SWFPut — Flash and HTML5 Video Plugin for WordPress

Ed Hynan <edhynan@gmail.com>

This 'README' serves as the main documentation for the **SWFPut** *WordPress* plugin, and as the conventional 'README' as well.

1. What is it?

SWFPut is a plugin for *WordPress*. It provides video player programs for the HTML video element and the flash plugin, and the means to configure the players with video sources and playback attributes. There are two separate components: the video players, and the *WordPress* plugin proper. The video players are delivered to site visitors by the plugin as either HTML5 video with a flash plugin program as fallback content, or inversely as flash video primary content with HTML5 video elements for fallback. The former arrangement is the default (as of version 2.1), but there is an option to enable the latter arrangement if flash is preferred. It is not necessary to provide for both types, one or the other type may be left out.

Video objects may be placed in posts and pages, or in the widget areas supported by your theme (i.e., the plugin includes a widget). Video is placed in posts and pages with a *shortcode*; if you do not know what a shortcode is, or do not want to deal with them, that's no problem. (In fact, it is preferable that the shortcodes *not* be hand-edited, and they will not be discussed in detail here.) The plugin adds to the post/page editor interfaces a button to insert video at the cursor, and an easy to use dialog-type setup interface similar to that used by *WordPress* core media, and also a full featured form in a "metabox" with additional advanced settings to setup and add, or edit, or delete video objects. The video widget has a similar full featured form for setup.

Video in posts may be displayed in the *WordPress* visual editor, if the default visual editor is used. If the default visual editor has been changed, for example by a plugin, the video display in the editor will probably not work. If there are problems, video display in the editor may be disabled on the **SWFPut** settings page.

SWFPut does not depend on *JavaScript* on the front end, but it is used for the HTML5 video player, and for proper resizing of the video display area. If *JavaScript* is not available, HTML5 video will have the interface provided by the web browser, and the flash video will be largely unaffected (it only uses *JavaScript* to check whether it is running on a mobile platform). "Responsive" display resizing will not be possible if *JavaScript* is not available, so reasonable dimensions should be given in the setup form.

The **SWFPut** flash video player has been coded to work well with the free *Gnash* web browser plugin, as well as the closed binary-only proprietary version in common use. As of this writing, *Gnash* does not handle **MP4** files well, even though it handles H.264 video and AAC audio if they are in an FLV container file. Therefore, it is a good idea to prepare FLV files for flash if you can.

2. Building From the Source

SWFPut is distributed from the *WordPress* plugin repository as a *ZIP* archive ready for installation on a *WordPress* site. Therefore, there is no need to build the plugin before use.

(You may skip forward to the **Usage** section if you don't intend to modify the player or plugin.)

All necessary sources are included in the distributed archive, but the necessary build tools are not. The plugin is maintained with a POSIX makefile, which in turn expects a POSIX Bourne shell. Additional requirements are the *PHP* command-line interface with the *Ming* extension to build the flash video player, *PERL* packages to minify *JavaScript*, *GNU* gettext for localization sources, *GNU* groff (and a small C program) to make the forms of this document, the portable

Info-ZIP zip command to make the archive, and various POSIX tools such as sed.

The actual plugin is composed of *PHP* code, and does not require compilation or link editing. The flash video player is a compiled program, but a binary is included in the installable package so that use does not require compilation by the user.

The Makefile default target builds as necessary and then creates the ZIP file. The Makefile is the build documentation.

3. Usage

SWFPut installs an item under the "Settings" menu named "SWFPut Plugin". Selecting that should produce the plugin's configuration page. The configuration page includes optional verbose help, and so it will not be described here.

In the post and page editors, the plugin adds a button, labelled "Add SWFPut Video," next to the "Add Media" button. This will insert a placeholder video at the cursor position. When selected, a pencil button will appear, which invokes a graphical setup dialog. This dialog is expected to be easy and intuitive, so it is not elaborated on here.

SWFPut also adds an interactive form in a new metabox with the title "SWFPut Video". This form has advanced settings not available in the graphical dialog. Directly under the title is a row of buttons. Under the row of buttons, the bulk of the form is placed in three sections entitled "Media", "Dimensions", and "Behavior". The title bar of each section has a button that will hide or show that section, which might help if the height of the form is greater than that of the display.

3.1. Form Buttons

- **Fill from post**: When the post (or page) already contains a **SWFPut** video object (i.e., shortcode), this will find it in the editor and fill the form with its details. A post may contain any number of **SWFPut** video objects. If there is more than one, then repeatedly using this button will cycle through each in turn.
- **Replace current in post**: When the form has been filled with the details of a video object using "**Fill from post**" (described above), or if it contains the details of a new video object that has just been added, the form items may be changed, and this button will edit the associated shortcode with the changes.
- **Delete current in post**: As described above for "**Replace current in post**", except that rather than changing the details of the shortcode, it is deleted.
- **Place new in post**: After making sure that the cursor (insertion point) in the editor is at the desired position, and filling out the form items, use this button to add a new shortcode (video).
- **Reset defaults**: Except for the "Caption" text field, all form items are set to default values, or cleared. It is assumed that text typed into the "Caption" field would be better maintained by hand, so that field is not cleared.

3.2. Form Sections

3.2.1. Media

- **Caption**: A video object is set in a page as an image would be, with the same border, and an optional caption, which may be set here. If this field is left blank, there will be no caption.
- **Flash video URL or media library ID**: A video URL may be given here, or an attachment ID valid for the *WordPress* database. Or more conveniently, this field may be set from the two drop-down lists described next. Acceptable protocols are *HTTP*, *HTTPS*, and *RTMP*. Support for *RTMP* is only partial and very limited. See "**Playpath (rtmp)**" below. Acceptable file (media) types are **FLV**, **MP4**.
- **Select flash video URL from uploads directory**: This is a drop-down list from which the URL field may be set. The *WordPress*

`uploads` directory is searched recursively for files with the suffixes **FLV**, **MP4**, and for each a URL is added to this list. This has the advantage that it will find files added by hand upload (rather than with the 'add media' interface) if they are placed in `uploads` or a directory under it.

- **Select ID for flash video from media library**: This is a drop-down list from which the URL field may be set, as above, with the difference that it searches the *WordPress* media database, and presents the suitable filenames with their media ID's.
- **HTML5 video URLs or media library ID's**: A series of URLs for the HTML5 video player. If more than one URL is given, they should be separated by the '`|`' (pipe) character. Each individual URL may be appended with an argument for the '`type`' attribute of the video element, separated from the URL by a '`?`' character (do not include the '`type`' keyword; give only the value that should appear between quotes in the type argument, and although many web examples show a space after the comma separating the video and audio codec names, browsers might reject the source because of the space, so it should be left out)[1]. If more than one is given they will appear in order. The browser should use the first type that it supports. The MP4, WEBM, and OGG types have varied support among web browsers, so ideally all three formats should be provided.
- **Select HTML5 video URL from uploads directory**: (See next item below.)
- **Select ID for HTML5 video from media library**: These selection items work much like the similarly named items pertaining to flash URLs, as described above. These show files with MP4 or OGG or OGV or WEBM extensions, suitable for the HTML5 video player. An important difference is that when you make a selection, the entry field is appended, rather than replaced, on the assumption that you are adding multiple resources for the necessary HTML5 video formats. When the URL field content is appended, a '`|`' (pipe) character is used as a separator. See "HTML5 video URLs or media library ID's" above.
- **Playpath (rtmp)**: If the URL field for flash video is given an **RTMP** URL, the 'playpath' is given here. Note that only the simplest RTMP connections are supported: those requiring only the playpath. This item has bearing on HTML5 video.
- **Url of initial image file (optional)**: A URL for a 'poster' image file may be placed here. When the player is loaded, if it is not set to play on load, this image is displayed until the play button is invoked. Accepted image types are **JPEG**, **PNG**, and **GIF**.
- **Load image from uploads directory**: This is a drop-down list from which the "**Url of initial image file (optional)**" field may be set. The *WordPress* `uploads` directory is searched recursively for files with the suffixes listed as acceptable for that field, and for each a URL is placed in this list. This has the advantage that it will find files added by hand upload (rather than with the 'add media' interface) if they are placed in `uploads` or a directory under it.
- **Load image ID from media library**: This is a drop-down list from which the "**Url of initial image file (optional)**" field may be set, as above, with the difference that it searches the *WordPress* media database, and presents the suitable filenames with their media ID's.
- **Use initial image as no-video alternate**: If an initial image was given, then also use it as fallback display when video is not supported. This

[1] For example:

```
videos/cat.mp4?video/mp4|
videos/cat.webm?video/webm;
codecs=vp8,vorbis|
videos/cat.ogv?video/ogg;
codecs=theora,vorbis.
```

option is on by default.

3.2.2. Dimensions

- **Pixel Width × Height**: Set these to the desired size of the video player's embedded window. This does not need to be the same as the display size of the video to be played, but the appearance will be best if the *aspect* of the player's display is the same as the display aspect of the video. For example, with a video of a size of 400×300 setting these to fields to 320×240 would look good, as the width:height ratios are the same.

  The player will scale the video to fit within its display, but it maintains the aspect ratio, so horizontal or vertical black (unused) areas will be visible if the aspect ratios do not match.

  These dimensions are not fixed unless scripts are disabled in the browser. Where scripting is available, the video display area is resized in concert with changes that the browser applies to the surrounding elements. This is particularly important on mobile platforms, where the browser will probably make extreme size adjustments for the small display size. This "responsive" behavior will depend on the current theme being responsive.

- **Mobile width**: This is to provide a width to use only if a mobile browser is detected; the height is automatically proportional, according to the regular "**Pixel Width × Height**" dimensions described above.

  This might be most useful for widgets placed on a 'sidebar,' because a sidebar might be placed below, rather than beside, the main content. In this case more effective display width might be available, and a wider display might be suitable. This feature is disabled with a value of '0,' which is the default, and has no effect if the browser provides a user agent string not known as a mobile platform.

- **Auto aspect**: This enables a feature meant to be helpful when the video to be played might have been prepared as DVD-Video (NTSC or PAL) for standard (non-widescreen) 4:3 display. Such video has non-square pixels; i.e., its actual width×height does not match its intended display aspect. With this check enabled, the video player will force display at 4:3 ratio if the video dimensions match one of the DVD-Video pixel sizes. This is not suitable for widescreen DVD-Video, which has one of the expected DVD-Video pixel sizes, but is meant to be displayed with a 16:9 aspect.

- **Display aspect**: Set the intended display aspect ratio in this field if you know that the video has non-square pixels. A value of 0 (zero) disables this field; otherwise, a value may be given as a decimal number (e.g., 1.33333333) or as a ratio using ':' or 'x' or '/' as separator (e.g., 4:3, or 16x9, or 20/11, etc.—several other characters will also be accepted as a separator, but it's sensible to use those listed here).

- **Pixel aspect**: Similar to "**Display aspect**" above, but this field takes the source (pixel) aspect ratio rather than the display aspect in the unlikely event that that value is more readily available. For example, video prepared for NTSC DVD at 720×480 pixels intended for standard (4:3) display has a pixel aspect ratio of 8:9, and at 352×240 a pixel aspect ratio of 10:11. As above, '0' disables this field.

3.2.3. Behavior

- **Alignment**: This is an exclusive selection with options 'left,' 'center,' 'right,' and 'none.' The default is *center*. This option will set the alignment of the video display within the page or post by adding a CSS class called one of '.alignleft,' '.aligncenter,' '.alignright,' or '.alignnone.' A *WordPress* theme should provide CSS definitions for the first three of those classes. The fourth, *.alignnone*, is meant to use the alignment that the

video markup inherits, whatever it might be. (If there is a CSS definition for *.alignnone* hopefully it will have a sensible effect.)

Because this option works with classes, which depend on CSS definitions, it is ultimately the user's *WordPress* theme that provide the effect of this option. For example, if a theme provides a left or right margin for the '.widget' class, then this plugin's video widget might not align according to this option.

- **Video preload**: This is an exclusive selection. HTML5 video allows a "preload" attribute with a value of "none" or "metadata" or "auto." This option provides those three values and one special selection: "per initial image." This special selection will use "none" if an "initial image file" is set (in the **Media** section of the form), or "metadata" if an initial image, or *poster*, is not set.

  The "metadata" selection allows the browser to fetch a small part of the video file with information such as video dimensions, duration, and the codec types[2]. This can be useful because a browser might also receive some of the video frames, and so it may display one frame as a 'poster.' (Whether a frame displayed this way is suitable is not certain.)

  If "none" is selected the browser will not fetch any of the video until it is played, and so without an initial image, the video region on the page will be solid black until played.

  The "auto" selection should be avoided unless you know what it does and that you need it. This is because with "auto" the browser may choose to fetch the entire video even before the visitor actively plays the video. Video files can be quite large, and a large unsolicited download might be unkind to your site's visitors; it might even cause a visitor additional charges depending on their connection service. Also consider your server and network load.

  The flash player does not have similar attributes, but will behave similarly with regard to an initial image: if one was not set, and the preload option is not "none," then the player will start playback and let it advance for a small random period, and then pause playback, leaving a visible frame to act as a 'poster.'

- **Initial volume**: The video player has a volume control that visitors can adjust. This field will set a default volume. If the web browser flash plugin is permitted to save values locally on a visitor's machine, then their adjustment will be saved, and will be used rather than the default when they visit again (or reload the page). User settings are not saved for the HTML5 video player presently.

- **Play on load**: This will cause the video to begin playing as soon as the player program is loaded. When this is not set, the player waits for the play button.

  Above, it was stated that if scripting is not available in the web browser, HTML5 video will have the default appearance and behavior provided by the web browser. The play on load option will not be in effect in that case, because while the video element can take an autoplay attribute, that cannot reliably work with the scripted HTML5 player since it might begin before the script gains control of the video element.

- **Loop play**: This will cause the medium to play again from the beginning each time it ends. When this is not set, the video plays once, and then pauses.

- **Hide control bar initially**: This will cause the control bar to hide a few seconds after the player loads (e.g., so that it does not obscure an initial image). Note that this also changes the control bar behavior in general: the bar will show whenever mouse movement is detected within the embedded window, and hide again

[2] In most of the supported video file types, the browser should be able to find the metadata without needing to retrieve too much data.

when there has been no mouse movement for a few seconds.

When this is not set, the control bar is left showing when the player loads, and thereafter is always shown when the mouse is within the embedded window, and is always hidden when the mouse is *detected* leaving the window (when the mouse is moved out of the player window with rapid motion the browser or plugin often fails to deliver a 'mouse has left' event to the player program, so hiding the bar is not always reliable).

- **Hide and disable control bar**: Enable this if the media should play through without interruption.
- **Allow full screen**: This enables a control bar button that will place the video in full-screen mode.
- **Control bar Height (30-60)**: The control bar height can be adjusted with this. The range 30-60 merely suggests useful sizes; it is not a fixed boundary. A good size would be influenced by the specified display dimensions on non-mobile platforms, but also by usability on mobile platforms. Test, please.

3.3. The Widget

The player can also be used as a widget. The "Appearance" menu "Widgets" item should produce the "Widgets" page which, after installation of **SWFPut**, should show "**SWFPut Video Player**" under "**Available Widgets**". After dragging this to a widget area the setup form should display (click the arrow near the title if necessary). The widget's form has the same items described under "**Form Sections**" above, although this form is not displayed in the three separate sections and does not have the buttons near the top. There is one additional item at the top of the widget form: a text field named "Widget title". Not surprisingly, the contents of that field will be displayed as a title above the widget on the pages that include the particular widget area used.

4. License

This program and all files included in the distribution archive are under the ***GNU GPL***, version 3. See the file `COPYING`, which should be present in the top-level directory of the distribution archive; or, see the license at `http://www.gnu.org/licenses/`.