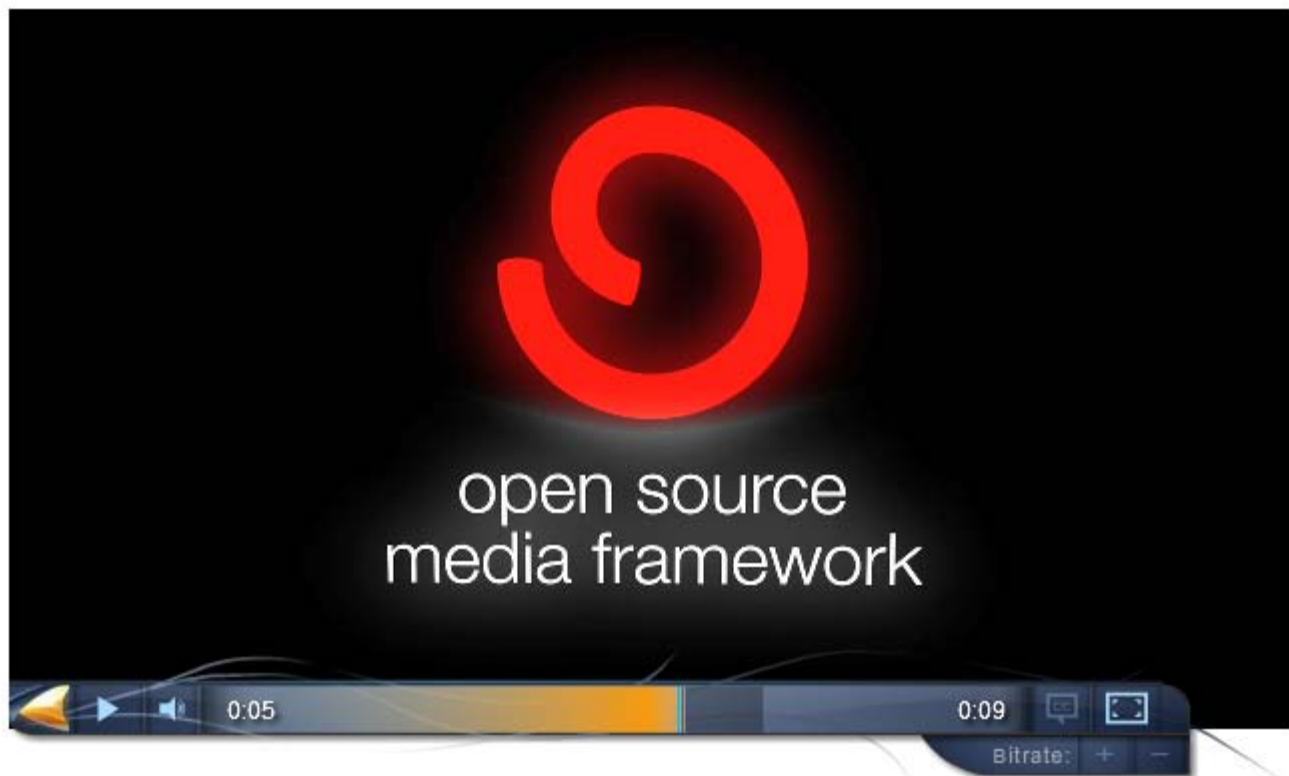


INTRODUCTION TO THE REALEYES OSMF PLAYER SAMPLE (REOPS)

Summary

The base of the RealEyes OSMF Player Sample (REOPS) offers an excellent starting point for creating a robust video player utilizing the Open Source Media Framework (OSMF) from Adobe. The REOPS project is meant to be a building block for developers as well as a visual representation to illustrate the capabilities and 'how to' of the OSMF framework.

The REOPS project includes a very extensible and robust control bar skinning solution and templates to help customize the control bar, as well as full-screen support, Closed Captioning from an external file, and OSMF dynamic plug-in support. The REOPS project can be used to deploy easily customized video players that support progressive video playback, video on demand streaming, live streaming and dynamic streaming. What is more, all of these features are configurable from an external XML file.



This article is an introduction to REOPS that will provide developers, designers, and/or implementers with the proper knowledge to customize and deploy a video solution based on the OSMF framework. This article is meant to be only a brief introduction into the visual and technical capabilities and implementation parameters of this new sample.

The sample contains the following resources:

- Flash Plug-in file (MXP) for easy install of the Flash IDE Component and the skin template files
- Flash implementation example for progressive, streaming, and dynamic streaming - including Closed Captioning for the basic streaming sample, and manual bit-rate switching for the dynamic streaming sample
- Flash Builder project archive file for the source code and base application setup

All the files for the REOPS project can be found at the Google Code page:

<http://code.google.com/p/reops/>

There are both downloads and SVN access available there.

NOTE: Future articles will dive deeper into the how REOPS uses the OSMF framework to build a robust media player solution. For now we will concentrate on getting you up and running with OSMF and the REOPS project.

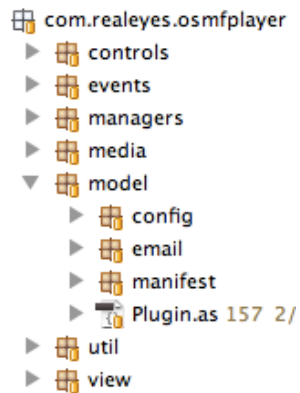


Getting Started

The REOPS Player Code

Download the ZIP file accompanying this article (REOPS.zip) or you can download the most current version of the code from the Google Code page <http://code.google.com/p/reops>.

The REOPS project is an ActionScript 3 project. The following describes the main structure of the REOPS project:



- **REOPS.as**: The main application file.
- **assets**: Contains the following directories
 - **data**: Default XML configuration file as well as additional configuration samples for:
 - Progressive
 - Streaming
 - Dynamic streaming
 - **skins**: Sample skins for the REOPS player
- **com.realeyes.osmfplayer**: is the main package for the REOPS classes. We'll cover details of the major classes in this and future articles.
- **libs**: Contains SWC libraries used in the project
 - **OSMF.swc**: **The OSMF framework**



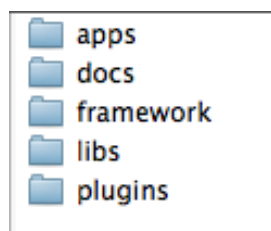
Download the OSMF Framework Code

The REOPS project comes with the OSMF.swc, but you may want to look at the actual OSMF framework code for yourself. There are a couple of ways to get the code:

- Download the OSMF source zip file from <http://opensource.adobe.com/wiki/display/osmf/Downloads>.
- If you are familiar with SVN you can checkout the source from SVN here as well <http://opensource.adobe.com/svn/opensource/osmf>

The OSMF Framework Files

The source files are organized into the following directories:



- **apps:** sample applications you can use to get familiar the OSMF framework
- **docs:** ASDocs for the framework classes. This will be in the SVN checkout, you can download the ASDocs from the download page as well.
- **framework:** The actual framework code with accompanying test code
- **libs:** Libs that are used with the samples and tests delivered with the OSMF framework code
- **OSMF.SWC:** The compiled code to use your projects. This will be in the downloaded zip file. If you check out the project from SVN you will need to compile the project to get the osmf.swc file.
- **plugins:** Sample OSMF plug-ins built for the OSMF framework



Setting up REOPS

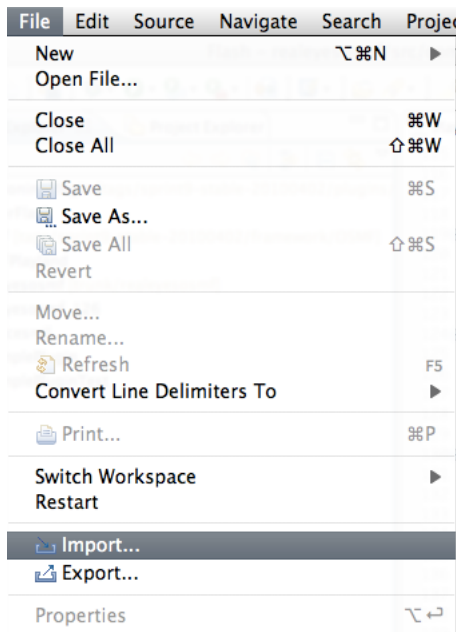
Now that you have downloaded the code you can set up Flash Builder and/or the Flash IDE to start working with the projects. The following sections will give you the details to get REOPS working in both environments.

Setting up the projects in Flash Builder

Import the REOPS project

Make sure you have downloaded and expanded the REOPS.zip or checked out the project code from the Google Code repository (<http://code.google.com/p/reops/>).

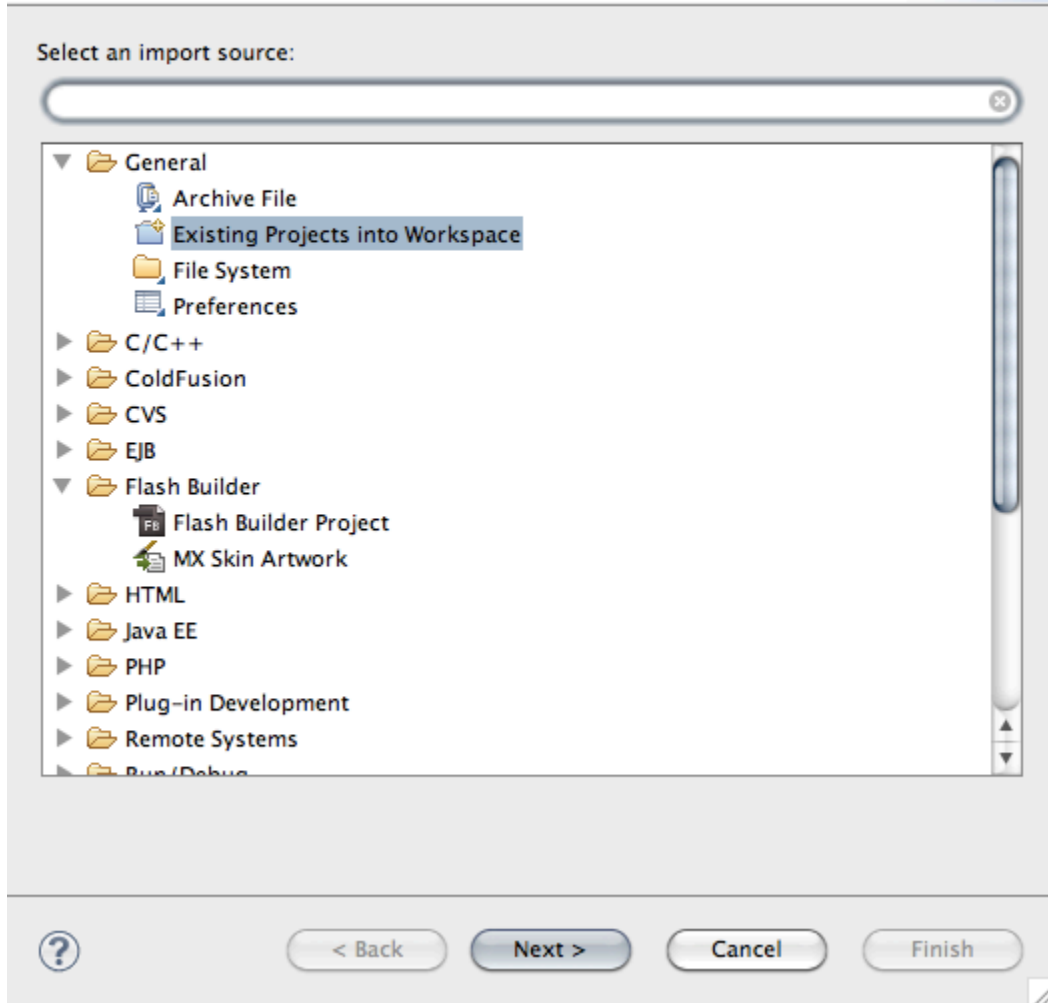
1. In the Flash Builder Main Menu select File -> Import



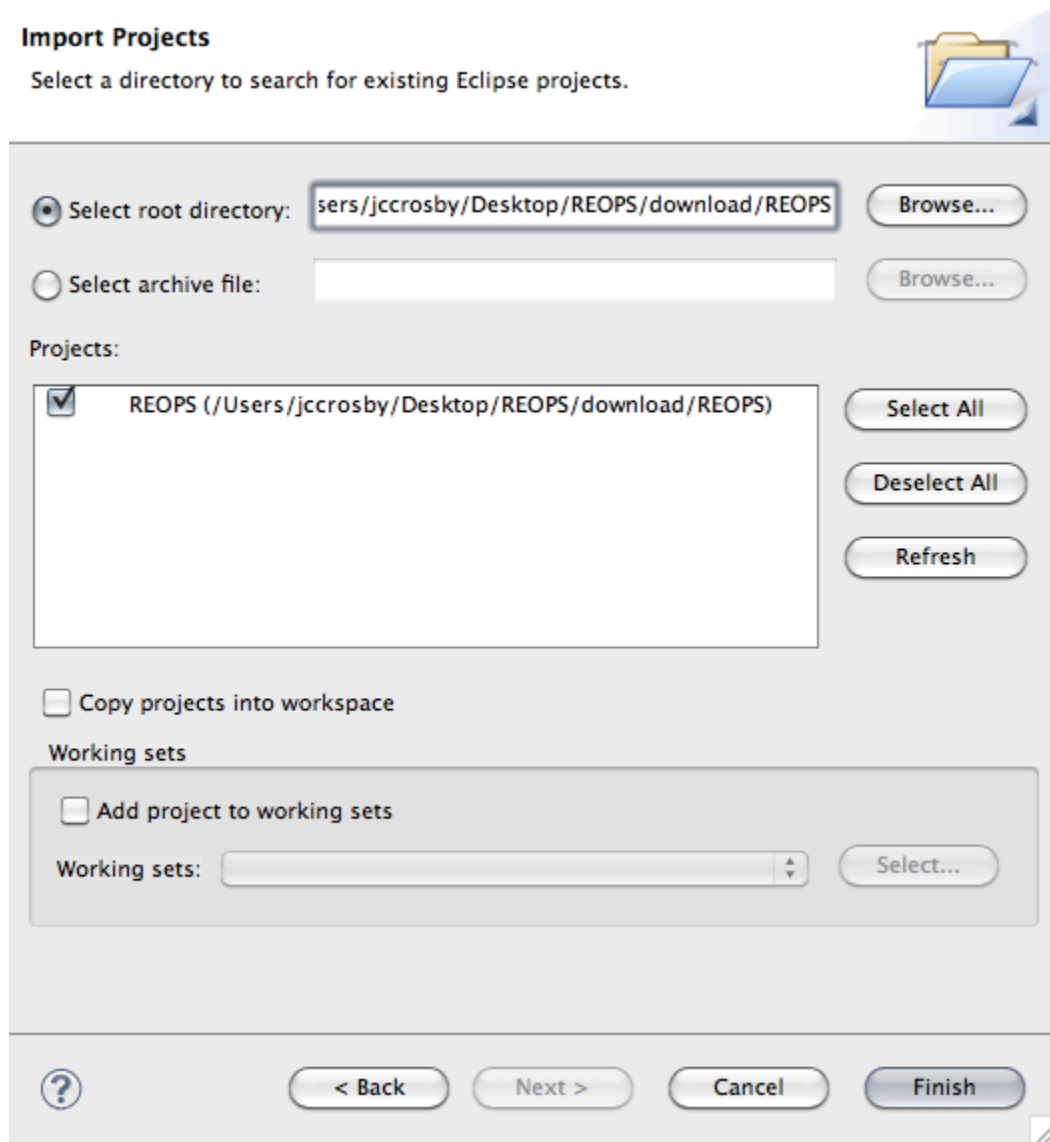
2. Next, in the 'Import' dialog select 'General' ->'Existing Projects into Workspace' in the tree list and click 'Next>'

Select

Create new projects from an archive file or directory.



3. In the next step of the 'Import' dialog click the browse button next to the 'Select root directory' radio button and browse to the Flash Builder project archive file or the REOPS project directory of your SVN working copy.



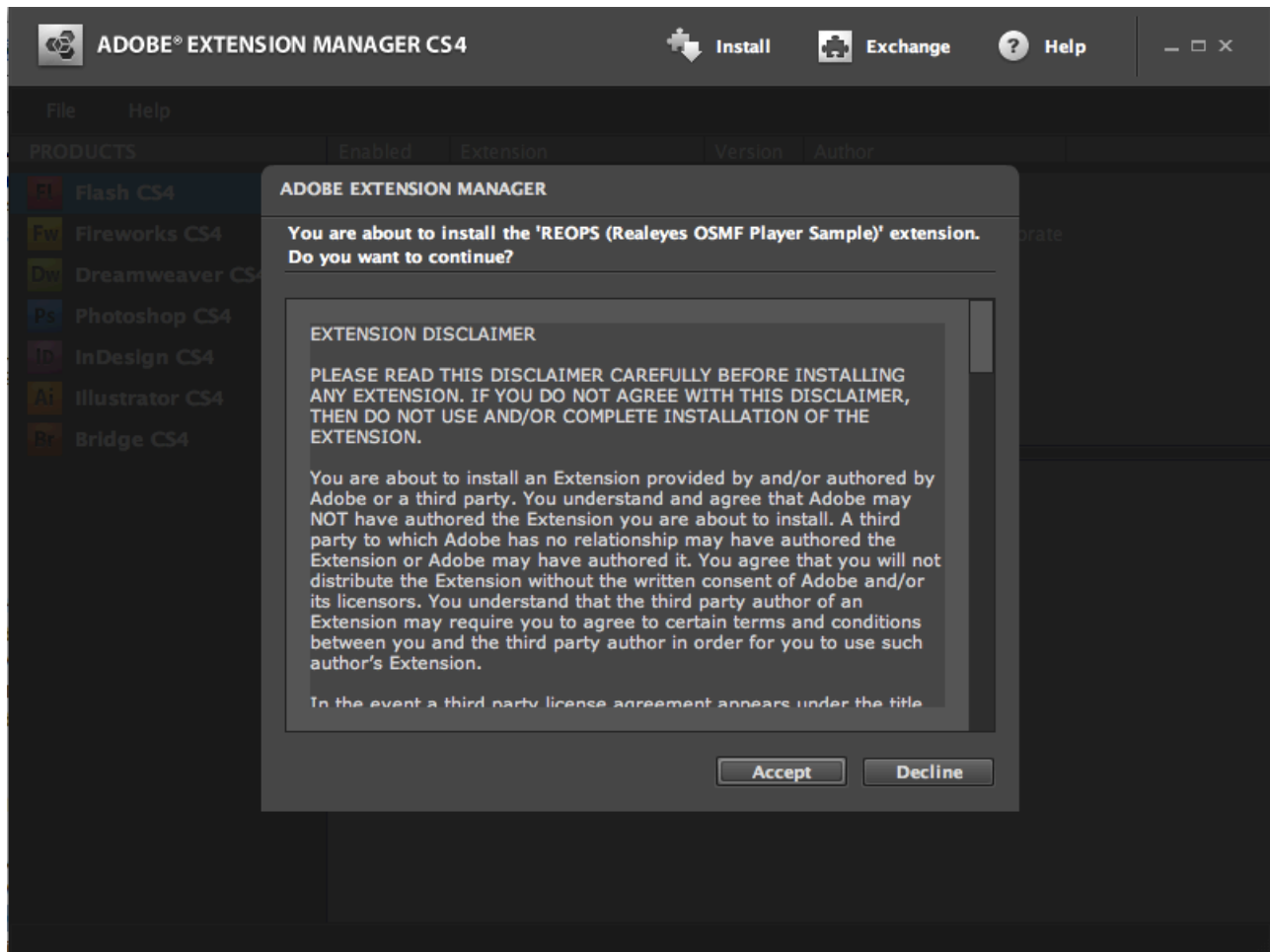
4. Select the REOPS project from the 'Projects' list and click 'Finish'.
5. The REOPS project should be successfully setup. To test the REOPS Player, from Flash Builder's main menu select 'Run' -> 'Run As...' -> 'Web Application'.
6. You should see the REOPS player start in the browser and play the sample video.
7. If you open the REOPS.as file in the default (root) package, after line 100 you will see additional configuration paths that you can use to test the different types of media playback:
 - o **PlayerConfig_Progressive.xml**: For progressive video
 - o **PlayerConfig_Streaming.xml**: For streaming video
 - o **PlayerConfig_DynStreams.xml**: For dynamic streaming video

NOTE: Later we will show you how to override the default configuration path using FlashVars

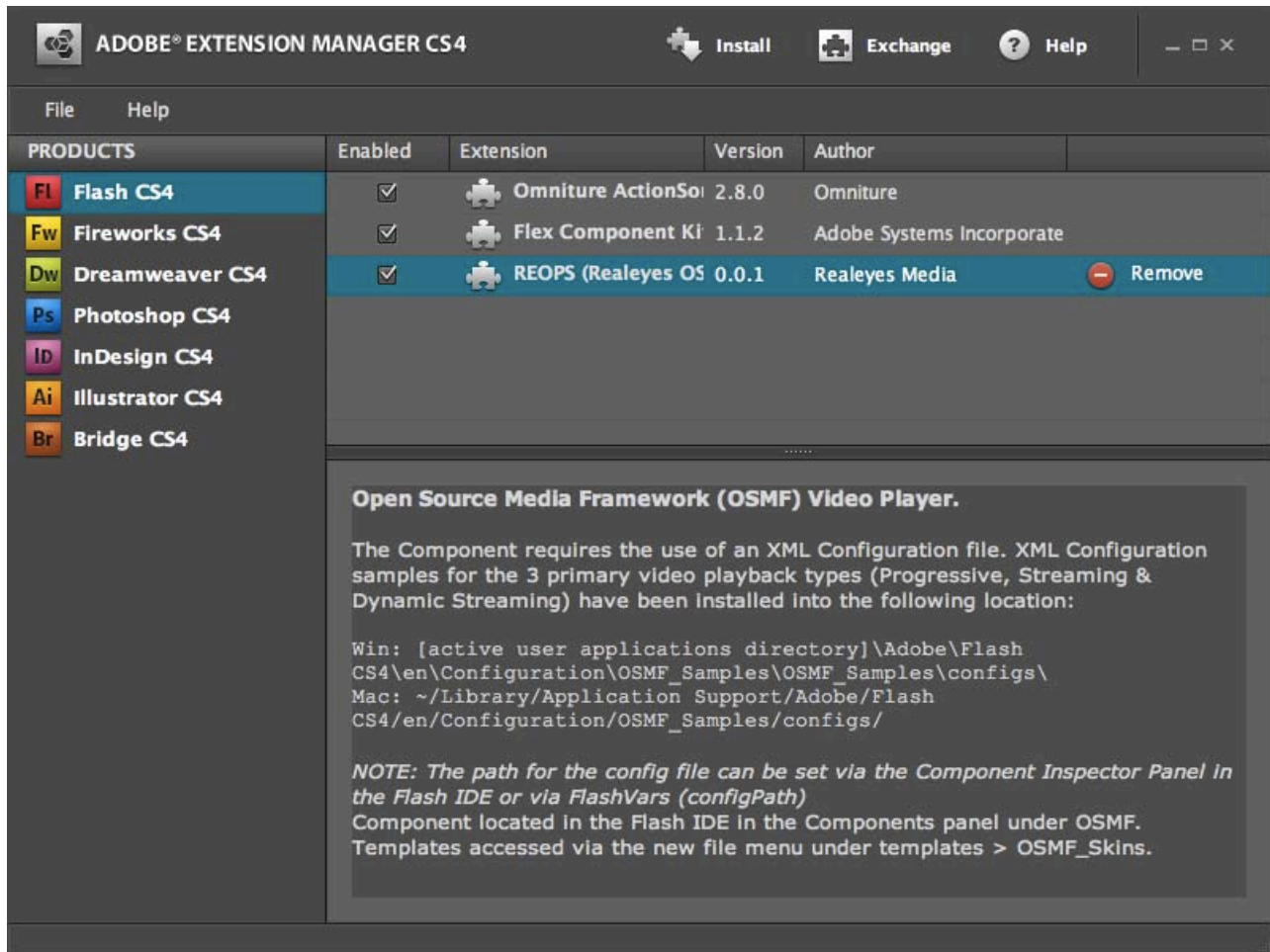


Setting up the project in the Flash IDE

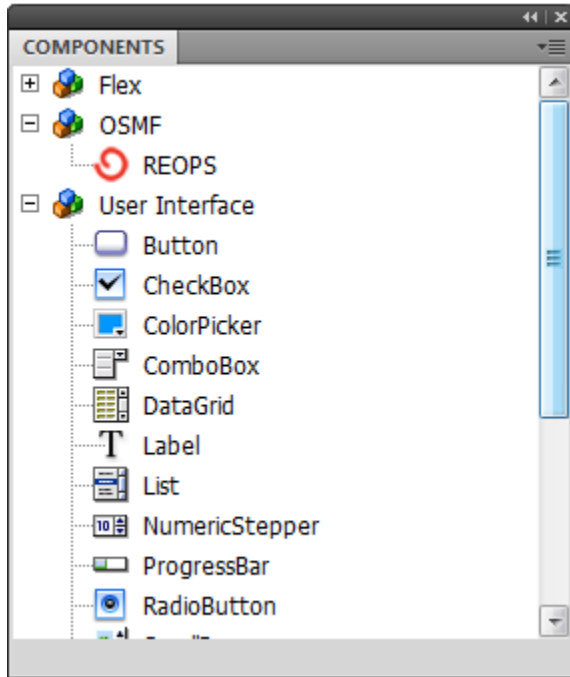
The first thing to do when getting setup is to install the REOPS.mxp file. The MXP file is an Adobe extension file and will automatically install the assets into the Adobe Flash IDE. Simply double click the MXP and as long as you have the Adobe Extension Manager installed (which is installed by default with Flash), it should launch automatically, and ask you to verify the user agreement.



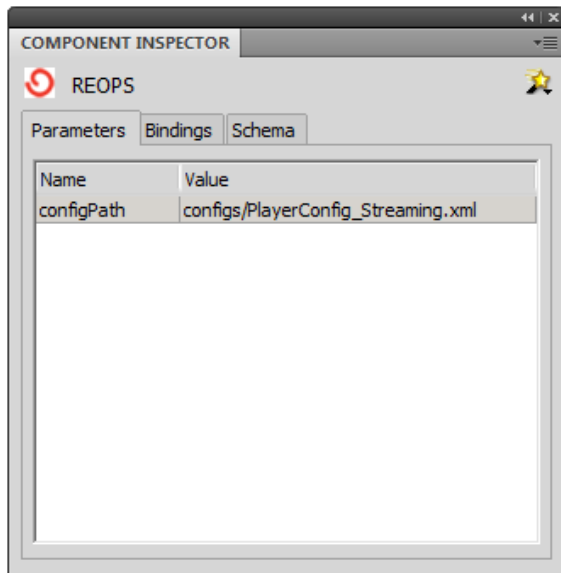
After you accept, you the extension should be installed and you should be prompted to restart Flash if it is already opened. In the Adobe Extension Manager, after the extension is installed it should be selected in the main display list, and you should see the install notes indicating what was installed and where.



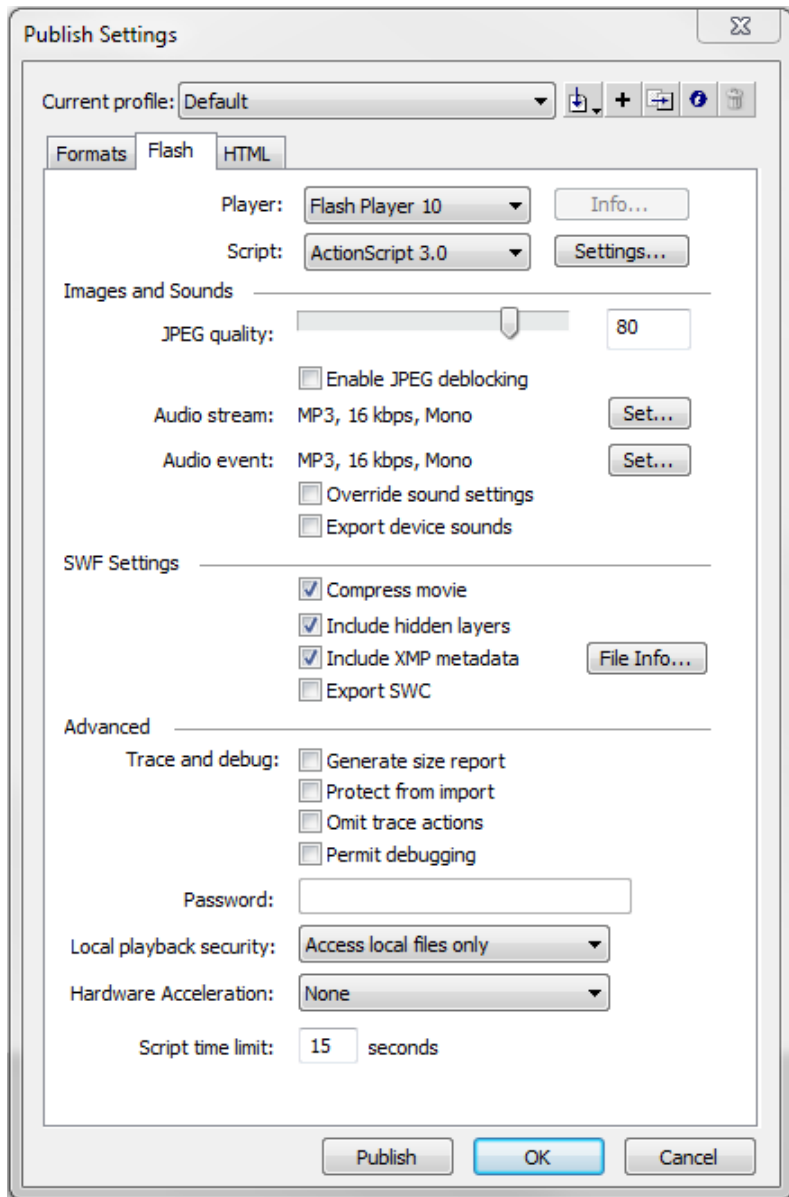
If you simply use the UI Component from the Flash IDE Components panel in a FLA file there will be no outside code dependencies as the necessary classes are all compiled into the component.



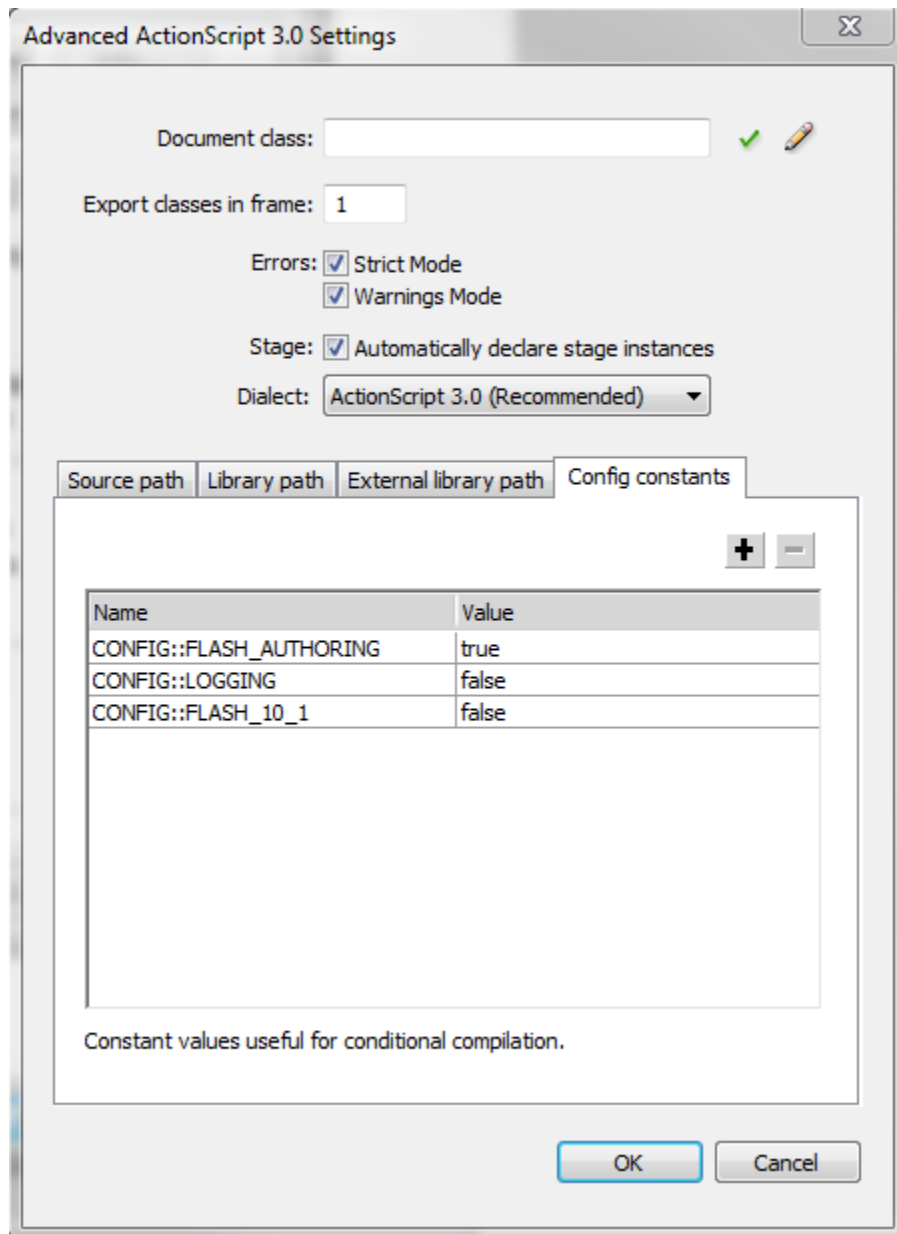
Besides a skin SWF containing the visual controls, the only external dependency when using the UI Component is the required external XML configuration file. That XML file, along with the skin, will need to be deployed with the player SWF file after it is compiled from the FLA. The path for the configuration file can be set from the Component Inspector Panel.



Although there are no external code dependencies, the OSMF framework and the Closed Caption application used by the REOPS component require 2 compiler config constants to be defined for the FLA source file. To set these constants, first open the publish settings dialogue box by clicking the File > Publish Settings menu option. From the Publish Settings panel select the Flash tab near the top.



Next click the ActionScript 3.0 Settings button near the top of the Flash tab's contents. Within the ActionScript 3.0 Settings panel, select the Config constraints tab. Inside the tab, add 2 variables by clicking the + icon button. The first should have a name of CONFIG::LOGGING and a value of false. The second should have a name of CONFIG::FLASH_10_1 and also a value of false. The end result of the panel should look like the below diagram.



If you want to use code base directly in the Flash IDE and not through the pre-compiled component, once you have downloaded the REOPS source code and OSMF source code you can add those to your application or environment class paths in Flash. You will still need the Config constants set above.

Understanding the Configuration File

When REOPS starts up, it loads an external XML configuration file. This file should contain configuration data for:

- The initial media the player will play
- Layout specifics for the player skin and controls
- Any dynamic plugins that REOPS will load

The REOPS XML configuration is made up of all or some of the following XML nodes:

- The **<player>** node (Required): The top level node that provides some of the basic settings the player needs to know about, such as width, height, scaleMode, autoPlay & hasCaptions.
- The **<playlist>** node (Required): Contains the initial media that the player will play. Children of this node can be of the following types:
 - The **<mediaElement>** node: Represents an explicit piece of media (Video, Audio, Image etc.).
 - The **<keyValueFacet>** node: Allows the configuration to specify data that will be added to the MediaElement's resource (we will cover more on the resource object later).
 - The **<serial>** and **<parallel>** nodes: Allow for multiple **<mediaElement>** child nodes and will create a SerialElement or ParallelElement for playback.
- The **<skin>** node (Required): Provides the path to the skin SWF for the Player as well as addition element data for layout and functionality.
 - The **<skinElement>** node: Provides data for different skin elements such as the control bar and loading icons.
- The **<plugin>** node: Provides a list of dynamic plugins that will be loaded via OSMF's PluginLoader. Static plugins are compiled into the REOPS player and will be addressed in a later article.

There are quite a few example XML configuration files provided in the REOPS project download. If you are using the Flash Builder archive or SVN checkout look in the `assets/data` directory. If you are using the MXP you can look in the following directory:

- **Win:** [active user applications directory]\Adobe\Flash CS4\en\Configuration\OSMF_Samples\OSMF_Samples\configs\
- **Mac:** ~/Library/Application Support/Adobe/Flash CS4/en/Configuration/OSMF_Samples/configs/

Feel free to open them up and explore the different configurations. We'll take a look at the `PlayerConfig.xml` to become more familiar with how the XML configuration provides data to the REOPS Player.



Open the PlayerConfig_Streaming.xml file. You will see all of the major XML nodes previously mentioned. We will review the major nodes here. Some of the values in this example may differ, but the main idea should be clear. First we will look at the <player> node:

```
<player
  width="640" height="480" scaleMode="stretch"
  isLive="false" autoPlay="true"
  updateInterval="250" hasCaptions="true">
```

Here the player node sets the width, height and scaleMode of the player. It lets the player know that it will not be playing live video. The video will play as soon as it is loaded (autoPlay="true") and have closed captioning (hasCaptions="true").

A note about closed captioning - If the hasCaptioning is set to true the CaptioningPlugin is loaded by the player and will look for a KeyValuePair specifying where to load the captioning data from. The Key value pair is set on the <mediaElement> node a child of the <playlist> node and something that we will cover in a future article. Let's look at the <playlist> node now.

```
<playlist>
  <mediaElement>
    <id>akamail0yearf8512K</id>
    <mimeType>video/x-flv</mimeType>
    <streamType>recorded</streamType>
    <deliveryType>streaming</deliveryType>
    <media      url="rtmp://www.server.com/app/filename"
              width="800" height="600" />
  </mediaElement>
</playlist>
```

The XML snippet above shows a single streaming FLV that has a closed caption file associated with it. This bit of XML is based on the initial F4M specification from the OSMF project (<http://opensource.adobe.com/wiki/display/osmf/Flash+Media+Manifest+File+Format+Specification>), but has been expanded to fit the needs of the REOPS player. The nodes to pay attention to here are:

- The **<media>** node: This specifies the rtmp path of the flv as well as the width and height.
- The **<keyValueFacet>** node: This node and its children specify data that is added to the MediaElement that is created when this data is loaded and provides the path to the closed caption file to load for the media when it plays.



The <skin> node provides the path to the skin swf file that will be used to set the chrome of the player controls. You can also provide child <skinElement> nodes to further enhance and adjust the look and functionality of the players controls. The skinning system will be covered in greater detail in a future article.

```
<skin path="assets/skins/RE_Skin.swf">
  <skinElement id="controlBar"
    elementClass="com.realeyes.osmfplayer.controls.ControlBar"
    initMethod="initControlBarInstance"
    scaleMode="NONE"
    hAdjust="0" vAdjust="0"
    vAlign="BOTTOM"
    autoPosition="true"
    draggable="true"
    autoHide="true" />
</skin>
```

The <skin> node above loads the RE_Skin.swf and provides the initial data for the ControlBar class specified by the elementClass attribute.

Finally, there is the <plugins> node (the reosmf_config.xml file doesn't contain a <plugins> node). The <plugins> node can have a list of <plugin> children that provide the REOPS Player with a list of URLs to load dynamic plugins from. The REOPS Player handles passing these plugins to OSMF's PluginLoader object. Plugins is a large subject on its own and we will be sure to provide some more detail in a future article.

Overriding the Default XML Configuration Path

Setting the XML configuration path via FlashVars

The default XML configuration path is relative to the player SWF file and will look for the assets/data/reosmf_config.xml file. You can override this location by providing a 'configPath' FlashVar and setting it to the path for the XML configuration you wish to load. This would look something like the following when using SWFObject:

```
var flashvars = { configPath: "assets/data/PlayerConfig_Progressive.xml" };
...
swfobject.embedSWF( "REOPS.swf", "flashContent",
  "640", "480", swfVersionStr, xiSwfUrlStr,
  flashvars, params, attributes);
```



Creating Custom Skins for the Sample

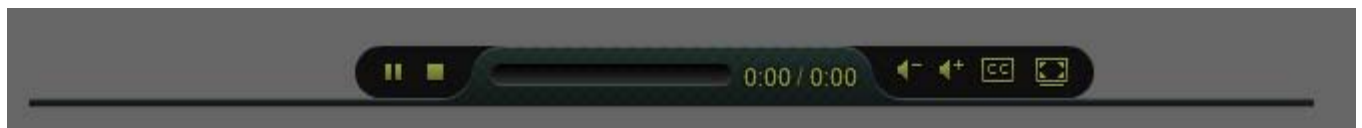
Overview of the Skinning Solution

The REOPS project offers a very powerful, extensible, and easily custom branded skinning solution. The skinning process is similar to the Flex Component Kit (FCK) for Flash based skinning. REOPS Skinning has support for easy template generation directly from the Flash IDE when the provided MXP file been installed, and even includes some top notch skins designed by Juan Sanchez (<http://www.scalenine.com>), illustrating some of the creative possibilities and freedom of the skinning system. Check out his blog post about the design process he used and the implementation with the REOPS project here:

<http://www.juanchez.com/2010/02/01/osmf-player-skins/>



Core Skin (Juan Sanchez)



Formula Skin (Juan Sanchez)



Lunar Skin (Juan Sanchez)

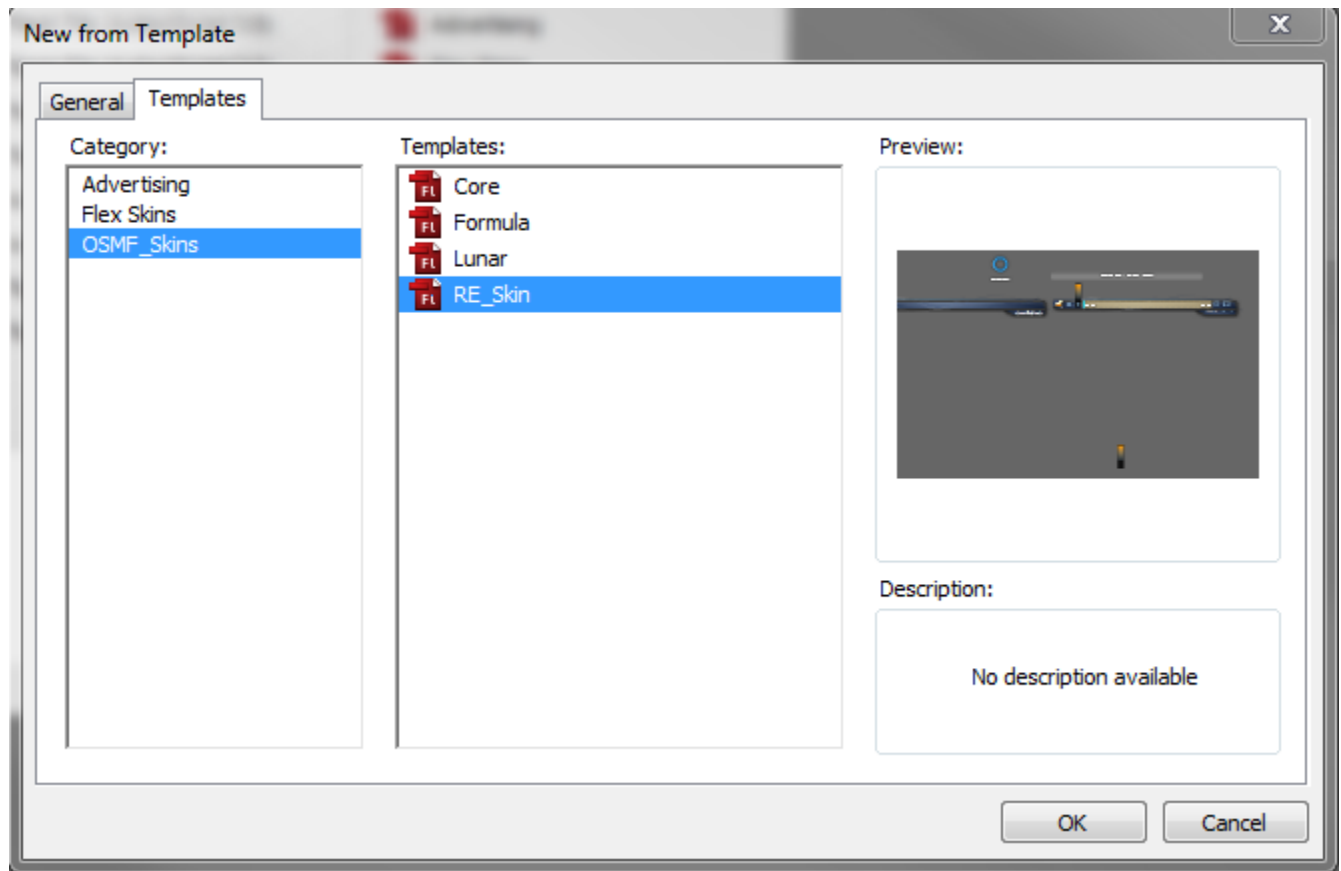
This skin was designed as a full template reference implementation by RealEyes:



RE-Skin (David Hassoun, using elements from Lunar)



All the above skins are installed as Flash templates with the REOPS Flash Extension (MXP). They are accessed in the initial start splash screen > Create from Template > OSMF_Skins or via the File > New > Templates (tab on top) > OSMF_Skins.



Aside from offering a similar graphic and component based skinning system as the FCK, including timeline driven states which include support for animated states and transitions, the REOPS skinning system has some very powerful and useful capabilities that many other skinning systems may lack. Some of the key differentiators for skinning are self-contained code and functionality, and a progressive componentization layout system. Since the skins use a compiled clip for the required code, they have the ability to be compiled or transferred to any other machine or designer/developer without the need to pass along any code or libraries unless code extension has been applied.

The progressive componentization starts with all the pieces of the skin broken into their simplest form (generally graphics that have a high potential of reuse – such as icons), then utilizes combinations of the graphics to create the basic components such as play/pause buttons. At this point Flash MovieClip timelines come in to play for state management (up, over, down, disabled, selectedUp, selectedOver, selectedDown, selectedDisabled). The 'selected' states are used for toggle buttons such as volume mute/on, and play/pause toggle. Then the components are laid out into composition skin elements such as the ControlBar or the LoadingIndicator. These components offer the freedom to layout the buttons and controls any way the design calls for, and is done within the stage of a MovieClip – or via ActionScript 3. The latter is possible if code is extended as might be desired for more complex solutions. This



process of allowing the whole skin element component to be constructed in the skin itself offers some interesting benefits, one of which is very valuable to the design phase itself. While the design is going on, the skin template FLA can be compiled at any time and the skin element components can be seen in a functional and relatively complete state. This allows the designer/developer uncompromised control over the functional and visual integration of the control bar without setting up complex test harnesses.

The skin elements are applied to the display within the REOPS application after the skin SWF is loaded in at runtime. The configuration file defines which elements to apply not by what is on the stage of the FLA and seen when the SWF is run, but rather by the class references for the skin elements in the library of the FLA. The full class paths for the desired skin elements must be defined in the configuration file within the player > skin > skinElement > elementClass node. There is one skinElement node for each skin element component you wish to apply. So for instance if you wanted to apply a Control Bar, a Loading Indicator, and a Closed Caption Field you would have 3 skinElement nodes. As an example the default elementClass for the ControlBar in the template is "com.realeyes.osmfplayer.controls.ControlBar". This can be found in the FLA by right clicking on the ControlBar component in the Library and selecting Properties. If you do not see the Class field, make sure you are in the 'Advanced' view, which is selected by a button labeled Advanced on the bottom right of the Basic 'Symbol Properties' panel. If you are in the Advanced view, simply copy the value from the 'Class' field and put it in the elementClass node. This is also where you could apply a custom class that extends the default noted above with additional custom functionality you create, thus offering a very powerful and standard way to add new functionality to your skin elements.

Customizing the visual elements of the skins is rather straightforward. The recommended approach is to adjust the graphical symbols such as the button background, the icons, and other such pieces like the progress bars. Then within the example components, which should provide the first level of componentization, adjust the states and transitions as you desire. All the example elements within the design portion of the FLA have already been applied with the proper instance names to assist with the next step. The next step is to add or remove any controls/components within the skin element components such as the ControlBar and adjust the positioning as desired. As long as all the subcomponents have the proper instance names, which they all do by default, then they should function as would be expected when utilized within the REOPS application.

Additional UI components can be added to the skin elements such as the ControlBar. If their code and functionality is self contained, then no special changes should be required. If they need to be integrated more closely with the ControlBar functionality and the REOPS, then you will likely have to extend some of the base classes and reapply the new classes to the components in the library. This will be covered in more depth in a future article, to help with such cases as wanting to add a share overlay, that is activated by a control bar button.



Deploying the Sample

To deploy the REOPS player to your web server you will need to upload the following files:

- The player SWF: **REOPS.swf**
- The configuration file: **resomf_config.xml**
- The skin SWF specified in the configuration file: **assets/skins/RE_Skin.swf**
- The hosting page: **index.html**
- The swobject.js file (<http://code.google.com/p/swfobject/>)

The index.html page should embed the player and if the config xml file path is different from the default path (assets/data/resomf_config.xml), also need to set the configPath FlashVar with the actual configuration file path.

Example index.html code using SWFObject:

```
<html>
<head>
  <title>REOPS Sample</title>
  <script type="text/javascript" src="swfobject.js"></script>
  <script type="text/javascript" charset="utf-8"></script>
  <script type="text/javascript">
    var params = {};
    params.quality = "high";
    params.bgcolor = "#FFFFFF";
    params.allowscriptaccess = "sameDomain";
    params.allowfullscreen = "true";
    var attributes = {};
    attributes.id = "REOPS";
    attributes.name = "REOPS";
    attributes.align = "middle";

    // Set the config path here
    var flashvars = { configPath: "path/to/reosmf_config.xml" };
    swfobject.embedSWF(
      "REOPS.swf",
      "flashContent",
      640, 480,
      "10.0.42",
      "playerProductInstall.swf",
      flashvars, params, attributes
    );
  </script>
</head>
<body>
  <div id="flashContent"></div>
</body>
</html>
```



Conclusion

The REOPS project is just beginning. This base and the future enhancements to the REOPS project should supply an excellent starting point for your own OSMF based media players, or be used out of box and simply select one of the existing skin templates or create your own. Already with what the OSMF project offers, the REOPS project is able to take advantage of the impressive capabilities it provides to give you a good starting point and stand as an example for some of the solutions that are possible with OSMF.

Although the skinning system has the least direct association with the OSMF, its initial capabilities provide a familiar process with enhanced benefits associated with the visual delivery and user interactions. A key principal of the skinning system is its extensibility – a later article in this series will delve more into the how-to of extending the visual side of the REOPS samples.

The external configuration system is also worth noting in regards to its extensibility. Already the basics offered have many uses but as the REOPS project continues, additional support and capabilities will be added to the external configuration capabilities. Integrating the REOPS sample with application server data services instead of the flat XML format used would be a relatively easy task, and that is just the beginning.

We are looking forward to providing a series of articles using the REOPS project to dive deeper into the OSMF framework. We'll discuss real world solutions, and ways to extended the current REOPS project with functionality from the OSMF project as it nears its 1.0 release.

Here are a few of the concepts and ideas that we will be covering in future articles:

- HTTP Streaming
- Live DVR
- DRM Protection with Adobe Flash Access 2.0
- External Playback API for episode/playlist user selection
 - External Interface integration for JavaScript
 - AS3 API
- Advertising plug-in integration (VAST/MAST)
 - In play
 - Overlay
 - Outside regions
- Tracking Plugin Integration
 - Generic JavaScript Invokers
 - Google Analytics

Check back to Adobe Devnet in the coming weeks to see what new articles and features have been published.



Author Profiles

David Hassoun

twitter: [@hotkeys](#)

blog: <http://david.realeyes.com>

David Hassoun is the founder of [RealEyes Media, LLC](#), a digital media firm based in Colorado that focuses on interactive motion media and advanced Flash and Flex platform applications. David has always had a passion for motion media, the power of video, and the challenges of usability and interactivity. David is an Adobe Certified Master Instructor, teaches advanced RIA classes at the University of Denver, serves as the Rocky Mountain Adobe User Group Manager, and has taught and developed advanced Flash and Flex application courses. As a consultant or while employed with other firms, he has worked for a wide range of companies such as American Express, Chase Manhattan, Qwest, Boeing, Macromedia, Adobe, US Air Force, Bechtel/Bettis, and many more. David regularly performs advanced code and technical best practices reviews, and has provided directional advice for international industry leaders over the past years—including many technical, courseware, and application reviews as an industry expert.

John Crosby

twitter: [@jccrosby](#)

blog: <http://thekuroko.com>

John Crosby is the Senior Solutions Architect and a Partner at [RealEyes Media, LLC](#). Always interested in new things, John has his fingers in many different projects and technologies as well as having helped to create training curriculum for Flash, Flash Mobile and Flex. As former professional foodie turned keyboard-jockey, John is always on the lookout for tools, concepts & ideas to make development more interesting, reliable, and fun.

