**SWFUpload v2.2.0.1 中文帮助手册**

**TOC**

1. [关于SWFUpload](http://www.phptogether.com/swfuploadoc/#swfupload)
2. [SWFUpload Version 2](http://www.phptogether.com/swfuploadoc/#v2)
3. [概述](http://www.phptogether.com/swfuploadoc/#overview)
4. [快速指南](http://www.phptogether.com/swfuploadoc/#gettingstarted)
5. [SWFUpload JavaScript对象](http://www.phptogether.com/swfuploadoc/#javascriptobject)
	1. [构造函数](http://www.phptogether.com/swfuploadoc/#constructor)
	2. [全局变量和常量](http://www.phptogether.com/swfuploadoc/#globals)
		1. [instances](http://www.phptogether.com/swfuploadoc/#instances)
		2. [movieCount](http://www.phptogether.com/swfuploadoc/#movieCount)
		3. [QUEUE\_ERROR](http://www.phptogether.com/swfuploadoc/#queue_error)
		4. [UPLOAD\_ERROR](http://www.phptogether.com/swfuploadoc/#upload_error)
		5. [FILE\_STATUS](http://www.phptogether.com/swfuploadoc/#file_status)
		6. [BUTTON\_ACTION](http://www.phptogether.com/swfuploadoc/#button_action)
		7. [CURSOR](http://www.phptogether.com/swfuploadoc/#button_cursor)
		8. [BUTTON\_WINDOW\_MODE](http://www.phptogether.com/swfuploadoc/#button_window_mode)
	3. [属性](http://www.phptogether.com/swfuploadoc/#properties)
		1. [customSettings](http://www.phptogether.com/swfuploadoc/#customSettings)
		2. [movieName](http://www.phptogether.com/swfuploadoc/#movieName)
	4. [方法](http://www.phptogether.com/swfuploadoc/#methods)
		1. [addSetting](http://www.phptogether.com/swfuploadoc/#addSetting) (不建议使用)
		2. [getSetting](http://www.phptogether.com/swfuploadoc/#getSetting) (不建议使用)
		3. [retrieveSetting](http://www.phptogether.com/swfuploadoc/#retrieveSetting) (在v2.1.0版本中被移除)
		4. [destroy](http://www.phptogether.com/swfuploadoc/#destroy) (在v2.1.0版本中添加)
		5. [displayDebugInfo](http://www.phptogether.com/swfuploadoc/#displayDebugInfo)
		6. [selectFile](http://www.phptogether.com/swfuploadoc/#selectFile)
		7. [selectFiles](http://www.phptogether.com/swfuploadoc/#selectFiles)
		8. [startUpload](http://www.phptogether.com/swfuploadoc/#startUpload)
		9. [cancelUpload](http://www.phptogether.com/swfuploadoc/#cancelUpload)
		10. [stopUpload](http://www.phptogether.com/swfuploadoc/#stopUpload)
		11. [getStats](http://www.phptogether.com/swfuploadoc/#getStats)
		12. [setStats](http://www.phptogether.com/swfuploadoc/#setStats)
		13. [getFile](http://www.phptogether.com/swfuploadoc/#getFile)
		14. [addPostParam](http://www.phptogether.com/swfuploadoc/#addPostParam)
		15. [removePostParam](http://www.phptogether.com/swfuploadoc/#removePostParam)
		16. [addFileParam](http://www.phptogether.com/swfuploadoc/#addFileParam)
		17. [removeFileParam](http://www.phptogether.com/swfuploadoc/#removeFileParam)
		18. [setUploadURL](http://www.phptogether.com/swfuploadoc/#setUploadURL)
		19. [setPostParams](http://www.phptogether.com/swfuploadoc/#setPostParams)
		20. [setFileTypes](http://www.phptogether.com/swfuploadoc/#setFileTypes)
		21. [setFileSizeLimit](http://www.phptogether.com/swfuploadoc/#setFileSizeLimit)
		22. [setFileUploadLimit](http://www.phptogether.com/swfuploadoc/#setFileUploadLimit)
		23. [setFileQueueLimit](http://www.phptogether.com/swfuploadoc/#setFileQueueLimit)
		24. [setFilePostName](http://www.phptogether.com/swfuploadoc/#setFilePostName)
		25. [setUseQueryString](http://www.phptogether.com/swfuploadoc/#setUseQueryString)
		26. [setDebugEnabled](http://www.phptogether.com/swfuploadoc/#setDebugEnabled)
		27. [setButtonImageURL](http://www.phptogether.com/swfuploadoc/#setButtonImageURL) (在v2.2.0版本中添加)
		28. [setButtonDimensions](http://www.phptogether.com/swfuploadoc/#setButtonDimensions) (在v2.2.0版本中添加)
		29. [setButtonText](http://www.phptogether.com/swfuploadoc/#setButtonText) (在v2.2.0版本中添加)
		30. [setButtonTextStyle](http://www.phptogether.com/swfuploadoc/#setButtonTextStyle) (在v2.2.0版本中添加)
		31. [setButtonTextPadding](http://www.phptogether.com/swfuploadoc/#setButtonTextPadding) (在v2.2.0版本中添加)
		32. [setButtonDisabled](http://www.phptogether.com/swfuploadoc/#setButtonDisabled) (在v2.2.0版本中添加)
		33. [setButtonAction](http://www.phptogether.com/swfuploadoc/#setButtonAction) (在v2.2.0版本中添加)
		34. [setButtonCursor](http://www.phptogether.com/swfuploadoc/#setButtonCursor) (在v2.2.0版本中添加)
	5. [事件](http://www.phptogether.com/swfuploadoc/#events)
		1. [flashReady](http://www.phptogether.com/swfuploadoc/#flashReady)
		2. [swfUploadLoaded](http://www.phptogether.com/swfuploadoc/#swfUploadLoaded)
		3. [fileDialogStart](http://www.phptogether.com/swfuploadoc/#fileDialogStart)
		4. [fileQueued](http://www.phptogether.com/swfuploadoc/#fileQueued)
		5. [fileQueueError](http://www.phptogether.com/swfuploadoc/#fileQueueError)
		6. [fileDialogComplete](http://www.phptogether.com/swfuploadoc/#fileDialogComplete)
		7. [uploadStart](http://www.phptogether.com/swfuploadoc/#uploadStart)
		8. [uploadProgress](http://www.phptogether.com/swfuploadoc/#uploadProgress)
		9. [uploadError](http://www.phptogether.com/swfuploadoc/#uploadError)
		10. [uploadSuccess](http://www.phptogether.com/swfuploadoc/#uploadSuccess)
		11. [uploadComplete](http://www.phptogether.com/swfuploadoc/#uploadComplete)
		12. [debug](http://www.phptogether.com/swfuploadoc/#debug)
	6. [SWFUpload实用对象](http://www.phptogether.com/swfuploadoc/#utility)
		1. [Settings Object](http://www.phptogether.com/swfuploadoc/#settingsobject)（设置对象）
		2. [设置说明](http://www.phptogether.com/swfuploadoc/#settingsdescription)
		3. [File Object](http://www.phptogether.com/swfuploadoc/#fileobject)（文件对象）
		4. [Stats Object](http://www.phptogether.com/swfuploadoc/#statsobject)（统计对象）
6. [SWFUpload插件](http://www.phptogether.com/swfuploadoc/#plugins)
7. [兼容性和bug](http://www.phptogether.com/swfuploadoc/#knownissues)

**关于SWFUpload**

[SWFUpload](http://www.swfupload.org/)是一个由 [Vinterwebb.se](http://www.vinterwebb.se/)原创开发的客户端文件上传工具。它采用Flash和JavaScript结合的方式来提供文件上传功能，而浏览器仅仅只需提供<input type="file" />标签。

SWFUpload的主要功能如下：

* 允许一次上传多个文件。
* 采用AJAX方式上传，页面无须刷新。
* 上传过程中的事件控制。
* 类使用了名称空间，稳定兼容其他JavaScript库(例如jQuery, Prototype等。)。
* 支持Flash 9和Flash 10。(2.2.0不再支持Flash8)

由于设计理念上见解和宗旨的差异，SWFUpload和其它基于Flash的上传工具不一样。SWFUpload将UI的表现交给浏览器，以给开发者尽可能多的发挥余地。开发者可以使用XHTML, CSS和JavaScript来装饰上传UI以满足他们站点的需要。上传状态的更新是通过一系列简单的JavaScript事件完成的。开发者可以在文件上传过程中使用这些事件来更新页面。

遗憾的是，Flash Player 10强制我们在一个flash电影中必须设置一个按钮来触发文件浏览窗口。SWFUpload通过JavaScript仍然向开发者提供按钮和文本覆盖的功能。

**SWFUpload v2**

SWFUpload拥有更先进的功能，更高的稳定性，Flash Player错误解决方案以及一系列非常有用的插件。新功能如下：

* 安全兼容Flash Player 10。
* 在上传时发送POST值。
* 按文件发送POST值。
* 完整的事件集合。
* 所有设置可动态更新。
* 从服务器获取数据。
* 不需要取消操作就可以停止上传。
* 可以任意顺序上传文件。
* 可以选择单个或多个文件。
* 可以对上传队列中的文件总数、已上传文件的大小和允许上传的文件大小做限制。
* 正确处理0字节文件。
* 上传之前可以通过事件进行校验。
* 对Flash和浏览器进行bug兼容。

**概述**

**HTML上传**

标准的HTML上传提供一个输入框和一个供用户选择单个文件的按钮。文件使用表单提交，整个文件必须在下一个页面显示之前上传完成，在上传之前不能对文件进行校验(如文件大小限制和是否有效的扩展名等)。上传时提供给用户的反馈信息非常少。

标准HTML上传的使用方式简单、直接并且支持几乎所有的浏览器。

**SWFUpload**

SWFUpload使用一个Flash电影来处理文件选择和上传。Flash电影提供一个自定义按钮，该按钮可以激活Flash的高级文件选择窗口。该文件选择窗口允许用户选择单个或多个文件，文件类型可以被限制，如此一来，用户就只能选择被允许的文件 (如\*.jpg;\*.gif)。

一旦被选择，每个文件就已经通过校验并且进入上传队列。当文件被Flash上传时，由开发者控制，用来更新页面UI的若干个JavaScript事件就会被触发，这些事件允许你提供上传的实时状态和错误消息。

上传完成的文件将脱离当前页面和表单并且被提交。每个文件都是单独被上传的，以此来使服务端处理上传的脚本不需要太复杂。由于Flash提供上传功能，因此页面无须被提交或重载。SWFUpload的使用方式更像一个AJAX应用而非标准的HTML表单，页面中的表单处理是和文件上传分开的。

**快速指南**

SWFUpload不是一个可拖放的上传工具。使用它，需要掌握JavaScript和DOM的相关知识。一些demo展示了SWFUpload可以做到的事情以及它是如何完成普通任务的。

SWFUpload由以下4个方面组成：

1. 初始化和设置(JavaScript)
2. JavaScript库: SWFUpload.js
3. Flash控制: swfupload.swf
4. 事件处理(JavaScript)

在部署SWFUpload时，大多数问题都是由错误设置、糟糕的事件处理、Flash/浏览器bug或者服务器配置引起的。

**初始化和设置**

SWFUpload必须在页面中初始化，这通常在window.onload事件中完成。SWFUpload构造函数使用一个设置对象作为参数，设置对象可以对象原型的形式被直接传递给构造函数。

必须使用一个对初始化的SWFUpload对象的引用，因为我们需要它来开始上传和控制SWFUpload。

**Example:**使用一个对象原型来初始化SWFUpload

var swfu;

window.onload = function () {

 swfu = new SWFUpload({

 upload\_url : "http://www.swfupload.org/upload.php",

 flash\_url : "http://www.swfupload.org/swfupload.swf",

 file\_size\_limit : "20 MB"

 });

};

**Example:**使用一个存储在变量中的设置对象来初始化SWFUpload

var swfu;

window.onload = function () {

 var settings\_object = {

 upload\_url : "http://www.swfupload.org/upload.php",

 flash\_url : "http://www.swfupload.org/swfupload.swf",

 file\_size\_limit : "20 MB"

 };

 swfu = new SWFUpload(settings\_object);

};

**JavaScript库**

JavaScript库文件(swfupload.js)必须被包含在用户上传页面中。

一旦SWFUpload对象被创建，开发者就可以访问一些函数来控制SWFUpload。

**Example:**添加SWFUpload.js到一个页面

<script type="text/javascript" src="http://www.swfupload.org/swfupload.js"></script>

**Example:**使用定义的设置初始化SWFUpload。

var swfu = new SWFUpload({

 upload\_url : "http://www.swfupload.org/upload.php",

 flash\_url : "http://www.swfupload.org/swfupload\_f9.swf",

 button\_placeholder\_id : "spanSWFUploadButton"

});

**Flash控制器**

SWFUpload JavaScript库动态地载入Flash控制器(swfupload.swf)。

Flash控制文件的路径必须在初始化时通过SWFUpload设置对象给出。

Flash控制器是一个Flash小电影，它被用来处理文件浏览、校验和上传。它作为一个按钮呈现在页面中，并且与JavaScript相联系将上传状态和其他事件通知给浏览器。

**事件处理器**

开发者必须创建一系列的JavaScript函数来处理SWFUpload事件，当不同的事件发生时这些函数将被调用。

通过处理SWFUpload事件，开发者可以根据上传过程、错误信息和上传完成提供反馈信息。开发者不应该重写存储在SWFUpload.prototype中的函数，而是应该创建你自己的一系列函数并且在设置对象中向它们传递参数。

**Example:**SWFUpload事件处理器和初始化。

//上传开始事件处理器，这个函数作为一个变量值在设置对象中被赋予upload\_start\_handler

var myCustomUploadStartEventHandler = function (file) {

 var continue\_with\_upload;

 if (file.name === "the sky is blue") {

 continue\_with\_upload = true;

 } else {

 continue\_with\_upload = false;

 }

 return continue\_with\_upload;

};

//上传成功事件处理器，这个函数作为一个变量值在设置对象中被赋予upload\_success\_handler

var myCustomUploadSuccessEventHandler = function (file, server\_data, receivedResponse) {

 alert("The file " + file.name + " has been delivered to the server. The server responded with " + server\_data);

};

//创建SWFUpload Object

var swfu = new SWFUpload({

 upload\_url : "http://www.swfupload.org/upload.php",

 flash\_url : "http://www.swfupload.org/swfupload.swf",

 file\_size\_limit : "200 MB",

 upload\_start\_handler : myCustomUploadStartEventHandler,

 upload\_success\_handler : myCustomUploadSuccessEventHandler

});

**SWFUpload JavaScript对象**

**构造函数**

**SWFUpload(***settings object***)**

**返回值:** 一个SWFUpload实例

var swfupload\_instance = new SWFUpload(settings\_object);

**全局变量和常量**

一些与SWFUpload对象相关的全局变量和常量对高级应用和错误处理很有帮助，它们只是可读而不可修改的。

**SWFUpload.instances**

SWFUpload.instances是一个含有“对页面中所载入的每个SWFUpload实例的引用”的对象 。Flash播放器依赖这个对象来调用相应的事件处理器。SWFUpload.instances按照movieName属性进行索引。

**SWFUpload.movieCount**

SWFUpload.movieCount是一个用以跟踪记录SWFUpload实例数目的全局变量 ，它可以帮助确保每个flash电影都被给予一个唯一的movieName。

**SWFUpload.QUEUE\_ERROR**

SWFUpload.QUEUE\_ERROR是一个含有上传队列错误代码的单一对象。它一般用于确定在fileQueueError事件中发送了哪条错误代码。

* QUEUE\_LIMIT\_EXCEEDED - 说明用户选择的文件数目超出了限制。一旦上传队列中的文件被更新并且移除出队列（译者注：这是针对于上传队列已满的情况），用户就又可以向上传队列增加文件。
* FILE\_EXCEEDS\_SIZE\_LIMIT - 说明被选择的文件大小超出了file\_size\_limit的限制。
* ZERO\_BYTE\_FILE - 说明被选择的文件是空的，Flash播放器无法处理空文件，因此该文件被拒绝。Windows快捷方式文件也可能导致这个错误。
* INVALID\_FILETYPE - 说明选择的文件类型（后缀）不符合file\_types的设置。用户可以通过手工输入文件名以绕过file\_types的限制。

**SWFUpload.UPLOAD\_ERROR**

SWFUpload.UPLOAD\_ERROR是一个含有上传错误代码的单一对象，它主要被用来确定在uploadError事件中发送了哪条错误代码。

* HTTP\_ERROR - 尝试上传，但是服务器没有返回200状态代码（译者注：200表示服务器已成功处理了请求）。
* MISSING\_UPLOAD\_URL - upload\_url没有被设置。
* IO\_ERROR - 当在读取或上传文件时，发生了错误，通常发生在服务器意外断开连接时。
* SECURITY\_ERROR - 上传动作违反了一个安全限制，该错误很少发生。
* UPLOAD\_LIMIT\_EXCEEDED - 用户尝试上传的文件数量超出了file\_upload\_limit的限制。
* UPLOAD\_FAILED - 在尝试初始化上传时发生了错误，该错误很少发生。
* SPECIFIED\_FILE\_ID\_NOT\_FOUND - 传递一个文件ID以开始上传，但是该ID未被找到。
* FILE\_VALIDATION\_FAILED - uploadStart事件返回了false。
* FILE\_CANCELLED - cancelUpload方法被调用。
* UPLOAD\_STOPPED - stopUpload方法被调用。

**SWFUpload.FILE\_STATUS**

SWFUpload.FILE\_STATUS是一个含有文件状态代码的单一对象，它可以用来检查文件状态。

* QUEUED - 说明当前文件正在队列中等待上传。
* IN\_PROGRESS - 说明当前文件正在被上传。
* ERROR - 说明当前文件引起了一个队列错误或者上传错误。
* COMPLETE - 说明当前文件已经被成功上传至服务器。
* CANCELLED - 说明当前文件已经被cancelUpload方法取消上传。

**SWFUpload.BUTTON\_ACTION**

SWFUpload.BUTTON\_ACTION是一个含有按钮行为代码的单一对象。 它与button\_action设置搭配使用，以决定基于Flash的文件对话按钮的行为。

* SELECT\_FILE - 当Flash按钮被点击时，文件对话只允许选择一个文件。
* SELECT\_FILES - 当Flash按钮被点击时，文件对话允许选择多个文件。
* START\_UPLOAD - 当Flash按钮被点击时，第一个进入队列的文件将被上传。

**SWFUpload.CURSOR**

SWFUpload.CURSOR是一个含有按钮光标（译者注：鼠标）代码的单一对象。它和button\_cursor搭配使用，以决定当鼠标划过Flash按钮时的形态。

* ARROW - 光标将呈现为一个箭头。
* HAND - 光标将呈现为一个手指/手掌。

**SWFUpload.WINDOW\_MODE**

SWFUpload.WINDOW\_MODE是一个含有Flash电影的wmode参数的单一对象。 它用来告知浏览器如何呈现Flash电影。

某些WINDOW\_MODE/WMODE设置在一些浏览器中可能会产生问题。查看[兼容性和bug](http://www.phptogether.com/swfuploadoc/#knownissues)。

* WINDOW是默认的模式，Flash电影将被置于页面中其他所有元素之上。
* OPAQUE将允许页面中的其他元素覆盖Flash按钮。
* TRANSPARENT设置按钮背景呈现为透明的，这样位于按钮下方的HTML元素就可以透过按钮被显示出来。

**属性**

下面的属性列表是为用户使用准备的，使用其他的属性或者修改只读属性可能会使SWFUpload运行不正常。

**customSettings (读写)**

customSettings属性是一个空的JavaScript对象，它可以被用来存储与SWFUpload实例相关的数据。customSettings对象的内容可以通过设置custom\_settings来初始化。

**使用举例:**

//使用一些自定义设置来初始化SWFUpload

var swfu = new SWFUpload({

 custom\_settings : {

 custom\_setting\_1 : "custom\_setting\_value\_1",

 custom\_setting\_2 : "custom\_setting\_value\_2",

 custom\_setting\_n : "custom\_setting\_value\_n",

 }

});

swfu.customSettings.custom\_setting\_1 = "custom\_setting\_value\_1"; //修改一个已有的设置

swfu.customSettings.myNewCustomSetting = "new custom setting value"; //新增一个自定义设置

//使用一个全新的对象来重写customSettings

swfu.customSettings = {

 custom\_setting\_A : "custom\_setting\_value\_A",

 custom\_setting\_B : "custom\_setting\_value\_B"

};

存储在customSettings对象中的值可以在事件处理器中被访问到：

function uploadSuccess(file, serverData, receivedResponse) {

 if (this.customSettings.custom\_setting\_A === true) {

 alert("Checked the custom setting!");

 }

}

**movieName**

包含SWFUpload实例所对应电影的唯一名称。这个值将被传递给Flash来帮助与JavaScript进行通信。除此之外，它还用来索引SWFUpload.instances数组中的实例。你不应该修改movieName的值。

**方法**

以下方法可以被用来操作TSWFUpload。一些方法与DOM元素点击事件绑定在一起，其他方法则只能在SWFUpload事件处理器内部使用。

**object addSetting(***setting\_name***,** *value***,** *default\_value***)**

**已过期** addSetting函数用来进行自定义设置 。如果value未定义，那么将使用默认值。此函数在SWFUpload初始化时被调用，使用addSetting方法更新设置不会改变Flash控制器的原有设置。

addSetting返回存储在设置对象中的值 。

**object getSetting(***setting\_name***)**

**已过期** getSetting函数用来获取特定设置项的值。 如果设置项没有被找到，将返回一个空字符串。

**object retrieveSetting(***setting\_value***,** *default\_value***)**

**在v2.1.0版本中被移除** retrieveSetting函数和 addSetting函数很相似，但是retrieveSetting函数不会修改内部设置对象。 retrieveSetting将返回设置项的值，如果该设置项没有定义，那么它将返回默认值。

这是一个很实用的函数。

**bool destroy()**

**在v2.1.0版本中添加**

移除SWF的DOM元素然后注销SWFUpload的内部引用。它被用来移除页面中的SWFUpload实例，除此之外，该方法还会试图阻止IE浏览器的内存溢出问题。

如果元素被成功移除，该函数返回true；如果发生错误，将返回false，并且SWFUpload实例和DOM元素将前后不一致。

**void displayDebugInfo()**

displayDebugInfo方法使用debug事件输出SWFUpload的设置项。如果debug被设置为'true',这个函数将在初始化之后自动被调用。

**void selectFile()**

**已过期。与Flash Player 10不兼容。**

selectFile函数将使Flash控制器显示出一个文件选择对话窗口，可以从对话窗口中选择单个文件。

调用selectFile函数将引发一系列的文件相关事件。

**void selectFiles()**

**已过期。与Flash Player 10不兼容。**

selectFiles函数将使Flash控制器显示出一个文件选择对话窗口，可以从对话窗口中选择多个文件。

调用selectFile函数将引发一系列的文件相关事件。

**void startUpload(***file\_id***)**

startUpload方法将根据file\_id来开始上传一个文件。 如果file\_id参数被忽略或者未定义，那么队列中的第一个文件将被上传。

调用startUpload函数将引发一系列的上传相关事件。

**void cancelUpload(***file\_id***,** *trigger\_error\_event***)**

cancelUpload方法根据file\_id参数来取消特定文件的上传动作，之后该文件将被移除出上传队列。

如果file\_id参数被忽略或者未定义， 那么队列中的第一个文件将被取消上传。

trigger\_error\_event参数是可选的，如果该参数设置为false，那么uploadError事件将不会触发。

**void stopUpload()**

stopUpload函数停止正在被上传的文件并且将重整上传队列。

正在被上传的文件被停止后，uploadError事件将被触发。如果当前没有文件被上传，将不会发生任何动作，事件也不会触发。

**object getStats()**

获取stats（统计）对象。

**void setStats(***stats\_object***)**

Stats对象可以被修改。如果你想在上传完成之后，改变成功上传的文件数量或者上传中产生的错误，这一特效将非常有用。

**object getFile(***file\_id***,** *index***)**

getFile可以用来从文件队列获取一个文件对象，你可以通过传入一个文件ID(文件对象的ID属性)或一个文件索引(文件对象的index【索引】属性)来获取。

当使用file\_id时，只能获取上传队列中的文件。如果文件没有被找到，将返回null。

当使用index时，所有文件都可以供获取。(包括那些产生队列错误而不在队列中的文件)。如果index超出了范围，将返回null。

**void addPostParam(***name***,** *value***)**

addPostParam将为文件上传添加name/value键值对，这些数据通过POST方法被发送。

name/value键值对将被添加至post\_params设置中。

**void removePostParam(***name***)**

removePostParam将为文件上传从POST数据中移除name/value键值对。

post\_params设置中的name/value键值对也将被移除。

**bool addFileParam(***file\_id***,** *name***,** *value***)**

addFileParam根据file\_id参数添加name/value键值对，这些数据将通过POST方法被发送。

name/value键值对只会和相应的文件一起被发送。如果要将所有的上传文件与它们相对应的name/value键值对一起发送，可以设置post\_param。

**bool removeFileParam(***file\_id***,** *name***)**

removeFileParam函数从一个文件上传移除由addFileParam函数添加的name/value键值对。

如果name/value键值对没有被找到，将返回 'false'。

**void setUploadURL(***url***)**

动态地修改upload\_url设置。

**void setPostParams(***param\_object***)**

动态地修改post\_params设置，任何之前设置的值都将被覆盖。param\_object应该是一个单一的JavaScript对象，所有的name和value必须是字符串。

**void setFileTypes(***types***,** *description***)**

动态地更新file\_types和file\_types\_description的设置,2个参数都是必须的。

**void setFileSizeLimit(***file\_size\_limit***)**

动态地修改file\_size\_limit设置。此设置对队列中所有将被上传的文件均有效。file\_size\_limit参数将接受一个单位 ，可用的单位有：B, KB, MB, and GB，默认单位是KB。

使用举例: 2147483648 B, 2097152, 2097152KB, 2048 MB, 2 GB

**void setFileUploadLimit(***file\_upload\_limit***)**

动态地修改file\_upload\_limit设置。如果设置为0，这说明“无限制”。

**void setFileQueueLimit(***file\_queue\_limit***)**

动态地修改file\_queue\_limit设置。如果设置为0，这说明“无限制”。

**void setFilePostName(***file\_post\_name***)**

动态地修改file\_post\_name设置。Linux系统中，Flash播放器将忽略该设置。

**void setUseQueryString(***use\_query\_string***)**

动态地修改use\_query\_string设置。当设置为true，将强制SWFUpload通过网址传值来发送POST参数，而不是通过POST数组。use\_query\_string参数必须是一个布尔值。

**void setDebugEnabled(***debug\_enabled***)**

动态地启用或者禁用debug输出，debug\_enabled参数应该是一个布尔值。

**void setButtonImageURL(***url***)**

动态改变Flash按钮的背景图。图片的路径必须相对于swfupload.swf文件，或者以斜杠开始的绝对路径(如/)，或者一个完整的url地址(如http://www.swfupload.org/buttonImage.png)。任何Flash支持的图像格式都可以被载入，最常见的图片格式为jpg, gif和png。

按钮背景图最好是一个按钮浮标(或者是几张图片堆叠在一起)，通过将图片上移或者下移以显示图片的不同部分来表现按钮的所有状态。这些状态包括：普通，鼠标划过，点击，禁用。你可以查看demo中的示例按钮。

**void setButtonDimensions(***width***,** *height***)**

动态改变Flash按钮的宽和高，宽、高值必须是数字型，并且不能含有任何单位。height参数的值应该是整个按钮背景图高度的1/4，这样按钮浮标才可以正确显示各个状态。

**void setButtonText(***text***)**

设置显示在Flash按钮上的文本。如果文本太长超出了按钮，将进行截取。

可以用Flash播放器支持的HTML标签来定义文本样式。([Adobe文档](http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/text/TextField.html#htmlText))

**void setButtonTextStyle(***css\_style\_text***)**

设置应用于Flash按钮文本的CSS样式，CSS的格式必须遵循Flash播放器文档中的要求。([Adobe文档](http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/text/StyleSheet.html))

此方法中定义的样式类，在button\_text设置中可以通过HTML标签引用。

**void setButtonTextPadding(***left***,** *top***)**

设置Flash按钮文本的顶边距和左边距，参数值可以是负数。

**void setButtonDisabled(***isDisabled***)**

当参数设置为'true'时，将把Flash按钮的状态设置为禁用，任何点击都无效。

**void setButtonAction(***buttonAction***)**

设置当Flash按钮被点击时所进行的动作，可选的动作可以从[BUTTON\_ACTION](http://www.phptogether.com/swfuploadoc/#button_action)中获得。

**void setButtonCursor(***buttonCursor***)**

设置当鼠标划过Flash按钮时的光标形态。可选的光标形态可以从[CURSOR](http://www.phptogether.com/swfuploadoc/#button_action)中获得。

**事件**

**译者注：**文档中给出的事件参数只是对其的描述，而不是既定的参数形式。在你的代码中可以自定义，如：fileQueued(file object)，在你的代码中可以是fileQueued(fobj)或者fileQueued(aaa))。

SWFUpload在操作中可以触发多个事件。这些事件可以被开发者用来更新UI，改变SWFUpload的行为或者报告错误。

所有的SWFUpload事件都将根据SWFUpload实例的上下文关系而调用。 比如，'this'对象是一个对触发当前事件的SWFUpload实例的引用。

SWFUpload事件只能在对象初始化时通过在设置对象中分配事件处理器函数来设置，你不应该重写属于SWFUpload.prototype对象的内部函数。

在一个文件上传中，事件通常以如下顺序被调用(上传事件序列)：

* uploadStart
* uploadProgress (当文件上传时被反复调用)
* uploadError (如果有错误发生就会被调用，文件上传会被取消或者停止)
* uploadSuccess (上传成功完成，可以获取从服务器返回的数据)
* uploadComplete (一个文件上传完毕，SWFUpload准备上传下一个文件)

**flashReady()**

flashReady是一个内部事件，它不能被重写。Flash控制器通过调用此事件来通知SWFUpload，告诉它Flash电影已经载入并且准备接受指令。

**swfUploadLoaded()**

swfUploadLoaded事件由flashReady触发，它是一个可设置的事件。swfUploadLoaded事件将告诉你，现在调用SWFUpload的方法是安全的。

**fileDialogStart()**

fileDialogStart事件在selectFile方法或selectFiles方法被调用后触发。在文件选择对话窗口出现之间，此事件就会立即触发。但是，此事件直到对话窗口关闭之后才会执行。

**fileQueued(***file object***)**

在文件选择对话窗口关闭之后，被选择的的每个文件都会触发fileQueued事件。

**fileQueueError(***file object***,** *error code***,** *message***)**

在文件选择对话窗口关闭之后，未进入上传队列中的每个文件都会触发fileQueueError事件。 一个文件未能进入上传队列的原因有很多，如：文件太大，文件为空，队列中的文件已满等。

通过error code参数可以识别引起队列错误的原因，error code对应于一个SWFUpload.QUEUE\_ERROR常量。

**fileDialogComplete(***number of files selected***,** *number of files queued***,** *total number of files in the queue***)**

fileDialogComplete事件在文件选择对话窗口关闭并且所有被选择文件都经过处理（校验）后 触发。'number of files queued'表示从文件选择窗口中选择的文件数量(区分实际进入上传队列中的文件数量，译者注：从之前章节的介绍中，我们得知并不是所有被选择的文件都会进入上传队列，比如你选择了5个，但是实际被上传的只有4个。但是根据译者测试，最后两个参数的值是一样的，读者如果发现了区别请告知)。

如果你想要自动开始上传，可以在此事件中调用'this.startUpload()'方法。

**uploadStart(***file object***)**

uploadStart事件在文件上传之前立即被调用。 此事件可以让你在文件被上传之前进行最后的校验、添加POST参数或者其他工作。

在uploadStart事件中你可以返回一个'false'值来取消上传，如果你返回'true'或者不返回任何值，上传将继续。返回'false'将触发uploadError事件。

**uploadProgress(***file object***,** *bytes complete***,** *total bytes***)**

uploadProgress事件将由Flash控制器定时触发（译者注：上传进度发生改变就会触发），该事件用于在页面中更新UI（译者注：如上传进度）。

**注意:** Linux系统下的Flash播放器只在整个文件上传之后触发一次uploadProgress事件，这是Linux系统下Flash播放器的一个bug， 我们无法避免。

**uploadError(***file object***,** *error code***,** *message***)**

当上传被中断或者没有成功完成时，就会触发uploadError事件，error code参数表示发生错误的类型。根据error code参数的值你可以得出SWFUpload.UPLOAD\_ERROR中的一个常量（译者注：表示具体错误）。

停止,取消或者uploadStart事件返回'false'都将触发uploadError事件。被取消上传但是仍然在上传队列中等候的文件，不会触发Uploaderror事件。

**uploadSuccess(***file object***,** *server data***,** *received response***)**

当文件上传完毕并且服务器返回200状态时，uploadSuccess事件将触发。任何由服务器输出的数据，都可以通过服务器数据参数获得。

由于Flash播放器的一些bug，可能无法捕获服务器的回应，uploadSuccess事件也不会触发。在这种情况下，将会检查assume\_success\_timeout的设置，查看是否过了触发uploadSuccess事件的时间，这时捕获的服务器回应值应该是false。

http\_success设置，允许即使在状态码不是200的时候也可以触发uploadSuccess事件。这种情况下，Flash播放器无法获得任何服务器数据。

这时候，当前上传并没有完成，其他上传无法通过uploadSuccess开始。

**uploadComplete(***file object***)**

uploadComplete总是在上传过程的最后触发 (在uploadError事件和uploadSuccess事件之后)。这时候，当前上传已经完成了，可以开始下一个上传。

如果你想自动开始下一个上传，可以在此事件中调用this.uploadStart()。如果你在uploadComplete事件中还写了取消所有上传的代码，那么在调用uploadStart时需要谨慎。

**debug(***message***)**

当debug设置为'true'时，debug事件将由SWFUpload库和Flash控制器一起触发。如果debug事件没有被重载，那么SWFUpload将向控制台（一个自动被添加至页面body部分最后的文本框）记录debug信息。

**SWFUpload实用对象**

**设置对象**

设置对象是一个向SWFUpload实例提供一系列设置的JavaScript对象。**每个设置都只应该出现一次。**许多设置都是可选的，并且如果忽略将自动加载默认值。请在[设置说明](http://www.phptogether.com/swfuploadoc/#settingsdescription)中查看必须和可选设置。

**Example:**

{

 upload\_url : "http://www.swfupload.org/upload.php",

 flash\_url : "http://www.swfupload.org/swfupload.swf",

 file\_post\_name : "Filedata",

 post\_params : {

 "post\_param\_name\_1" : "post\_param\_value\_1",

 "post\_param\_name\_2" : "post\_param\_value\_2",

 "post\_param\_name\_n" : "post\_param\_value\_n"

 },

 use\_query\_string : false,

 requeue\_on\_error : false,

 http\_success : [201, 202],

 assume\_success\_timeout : 0,

 file\_types : "\*.jpg;\*.gif",

 file\_types\_description: "Web Image Files",

 file\_size\_limit : "1024",

 file\_upload\_limit : 10,

 file\_queue\_limit : 2,

 debug : false,

 prevent\_swf\_caching : false,

 preserve\_relative\_urls : false,

 button\_placeholder\_id : "element\_id",

 button\_image\_url : "http://www.swfupload.org/button\_sprite.png",

 button\_width : 61,

 button\_height : 22,

 button\_text : "<b>Click</b> <span class="redText">here</span>",

 button\_text\_style : ".redText { color: #FF0000; }",

 button\_text\_left\_padding : 3,

 button\_text\_top\_padding : 2,

 button\_action : SWFUpload.BUTTON\_ACTION.SELECT\_FILES,

 button\_disabled : false,

 button\_cursor : SWFUpload.CURSOR.HAND,

 button\_window\_mode : SWFUpload.WINDOW\_MODE.TRANSPARENT,

 swfupload\_loaded\_handler : swfupload\_loaded\_function,

 file\_dialog\_start\_handler : file\_dialog\_start\_function,

 file\_queued\_handler : file\_queued\_function,

 file\_queue\_error\_handler : file\_queue\_error\_function,

 file\_dialog\_complete\_handler : file\_dialog\_complete\_function,

 upload\_start\_handler : upload\_start\_function,

 upload\_progress\_handler : upload\_progress\_function,

 upload\_error\_handler : upload\_error\_function,

 upload\_success\_handler : upload\_success\_function,

 upload\_complete\_handler : upload\_complete\_function,

 debug\_handler : debug\_function,

 custom\_settings : {

 custom\_setting\_1 : "custom\_setting\_value\_1",

 custom\_setting\_2 : "custom\_setting\_value\_2",

 custom\_setting\_n : "custom\_setting\_value\_n",

 }

}

**设置说明**

upload\_url

upload\_url的值可以是一个完整的url或者相对于当前脚本的路径，upload\_url应该和Flash控制器处于同一个域下以保证稳定性。

如果preserve\_relative\_urls设置为false，SWFUpload会将相对路径转换为一个绝对路径以避免不同平台的Flash播放器对URL的不同识别问题。 如果你想禁用SWFUploads对URL的转换，相对路径应该相对于swfupload.swf文件。

file\_post\_name

file\_post\_name用来设置提交文件的键名，这个名称跟文件本身的名称没有关系。默认值是'Filedata'（译者注：在上传脚本中，使用$\_FILES['Filedata']来获取文件的相关信息），我们建议使用默认值以保证最好的兼容性。

post\_params

post\_params定义一个将随每个已上传文件一起被提交的name/value键值对，该项的值是一个单一的JavaScript对象。如何定义多个name/value键值对，在上面设置对象例子中有示范。Value可以是字符串或者数字(可以用JavaScript的typeof函数来验证其类型)。

注意: Flash Player 8不支持发送额外的post参数。SWFUpload将自动把post\_params作为网址传值的一部分发送。

use\_query\_string

use\_query\_string的值为true或false。此项的值告知SWFUpload是通过网址传值的形式发送post\_params和文件参数（译者注：可以使用$\_GET数组和$\_REQUEST数组获取）还是直接通过POST（译者注：可以使用$\_POST数组和$\_REQUEST数组获取）发送。此设置从SWFUpload v2.1.0版本中引进。

preserve\_relative\_urls

该项的值告诉SWFUpload是否将upload\_url的相对路径转换为绝对路径。如果设置为true，SWFUpload将不会修改url路径 ，默认值是false。

requeue\_on\_error

requeue\_on\_error的值可以是true或false。当此项的值被设置为true时，任何发生uploadError错误(除了fileQueueError和FILE\_CANCELLED)的文件都将返回队列的前面，而不会被丢弃。如果需要，文件可以再次被上传。要从上传队列移除文件，应该调用cancelUpload方法。

所有与上传失败相关的事件仍会被调用，所以重整上传失败的文件可能会和队列插件有冲突(或者是用来上传整个队列的自定义代码)。如果没有采取措施来取消已经上传失败的文件，自动上传下一个文件的代码将会重复上传已经失败的文件。

该设置项从SWFUpload v2.1.0版本中引进。

http\_success

此项是一个数组，它定义了一组用以表示执行成功的HTTP状态代码，状态码200通常意味着服务器成功响应。在服务器返回数据中，只给出了状态码200.

当返回或者接受一个不是200的HTTP状态码时，服务器无须返回内容。

assume\_success\_timeout

在文件上传完毕后，SWFUpload等待Flash探测服务器响应的时间（单位：秒）。此项可以让你规避Flash播放器的bug，这些bug通常是由于长时间运行的服务器端脚本使Flash忽略了服务器响应或者Mac平台下的Flash播放器忽略了无内容的服务器响应而引起的。

测试表明，在上一次uploadProgress事件发生之后，Flash将会忽略那些超过30秒的服务器响应。

默认值为0，这表示禁用此特性。 SWFUpload将不停地等待Flash播放器触发uploadSuccess事件。

file\_types

file\_types设置项的值是一系列以分号隔开，允许用户选择的文件后缀列表。如'\*.\*'将允许选择所有的文件类型。

file\_types\_description

在文件浏览窗口中呈现给用户的关于文件类型的文本描述。

file\_size\_limit

file\_size\_limit定义了上传文件的大小限制，它的值必须带有单位。可选单位有B, KB, MB和GB。如果没有给出单位，默认使用KB。设置为0，表示文件大小不受限制。

注意: 此设置只作用于用户的浏览器，它不会影响web服务器上的设置（译者注：为了防范恶意用户绕过前台脚本，你应该在后台脚本中加以验证）。

file\_upload\_limit

定义SWFUpload一次最多可以同时上传的文件数量，此项值必须依赖于file\_queue\_limit的设置。一旦用户已经上传的文件数量或者上传队列中的文件数量达到了最大值，他就不能再添加文件。默认值为0，表示无限制。只有成功上传的文件，才会被计算进file\_upload\_limit(触发了uploadSuccess事件的上传)。setStats函数可以用来修改已成功的上传的数目。

注意: 注意，此项的值无跨法页面进行跟踪，当页面刷新时它的值将被重置。允许同时上传的文件数量，应该根据web服务器的负载进行设置。

file\_queue\_limit

定义上传队列中的文件最大数量。如果一个文件已经上传完成或者上传出错或者被取消上传，你可以向上传队列新增一个文件，只要不超出限制。如果file\_upload\_limit的值比file\_queue\_limit的值小，那么此设置将采用file\_upload\_limit的值。

flash\_url

此项的值是Flash控制文件的完整或相对路径。一旦SWFUpload初始化，该设置无法更改，相对路径是指相对于当前页面的url。

flash\_width

**(在v2.1.0版本中被移除)**定义存放flash的HTML元素的宽度，如果此项的值小于1px，某些浏览器可能显示不正常。此设置是可选的，默认值是1px。

flash\_height

**(在v2.1.0版本中被移除)**定义存放flash的HTML元素的高度，如果此项的值小于1px，某些浏览器可能显示不正常。此设置是可选的，默认值是1px。

flash\_color

**(在v2.2.0版本中被移除)**设置存放flash的HTML元素的背景色，默认值是'#FFFFFF'。

注意:在所有浏览器中，如果对存放flash的元素采用"skinning"1px，将不会起做用（译者注：查资料，skinning一词在IT行业有蒙皮的意思。译者不确定，所以没有直接翻译，如果有热心读者知道，请告知。）。

prevent\_swf\_caching

**(从v2.2.0版本引进)** 此项的值是布尔型。它用来指示是否应该向Flash的URL添加一个随机值来防止浏览器缓存SWF电影。对一些基于IE引擎的浏览器来说，此设置可以避免bug。

注意:向flash的URL添加随机数的算法非常简单，并且无法处理已经带有参数的url地址。

debug

此项的值为布尔型，定义是否触发debug事件。

button\_placeholder\_id

**(从v2.2.0版本引进)** 此项的值是一个将被Flash按钮**替换**的DOM元素的ID。此设置会重载button\_placeholder，Flash按钮的样式可以使用CSS类'swfupload'来实现。

button\_placeholder

**(从v2.2.0版本引进)** 此项的值是一个将被Flash按钮**替换**的DOM元素，只有当设置了button\_placeholder\_id时，才会起作用。Flash按钮的样式可以使用CSS类'swfupload'来实现。

button\_image\_url

**(从v2.2.0版本引进)** Flash按钮的背景图片，可以是一个完整有效的URL地址或者相对路径。任何Flash支持的图像格式都可以使用 (另一个SWF文件或者gif, jpg和png)。

该图像URL受preserve\_relative\_urls设置的影响，并且应该和upload\_url的规则一样（译者注：都是用相对路径或绝对路径）。

按钮背景图像采用CSS精灵的方式处理，在一个按钮背景图片中应该含有4种按钮状态。每个按钮状态图像都应该堆叠在另一个状态图像之上，按如下顺序:普通, 鼠标划过, 按下/点击,禁用。

button\_width

**(从v2.2.0版本引进)** Flash按钮的宽度。

button\_height

**(从v2.2.0版本引进)** Flash按钮的高度，此项的值应该是整个按钮图片高度的1/4（译者注：文档上文提到了，一个Flash按钮图片里应该包含按钮的四个状态，也可以理解为是4张单独的按钮图片拼合在一起的）。

button\_text

**(从v2.2.0版本引进)** 显示在Flash按钮上的文本，可以为空。文本可以使用CSS和button\_text\_style的设置来定义样式。查看[Adobe Flash的详细文档](http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/text/TextField.html#htmlText)。

button\_text\_style

**(从v2.2.0版本引进)** 定义如何显示的button\_text的CSS样式，字符串形式。查看[Adobe Flash的详细文档](http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/text/StyleSheet.html)。

button\_text\_top\_padding

**(从v2.2.0版本引进)** 设置Flash按钮的文本在垂直方向上的位置，可以使用负数。

button\_text\_left\_padding

**(从v2.2.0版本引进)** 设置Flash按钮的文本在水平方向上的位置，可以使用负数。

button\_action

**(从v2.2.0版本引进)** 定义当Flash按钮被点击时所执行的动作。动作有效值可以在swfupload.js文件的BUTTON\_ACTION对象源码中找到。

button\_disabled

**(从v2.2.0版本引进)** 布尔型，设置Flash按钮是否处于禁用状态。当禁用时，按钮将不执行任何动作。

button\_cursor

**(从v2.2.0版本引进)** 定义当鼠标划过Flash按钮时，光标呈现何种形态。

button\_window\_mode

**(从v2.2.0版本引进)** 设置Flash电影的WMODE属性。有效值可以查看SWFUpload.WINDOW\_MODE常量中查看。

custom\_settings（自定义设置）

custom\_settings允许开发者安全地向SWFUpload实例添加额外信息而不用担心影响SWFUpload内部的值或者新版本变更。此设置的值应该是一个JavaScript对象。

一旦实例化，就可以通过SWFUpload的'customSettings'属性来访问自定义设置。

var swfu = new SWFUpload({

 custom\_settings : {

 "My Setting" : "This is my setting",

 myothersetting : "This is my other setting",

 integer\_setting : 100,

 a\_dom\_setting : document.getElementById("some\_element\_id")

 }

});

var my\_setting = swfu.customSettings["My Setting"]);

swfu.customSettings["My Setting"] = "This is my new setting";

swfu.customSetting.myothersetting = "another new value";

swfu.customSetting.integer\_setting += 25;

swfu.customSetting["a\_dom\_setting"].style.visibility = "hidden";

事件处理器

其余的设置，用来定义SWFUpload操作过程中被调用的事件处理器。当需要时，应该定义一系列JavaScript函数来处理这些事件。

**File Object(文件对象)**

文件对象被传递给多个事件处理器，它包含有文件的相关信息。一些操作系统并不提供以下所有值(尤其是创建日期和修改日期)。

{

 id : 字符串, //SWFUpload文件的id, 用来开始或取消上传

 index : 数字, //当前文件的索引，供getFile()方法使用。

 name : 字符串, //文件名，不包含文件路径。

 size : 数字, //文件大小，单位为字节。

 type : 字符串, //客户端操作系统所描述的文件类型

 creationdate : 日期格式, //文件被创建的日期

 modificationdate : 日期格式, //文件的最后修改日期

 filestatus : 数字, //文件的当前状态码，使用SWFUpload.FILE\_STATUS来获取文件的状态描述。

}

**Stats Object(统计对象)**

Stats对象提供关于上传队列的信息。

stats对象包含以下属性:

{

 in\_progress : 数字 //1或0表示一个文件是否处于上传中

 files\_queued : 数字 //当前上传队列中的文件数量

 successful\_uploads : 数字 //上传成功的文件数量 (上传成功将触发uploadSuccess事件)

 upload\_errors : 数字 //发生错误的文件数量 (除了取消文件)

 upload\_cancelled : 数字 //被取消的文件数量

 queue\_errors : 数字 //触发fileQueueError事件的文件数量

}

除了*in\_progress*和*files\_queued*之外，所有这些值都可以通过setStats()方法来更新。

**SWFUpload插件**

在SWFUpload v2.0中，引入了多个插件。他们可以与SWFUpload一起，帮助完成很多普通功能。

Currently most of the documentation for using the plugins is contained in the plugin JavaScript file.

**SWFObject**

SWFObject插件使用[SWFObject库](http://code.google.com/p/swfobject/)来将SWFUpload的Flash组件嵌入页面。

此插件也支持文件加载和Flash版本检测，插件中含有详细使用说明。你不能将SWFObject的文件载入功能和其他库的文件载入功能混合使用，只能选择其中一种。

**Flash Player 10:** 由于Flash Player 10要求SWFUpload的swf文件表现为一个按钮，为了成功载入SWFUpload，flash电影必须是可见的。如果button\_placeholder\_id所对应的元素是隐藏的。(visibility设置为hidden或者display设置为none)，SWFUpload将会载入失败。

**Cookies**

为了规避Flash的Cookie bug，Cookies插件将自动读取浏览器的cookies并且在上传时以POST或者GET变量的方式发送到上传url地址。

注意，插件在发送cookies时，它们是以name/values键值对的形式存在于POST或者GET数组中。在服务端，它们将无法作为cookie被访问，一些会检查会话的cookies或者授权值的框架也无法获取这些值。

**Queue Handling**

该插件提供队列处理机制，如：一次性上传整个队列、一次性取消上传整个队列、自动上传。

**Speed**

此插件继承'file'对象，它拥有若干描述当前上传的属性。包括：当前上传速度、平均速度、已过时间、上传剩余时间和其他。

**常见问题**

Flash播放器和许多浏览器都有bug，这些bug将直接反应在SWFUpload的表现上。当我们在努力规避/修复这些bug的同时，仍然有一些是我们不能避免的。

**Cancelling in Linux**

Linux系统下，如果你取消上传，旧版本的Flash 9将使浏览器崩溃，较新版本的Flash 9改善了该问题。

**Upload Progress in Linux**

Linux系统中，在文件上传结束后，Flash仅仅触发一次uploadProgress事件。

在一些版本的Linux系统中，在上传的过程中，整个浏览器将不可用。

**Upload Progress in OS X**

有一些报告指出，uploadProgress事件在MAC平台的Flash中不会触发。尽管还没有被确认，但是开发者应该注意此问题。

**MIME Type**

不论文件的实际MIMIE类型是什么，Flash播放器都将使用*application/octet-stream*类型来上传文件。

**Maximum number of selected files**

Flash播放器并不强制一次只能选择多少文件，但是，它会建立一个有长度限制的文件序列，该文件序列是由文件名和分隔符[引号][空格][引号]组成的。可以选择的文件数量，是由文件名长度、前后引号、被选择文件数量减去1再乘以3这三者的和来决定的。

由于平台的不同，此限制也可能不同。如果用户选择了太多的文件，他们将会收到一条由Flash发出的警告而当前对话将仍然停留文件选择窗口。

**Proxies**

使用代理，Flash播放器可能工作不正常。它无法很好地识别代理，有时候还会崩溃。

一些反病毒软件使用代理服务器来扫描上传文件，这样会误使SWFUpload相信整个文件已经被上传了。只要上传进度达到100%，SWFUpload将迅速触发uploadProgress事件，在代理服务器将文件上传完毕之前，SWFUpload看上去就像暂停了一样。

**Apache mod\_security**

Apache的mod\_security模块，会校验发送给服务器的POST数据。Flash播放器对此模块很敏感(目前，关于这一点是否无效，还没有定论)，如果服务器开启了mod\_security，POST提交的上传文件将被拒绝。mod\_security可以通过你的.htaccess文件禁用。

**SSL**

有一些报告指出，Flash播放器不能通过SSL协议上传。此问题还没有确认，但是绕过SSL上传的文件可能是不可靠的。尤其要注意的是，使用自行签约的SSL证书可能存在问题。

另外，由于Flash没有提供用于接受SSL证书的方法，那么从不被信任的认证机构获取的SSL证书将无效。我们已经注意到此问题，就像cookie bug一样,Windows平台上的Flash可以通过互联网浏览器来获取被信任的CA。

**Authentication**

Flash对HTTP身份认证的支持并不完美。最新版本的Flash对此有了改善，老版本的Flash将会使浏览器崩溃。

**Prematurely terminated connections**

过早地结束响应(如ASP.Net中的Response.end())有时候可能会将成功的上传报告为失败。

**Filedata in Linux**

Linux系统中，改变Filedata的值(file\_post\_name设置)将被忽略。

**Cookie issue**

在Windows平台中，非IE浏览器的Flash插件(FireFox, Opera, Safari等)将发送IEcookies，而不顾浏览者的使用。这破坏了许多服务端脚本的身份认证和会话控制。

如果开发者希望使用Session，那么他们应该手工传递Session和有关身份认证的cookie信息并且手工存储Session信息。

SWFUpload源码包含有PHP和ASP.Net的规避bug的示例代码。

**ExternalInterface bugs**

当与浏览器/JavaScript通信时，Flash无法正确的转义数据。SWFUpload为了解决这个问题做了诸多努力。如果在以后的Flash/浏览器中解决此bug，那么SWFUpload就可以发送大量的转义数据。

**Server Data length bugs**

在MAC和Linux系统的flash中，服务器数据太长容易被损坏，数据可能会被清空，错乱，或重复。我们建议服务器返回的数据尽量精简。

**Avant Browser**

一些使用Avant浏览器的用户可能会发现，在Flash控制器被缓存后SWFUpload无法工作。SWFUpload开发者已经反复提到了此问题，但是Avant浏览器的开发团队就像没事人一样。

当页面被重载时SWFUpload将会载入并触发swfupload\_loaded事件。但是，没有任何内部回调函数被定义来处理存放Flash的DOM元素。SWFUpload v2.0.2版本中添加了检测机制，如果没有检测到回调函数，将阻止swfupload\_loaded被触发。

**SWFUpload v2.2.0引进了prevent\_swf\_caching设置，以此来解决Avant浏览器中flash被缓存而无法工作的问题。**

**File Dialog & Page Changing**

当文件浏览窗口处于打开状态时离开或重载页面会导致浏览器崩溃(包括所有浏览器和所有操作系统，译者注：在谷歌浏览器测试没有崩溃，可能是官方文档没有更新此条目)。通常，这是由于在“一个onclick事件中调用了selectFile或selectFiles函数的”<a>标签上取消点击事件失败时造成的。

**Long Running Upload Scripts**

在Flash将文件上传至web服务器之后，上传脚本将被执行。这个脚本可以控制是否保存文件、创建缩略图、扫描病毒等。如果上传脚本在30秒内没有返回任何数据，Flash将断开连接并返回一个IO错误。当脚本运行是你可以通过返回字符或数据来防止该错误(如果可以)对PHP来说，即使Flash中断了连接脚本仍然会继续执行并且成功完成。在Flash断开连接后返回的任何数据都将丢失。

**WMODE / BUTTON\_WINDOW\_MODE**

在一些浏览器中，如果Flash控制器不在频幕的可见范围中，当前的WMODE(使用BUTTON\_WINDOW\_MODE设置)可能会组织Flash控制器载入。当用户滚动页面让控制器可见时，它才会载入。

这可能会对SWFObject插件产生不利影响。不会有任何SWFUpload事件触发，而且按钮背景也不会载入，直到flash控制器可见。

**Memory Leaks**

当JavaScript通过SWFUpload使用的内部接口类进行通信时，一些浏览器(如IE)无法释放由Flash播放器使用的内存。创建多个SWFUpload实例或者多次重载页面将会导致浏览器耗费更多的内存直到浏览器崩溃或对系统产生危害。

v2.2.0版本中，我们为SWFUpload部署一些安全措施。其中一些安全措施会自动运行，但是我们仍然建议你在离开页面时调用destroy()方法。如果在你的页面中存在成百上千的SWFUpload实例，你应该谨慎使用并且仔细的进行内存泄漏测试。

**Other Mac Issues**

* 有用户报告称Mac平台的Flash播放器在上传文件到子域时工作不正常。
* 有用户报告称，Mac平台的Flash播放器不处理重定向(HTTP状态码为302)，Windows客户端目前看来并无此问题。
* flash文档声明在OS X 10.3之前的版本中，载入的字节数总是被报告为-1。SWFUpload将该值转换为0，但是总的字节数不会被发送，而且进度条始终达不到100%。用户必须在uploadSuccess或uploadComplete事件中更新UI以显示100%来保持进度条的连续性。
* 一些用户报告称，如果upload\_url中含有空格，Mac平台下的Flash播放器会出问题。应该避免空格或者用'+'或'%20'符合来替换。
* 有用户报告称，Mac平台的Flash播放器会向HTTP\_HOST变量添加端口号(如：http://www.example.com:80)。如果你在服务端脚本检查此变量，请注意此问题。 -