

# SKPlayer

Copyright © SteelSky Software, 2011

*Руководство  
пользователя*

## Содержание

Введение.....	3
Требования к клиентской системе .....	3
Базовые возможности .....	3
Использование базовых возможностей .....	4
Проигрывание видео в MP4 или FLV формате, или MP3 аудио. ....	4
Шаг 1: HTML.....	4
Шаг 2: JavaScript + CSS .....	4
Шаг 3: Конфигурация плеера.....	4
Параметры конфигурации .....	6
Параметры конфигурационного файла.....	6
Использование изображений для предварительного просмотра содержимого разделов видео на временной линии (timeline thumbnails).....	8
Использование собственного логотипа .....	9
Плагины плеера .....	11
Общие параметры конфигурации плагинов .....	11
CSS-свойства плагинов .....	11
Overlay – швейцарский армейский нож рекламы .....	12
Параметры плагина .....	12
Показ встроенного HTML .....	12
Показ HTML из внешней ссылки.....	13
Показ изображения или SWF-файла .....	13
Если пользователь кликнул на оверлее... ..	14
Как заставить оверлей появиться – и скрыться.....	15
Share this video – распространяем видео по сети .....	16
Внешний вид плеера .....	18
Архитектура «шкурок».....	18
Создание своего оформления плеера. ....	18
SKPlayer JavaScript API.....	19
Использование плагинов OSMF-ядра .....	20
HttpPseudoStreaming.....	20

YouTube .....	22
MAST.....	22
Advertisement.....	24
SMIL .....	25
Google Analytics.....	26
Captioning – субтитры в видео.....	27
Интеграция со сторонними приложениями и CMS .....	30
Интеграция с Wordpress .....	30
Установка и настройка SKPlayer в Wordpress.....	30
Интеграция с DLE.....	32
Установка SKPlayer в DLE.....	32
SKPlayer ЧаВо и стрельба по проблемам .....	34

## Введение

SKPlayer является Flash-плеером для WWW нового поколения. Прочное и гибкое основание в виде ActionScript 3, кастомных GUI-компонентов, в сочетании с современной версией Adobe Open Source Media Framework позволили создать продукт как легкий в расширении, так и надежный, способный к гибкой кастомизации.

Плеер может проигрывать различные поддерживаемые Flash Player 10 форматы мультимедиа, в том числе и видео формата h.264 как в MP4, так и в FLV контейнерах, а множество плагинов обеспечивают дополнительную функциональность, достаточную, чтобы удовлетворить самые взыскательные запросы.

## Требования к клиентской системе

Adobe Flash Player версии 10.0 или выше. С плеером поставляется SWFObject – написанная на JavaScript обертка, способная проверять версию клиентского Flash Player и предложить клиенту апгрейд версии, если это необходимо.

## Базовые возможности

- Проигрывание видео в FLV и MP4 контейнерах, поддержка MP3 аудио.
- Поддержка статических заставок, показываемых перед видео.
- Поддержка серверных протоколов – HTTP-прогрессивный даунлоад, RTMP.
- JavaScript API, позволяющий осуществлять управление плеером из JavaScript страницы.
- Базовый DRM – вы можете задавать количество просмотров за единицу времени для всех пользователей. Данные о просмотрах хранятся на стороне пользователя, но не стираются при очистке обычных HTTP-cookies.
- Поддержка timeline thumbnails – возможности увидеть содержимое экрана при поднесении курсора мыши к временной линии видео<sup>1</sup>.
- Генерация эмбеддов и полная сохранение работоспособности плеера и всех его настроек на всех сайтах, где плеер был показан – включая 100% сохранение конфигурации плагинов.
- Легкость расширения. Дополнительные возможности осуществляются плагинами, как предназначенными для показа в рабочей области, так и плагинами, предназначенными для использования новых источников потоков.

---

<sup>1</sup> Требуется предварительная обработка на стороне сервера. Скрипт для обработки входит в комплект поставки

## Использование базовых возможностей

### Проигрывание видео в MP4 или FLV формате, или MP3 аудио.

SKPlayer добавляется в HTML-страницу либо статическим, либо динамическим образом. Различие между ними подробно описано в [документации SWFObject](#). Вкратце – статический метод позволяет вам определить заранее, что именно будет размещено в странице, определить альтернативный контент (то есть тот, который будет виден в случае, если у пользователя не установлен Flash Player). Динамический метод позволяет вам легко заменить любой объект в странице (к примеру, div или span) на плеер. Ниже мы рассмотрим динамический способ как наиболее легкий, по шагам<sup>2</sup>.

#### Шаг 1: HTML

Создайте валидный HTML, в котором предусмотрите место для вставки плеера. Самый простой способ – предусмотрите элемент div и задайте ему id, скажем, skplayer. Заодно впишите туда альтернативный контент:

```
<div id="flashContent" type="application/x-shockwave-flash">
  <p>
    To view this page ensure that Adobe Flash Player version
    10.1.0 or greater is installed.
  </p>
  <script type="text/javascript">
    var pageHost = ((document.location.protocol == "https:") ? "https://" :
"http://");
    document.write("<a href='http://www.adobe.com/go/getflashplayer'><img src='"
+ pageHost +
"www.adobe.com/images/shared/download_buttons/get_flash_player.gif' alt='Get Adobe
Flash player' /></a>");
  </script>
</div>
```

Добавьте подобный код в любое место в части body страницы – там, где вы хотите, чтобы плеер был виден.

#### Шаг 2: JavaScript + CSS

Добавьте в страницу (раздел head) ссылки на необходимые файлы из комплекта поставки:

```
<link rel="stylesheet" type="text/css" href="history/history.css"/>
<script type="text/javascript" src="history/history.js"></script>
<script type="text/javascript" src="swfobject.js"></script>
```

#### Шаг 3: Конфигурация плеера

Добавьте плеер в страницу с нужными вам параметрами.

---

<sup>2</sup> В комплект поставки входит несколько примеров добавления плеера в страницу.

К примеру:

```
<script type="text/javascript">
  var swfVersionStr = "10.0.0";
  var xiSwfUrlStr = "expressInstall.swf";

  var flashvars = {
    movieURL:'http://stream.example.com:14080/test/city_traffic_hd.flv',
    thumbURL: "http://www.example.com/video/poster.jpg",
    config: " http://www.example.com/skplayer3/config.xml"
  };

  var params = {};
  params.quality = "high";
  params.bgcolor = "#000000";
  params.allowscriptaccess = "always";
  params.allowfullscreen = "true";
  params.wmode = 'direct';

  swfobject.embedSWF(
    "skplayer.swf", "flashContent",
    "720", "558",
    swfVersionStr, xiSwfUrlStr,
    flashvars, params);
  <!-- JavaScript enabled so display the flashContent div in case it is not
  replaced with a swf object. -->
  swfobject.createCSS("#flashContent", "display:block;text-align:left;");
</script>
```

На что стоит обратить внимание:

1. `params.wmode` – в данном примере используется `direct`, который обеспечивает акселерацию с использованием GPU для проигрывания h.264-контента. Если вы собираетесь использовать, к примеру, рекламу в HTML-формате поверх плеера или вообще использовать какие-то элементы страницы поверх плеера – вам необходимо будет сменить `wmode` на `opaque`. Также возможен режим с прозрачным фоном `wmode="transparent"`, но в этом случае Flash Player использует довольно сложный композитный режим, который может плохо сказаться на производительности. Если вы вообще убираете данный параметр из параметров `swfobject`, то по умолчанию используется режим `“window”` – режим максимальной совместимости.
2. `params.bgcolor` необязателен – он задает фоновый цвет плеера. Если он отсутствует, то плеер не будет иметь никакого фонового цвета – будет прозрачным.
3. `var swfVersionStr = "10.0.0";` - эта строка задает минимальную версию Flash Player. SKPlayer поддерживает версии 10.0 и выше, и эта строка произведет апгрейд версии ниже до нужной – с минимальным задействованием пользователя. В большинстве случаев вам не потребуется ничего менять в этой строке, но если вам понадобятся какие-то продвинутые возможности плееров более высоких версий с учетом контента вашего сайта – к примеру, DRM – то эту строку надо будет соответственно поменять.
4. Как видите, все URLы являются абсолютными. Хотя это и не строго обязательно для плеера, но весьма желательно – для того, чтобы не было путаницы при вставке кода плеера на другие сайты (см. ниже).

- Учитывайте высоту управляющих элементов при задании высоты плеера (в данном случае 558), если вы хотите, чтобы изображение точно входило в размеры плеера без зазоров.

## Параметры конфигурации

Конфигурация SKPlayer состоит из переменных, задаваемых в двух возможных местах – в переменных flashvars, передаваемых встроенному объекту Flash, а также в файле конфигурации в формате XML. Это дает возможность передавать в странице только те переменные, которые меняются, оставляя постоянные значение в конфигурационном файле, повышает читаемость конфигурации. В свете разнообразия возможностей, предоставляемых плеером, читать огромное количество flashvars было бы трудно, и вело бы к возможным ошибкам. Это также дает возможность писать, к примеру, JavaScript-реакции на те, или иные события прямо в код конфигурационного файла.

Итак, справка по конфигурационным параметрам SKPlayer. Все URLы могут быть как относительными, так и абсолютными.

- movieURL** – URL видео. Обязателен.
- thumbURL** – URL «постера», который будет отображен в рабочей области сразу после загрузки плеера, до того, как начнется воспроизведение. Необязателен.
- config** – URL файла конфигурации. Обязательный параметр. Файл конфигурации представляет собой XML-файл следующего вида:

```
<?xml version="1.0" encoding="UTF-8" ?>
<skplayer>
... здесь мы задаем все конфигурационные данные, которые описаны чуть ниже
</skplayer>
```

- replace.\*** - параметры замены, которые будут использованы в конфигурационном файле. К примеру, если вы укажете `replace.id = MYID` во flashvars, внутри конфигурационного файла все вхождения {id} во всех параметрах конфигурации как плеера, так и его плагинов, будут заменены на MYID. Это дает вам, к примеру, возможность передавать id аффилиейта в URL клика на рекламе.

## Параметры конфигурационного файла

Хотя данные переменные вы можете также передавать через flashvars, тем не менее, обычно вы будете их использовать из конфигурационного файла. Переменные представляют собой стандартные узлы XML и добавляются в файла в виде `<имя_переменной>значение</имя_переменной>`.

Минимальный конфигурационный файл содержит только лицензионный ключ. Все остальные значения будут использованы по умолчанию.

```
<?xml version="1.0" encoding="UTF-8" ?>
<skplayer>
  <licenseKey>
    ... Your key
  </licenseKey>
</skplayer>
```

- **licenseKey** – ключ лицензии. Поместите его сюда после покупки, чтобы избавиться от сообщения о незарегистрированной версии. Для удобства вы можете разбить его переносами строки, он все равно будет работать.
- **autoPlay** – 0 или 1 задают то, будет ли видео проигрываться сразу после загрузки плеера. Значение по умолчанию – 0.
- **autoLoad** – 0 или 1 задают то, начнется ли подгрузка видео сразу после загрузки плеера. Значение по умолчанию – 0.
- **autoPlayOnFirstSeek** – 0 или 1 задают то, начнет ли плеер проигрывание после первого клика на временной линии с места, на котором кликнули. Включение параметра также означает включение **autoLoad**. Значение по умолчанию – 0.
- **scaleMode** – задает режим масштабирования изображения в рабочей области плеера. Возможные значения:
  - **none** – сохраняется оригинальный размер изображения
  - **stretch** – изображение растягивается на всю рабочую область экрана. Возможно искажение пропорций.
  - **letterbox** – изображение будет растянуто на всю рабочую область с сохранением пропорций. Добавляется небольшой зазор для того, чтобы при масштабировании не потерялась часть изображения. *Является значением по умолчанию.*
  - **zoom** – изображение будет растянуто на всю рабочую область с сохранением пропорций. В отличие от режима letterbox, зазоров не добавляется, так что малая часть изображения с одного из краев может быть потеряна.
- **externalEventCallback** – имя JavaScript функции, которая будет вызываться плеером каждый раз, когда происходят внутренние события. Параметры вызова – ( ID плеера, тип события). См. справку по JavaScript API.
- **viewsLimit** – численный параметр, задает предел количества просмотров для пользователя. По умолчанию количество просмотров не ограничено.
- **viewCountedAt** – численный параметр, задает то, на какой секунде видео считается просмотренным. До этого предела количество просмотров не увеличивается. Задается в секундах.
- **callOnLimitReached** – имя JavaScript функции, которая будет вызываться по достижению предела количества просмотров. Параметр вызова – ID плеера.
- **keepViews** – задает (в часах) то время, сколько хранится информация о просмотре, используемая для определения количества просмотров. По умолчанию считаются просмотры за последние 24 часа.

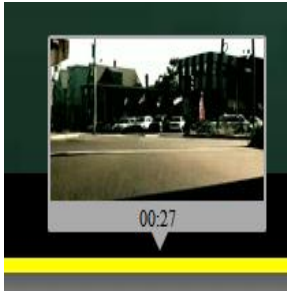
- **thumbServiceURL** - задает URL сервиса, выдающего изображения для предварительного просмотра содержимого видео на временной линии. См. раздел ниже.

Также в конфигурационном файле могут присутствовать записи по конфигурации внешнего вида, конфигурация мультимедийного ядра плеера, загруженного в данный момент, его плагинов, а также плагинов самого плеера. Об этом будет рассказано немного позже.

**ПОМНИТЕ:** XML чувствителен к регистру символов – поэтому параметры необходимо указывать именно в том виде, в каком они приведены в документации. Не KeepViews, не keepviews – а именно keepViews, к примеру.

### Использование изображений для предварительного просмотра содержимого разделов видео на временной линии (timeline thumbnails).

SKPlayer поддерживает возможность просмотра содержимого видео на временной шкале с заранее заданным шагом. В случае использования этой возможности при движении курсором мыши вдоль временной шкалы вы увидите не только время данного участка, но и приблизительное содержание.



Для того, чтобы задействовать данную возможность, необходимо обработать видео на сервере предлагающемся к плееру скриптом thumbgen.pl и затем сделать результаты обработки доступными с WWW.

#### Подготовка данных

Для того, чтобы подготовить на сервере данные для просмотра, вам нужно следующее ПО:

1. perl – версии 5.6 или выше.
2. модули perl JSON и Config::Properties
3. ffmpeg с поддержкой как кодеков видео, которое вы собираетесь обрабатывать, так и с поддержкой JPEG.

Скопируйте thumbgen.properties.example в thumbgen.properties и отредактируйте его. Файл с примерной конфигурацией содержит описание всех параметров и содержит примеры этих параметров.

Запустите thumbgen.pl:

```
perl thumbgen.pl
```

Скрипт сообщает о прогрессе своей работы. При удачном завершении в выходном директории, указанном как `thumbs_storage_root` в файле конфигурации, вы видите:

1. Набор файлов с расширением `.json` – это описание коллекций «тумб» для просмотра.
2. Некоторое количество директориев, имена которых совпадают с именами видео-файлов: `movie1.flv`, `movie2.mp4` и т.п. Внутри них должны находиться «тумбы» - сами изображения заданного в параметре `thumbs_height` размера.

Настройка плеера

Теперь необходимо внести изменения в конфигурацию плеера. Сначала добавим в файл конфигурации параметр, описывающий то, где именно нам приходится искать данный файл – `thumbServiceURL`:

```
<skplayer>
...   <thumbServiceURL>http://example.com/thumbs/%id%.json</thumbServiceURL>
</skplayer>
```

Параметр `id` – это то, что будет заменено на ID коллекции тумб позже, уже для каждого конкретного видео.

Меняем параметры SKPlayer в JavaScript:

```
var flashvars = {
    movieURL: 'http://stream.example.com:14080/test/city_traffic_hd.flv',
    thumbURL: "http://www.example.com/video/poster.jpg",
    thumbID: "movie1.flv",
    config: " http://www.example.com/skplayer3/config.xml"
};
```

В данном случае это приведет к тому, что с сервера будет запрошен файл коллекции изображений для просмотра <http://example.com/thumbs/movie1.flv.json> - который вы подготовили в предыдущем пункте.

## Использование собственного логотипа

Замена лого плеера на собственный бренд – возможность, доступная только в зарегистрированной версии SKPlayer. В качестве лого может работать как обычный графический файл в форматах JPEG, PNG или GIF, так и SWF-анимация. Лого будет показано в правом верхнем углу во всех режимах плеера. Логотип задается параметром конфигурации `logo`, в котором вы указываете URL, из которого плеер забирает ресурс.

```
<?xml version="1.0" encoding="UTF-8" ?>
<skplayer>
  <licenseKey>
    ... Your key
  </licenseKey>
  <logo>http://www.example.com/images/player_logo.swf</logo>
</skplayer>
```

## Плагины плеера

Плеер поддерживает внешние плагины пользовательского интерфейса. Плагины описываются секцией “uimodules” файла конфигурации. Секция может содержать произвольное количество записей конфигурации, каждая запись должна называться по имени плагина, который описывает. Один и тот же плагин может указываться в файле конфигурации несколько раз, с разными параметрами.

### Общие параметры конфигурации плагинов

- **url** – необязательный параметр, указывающий URL, с которого происходит загрузка плагина. По умолчанию, загрузка осуществляется из поддиректория uimodules директория установки плеера.
- **css** – необязательный параметр, задает URL дополнительного CSS, используемого плагином.
- **cssData** – необязательный параметр, позволяет указать стили оформления плагина прямо в конфигурационном файле.
- **autoHide** – плагин будет автоматически скрываться, когда пользователь не производит никаких действий в течение нескольких секунд.

### CSS-свойства плагинов

#### Общий вид

- **backgroundColor** – цвет фона плагина, стандартный CSS-цвет.
- **backgroundAlpha** – alpha-параметр, определяющий степень прозрачности фона. Десятичная дробь. 0 значит полностью прозрачный фон, 1 – полностью непрозрачный.
- **color** – основной цвет плагина, к примеру – цвет текста по умолчанию.
- **alpha** – степень прозрачности всего плагина, с его содержимым.

#### Геометрия

Плагины располагаются в рабочей области в соответствии с заданными координатами. Задаются координаты описанными ниже параметрами, все параметры *необязательны*, по умолчанию плагин располагается в верхнем левом углу с края, его размеры определяются размером содержимого. Все размеры абсолютны<sup>3</sup> и указываются в обычном для CSS стиле – к примеру, ширина в 200 пикселей описывается как width: 200px.

- **left** – расстояние от левого края рабочей области до левого края плагина
- **right** – расстояние от правого края рабочей области до правого края плагина
- **top** – расстояние от верхнего края рабочей области до верхнего края плагина.
- **bottom** – расстояние от нижнего края рабочей области до нижнего края плагина

---

<sup>3</sup> Поддержка относительных размеров планируется в будущих версиях

- **verticalCenter** – смещение середины плагина по вертикали от середины рабочей области по вертикали. К примеру, verticalCenter: 0 означает, что плагин будет располагаться всегда точно на средней линии по вертикали рабочей области.
- **horizontalCenter** – то же самое, только для горизонтальных координат
- **width** - ширина плагина.
- **height** - высота плагина.

## Overlay – швейцарский армейский нож рекламы

Плагин overlay является универсальным инструментом для показа HTML-текста, изображений, а также списков изображений поверх рабочей области плеера. Плагин после загрузки образует на рабочей области прямоугольную область, в которой находятся указанные объекты. То, когда именно показывается и скрывается область, ее содержимое, меняются в самых широких пределах.

### Параметры плагина

- **id** - который позволит вам различать его копии в случае использования нескольких копий одновременно. Этот id вы, к примеру, можете использовать в CSS-файле шкурки для того, чтобы задать разным копиям разные параметры. Имя CSS-класса, который вы также можете использовать – OverlayModule.

### Показ встроенного HTML

Если плагину указан параметр «html», то все содержимое параметра будет показано внутри рабочей области плагина. Для отображения HTML используются внутренние возможности Flash Player, поэтому стоит учитывать ограничения его возможностей.

Поддерживаемая разметка HTML:

- <a> - поддерживаются параметры href и target
- <b>
- <br>
- <font> - поддерживается параметр color (только шестнадцатеричные значения, такие как #FFFFFF), face, size.
- <img> - поддерживаемые атрибуты
  - src
  - width
  - height
  - align
  - hspace
  - vspace
- <i>
- <li>
- <p>

- `<span>`
- `<textformat>` - нестандартный тэг, задающий формат текста параграфа. Поддерживаются параметры
  - `blockindent` - инdentация блока в пойнтах.
  - `indent` - инdentация первого символа.
  - `leading` - вертикальное расстояние между строками.
  - `leftmargin` - левый марджин параграфа
  - `rightmargin` - правый марджин параграфа
  - `tabstops` - массив целых чисел - параметров табуляции.
- `<u>`

Для более подробного ознакомления с HTML, поддерживаемым Flash Player, рекомендуется ознакомление с документацией по ActionScript 3.0 по классу [TextField](#), свойство `htmlText`.

Пример использования:

```
<skplayer>
...
  <uimodules>
    <overlay>
      <id>text_ad</id>
      <html><![CDATA[
        <b>test</b>
      ]]></html>
    </overlay>
  </uimodules>
</skplayer>
```

### Показ HTML из внешней ссылки.

В случае использования параметра `htmlSource`, плагин будет загружать HTML из указанного URL. Также поддерживается атрибут `refresh`, который указывает интервал (в секундах), с которым указанный URL будет запрашиваться повторно.

Пример использования

```
<skplayer>
  <uimodules>
    <overlay>
      <id>text_ad</id>
      <htmlSource refresh="5">http://example.com/test/1.html</htmlSource>
    </overlay>
  </uimodules>
</skplayer>
```

### Показ изображения или SWF-файла

Параметр `image` задает URL внешнего изображения, которое будет отображаться в рабочей области плагина. Это может быть как статическое изображение в формате JPEG, PNG или GIF, так и SWF-файл.

Пример использования

```
<skplayer>
...
  <uimodules>
    <overlay>
      <id>text_ad</id>
      <image>http://example.com/test/banner.gif</image>
    </overlay>
  </uimodules>
</skplayer>
```

### Если пользователь кликнул на оверлее...

В дополнение к возможностям, поддерживаемым встроенным HTML и его тэгом <a>, сам плагин поддерживает несколько типов реакций на клик в рабочей области плагина.

#### *Навигация к заданному URL*

Задается параметром clickURL плагина. Как следует из названия – в новом окне браузера будет открыт URL с данным адресом.

Пример использования

```
<skplayer>
...
  <uimodules>
    <overlay>
      <id>text_ad</id>
      <image>http://example.com/test/banner.gif</image>
      <clickURL>http://example.com/url.php</clickURL>
    </overlay>
  </uimodules>
</skplayer>
```

#### *Исполнение JavaScript-кода*

Плагин поддерживает исполнение JavaScript кода в браузере в ответ на клик в рабочей области плагина. Для удобства объект плеера экспортируется плагином в то же адресное пространство, что и запускаемый скрипт, под именем «skplayer», и вы можете использовать стандартный JavaScript API SKPlayer для работы с ним там.

Пример использования – клик на имидже покажет окно сообщения «Hello from JavaScript!» и запустит проигрывание видео.

```
<skplayer>
...
  <uimodules>
    <overlay>
      <id>text_ad</id>
      <image>http://example.com/test/banner.gif</image>
      <click><![CDATA[
        alert('Hello from JavaScript!');
        skplayer.playMovie();
      ]]></click>
    </overlay>
  </uimodules>
</skplayer>
```

### Как заставить оверлей появиться – и скрыться.

Оверлейный плагин также может быть в широких пределах сконфигурирован для показа и скрытия его рабочей области в различные моменты времени, в ответ на различные события.

Задаются моменты показа и скрытия модуля следующими параметрами:

- **showAt** – конфигурация логики показа оверлея. Вложенные возможные параметры:
  - **event** – задает то, в момент каких событий данный оверлей будет показан. Возможные значения:
    - **finished** – когда видео доиграло до конца
    - **paused** – когда видео приостановлено.
    - **resumed** – когда проигрывание снято с паузы.
    - **mouseOver** – когда указатель мыши прошел над рабочей областью плеера.
    - **viewsLimit** – когда достигнут предел количества разрешенных просмотров.
  - **time** – задает, на какой секунде с момента начала просмотра, оверлей будет показан.
- **hideAt** – конфигурация логики скрытия оверлея. Вложенные возможные параметры:
  - **event** – задает то, в момент каких событий данный оверлей будет скрыт. Возможные значения:
    - **finished** – когда видео доиграло до конца
    - **paused** – когда видео приостановлено.
    - **resumed** – когда проигрывание снято с паузы.
    - **mouseOut** – когда указатель мыши прошел над рабочей областью плеера.
  - **time** – задает, на какой секунде с момента начала просмотра, оверлей будет скрыт.

Примеры использования

Показать оверлей в момент паузы и скрыть, как только воспроизведение продолжается.

```
<skplayer>
...
  <uimodules>
    <overlay>
...
      <showAt>
        <event>paused</event>
      </showAt>
      <hideAt>
        <event>resumed</event>
      </hideAt>
    </overlay>
  </uimodules>
</skplayer>
```

Показать оверлей на 5й секунде и скрыть на 15й

```
<skplayer>
...
  <uimodules>
    <overlay>
...
      <showAt>
        <time>5</time>
      </showAt>
      <hideAt>
        <time>15</time>
      </hideAt>
    </overlay>
  </uimodules>
</skplayer>
```

## Share this video – распространяем видео по сети

Плагин обеспечивает «вирусоподобное», легкое распространение видео как по сайтам, так и по социальным сетям. Единственная конфигурация, которая может ему понадобиться – это расположение его в рабочей области плеера. Плагин добавляется в конфигурацию путем добавления секции <share> в <uimodules>

```
<skplayer>
...
  <uimodules>
    <share>
      <cssData><![CDATA[
#share {
  top: 20px;
  left: 20px;
}
      ]]></cssData>
      <autoHide>true</autoHide>
    </share>
  </uimodules>
</skplayer>
```

В данном примере также задействован параметр `autoHide=true`, что приводит к тому, что плагин будет немедленно скрыт как только указатель мыши покидает рабочую область плеера или бездействует некоторое время.

## Внешний вид плеера

SKPlayer предоставляет возможности по изменению его внешнего вида в широких пределах без каких-либо навыков программирования и, в большинстве случаев, без каких-либо специальных инструментов.

### Архитектура «шкурки».

Внешний вид определяется задействованной на данный момент «шкуркой» плеера. Пример таковой – шкурка плеера, задействованная по умолчанию. Вы можете найти ее в директории skins/default – там, где установили сам плеер. Там вы найдете несколько файлов:

- **skin.xml** - главный файл, в котором описаны компоненты шкурки. Фактически это просто перечисление всего, что входит в оформление плеера. Параметр `css` указывает на относительное расположение файла со стилями, параметр `decal` задает местоположение файла, описывающего трафареты, параметр `fonts` задает местоположение SWF-файла, в котором содержатся шрифты. Включение шрифтов в оформление необязательно – в этом случае плеер будет использовать шрифты, установленные на машине клиента, но в этом случае следует быть внимательным – поскольку использованных вами шрифтов на клиентской машине может не оказаться.
- **decal.xml** – файл, который задает как имиджи, использованные во внешнем оформлении, так и то, какие их части использовать для различных частей.
- **CSS-файл** – описывает внешнее оформление плеера и его плагинов. Представляет собой обычный CSS-файл, но может содержать нестандартные поля, а также не поддерживает все возможности полного CSS, хотя и достаточно близок.

### Создание своего оформления плеера.

В процессе

## SKPlayer JavaScript API

DOM-объект плеера предоставляет для взаимодействия с ним набор функций, которые вы можете использовать для управления плеером, получения информации о его состоянии. Также вы можете получать обратную связь в вашей HTML-странице в те моменты, когда внутри плеера происходят различные события.

Плеер экспортирует следующие функции:

- `playMovie()` – плеер начинает играть текущее видео
- `stopMovie()` – плеер останавливает видео.
- `hideModule(moduleName:String)` – скрывает модуль, указанный в конфигурации под именем `moduleName`
- `showModule(moduleName:String)` – делает указанный модуль видимым
- `getState():String` – возвращает значение текущего статуса плеера. Возможные значения
  - будет описано позже
- `getEmbedCode():String` – возвращает код эмбеда.

Плеер вызывает функцию `skplayerReady(id:String):void` в момент, когда он завершил загрузку конфигурации. В `id` передается ID объекта плеера – к примеру, чтобы вы могли отличить, какой из нескольких плееров на странице произвел загрузку.

Также плеер будет вызывать на всех событиях, происходящих внутри ядра, JS функцию, имя которой указано в параметре конфигурации **`externalEventCallback`**. В функцию передаются два параметра – ID плеера и тип события. К примеру, вот как на демонстрационном сайте функция `skplayerEvent` показывает рекламу из `iframe` на паузе или в конце фильма, и скрывает, как только воспроизведение началось или продолжилось:

```
function skplayerEvent(player_id, event_type) {
    var adx = document.getElementById("player_ads");
    switch (event_type) {
        case "skplayerPaused":
        case "skplayerMovieEnded":
            adx.style.visibility = 'visible';
            break;
        case "skplayerResumed":
            adx.style.visibility = 'hidden';
            break;
    }
}
```

Возможные типы событий:

(В процессе)

## Использование плагинов OSMF-ядра

OSMF-ядро плеера может загружать также сторонние плагины для OSMF (Open Source Media Framework – продукт Adobe с открытым исходным кодом). У всех таких плагинов есть стандартный способ конфигурации – они используют для этого метаданные, которые передает им ядро. Вы можете описывать эти метаданные в конфигурационном файле, для каждого плагина, который используете.

Добавляются плагины в секции `core` конфигурации плеера. Запись конфигурации плагина включает в себя элемент конфигурации с произвольным именем, в котором указан URL плагина (относительно директория, где размещено ядро плеера, или абсолютный) и прочие параметры. Параметр URL, тем не менее, вы можете тоже пропустить, если имя секции конфигурации совпадает с именем SWF плагина. Ниже приведены случаи применения и описаны необходимые примеры.

## HttpPseudoStreaming

Плагин является проприетарной разработкой SteelSky Software и обеспечивает псевдостриминг по HTTP-протоколу. Псевдостриминг отличается от обычной прогрессивной загрузки тем, что обеспечивает позиционирование не только в той части видео, которая уже загружена, но и также переход к тем частям видео, которые еще не загружены. Для того, чтобы плагин работал, вам **необходимы**:

### *Специальным образом обработанное видео.*

Для того, чтобы плеер мог запросить у сервера тот отрезок видео, который еще не загружен на плеер, ему необходимо иметь информацию о ключевых точках во всем видеофайле. Эта информация содержится в метаданных, добавляемых в видео специальными программами. К примеру, метаданные в FLV вы можете добавить программами `flv2tool` или [yamdi](#). Для MP4 удобнее всего пользоваться [MP4Box](#) версии по крайней мере 0.2.4.

Добавление метаданных в FLV – берем файл `infile.flv`, и получаем `movie.flv`, который можно выкладывать на сервер:

```
yamdi -i infile.flv -o movie.flv
```

Добавление метаданных в MP4.

```
MP4Box -inter 0.5 movie.mp4
```

### *Поддержка псевдостриминга на стороне сервера.*

Как правило, для этого используются серверы `nginx` или `lighttpd` с соответствующими модулями.

### [Поддержка FLV в nginx.](#)

Поддержка FLV в lighttpd – включенным в поставку lighttpd модулем mod\_flv.

[Поддержка MP4 в nginx.](#)

[Поддержка MP4 в lighttpd.](#)

Подключение плагина плеера

Плагин находится в SWF core/plugins/HttpPseudoStreamingPlugin.swf и загружается следующей конфигурационной записью:

```
<skplayer>
...
  <core>
    <plugins>
      <HttpPseudoStreamingPlugin/>
    </plugins>
  </core>
</skplayer>
```

Также вы можете изменить некоторые параметры конфигурации плагина. Доступные параметры

- **dynamicBuffer** – если выставлен в true, то плагин будет менять размер буфера, в котором хранится воспроизводимое видео, меняя его размер по мере «провисаний» в процессе проигрывания. По умолчанию буфер статический, и буферизуется видео продолжительности, достаточной для 1 секунды воспроизведения.
- **startFormat** – формат, в котором плеер добавляет параметр в URL, передающий серверу смещение, с которого необходимо начать воспроизведение. По умолчанию формат имеет вид «start=\${start}» - где \${start} меняется на величину смещения, передаваемую серверу в URL запроса.

Для примера включим динамическую буферизацию и предположим, что наш стрим-сервер ожидает смещение в параметре startFrom:

```
<skplayer>
...
  <core>
    <plugins>
      <HttpPseudoStreamingPlugin>
        <metadata ns="http://steelskysoft.com/skplayer/http_pseudostreaming">
          <dynamicBuffer>true</dynamicBuffer>
          <startFormat>startFrom=${start}</startFormat>
        </metadata>
      </HttpPseudoStreamingPlugin>
    </plugins>
  </core>
</skplayer>
```

Все. Теперь достаточно указать плееру на видео с прописанными в него метаданными, и пользователь может легко в любой момент прокрутить видео на любую позицию.

## YouTube

Плагин обеспечивает воспроизведение видео, размещенных на популярном сервисе [YouTube](#). Используя этот плагин, вы соглашаетесь с [условиями использования YouTube API](#).

```
<skplayer>
...
  <core>
    <plugins>
      < YouTubePlugin/>
    </plugins>
  </core>
</skplayer>
```

Плагин не имеет никаких конфигурационных параметров – вам достаточно теперь указать плееру URL страницы YouTube, где размещено видео.

```
var flashvars = {
    movieURL: 'http://www.youtube.com/watch?v=MCLsktSGOVg&feature=related',
    config: " http://www.example.com/skplayer3/config.xml"
};
```

## MAST

MAST (Media Abstract Sequencing Template) является предложенным Akamai открытым стандартом описания рекламы в видео. Его полное описание доступно по [этому адресу](#). Фактически вы можете считать, что MAST есть разновидность «плейлиста» для рекламы, который задает когда и где показывать рекламу выбранных провайдеров. Он является более высокоуровневым описанием рекламы, чем, к примеру, стандарты VAST и VPAID, которые не предназначены для чтения или составления человеком. Плеер берет из MAST-документа, который вы создали, ссылки на источники рекламы в форматах VAST или VPAID, в нужные моменты берет от них данные по самому рекламному ролику и показывает пользователю, отслеживает все события, передавая их рекламному центру, обрабатывает клики. Таким образом, ваша задача состоит из нескольких частей:

1. Выбрать рекламных провайдеров и узнать у них ссылки на URLы доставки VAST/VPAID.
2. Составить файл MAST в XML-формате, который будет описывать: когда, где и чью рекламу вы хотите показать, вписав в него URLы доставки рекламы с нужными параметрами.
3. Дать плагину ссылку на этот документ.

Разберем небольшой пример работы с OpenX – добавим зону pre-roll, которая будет показана пользователю перед каждым видео.

Работа с VAST описана достаточно подробно в [официальной документации](#), так что эту часть мы затронем лишь немного: вам необходимо создать зону в Inventory->Zones->Add new zone, выбрать тип Inline Video или Overlay, добавить в нее баннеры. Теперь главное – ID зоны. Вы его видите в списке зон, когда, к примеру, проводите мышью над ссылками, он виден как

“zoneid=число”. Предположим, что OpenX установлен у нас по адресу <http://www.example.com/openx/> - тогда необходимый URL доставки рекламы будет у нас такой:

[http://www.example.com/openx/www/delivery/fc.php?script=BannerTypeHtml:vastInlineBannerTypeHtml:vastInlineHtml&zones=<ZONE\\_IDS>&block=1&format=vast&charset=UTF-8](http://www.example.com/openx/www/delivery/fc.php?script=BannerTypeHtml:vastInlineBannerTypeHtml:vastInlineHtml&zones=<ZONE_IDS>&block=1&format=vast&charset=UTF-8)

Замените ZONE\_IDS на список зон, из которых вы хотите взять рекламу, разделенных вертикальной чертой.

Теперь создадим документ MAST. В нем мы укажем наши URLы доставки.

Вот примерный документ для нашего случая (предположим, что ID зон, которых мы хотим показать перед фильмами – 10 и 14):

```
<?xml version="1.0" encoding="UTF-8" ?>
<MAST xsi:schemaLocation="http://openvideoplayer.sf.net/mast
http://openvideoplayer.sf.net/mast/mast.xsd"
xmlns="http://openvideoplayer.sf.net/mast"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <triggers>
    <trigger id="preroll" description="preroll" >
      <startConditions>
        <condition type="event" name="OnItemStart" >
        </condition>
      </startConditions>
      <sources>
        <source
uri="http://www.example.com/openx/www/delivery/fc.php?script=BannerTypeHtml:vastInlineBannerTypeHtml:vastInlineHtml&zones=10|14&block=1&format=vast&charset=UTF-8" format="vast">
          <sources /> <!--Child sources, in case we had any
that were dependant on this one -->
          <targets>
            <target region="linear" type="linear"/>
          </targets>
        </source>
      </sources>
    </trigger>
  </triggers>
</MAST>
```

Обязательно обратите внимание – все амперсанды в URL были заменены на &amp; - это требование относится к любым строкам в XML.

Теперь сделаем так, что реклама будет работать – легким движением руки! Вот фрагмент конфигурационного файла:

```
<skplayer>
...
  <core>
    <plugins>
      <MASTPluginNew>
        <metadata ns="http://www.akamai.com/mast/1.0" applyTo="stream">
          <uri>http://www.example.com/skplayer/wwwdata/mast.xml</uri>
        </metadata>
      </MASTPluginNew>
    </plugins>
  </core>
</skplayer>
```

Здесь мы подгружаем плагин и передаем ему метаданные. URL <http://www.akamai.com/mast/1.0> является знаком плагину, что данные относятся к нему. applyTo говорит ядру, что «посылку» с конфигурацией оно должно оставить для плагина через поток видео. Эти два момента останутся всегда одними и теми же с данным плагином, так что вы можете просто копировать их в свои файлы конфигурации. Все, что вам нужно поменять – это параметр uri (не URL, а URI – будьте внимательны) – в котором вы указываете, где плеер должен взять конфигурацию для рекламы – тот самый документ, составленный нами выше.

Теперь указываем плееру на составленный нами файл конфигурации, начинается воспроизведение любого видео – и перед просмотром видео вы увидите один из рекламных роликов, добавленных в зоны рекламы. В OpenX запишется показ баннера, и вы сможете, кликнув по экрану плеера, на котором курсор мыши на момент воспроизведения рекламы меняется на заманчивый занесенный палец, перейти на целевой сайт. В OpenX при этом будет засчитан клик – все, как полагается.

Еще раз напоминаем – подробное описание стандарта MAST вы можете получить [здесь](#). Там описана работа с оверлеями, сопутствующими баннерами и прочими атрибутами развитой рекламной системы.

## Advertisement

Плагин разработан Adobe Systems Incorporated, и позволяет вставку видео-рекламы в произвольные моменты, как в виде линейных вставок, так и в виде оверлеев.

Использовать плагин чрезвычайно просто – конфигурация задает, что именно и в какие моменты показывать. К примеру, чтобы продемонстрировать рекламный ролик на 10й секунде видео, добавьте следующий фрагмент в конфигурацию:

```
<skplayer>
...
  <core>
    <plugins>
      <AdvertisementPlugin>
        <url>plugins/AdvertisementPlugin.swf</url>
        <metadata
name="midroll">http://gcdn.2mdn.net/MotifFiles/html/1379578/PID_938961_1237818260000_w
omen.flv</metadata>
        <metadata name="midrollTime">10</metadata>
      </AdvertisementPlugin>
    </plugins>
  </core>
</skplayer>
```

Возможные параметры, которые вы можете использовать:

- **preroll** – URL рекламного клипа, показываемого перед видео.
- **postroll** - URL рекламного клипа, показываемого после видео
- **midroll** – URL клипа, показываемого в произвольный момент.
- **midrollTime** – в какой именно момент необходимо запустить вставку midroll, в секундах. Плагин не обеспечивает абсолютной точности, так что если вы укажете 10ю секунду, воспроизведение может начаться парой секунд позже.
- **overlay** – URL рекламного ролика, который будет показан поверх видео.
- **overlayTime** – время, когда необходимо начать воспроизведение рекламного ролика в оверлее.

## SMIL

SMIL (Synchronized Multimedia Integration Language) является [стандартом W3C](#), предназначенным для описания интерактивных мультимедийных презентаций. Описание являет собой XML-документ, предназначенный как для написания человеком, так и для машинного создания. Вы можете задавать с его помощью списки воспроизведения как видео, так и аудио, презентации, слайд-шоу. Подробное описание формата выходит далеко за рамки данной документации, рекомендуется прочтение официального руководства.

Краткий пример: сначала мы воспроизводим mp3, одновременно с которым последовательно, в течение 5 секунд, каждый, демонстрируем слайды. Затем начинается стриминг по RTMP с сервера:

```

<smil xmlns="http://www.w3.org/2005/SMIL21/Language">
  <head>
    <layout>
      <region id="content"/>
    </layout>
  </head>
  <body>
    <seq>
      <par>
        <seq>
          
          
          
        </seq>
        <audio src=" http://example.com/audio/test.mp3"/>
      </par>
      <video region="content"
src="rtmp://cp67126.edgefcs.net/ondemand/mp4:mediapm/osmf/content/test/sample1_700kbps
.f4v"/>
    </seq>
  </body>
</smil>

```

Важно сохранить данный файл с расширением \*.smil или \*.smi – в противном случае плагин не сможет определить, что этот файл относится именно к нему.

Теперь добавим плагин в плеер:

```

<skplayer>
...
  <core>
    <plugins>
      <SMILPlugin/>
    </plugins>
  </core>
</skplayer>

```

Все – мы можем указать плееру путь к файлу:

```

var flashvars = {
    movieURL: 'http://www.example.com/shows/example.smil',
    config: " http://www.example.com/skplayer3/config.xml"
};

```

## Google Analytics

Плагин обеспечивает интеграцию с Google Analytics, отслеживание просмотров и событий. Плагин имеет открытый исходный код, полное описание находится [тут](#). Ниже приведен пример конфигурации – она применяется на демонстрационном сайте плеера:

```

<skplayer>
...
  <core>
    <plugins>
      <track>
        <url>plugins/GTrackPlugin.swf</url>
        <metadata ns="http://www.realeyes.com/osmf/plugins/tracking/google">
          <value key="reTrackConfig" type="class"
class="com.realeyes.osmf.plugins.tracking.google.config.RETrackConfig">
            <!-- Set your analytics account ID -->
            <account>UA-7945510-1</account>
            <!-- Set the url that you registered with your GA account -->
            <url><![CDATA[http://demos.steelskysoft.com]]></url>
            <!-- Set up the percent based tracking -->
            <event name="percentWatched" category="video"
action="percentWatched">
              <marker percent="0" label="start"/>
              <marker percent="25" label="view"/>
              <marker percent="50" label="view"/>
              <marker percent="75" label="view"/>
            </event>
            <!-- Set up the event tracking for the completed event -->
            <event name="complete" category="video" action="complete"
label="trackingTesting" value="1"/>
            <!-- Set up the event tracking for the completed event -->
            <event name="pageView"/>
            <debug>true</debug>
            <!-- How often you want the timer to check the current
position of the media (milliseconds) -->
            <updateInterval>250</updateInterval>
          </value>
        </metadata>
      </track>
    </plugins>
  </core>
</skplayer>

```

## Captioning – субтитры в видео

Плагин предоставляет возможность отображения субтитров в видео. Плагин поддерживает W3C стандарт [DXFP](#) (неполная поддержка). Плагин OSMF-ядра осуществляет обработку SMIL-документов и управление показом и скрытием субтитров в определенный момент времени, для собственно отображения субтитров используется плагин overlay.

Пример DXFP документа:

```

<?xml version="1.0"?>
<tt xmlns="http://www.w3.org/2006/04/ttaf1" xml:lang="en">
  <body>
    <p begin="00:00:02" end="00:00:06" class="caption">Hello there!</p>
    <p class="caption" begin="00:00:10" end="00:00:15">This subtitles brought to
you</p>
    <p class="caption" begin="00:00:18" dur="7">by SKPlayer 3.0 and Captioning
plugin</p>
  </body>
</tt>

```

Теперь сконфигурируем плеер и два плагина – CaptioningPlugin и Overlay:

```

<?xml version="1.0" encoding="UTF-8" ?>
<skplayer>
...
  <uimodules>
    <overlay>
      <id>captions_container</id>
      <css>/skplayer/wwwdata/captions.css</css>
      <html/>
      <useCaptions/>
    </overlay>
  </uimodules>
  <core>
    <plugins>
      <HttpPseudoStreamingPlugin/>
      <CaptioningPlugin>
        <metadata ns="http://www.osmf.org/captioning/1.0" applyTo="stream">
          <uri>http://example.com/skplayer/wwwdata/test_dfxp.xml</uri>
        </metadata>
      </CaptioningPlugin>
    </plugins>
  </core>
</skplayer>

```

CaptioningPlugin в качестве параметра допускает “uri”, который, собственно, и указывает URI документа в формате DFXP. Плагину overlay передаются его ID и дополнительный CSS, используемый для отображения как самого плагина, так и текста субтитров (показан ниже). Также выставлен флаг useCaptions, который заставляет плагин вести себя как окно отображения субтитров.

Примерный CSS для отображения субтитров:

```
#captions_container {
  backgroundColor: #000000;
  backgroundColorAlpha: 0;
  width: 800px;
  bottom: 80px;
  horizontalCenter: 0;
}

.caption {
  color: #FFFFFF;
  fontSize: 16;
  fontFamily: Arial;
}
```

Первый селектор CSS влияет на отображение плагина overlay, заданного с ID captions\_container. Второй – влияет на собственно наши субтитры (которым в документе придан класс caption). Вы можете использовать больше классов в DFXP-документе и придать им другие стили, которые также могут быть заданы в CSS.

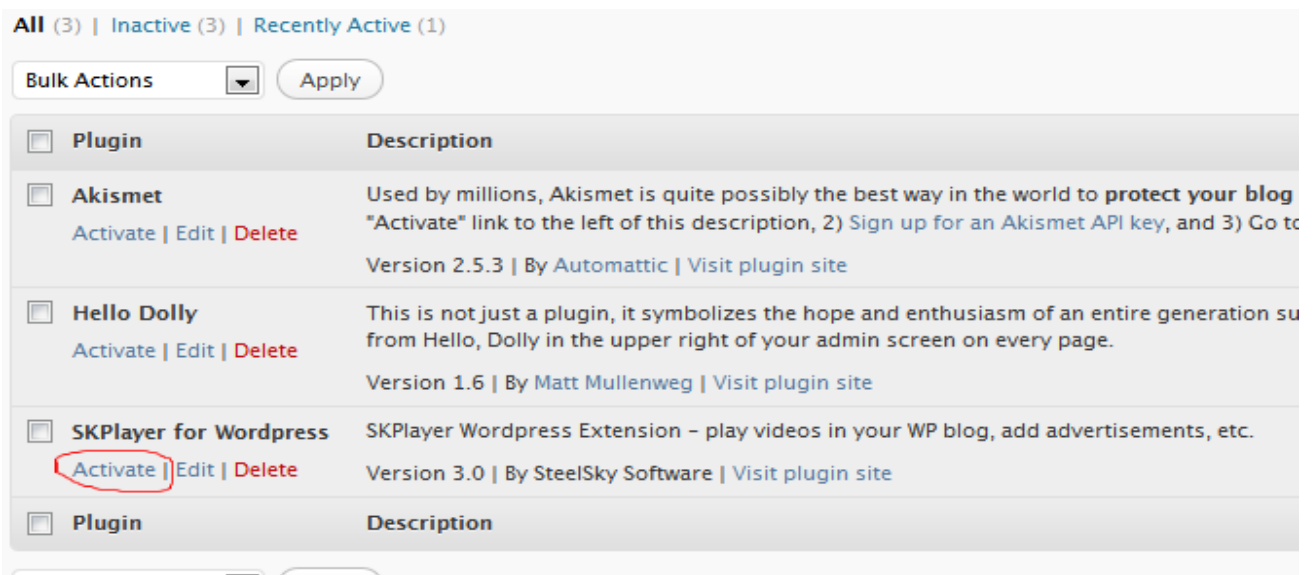
## Интеграция со сторонними приложениями и CMS

### Интеграция с Wordpress

SKPlayer поддерживает полную интеграцию с Wordpress, позволяя добавлять видео в любой пост. Поддерживаются все возможности плеера, дополнительные переменные, вы можете редактировать конфигурацию плеера прямо в WP, выбирать цвет фона.

#### Установка и настройка SKPlayer в Wordpress

Архив skplayer\_wp.zip разворачиваем в директорий wp-content/plugins. После этого в административной панели в разделе Plugins появляется новый пункт:



Кликните на Activate, чтобы активировать плагин.

Затем произведите начальную настройку плеера. После активации плагина в меню settings появился новый пункт – “SKPlayer”. Кликните на нем, чтобы открылась страница настроек плеера. Там вы увидите три элемента:

1. Flashvars по умолчанию. О них – немного ниже. Вы можете оставить это поле пустым.
2. Конфигурация плеера. Это – тот содержимое конфигурационного файла, но только в данном случае вам не приходится иметь дело с файлом – конфигурацию SKPlayer получает непосредственно из Wordpress. Самое минимальное содержимое поля: «<<skplayer/>» - пустой конфигурационный файл.
3. Цвет фона для плеера. Вы можете ввести его в виде шестнадцатеричного RGB значения (RRGGBB), так и воспользовавшись возможностью выбрать цвет из палитры. Можете не выбирать цвет – в таком случае фон плеера будет прозрачным.

Теперь вы можете добавлять видео в посты. Для этого в текст поста добавляется следующий код:

```
[SKPLAYER=media_url|thumb_url|WIDTHxHEIGHT|flashvars]
```

1. media\_url – URL к файлу, который вы хотите воспроизвести. Обязательный параметр.
2. thumb\_url – URL постера, который будет показан после загрузки плеера, до воспроизведения.
3. размеры плеера – ширина x высота.
4. flashvars.

Упомненные уже выше flashvars являются дополнительными переменными, которые передаются плееру. Они уже упоминались в разделе «параметры конфигурации», и выглядят как пары ИМЯ1=ЗНАЧЕНИЕ1,ИМЯ2=ЗНАЧЕНИЕ2 и т.д, через запятую. Наиболее очевидное их назначение – использование переменных замещения. Это, к примеру, позволит вам прокрутить перед видео произвольный рекламный ролик, указанный в посте, при этом задав в конфигурации рекламный ролик по умолчанию, который будет использоваться, если вы не указали другого. Разберем в качестве примера этот случай – мы хотим перед каждым видео на сайте прокручивать ролик ad\_1.flv, но в одном из постов перед видео показать ролик ad\_2.flv

1. Редактируем конфигурационный файл – добавляем туда поддержку плагина «Advertisement»:

```
<skplayer>
  <licenseKey>...</licenseKey>
  <core>
    <plugins>
      <advertisement>
        <url>plugins/AdvertisementPlugin.swf</url>
        <metadata name="preroll">{preroll}</metadata>
      </advertisement>
    </plugins>
  </core>
</skplayer>
```

Обратите внимание на переменную {preroll} – она будет заменена на реальное значение при помощи переданных плееру flashvars.

2. Теперь добавим ролик по умолчанию. В конфигурации плеера зададим:

```
replace.preroll=http://stream.example.com/ad_1.flv
```

Это ролик, который будет показываться перед всеми фильмами по умолчанию. То есть, если вы добавляете в пост код:

```
[SKPLAYER=http://stream.example.com/movie.flv|http://www.example.com/thumb.jpg|640x480]
```

3. Теперь создадим пост со вставленным кодом:

```
[SKPLAYER=http://stream.example.com/movie_2.flv|http://www.example.com/thumb2.jpg|640x480| replace.preroll=http://stream.example.com/ad_2.flv]
```

В нем будет показан фильм movie\_2.flv и перед ним пользователям будет показан ролик ad\_2.flv.

## Интеграция с DLE

SKPlayer поддерживает полную интеграцию с популярной CMS [DataLife Engine](#) (DLE). SKPlayer полностью заменяет стандартный плеер DLE (для проигрывания видео, аудио, YouTube) и использует часть его настроек (размеры, автостарт проигрывания при загрузке), которые вы можете выставить в административной панели.

### Установка SKPlayer в DLE

Архив skplayer\_dle.zip разворачиваем прямо на сайте в директорий **“engine”** DLE, либо разворачиваем на своей локальной машине и потом содержимое архива, к примеру, по FTP, загружаем на сервер – опять же, в директорий «engine». Таким образом, в директории DLE\_DIR/engine/modules должен появиться директорий skplayer.

Теперь добавляем шаблон SKPlayer в шаблоны тех страниц, где будет использоваться видео. Можно просто добавить его в main.tpl, раздел <head>...</head>.

```
{include file="engine/modules/skplayer/skplayer.php"}
```

Теперь, в лучших традициях DLE, мы начинаем править исходный код. Открываем файл engine/classes/parse.class.php. Находим там функции: build\_youtube, build\_video, build\_audio. Удаляем их и заменяем функциями из файла hack/add\_to\_parse.class.php.

Теперь в директории engine/modules/skplayer/player находим config.template.xml, копируем его в config.xml и правим. Самый простой файл конфигурации, который может вам пригодиться для DLE, приведен ниже – он использует HTTP-псевдостриминг и плагин для YouTube:

```
<skplayer>
  <licenseKey>
...Ваш ключ
  </licenseKey>
  <core>
    <plugins>
      <HttpPseudoStreamingPlugin/>
      <YouTubePlugin/>
    </plugins>
  </core>
</skplayer>
```

Все – после проделанных операций SKPlayer полностью подменит стандартный плеер DLE. **Внимание** – это произойдет только для тех публикаций, которые вы заново отредактируете.

Это связано с той особенностью DLE, что скрипт создает готовые HTML-версии страниц только на редактировании и потом хранит их в кэше.

Как уже сказано, SKPlayer полностью подменяет стандартный плеер и с ним совместим. Тем не менее, существуют определенные отличия.

1. SKPlayer поддерживает списки воспроизведения только в SMIL-формате.
2. Конструкция [video=...] BBCode была расширена, чтобы поддерживать «постеры» - изображения с превью видео. URL постера указывается после URL видео, через вертикальную черту. Например:

```
[video=http://example.com/video.flv|http://example.com/poster.jpg]
```

## SKPlayer ЧaBo и стрельба по проблемам

Q. Как мне отобразить первый фрейм видео как превью?

A. **autoLoad = true** в конфигурационном файле. Учтите – эта команда подгружает не только первый фрейм – она начинает буферизацию всего видео. Так что подумайте хорошо – возможно, вам все-таки лучше будет создать для каждого видео картинку в качестве превью.