Advanced Boomerang Ad Wrapper Site Integration

The SHE Media partner dashboard provides tags that serve as the basic integration with the Boomerang ad wrapper. For those sites that need advanced integration, this guide walks through advanced methods available in Boomerang. The advanced integration will allow you to customize the ads displayed on the site beyond the basic ad tags as well as provide extra control over various parts of the ad calls.

Header Integration

The dashboard provides the basic snippet of code that starts up Boomerang. This code is designed to be place In the `<head>` portion of the HTML template for your site. This snippet includes the object and queue setup, then calls our primary Boomerang library, `blogherads.js`, and the `header.js` file specific to your site.

If you need to perform extra setup for your page(s), or you wish to fully control the order in which ads are called when the final request to DFP is made, you can add an extra bit of code to the end of the first script block in that snippet where the blogherads object and adq are set up. An example of doing this is included here.

```
<head>
<script type="text/javascript">
  var MySlots = {};
  var blogherads = blogherads || {};
  blogherads.adq = blogherads.adq || [];
  // See notes below about the Push Function
  blogherads.adq.push(function() {
    // For responsive slots, this push function method must be used
    MySlots.sidebar1 = blogherads.defineResponsiveSlot([
      // mapping
      // First param is width, second is height. Text field is ad type
      [[900, 0], 'banner'], // This means anything 900 pixels wide and
above
      [[350, 0], 'tinybanner'], // This defines anything 350 pixels wide
to 899 pixels wide
      [[0, 0], 'none'], // This defines no ad to show at all. Assuming
we don't want to show any ads from 0 pixels wide to 349 pixels wide
    ], 'CONTAINER-ID');
    MySlots.inline1 = blogherads.defineSlot('medrec', 'CONTAINER2-ID');
    // ...More blogherads.defineSlot() calls

 // If the site needs to override the vertical/ad unit path, this call
allows the site to do that.
    blogherads.setConf('vertical', 'fashion');
  });
</script>

<script type="text/javascript" async="async" data-cfasync="false"
src="https://ads.blogherads.com/static/blogherads.js"></script>
<!-- The numbers 19/198 are specific to the blog/site. The specific
header.js URL for your site will be provided by SheKnows. -->
<script type="text/javascript" async="async" data-cfasync="false"
src="https://ads.blogherads.com/19/198/header.js"></script>
</head>
```

Advanced functionality demonstrated above

`blogherads.adq.push();`

This function is the wrapper to every action we perform in Boomerang. If you have worked with the googletag GPT library before, it is synonymous with the `googletag.cmd.push();` wrapper function. This function takes 3 forms:

1. `blogherads.adq.push(function() { /* boomerang calls go here */ });`
    a. This is the form of this method that most closely mimics calling `googletag.cmd.push();`. A Javascript closure function is passed in, and this function is executed once Boomerang has started up. Since Boomerang is an asynchronous library, this ensures you don't make calls to Boomerang functions before Boomerang is ready. For advanced integrations, you will probably make use of this. This is used in the example above to enclose calls to define custom slots.
    b. **NOTE**: You can include as many calls as you need to inside the closure. You do not need a closure for each Boomerang call you make:

```
blogherads.adq.push(function() {
  blogherads.setConf('vertical', 'home');
  blogherads.defineSlot('medrec', 'skm-ad-medrec').display();
  var MyTopSlot = blogherads.defineSlot('flexbanner',
'skm-ad-flexbanner');
  MyTopSlot.display();
});
```

2. `blogherads.adq.push(['SlotType', 'SlotDivID']);`
    a. This is the most common format you're likely to see and is the format used in most of the tags you will see in the dashboard. This form is used to call a single ad using all of the default settings for that ad. Instead of passing a Closure, in this form, you pass a Javascript array with 2 elements. The first element is the type of ad (see the list of ad types later in this article). The second element is the ID of the container div that will hold this ad. The div element needs to be created by you somewhere in the page for this call to work.
    b. Example: Create a div like this in the page: `<div id="my-ad-slot-1"></div>`. Then, add the following to populate this add with a 300x250 ad: `<script>blogherads.adq.push(['medrec', 'my-ad-slot-1']);</script>`. Note that `my-ad-slot-1` is used as the div ID and is what's passed as the second item in the array passed to `blogherads.adq.push();`.
3. `blogherads.adq.push('SlotType');`
    a. This is the final format of this function. In this format, you **only** pass a string to `blogherads.adq.push();`. This tells Boomerang that you would like to call an ad with the `SlotType` passed to the function. This form makes the assumption that you have created an div with the default div ID of `skm-ad-SlotType`. For instance, if `SlotType` is `medrec`, then Boomerang would expect this div with the ID `skm-ad-medrec` to exist on the page: `<div id="skm-ad-medrec"></div>`, and it would place the ad in that div.

Available Slot Types

- `banner` - 728x90
- `flexbanner` - Multisize: 728x90, 970x90, 970x250
- `medrec` - 300x250
- `flexrec` - Multisize: 300x250, 300x600, 300x1050
- `sky` - 160x600
- `tinybanner` - Multisize: 300x50, 320x50 (Mobile banner)

Defining slots using advanced methods

If you need more control than simply passing an array to `blogherads.adq.push();` as explained above, then you can use the Closure form of `blogherads.adq.push();` and, inside the closure, define ads using the `blogherads.defineSlot();` and `blogherads.defineResponsiveSlot();` calls, as seen in the example at the top. These methods give you greater control over the instantiation of each ad by returning a reference to the Boomerang slot object so you can control when the ad is displayed, apply per-slot targeting, etc.

`blogherads.defineSlot('SlotType, 'divID');`

This method is used to define a slot and return a reference to it. It takes 2 parameters that map to the same to elements in the array passed to that form of the `blogherads.adq.push();` call: `SlotType` and `divID`. This form returns a reference to the slot object, which you can use to set targeting for the slot and display the slot:

Examples:

- Define and display an ad:

```
blogherads.adq.push(function() {
  blogherads.defineSlot('flexrec', 'skm-ad-flexrec').display();
});
```

- Define an ad to be displayed later:

```
<!-- In the <head>: -->
<script>
  var MyLeaderboard;
  blogherads.adq.push(function() {
    MyLeaderboard = blogherads.defineSlot('flexbanner',
'my-leaderboard-slot');
  });
</script>


<!-- Later in the page: -->
<div id="my-leaderboard-slot"></div>
<script>
  blogherads.adq.push(function() {
    MyLeaderboard.display();
  });
</script>
```

- Define an ad, set some custom targeting for that one ad slot, then display the ad:

```
<div id="skm-ad-flexrec"></div>
<script>
  blogherads.adq.push(function() {
    blogherads.defineSlot('flexrec',
'skm-ad-flexrec').setTargeting('targetingKey',
'targetingValue').display();
    // Note, you can also pass an array to setTargeting like so:
    blogherads.defineSlot('flexrec',
'skm-ad-flexrec').setTargeting('targetingKey', ['targetingValue1',
'targetingValue2']).display();
  });
</script>
```

`blogherads.defineResponsiveSlot();`

The Boomerang ad wrapper also supports definition of responsive slots. These slots observe the size of the browser viewport to choose the appropriate size of ad based on the size mapping that is passed into the call. In addition, Boomerang watches for changes to the viewport dimensions and responds accordingly by reloading the ad at the new size of a size boundary is crossed. An example of a responsive slot is given at the top of this article. What follows is more detail into how the responsive slot call is structured. Like `blogherads.defineSlot();`, `bloghe` `rads.defineResponsiveSlot();` returns a slot object which you can `setTargeting()` on and `display()`. The difference between the two slot defintions is that, whil defineSlot() takes a slot type as its first parameter, defineResponsiveSlot() takes a slot **mapping** as its first parameter. This mapping is a multi-level array that defines viewport sizes and the slot types that will be utilized at the corresponding viewport sizes.

As mentioned, the mapping is an array of arrays. Each array item contains the following items: An array of minimum height and width, and a string containing the slot type associated with this size. Example:

```
// Use flexbanner (970x250) if the viewport is at least 1050px wide:
[[1050, 0], 'flexbanner']
```

Each of the arrays from above are combined into a larger array to form the **mapping**. Boomerang processes the mapping **in order**, and utilizes the first match, so you must place your largest breakpoints first. For instance, if the viewport is 1050px wide, and you have a mapping of 900px followed by 1050px, the 900px will match, because the viewport is at least 900px wide, so that line will be used. In this case, the 1050px line will never be used, because the 900px line will always match.

In addition to the slot types above, there is a special slot type that can be utilized when the desire it to **NOT** display any ad at all: `none`. For instance, you might set a smallest breakpoint like so: `[[0, 0], 'none']`. This means that from 0px by 0px up to your next breakpoint, no ad will be displayed, presumably because the viewport is too small to display an ad.

Here's an example of defining a responsive leaderboard. In this example, if the viewport is at least 1050px wide, we show the 970x250 ad (flexbanner), when the viewport is at least 800px wide, we show the 728x90 ad (banner), when the viewport is at least 320px wide, we show the mobile banner ad (tinybanner), and, below that, we show no ad at all:

```
<div id="responsive-leaderboard-ad"></div>
<script>
blogherads.adq.push(function() {
  blogherads.defineResponsiveSlot([
    [[1050, 0], 'flexbanner'],
    [[800, 0], 'banner'],
    [[320, 0], 'tinybanner'],
    [[0, 0], 'none']
  ], 'responsive-leaderboard-ad').display();


  // Note, you can also chain a setTargeting() call
  // before .display() here just like with a standard
  // Boomerang slot.
});
</script>
```

Reloading/Refreshing Ad Slots - `blogherads.reloadAds();`

Boomerang provides the `blogherads.reloadAds();` method that can be called to refresh any Boomerang slot on the page. There are two ways to call `reloadAds()`:

1. `blogherads.reloadAds();`
2. `blogherads.reloadAds([slot1, slot2, slot3]);`

When **no** arguments are passed to `blogherads.reloadAds();`, Boomerang will refresh **all** Boomerang slots that are **currently in view** (a slot is in view when at least 50% of the slot's pixels are visible in the browser's viewport). As an example of this, imagine a page with four ads: A 970x250 at the top, a 300x600 at the top of the sidebar, a 300x250 at the bottom of the sidebar, and a 728x90 in the footer of the page. If the user is scrolled to the top of the page, then the entire 970x250 slot would be in view, and at least a portion of the 300x600 would be in view. If the full width of the 300x600 ad is in the viewport, and at least 300 of the 600 rows of pixels of the slot are within the viewport, then, when a `blogherads.reloadAds();` call is issued on the page, both the 970x250 and 300x600 slots will be refreshed. Since the 300x250 and 728x90 slots are both outside of the viewport, they will **not** be refreshed when this call is made.

When an array of Boomerang slots is passed to `blogherads.reloadAds();`, then Boomerang will refresh **only** the slots that were included in the array, and those slots will be refreshed **regardless** of the current in-view status of those slots. So, taking the previous example, if the same slot configuration existed, and the call was made to refresh those slots using this method (ie `blogherads.reloadAds([top970Slot, top300600Slot, bottom300250Slot, bottom728Slot]);`), then all four slots would be refreshed regardless of the scroll position in the browser and regardless of which slots were currently visible in the viewport. **NOTE**: In order to use this method, you **MUST** retain a reference to the Boomerang slot objects that you wish to forcefully refresh. This reference is the object that is returned by making calls to `blogherads.defineSlot();` and `blogherads.defineResponsiveSlot();`.

Examples:

```
// Hold references to our slots.
var Slots = {};

// Define the slots
blogherads.adq.push(function() {
  Slots.bannerTop = blogherads.defineSlot('flexbanner',
'skm-ad-flexbanner');
  Slots.sidebarTop = blogherads.defineSlot('flexrec', 'skm-ad-flexrec');
  Slots.sidebarBottom = blogherads.defineSlot('medrec',
'skm-ad-medrec');
  Slots.bannerBottom = blogherads.defineSlot('banner', 'skm-ad-banner');
});

// Display the slots
blogherads.adq.push(function() {
  Slots.bannerTop.display();
  Slots.sidebarTop.display();
  Slots.sidebarBottom.display();
  Slots.bannerBottom.display();
});

// Reload all slots currently visible in the viewport
blogherads.adq.push(function() {
  blogherads.reloadAds();
});

// Reload the top banner and top sidebar ads regardless of viewable
status
blogherads.adq.push(function() {
  blogherads.reloadAds([Slots.bannerTop, Slots.sidebarTop]);
});

// Reload all slots defined regardless of viewable status
blogherads.adq.push(function() {
  blogherads.reloadAds([Slots.bannerTop, Slots.sidebarTop,
Slots.sidebarBottom, Slots.bannerBottom]);
});
```

Other utility functions

- Set global targeting on a page (this is similar to `setTargeting()` on a slot):

```
blogherads.adq.push(function() {
  // Single value targeting
  blogherads.setTargeting('key', 'value');
  // Multi-value targeting:
  blogherads.setTargeting('key', ['value1', 'value2', 'value3']);
});
```

- Set a given configuration setting using `blogherads.setConf()`:

```
blogherads.adq.push(function() {
  // Currently, we only support setting the vertical via setConf:
  blogherads.setConf('vertical', 'value');
  // Multi-level vertical (note: only 1 or 2 levels are supported):
  blogherads.setConf('vertical', 'value1/value2');
});
```

Tracking Synthetic Pageviews

In order to accurately calculate RPM statistics, Boomerang must be aware of all page navigation events. In traditional web applications, this is already handled. Boomerang automatically tracks a pageview event when it loads. On more custom experiences, this can pose a challenge, since a user navigation event may have occurred, but the page may not have been fully reloaded for the new view. Without an explicit signal, Boomerang won't record a new pageview event in these circumstances. Examples where this might happen include SPAs (Single Page Applications), infinite scroll pages, and gallery/slideshow pages. In order for Boomerang to handle this scenario appropriately, the developer must signal Boomerang to record a synthetic pageview. This is similar to forcing Google Analytics to track a new pageview after the page has loaded. For Boomerang, tracking this synthetic pageview can be accomplished by using the `trackPageView` method:

```
blogherads.adq.push(function() {
  blogherads.trackPageView();
});
```

Receiving Events

Boomerang offers the ability for the page to respond to certain events by utilizing the following call:

```
blogherads.addEventListener(EVENT_TYPE, EVENT_CALLBACK);
```

Parameters:

- `EVENT_TYPE` (`String`) - The type of event that will trigger this listener.
- `EVENT_CALLBACK` (`Function`) - A function to handle the event. The callback is passed a single object containing any extra information pertaining to the event. In addition, the callback may be fired with the object context (ie: `this`) set to the context of the event that triggered the listener. This is especially true for events that are specific to a given Slot. In that case, `this` will be the Boomerang slot for which the event fired.

Available Event Types:

- `gptSlotRenderEnded` - This event is fired in response to the `slotRenderEnded` event fired by the GPT library. For this event, the `this` object is set to the Boomerang slot that was rendered. The `slot` property of the event object also references the Boomerang slot that was rendered. More information on the rest of the event properties that are passed with this event can be found at https://developers.google.com/doubleclick-gpt/reference#googletageventsslotrenderendedevent.
- `gptSlotOnload` - This event is fired in response to the `slotOnload` event fired by the GPT library. For this event, the `this` object is set to the Boomerang slot that completed loading. The `slot` property of the event object also references the Boomerang slot that completed loading. More information on the rest of the event properties that are passed with this event can be found at https://developers.google.com/doubleclick-gpt/reference#googletageventsslotonloadevent.
- `gptImpressionViewable` - This event is fired in response to the `impressionViewable` event fired by the GPT library. For this event, the `this` object is set to the Boomerang slot that became viewable. The `slot` property of the event object also references the Boomerang slot that became viewable. More information on the rest of the event properties that are passed with this event can be found at https://developers.google.com/doubleclick-gpt/reference#googletageventsimpressionviewableevent.
- `gptSlotVisibilityChanged` - This event is fired in response to the `slotVisibilityChanged` event fired by the GPT library. For this event, the `this` object is set to the Boomerang slot that changed visibility status. The `slot` property of the event object also references the Boomerang slot that changed visibility status. More information on the rest of the event properties that are passed with this event can be found at https://developers.google.com/doubleclick-gpt/reference#googletageventsslotvisibilitychangedevent.

Examples:

```
blogherads.adq.push(function() {
  // Log when a slot's visible area changes.
  blogherads.addEventListener('gptSlotVisibilityChanged',
function(event) {
    // Note: For this event, "this" is the same as event.slot, so
"this.logDesc()" is the same as "event.slot.logDesc()".
    console.log('Slot ' + this.logDesc() + ' is now ' +
event.inViewPercentage + '% in view.');
  });


  // Log when a slot completes its render.
  blogherads.addEventListener('gptSlotRenderEnded', function(event) {
    // Note: For this event, "this" is the same as event.slot, so
this.logDesc() is the same as "event.slot.logDesc()".
    console.log('Slot ' + this.logDesc() + ' was rendered' +
(event.isEmpty ? ' EMPTY' : ''));
  });
});
```

SHE Media AMP Ad Tags Integration:

Your site must be an active member of the SHE Media Partner Network. Please contact Support for specific tags for your site.

Available Slot Types for AMP:

- `banner` - 728x90
- `flexbanner` - Multisize: 728x90, 970x90, 970x250
- `medrec` - 300x250
- `flexrec` - Multisize: 300x250, 300x600, 300x1050
- `sky` - 160x600
- `tinybanner` - Multisize: 300x50, 320x50 (Mobile banner)

Configuration Examples:

Required parameters:

- `data-slot-type` - SHE Media slot type.
- `data-boomerang-path` - Boomerang path.

Slot Type: **medrec - 300x250**

```
<amp-ad width="300" height="250"
      type="shemedia"
      data-slot-type="medrec"
      data-boomerang-path="/amp-example/26403"
  </amp-ad>
```

Slot Type: **`banner` - 728x90**

```
<amp-ad width="728" height="90"
      type="shemedia"
      data-slot-type="banner"
      data-boomerang-path="/amp-example/26403"
  </amp-ad>
```

Optional parameters

- `json` - Boomerang configuration key values can be passed using the **`boomerangConfig`** property. Custom targeting key values can be passed to Boomerang using the **`targeting`** property.

Boomerang config examples:

```
<amp-ad width="300" height="250"
      type="shemedia"
      data-slot-type="medrec"
      data-boomerang-path="/amp-example/26403"
      json='{"boomerangConfig": {"vertical": "parenting"}}'>
  </amp-ad>
```

```
<amp-ad width="300" height="250"
      type="shemedia"
      data-slot-type="medrec"
      data-boomerang-path="/amp-example/26403"
      json='{"boomerangConfig": {"vertical": "parenting"}}'>
  </amp-ad>
```

Custom targeting examples:

```
<amp-ad width="300" height="250"
      type="shemedia"
      data-slot-type="medrec"
      data-boomerang-path="/amp-example/26403"
      json='{"targeting":{"ct":["articles"]}}'>
  </amp-ad>
```

```
<amp-ad width="300" height="250"
      type="shemedia"
      data-slot-type="medrec"
      data-boomerang-path="/amp-example/26403"
      json='{"targeting":{"st":["mrec"]}}'>
  </amp-ad>
```