

Flash Integration with Ad Serving

PREFACE

This document is intended for Flash developers who are building advertising elements that will be delivered through ad servers. Although the concepts presented are relatively simple, a basic familiarity with Flash construction using layers, buttons, and the Actions Window is assumed. If those topics are unfamiliar, please refer to the Macromedia Rich Media Advertising Topic Center located at: <http://www.macromedia.com/resources/richmedia/tracking/>

CONSIDERATIONS

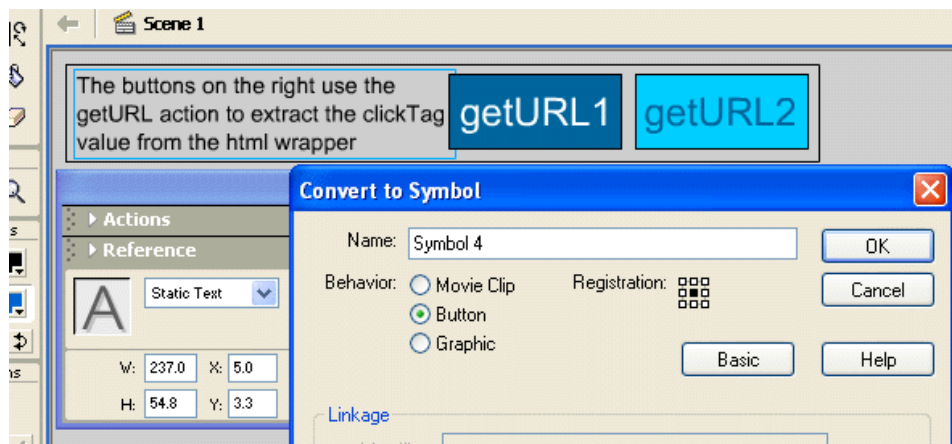
Some forethought should be given during the design and development phase of a Flash creative about the actual implementation process. Such considerations include:

- What area is meant to be clickable
- How many destination URL's will be used
- How to handle non-Flash users
- Are there embedded movies
- Is the ad dependent on external movie loads

Clickable Areas

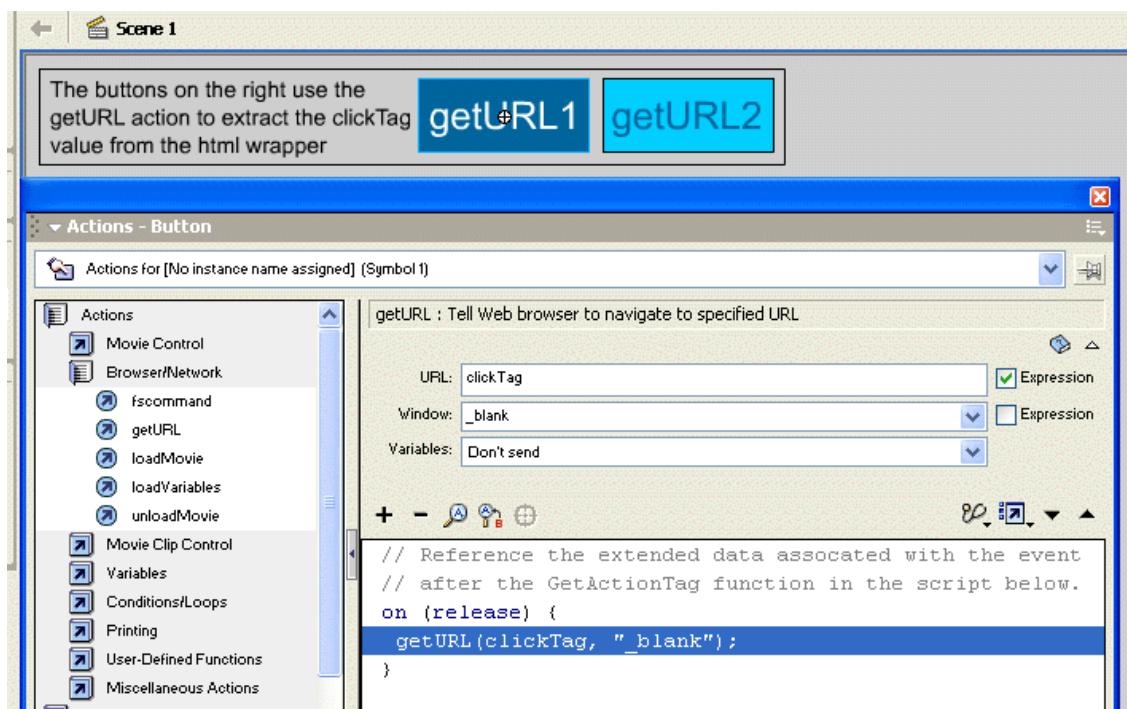
Smooth animations, clean graphics, and multi-dimensional areas may be the benefits of using Flash, but the benefit to Advertisers is a user that clicks through. Since ad servers have proprietary tracking and reporting mechanisms, it is necessary to utilize a sufficiently generic methodology that will allow an ad server to append its tracking to an .swf file.

Although Flash MX has made text linking with URL's much more user-friendly, for the purpose of ad-serving all text should be converted to a button using the "Convert to Symbol" option. Likewise, any other graphics or mask layers that are intended to be clickable should be converted to a button.



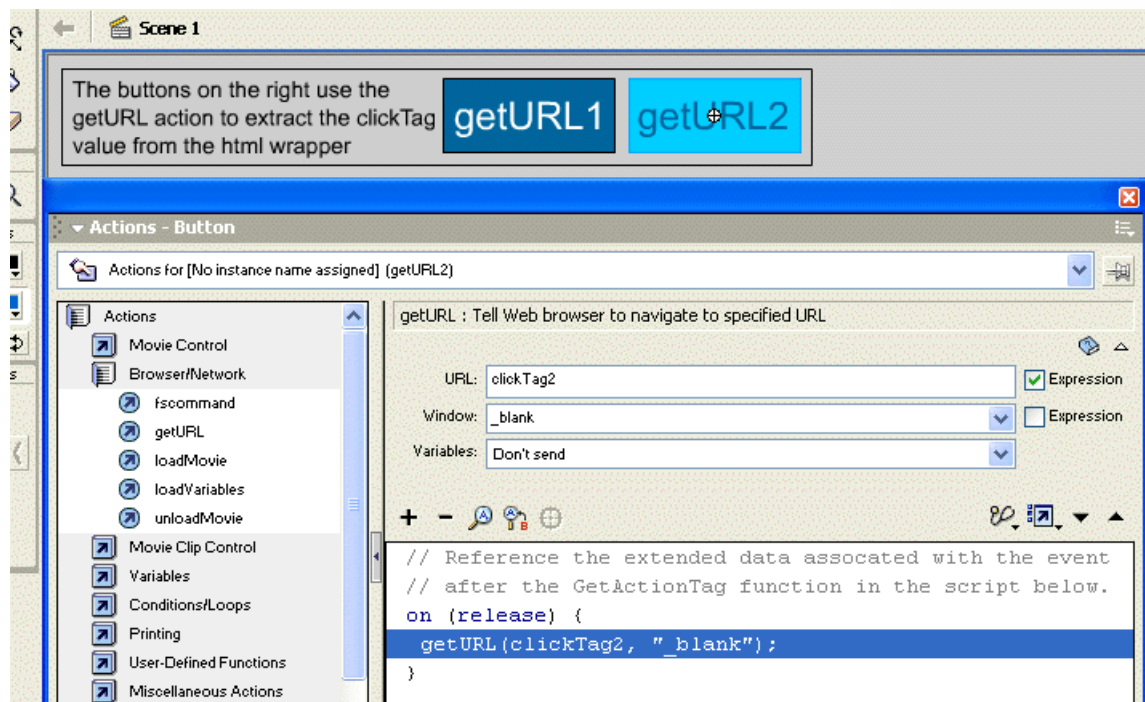
A mouse action can now be assigned to the button. It is imperative that URLs not be hard-coded into the creative, since an ad server will be unable to modify an exported .swf file. However, by utilizing a variable name within the `getURL ()` action for the button, Flash is able to reference the destination URL from its HTML wrapper, where an ad server can freely interact with the content. Atlas specifies this URL by the variable name "clickTag1".

After highlighting the button to be clicked, open the Actions Window, and select the `getURL` method for the button. Under the options portion of the Actions Window, specify "clickTag1" for the URL, and check the "Expression" box next to it. This designates clickTag1 as a variable name not to be interpreted as a literal string, so that Flash will evaluate the destination URL properly. Set the dropdown box for "Window" to "_blank", which has the effect of opening a new browser window.



Multiple Destinations

If the creative contains several areas that are clickable but point to the same destination, the `clickTag1` variable can be reused. However, if each clickable area needs to be tracked separately, or if the creative contains unique destinations for each button, then the `clickTag` variable needs to be unique as well. This can be accomplished by placing an integer at the end of each `clickTag` that will increment for each corresponding area. Thus, if there are three clickable areas, there will be three unique `clickTag` values: `clickTag1`, `clickTag2`, and `clickTag3`.



During the ad creation process within Atlas, each of these destinations are created and assigned their respective destinations. They are then automatically added to the flash HTML wrapper and given tracking codes.

Non-Flash Content

Although Flash plugins are nearly ubiquitous, there will undoubtedly be platform combinations that are inhospitable towards Flash creatives. Standard gifs of the same size and message are a mandatory deliverable as alternate content for browsers with incompatible plugins, or don't support javascript and/or iframes.

Embedded Movie Clips

If a portion of the Flash creative contains an animation that is compiled into a separate movie clip, and this animation contains a clickable area, it will follow a slightly different syntax in the getURL method. In order for the animation to reference the click-through, it must trace the hierarchy up to the parent timeline where the variable values are loaded. This is achieved by appending "_root." to the variable "clickTag", as in: getURL(_root.clickTag1).

Domain Issues

With the distribution of the Flash 6/MX plug-in, Macromedia instituted a security policy known as the "Sandbox Security Model". The intention is to prevent unwanted or malicious code from attaching itself to a flash file. The requirement stipulates that any subsequent movie loads must originate from the same domain as the parent movie.



Compiling all flash elements into a single creative is the simplest method for compliance with this policy. However, if a creative is sufficiently complex that a loadMovie() method is necessary to meet publisher file size requirements, there are steps to circumvent this policy.

In order to serve movies with subsequent file loads through Atlas, all Flash .swf files reside on the Atlas Direct Server. References to these movies within the creative should use the dependent filenames only. Atlas supplies the "base" parameter in the html template, which serves to construct the url location for any file locations that aren't explicitly defined. This satisfies the flash security restrictions.

Flash MX2004

Flash MX2004 (aka. Flash 7) introduces two new stipulations with respect to third-party adserving:

- 1) Flash is now case-sensitive. "clickTag1" and "clickTAG1" will be treated as two distinct variables. The standard Atlas syntax uses the spelling "clickTag", therefore Flash variable construction should follow suit.
- 2) Macromedia has expanded the definition of the Security Sandbox Model to exclude sub-domains off of the parent domain. For example, files located on image.atdmt.com will trigger a sandbox violation if the parent file is located on img.atdmt.com. Sub-domains were allowed in the Flash 6 security model. Since Atlas stores all .swf files on the same domain, this has not been an issue for adserving. However, if a creative has special requirements, it may be possible to explicitly enable other domains through the use of the security.sandbox.allowDomain() function.

Use of the on(press) Event

Windows XP Service Pack 2 now includes a default pop-up blocker that will disable attempts to open destination browsers when on(press) is used in conjunction with the getURL() command. In order to bypass the pop-up blocker and provide a proper clickthrough to a destination URL, use the on(release) event instead. Use of on(press) for in-banner events such as navigation is not affected.