YuMe HTML5 Plug-in (1.2.7.5)

YuMe, Inc.

YuMe, Inc.
1204 Middlefield Road
Redwood City, CA 94063
650.591.9400
www.yume.com

CONTENTS

Contents	3
Preface	4
Technical Skills Required	5
Process Overview	6
Integration Instructions	7

PREFACE

Welcome to YuMe, and thank you for deciding to become a YuMe Publisher. To start working with YuMe, you will need to equip your video player with YuMe's ACE technology. Publishers with ACE-equipped players can apply to become YuMe Publishers and monetize their content through YuMe's video ad network (following a YuMe content standards review), and can choose to take control of their ad operations with ACE for Publishers.

This document provides specific instructions for web publishers to integrate YuMe's HTML5 Plug-in with publishers' HTML5 video player, and provides a high level overview of the steps you will need to take to become a YuMe Publisher.

We want to make this process as fast and easy as possible, so please email us at yume_support@yume.com if you run into any integration issues, or if you have any questions about the process of becoming a YuMe Publisher.

TECHNICAL SKILLS REQUIRED

Integration requires an understanding of HTML5, CSS and JavaScript. You should be familiar with the fundamentals of XML and have a basic knowledge of online media formats including image and video. You can use your preferred text editing program to access most source files.

You will need to have access to a web server in order to test your player. During the QA process, we will ask you to host your YuMe integrated player on a semi-public web page so that we can validate it.

PROCESS OVERVIEW

Step 1: Complete the integration – Publisher's responsibility

Detailed integration instructions are included later in this document—most publishers complete the process in 2-3 days.

Step 2: Integration QA – Joint effort, YuMe & Publisher

In order for YuMe to certify your player, we need to run a few tests on it. This includes making sure that each of our ad units work in your player. If you have a public web server, we recommend that you place a copy of your player on this server in a hidden folder where we can access it. Then email us at yume_support@yume.com and be sure to include the URL to your test page.

Step 3: Content standards review – YuMe's responsibility

During the testing process, we will also review the video content on your site to ensure that it does not violate our network standards. We primarily work with major brand marketers who are used to buying ads on TV, which means that we will not be able to help you monetize your content if it is user-generated, pornographic, or contains foul language or vulgarity. We can also not help to monetize below-the-fold, auto-play content.

Step 4: Start monetizing your content

As a YuMe Publisher, you will be able to receive pre-rolls and other high-impact ad units from YuMe.

Once your integration has passed QA and we have reviewed your content and verified that it does not violate our standards, we can give you a login for our web-based campaign reporting console and start sending you ads. We will also work with you to determine which content and audience channels your videos should be assigned to, and to determine how much inventory you will want YuMe to monetize on a daily basis. Once your content has been assigned to YuMe channels, we will be able to start sending you ads from the YuMe Network.

INTEGRATION INSTRUCTIONS

Introduction:

These integration instructions address the following at a high level:

- Updating your HTML5 player to use the HTML5 Plug-in to request and play ads
- Updating your banner slots to run a companion banner (if applicable)

NOTE: Please observe that some YuMe HTML5 Plug-in function names appear to refer to a YuMe SDK. There is no separate YuMe HTML5 SDK—those references are to the YuMe HTML5 Plug-in. We apologize for the confusion.

Required files:

You will need to host the following files (included in the HTML5 Plug-in download) on your web server:

- empty.html
- yume_html5_sdk.js

Known issues:

The Plug-in might not support video rendering on Firefox browser. We suggest using a Flash video player when a user visits your page with the Firefox browser.

HTML UI elements required from Web Publisher:

Your page will need to include an HTML5 video player enclosed in a tag with the properties described below---you can change the size and positioning of the container (in our case “video_box”) to change your needs. In order to display HTML5 video ads from YuMe, you will need to enclose your <video> element in an anchor <a> tag that allows YuMe to properly handle click events on video ads being rendered in the player.

Sample code for the HTML5 video player, companion, and tracking elements:

```
<div style="width: 480px; height: 350px; float:left; position:absolute"
id="video box">
<a href="javascript:void(0)" id="hRefClickTag" style="cursor: default">
<video id="html5 video" poster="images/yume logo.gif" width=480 height=320>
Your Browser doesn't support HTML5 !!!</video></a></div>
```

You will also need to add the following element somewhere on the page to allow YuMe to properly handle tracking clicks:

```
<div id="trackerDiv" style="display:none"></div>
```

You have the option to add one or more 300x250 companion banner elements to your page by inserting the following element adjacent to your video player:

```
<div id="cb_medrect1_div" style="border:0; margin-left:490px; width: 300px; height: 250px;
position:absolute; overflow:hidden"> <iframe name="cb_medrect1_frame"
id="cb_medrect1_frame" src="empty.html" frameborder="0" scrolling="no" marginheight="0"
marginwidth="0" topmargin="0" leftmargin="0" allowtransparency="true" width="300"
height="250"></iframe></div>
```

NOTE: Your companion banner element will need to contain the path to the version of empty.html hosted on your servers.

HTML5 Plug-in Inclusion:

To include the HTML5 Plug-in in your page, add the following code to the <head> element:

```
<script type="text/javascript" src="js/yume_html5_sdk/yume_html5_sdk.js"></script>
```

NOTE: Your code will need to contain the path to the version of yume_html5_sdk.js hosted on your servers. yume_html5_sdk.js represents YuMe HTML5 Plug-in module. It also includes 2 utility modules: one for adding / removing event listeners and another for detecting the user's browser user agent at player at runtime.

HTML5 Plug-in Initialization:

Before utilizing the YuMe HTML5 Plug-in for ad fetching, you must instantiate & initialize the Plug-in by performing the following steps:

1. Create an instance of YuMeHTML5SDK.
2. Create an instance of YuMeHTML5SDKInitObject and set the required parameters.
3. Call YuMe HTML5 Plug-in's yume_init() function with the YuMeHTML5SDKInitObject populated with the required parameters. yume_init() will return false in case of any errors and also send an 'AD_ERROR' with appropriate error information.

NOTE: Initialization needs to be performed only once. After that any number of Ad requests can be made to the YuMe Plug-in before de-initialization.

NOTE: The ID attributes used when initializing the HTML5 SDK must exactly match with the ID attributes of the relevant and <iframe> elements included on the page.

Sample code for creating a new Plug-in instance and setting the required parameters:

```
var yumeSDKInstance = new YuMeHTML5SDK();

var yumeInitObj = new YuMeHTML5SDKInitObject();

yumeInitObj.adDomainUrl = "__YUME_EX_AD_SERVER_SITE_DOC_ROOT";
// NOTE: For testing this can be set to "http://shadow01.yumenetworks.com/"

yumeInitObj.domainId = "__YUME_EX_DOMAIN";
// NOTE: For testing this should be set to "211DaVuJgGj"

yumeInitObj.qsParams = "";
//NOTE: When you go live, YuMe will let you know what additional parameters to pass

yumeInitObj.html5VideoId = "html5_video";
//this value needs to be the same as the HTML5 video element's id on the page

yumeInitObj.html5VideoDivId = "html5_video_div"; //this value needs to be the same as the HTML5 video div element's id on the page

yumeInitObj.clickTagHRefId = "hRefClickTag";
//this value needs to be the same as the anchor tag's id around the video element
```

NOTE: "__YUME_EX_AD_SERVER_SITE_DOC_ROOT" and "__YUME_EX_DOMAIN" should be replaced with appropriate values of YuMe Ad Server Url and web publisher domain respectively.

Sample code for setting up a companion banner:

```
yumeInitObj.cbMedRectDivId = "cb_medrect1_div";
//this value needs to be the same as the MedRect div's id

yumeInitObj.cbMedRectIFrameId = "cb_medrect1_frame";
//this value needs to be the same as the MedRect frame's id

yumeInitObj.cbFullBannerDivId = "cb_fullban1_div";
//this value needs to be the same as the FullBanner div's id

yumeInitObj.cbFullBannerIFrameId = "cb_fullban1_frame";
//this value needs to be the same as the FullBanner frame's id

yumeInitObj.cbLeaderBoardDivId = "cb_leaderboard1_div";
//this value needs to be the same as the LeaderBoard div's id
```

```
yumeInitObj.cbLeaderBoardIFrameId = "cb_leaderboard1_frame";
//this value needs to be the same as the LeaderBoard frame's id

yumeInitObj.cbWideskyDivId = "cb_wideskyscraper1_div";
//this value needs to be the same as the Widesky div's id

yumeInitObj.cbWideskyIFrameId = "cb_wideskyscraper1_frame";
//this value needs to be the same as the Widesky frame's id

yumeInitObj.cb31RectDivId = "cb_31rect_div";
//this value needs to be the same as the 31Rect div's id

yumeInitObj.cb31RectIFrameId = "cb_31rect_frame";
//this value needs to be the same as the 31Rect frame's id
```

Sample code for creating ad playlists:

```
yumeInitObj.prerollPlaylist = "dynamic_preroll_playlist.json?imu=medrect";
yumeInitObj.midrollPlaylist = "dynamic_midroll_playlist.json?imu=medrect ";
yumeInitObj.postrollPlaylist = "dynamic_postroll_playlist.json?imu=medrect ";
```

Sample code for setting up a backfill banner:

```
yumeInitObj.iframeBannerPlaylist = "dynamic_banner_iframe.html";
```

Sample code for prefetch support disabling:

By default prefetching is enabled in YuMe HTML5 SDK

```
yumeInitObj.bPrefetchMidrollAd = false;
//this value needs to be set to false, if prefetching midroll ads needs to be disabled in YuMe HTML5 SDK

yumeInitObj.bPrefetchPostrollAd = false;
//this value needs to be set to false, if prefetching postroll ads needs to be disabled in YuMe HTML5 SDK
```

Sample code for setting Android's video format:

Set the video format for Android platform to be 3gp - by default, mp4 format will be set in YuMe HTML5 SDK

```
yumeInitObj.bAndroidVideoFormat = YuMeHTML5SDK.prototype.yume_videoFormat.FORMAT_3GP;
//this value needs to be set, only if 3gp format is required for Android
```

Sample code for Max Video Bitrate:

By default, a value of 384 will be set in YuMe HTML5 SDK

```
yumeInitObj.maxVideoBitRate = 384;
```

Sample code for video timeout poll interval:

By default, a value of 500ms will be set in YuMe HTML5 SDK

```
yumeInitObj.videoTimeoutInterval = 4;  //seconds
yumeInitObj.videoTimeoutPollInterval = 500;  //milliseconds
```

```
var bYumeSDKInit = yumeSDKInstance.yume_init(yumeInitObj);
        if(!bYumeSDKInitz{
        alert("yume_init() failed !!!");
        return;
}
delete yumeInitObj;
```

Sample code for using the EventAdderRemover object and BrowserDetection utility:

```
var eventAdderRemover = yumeSDKInstance.yume_getEventAdderRemoverObject();
var browserDetector = yumeSDKInstance.yume_getBrowserDetectorObject();
```

NOTE: The HTML5 Plug-in creates its own instances of the EventAdderRemover object and BrowserDetection utility during initialization. Publishers can use those objects in their HTML page by the HTML5 Plug-in functions above.

HTML5 Plug-in Ad Functions:

Whenever the publisher needs to display an ad from YuMe, the HTML5 Plug-in function yume_startAd() needs to be called with the ad slot information.

Sample code for making an ad call:

```
yumeSDKInstance.yume_startAd(YuMeHTML5SDK.prototype.yume_adBlockType.PREROLL_BLOCK);
//for fetching Preroll Ads
yumeSDKInstance.yume_startAd(YuMeHTML5SDK.prototype.yume_adBlockType.MIDROLL_BLOCK);
```

```
//for fetching Midroll Ads
yumeSDKInstance.yume_startAd(YuMeHTML5SDK.prototype.yume_adBlockType.POSTROLL_BLOCK);
//for fetching Postroll Ads
```

HTML5 Plug-in Events:

When the HTML5 Plug-in receives the yume_startAd() request, it fetches a playlist from the YuMe ad server and plays the ad immediately in the video element that was passed in during Plug-in initialization. The HTML5 Plug-in also sends events to the publisher's video player to indicate the status of the ad serving and rendering process. In order to listen for various ad events from the HTML5 Plug-in, the publisher needs to implement a function called yumeSDKAdEventListener(). The HTML5 Plug-in automatically fires 0%, 25%, 50%, and 100% impression beacons—publishers do not need to write any additional code to handle impression beaconing.

When YuMe returns an ad in response to an ad request, the player will send an AD_PRESENT event when an ad is present and about to be rendered, followed by an AD_PLAYING event when the ad is being rendered, and then an AD_COMPLETED event when the ad has finished rendering. Appropriate companion banners received in the playlist will also be displayed if the companion(s) were configured during setup. If companion banners are not present and the publisher has configured a backfill, then one backfill banner request will be sent by the Plug-in for each of the companion banners requested. If an appropriate backfill banner is received, it will be displayed.

When YuMe does not serve an ad in response to an ad request, the HTML5 Plug-in will send an AD_ABSENT event followed by an AD_COMPLETED event.

When there is an error, the HTML5 Plug-in will send an AD_ERROR event followed by an AD_COMPLETED event.

Complete list of HTML5 Plug-in events:

Ad Event	Description
AD_PRESENT	Sent when ad is present and about to be rendered
AD_ABSENT	Sent when no ad is served in response to an ad request
AD_PLAYING	Sent when an ad is being rendered
AD_COMPLETED	Sent when ad playback is completed *Suggested action: change the pause button status to play status when this event fires for a post-roll ad to allow users to replay the content.*
AD_ERROR	Sent when any of the mandatory parameters are sent as null or empty during

	SDK initialization, or when the yume_start_Ad is called before the yume_init function has been called
AD_CLICKED2SITE	When a video or image ad over the video is clicked for launching a site (in IPad). [Suggested action: Bring the play / pause button to "pause" state and disable the scrubber bar controls]
AD_CLICKED2VIDEO	When an image ad over the video, whose clicktag points to another video ad is clicked for fetching a new ad. [Suggested action: Bring the play / pause button to "play" state and reset the control bar]
PIP_VIDEO_PLAYING	When a video ad that was fetched by clicking on an image ad is being rendered. [Suggested action: Bring the play / pause button to "pause" state and update the control bar to track the play progress]
AD_CB_FLASH	This placeholder event is currently not in use
AD_CB_IFRAME	Sent when an iframe companion banner is available

Sample code for handing Ad events from HTML5 plug-in:

```
function yumeSDKAdEventListener(yume_event, yume_eventInfo) {
        switch(yume_event) {
                case YuMeHTML5SDK.prototype.yume_adEvent.AD_PRESENT:
                        alert("Received AD_PRESENT event from YuMe HTML5 SDK...");
                        //do something
                        break;
                case YuMeHTML5SDK.prototype.yume_adEvent.AD_ABSENT:
                        alert("Received AD_ABSENT event from YuMe HTML5 SDK...");
                        //do something
                        break;
                case YuMeHTML5SDK.prototype.yume_adEvent.AD_PLAYING:
                        alert("Received AD_PLAYING event from YuMe HTML5 SDK...");
                        //do something
                        break;
                case YuMeHTML5SDK.prototype.yume_adEvent.AD_COMPLETED:
                        alert("Received AD_COMPLETED event from YuMe HTML5 SDK...");
                        //do something
                        break;
                case YuMeHTML5SDK.prototype.yume_adEvent.AD_ERROR:
                        alert("Received AD_ERROR event from0 YuMe HTML5 SDK, Error Info:
                        " + yume_eventInfo);
                        //do something
                        break;
                case YuMeHTML5SDK.prototype.yume_adEvent.AD_CB_IFRAME:
                        alert("Received AD_CB_IFRAME event from YuMe HTML5 SDK..." +
```

```
                        yume_eventInfo);
                        //do something
                        break;
                default:
                        break;
        }
}
```

HTML5 Plug-in De-Initialization

It is necessary that HTML5 Plug-in be de-initialized once its usage is not necessary. This can be achieved by calling the Plug-in function yume_deInit() and is typically called during "onunload".

Sample code for plug-in de-initialization:

```
if(yumeSDKInstance) {
        yumeSDKInstance.yume_deInit();
        delete yumeSDKInstance;
}
```

Submitting your sample player to YuMe for QA

When you have completed your test integration, please send an email yume_support@yume.com that includes your name, contact information, and a link to your test page. If you run into any problems during your integration, please do not hesitate to email us.