SWFPut — Flash Video Player 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 the popular *WordPress* weblog software. It provides a video player program for the flash plugin and the means to configure an instance with a video source and playback attributes. There are two separate components: the flash video player, and the *WordPress* plugin proper. The video player is delivered to site visitors by the plugin in the traditional <object ...> block with the necessary arguments. Flash 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 administrative interface a full featured form to setup and add, or edit, or delete video objects, so the user does not need to be troubled with shortcodes (they will be visible in the editor; you will get used to them). The flash video widget has a similar full featured form.

The plugin does not add any *JavaScript* to the pages generated for your visitors, which might be helpful if you try to keep your pages useful to those who disable JavaScript in their browsers. (Such visitors might need to explicitly enable the flash web browser plugin, but that is another, unavoidable, issue.) JavaScript is used in the administrative interface for the forms and manipulation of shortcodes in the editor; but of course you must have JavaScript enabled when you log in to your *WordPress* site — this does not affect your visitors.

(Note that the **SWFPut** 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.)

2. Building From the Source

SWFPut is distributed as a *ZIP* archive prepared for installation on a *WordPress* site with the "Add New" item under the "Plugins" menu. Therefore, there is no need to build the package before use.

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

The actual plugin is composed of *PHP* code, and JavaScript for the administrative parts, and neither of those requires compilation or link editing. The flash video player is a compiled program, but binaries are included in the installable package so that use does not require compilation by the user. Of course, the source code is included and the binaries may be built if necessary (or desired). Compiling the flash program will require the *Ming* PHP extension. See the files `Makefile`, `mingtest/mingput.php`, and `mingtest/mainact.inc.php` if you wish to learn to build the player.

If you wish to change the JavaScript code, edit `js/formxed.dev.js`, rather than `js/formxed.js`. The latter is merely a 'minified' version of the former; see `Makefile`.

This file (README*) is built with the *GNU* roff *groff* with *ms* macros; see `docs/Makefile`.

The Makefiles require a *Unix*-like or *POSIX* system. The default target builds as necessary and then creates the ZIP file.

3. Usage

A logged in session is assumed.

SWFPut installed will add 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.

When editing posts or pages, below the editor the plugin will have placed an interactive form with the title "SWFPut Flash Video Shortcode". 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 the display.

3.1. Form Buttons

- **Fill form from editor**: When the post (or page) already contains a **SWFPut** flash 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 editor**: When the form has been filled with the details of a video object using "**Fill form from editor**" (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 (video object) with the changes.
- **Delete current in editor**: As described above for "**Replace current in editor**", except that rather than changing the details of the shortcode, it is deleted.
- **Place new in editor**: After making sure that the cursor (insertion point) in the editor is at the desired position, and setting the form items, use this button to add a new shortcode (video).
- **Reset default values**: 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.
- **Url or media library ID**: A fully qualified URL may be given here, or an 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** (video), and **MP3** (audio)[1].
- **Url from uploads directory**: This is a drop-down list from which the "**Url or media library ID**" field may be set. The *WordPress* `uploads` directory is searched recursively for files with the suffixes **FLV**, **MP4**, and **MP3**, and for each a URL is placed in this list. This has the advantage that it will find files added by hand (rather than with the 'add media' interface) if they are placed in `uploads` or a directory under it.

[1] For **MP3** files, you may try placing a video URL in the "**Playpath (rtmp)**" field to play along with the audio. If the video has an audio stream, that will mix in, so it should probably be a video only file. This is an experimental and **unsupported** feature.

- **Select ID from media library**: This is a drop-down list from which the "**Url or media library ID**" field may be set, as above, with the difference that it searches the *WordPress* media database, and presents the suitable filenames their media IDs.
- **Medium is audio**: To determine whether to play video or audio, the file suffix of the medium is checked. If the file suffix is not known, the medium is assumed to be video. If the medium is an mp3 audio file, but it does not have the **MP3** suffix, check this.
- **URLs for alternate HTML5 video**: An optional series of URLs for an alternate HTML5 video element that a web browser can try if flash video is not available or is not supported. 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, Firefox up to at least version 16 will reject the source because of the space, so it should be left out)[2]. If more than one is given they will appear in order. The browser should use the first type that it supports (if any, and some older versions of browsers might not consider any source but the first to appear). The MP4, WEBM, and OGG types have varied support among web browsers, so ideally all three formats should be provided.
- **Playpath (rtmp)**: If the "**Url or media library ID**" field is given an **RTMP** URL, the 'playpath' is given here. Note that only the simplest RTMP connections are supported: those requiring only the playpath.
- **Url of initial image file (optional)**: An **HTTP** or **HTTPS** URL for an 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**, and **SWF**[3].
- **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 (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 their media IDs.
- **Use initial image as non-flash alternate**: If an initial image was given, then also use it as an alternative display when flash is not supported. If optional HTML5 video URLs were given, they will remain the first alternate, and the initial image will be an alternate to that. This option is on by default.

[2] For example:
`foo/alternate.mp4?video/mp4|foo/alternate.webm?video/webm;codecs=vp8,vorbis|foo/alternate.ogv?video/ogg;codecs=theora,vorbis`.

[3] **SWF** files may be stills or animations. This type is a flash *program* that the flash web browser plugin will be executing, so, of course, it must not interfere with the **SWFPut** player. Test thoroughly if this is used.

3.2.2. Dimensions

- **Pixel Width × Height**: set these to the desired size of the 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, if set for a video with a size of 400×300, then setting these to fields to 320×240 would look good (the width:height ratio is the same). In any case, 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. It is also important to note that the browser will (probably) honor a maximum width for a page column set in *CSS*, and force the browser's flash plugin to display at a smaller width than the user specified. For example, if you set these fields to 640×480, but the column in which posts appear has a width of 600, the display would be at 600×480. In such a case, you might try 600×450 to maintain the aspect so that the video matches display size. The above assumes a 4:3 aspect; you would use the correct numbers, of course.
- **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, of 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

- **Initial volume**: The video player has a volume control that visitors can adjust, but 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).
- **Play on load**: This will cause the video (or audio) to begin playing as soon as the player program is loaded. When this is not set, the player waits for the play button.
- **Loop play**: This will cause the medium to play again from the beginning each time it ends. When this is not set, the media 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 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 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 (20-50)**: The control bar height can be adjusted with this. Actually, there is a fixed number precompiled sizes in the player binaries that are distributed with the package, and this field causes the nearest size to be selected. If the *WordPress* host's *PHP* configuration includes the *Ming* extension, then you may select an option on the **SWFPut** configuration page called "Dynamic SWF generation". This produces the video player on the fly, and in this case the value of this field builds the control bar at the requested size rather than using a near match.

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 Flash Video**" 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.

The description under "**Pixel Width** × **Height**" above mentioned the maximum column widths set by the theme's CSS, and how the video window width might be limited by the those. The same applies to widget areas, but the maximum width is likely much less than that in the posts area. These fields have defaults in the widget form that have been useful with the themes distributed with *WordPress*, but the user should certainly do some testing along with the chosen theme.

4. License