Running A9 in Parallel with OpenWrap

Introduction

The A9 Server-to-Server (S2S) wrapper solution and client-side code is not a component of any other wrapper (e.g., Prebid). This requires running the code in parallel with OpenWrap. OpenWrap manages the entire process of detecting slots and attaching bids to GAM and does not expose any functionality to request a bid that can be consumed. Therefore, the code cannot run in parallel using callbacks and custom handling.

Solution

Handling Pre-Timeout Call to GAM in Case of External Partners

When we introduce partners outside of OpenWrap, any notification and synchronization between OpenWrap and external bidders are not available. This means the only point of synchronization is the OpenWrap timeout event when a call to GAM is fired.

To handle this situation, there should be tight coupling between external bidders for notification for each slot when a bid is available. Unfortunately, building this tight integration is complicated and require many events definitions on A9/Publisher end as well as ingestion of APIs in OpenWrap.

OpenWrap API Interfaces

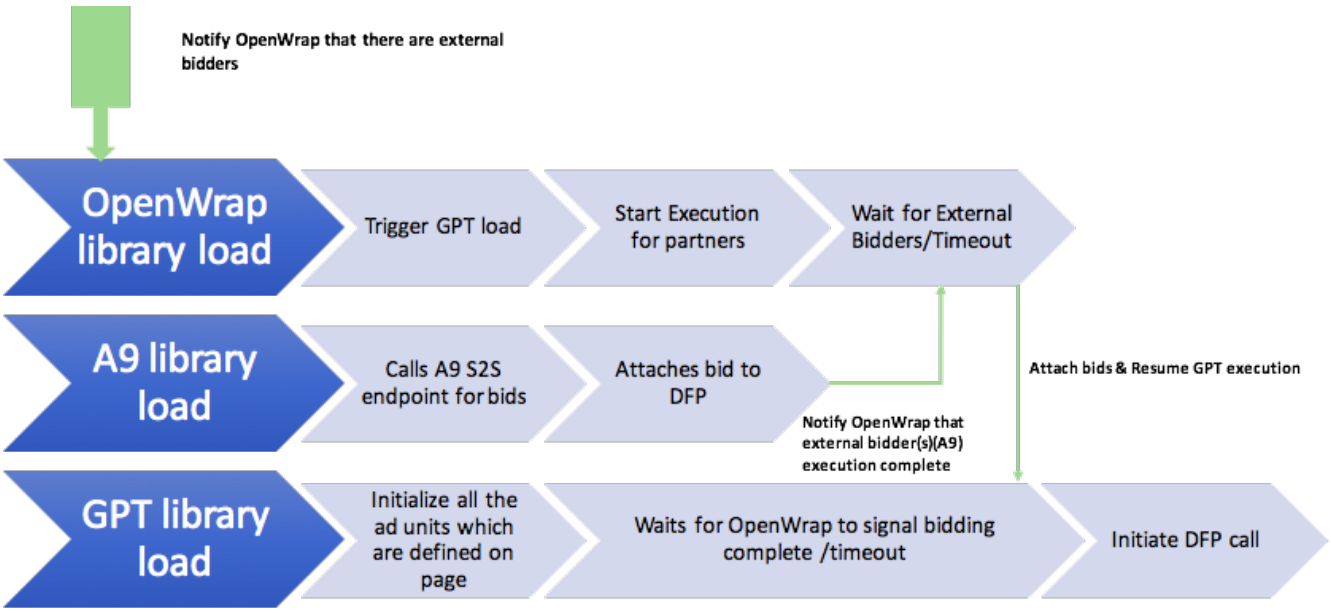
```
function registerExternalBidders(<Array of divIds:[string]>){}::return notifyId <integer>

function notifyExternalBiddingComplete(notifyId<integer>){} :: Nothing
```

1. **registerExternalBidders()** – This function will notify OpenWrap that there are one or more eternal bidders present for wrapper needs to wait before calling GAM.
   The call to this function should happen as soon as OpenWrap script loads.
2. **notifyExternalBiddingComplete()** – Will notify OpenWrap that all external bidders have completed execution, once per page. This function should be called as soon as bids from all external bidders (A9+ any bidders applicable) is available.

High Level Execution Flow

Standard GPT pages

For standard GPT pages with no time-based refresh/on-the-fly define-display for infinite scroll *(For non-advanced GPT where all the slots are defined once and the display API is called)*, the implementation can be handled using two global notification functions:



Sample Code with the Implementation. You can check the live working example here.

> **DFP is now called, Google Ad Manager (GAM)…**
>
> When you see DFP mentioned in this guide, remember that it is referring to GAM.

```
<!DOCTYPE HTML>
<html lang="en-us">
<head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <title>Wrapper Tag GPT Async Demo with A9</title>

    <h4>GPT implementation is ASYNC SRA <a href="https://support.google.com/dfp_premium/answer/1638622?hl=en">
(Refer this for types of GPT Tags)</a></h4>
    <!-- OpwenWrap Script begins here -->
    <script type="text/javascript">
    var PWT={}; //Initialize Namespace
    var notifyId;
    var googletag = googletag || {};
    googletag.cmd = googletag.cmd || [];
    PWT.jsLoaded = function(){ //PubMatic pwt.js on load callback is used to load GPT
        (function() {
            var gads = document.createElement('script');
            var useSSL = 'https:' == document.location.protocol;
            gads.src = (useSSL ? 'https:' : 'http:') + '//www.googletagservices.com/tag/js/gpt.js';
            var node = document.getElementsByTagName('script')[0];
            node.parentNode.insertBefore(gads, node);
        })();
    };
    (function() {
        var purl = window.location.href;
        var url = '//ads.pubmatic.com/AdServer/js/pwt/9999/1';
        var profileVersionId = '';
        if(purl.indexOf('pwtv=')>0){
            var regexp = /pwtv=(.*?)(&|$)/g;
            var matches = regexp.exec(purl);
            if(matches.length >= 2 && matches[1].length > 0){
                profileVersionId = '/'+matches[1];
            }
        }
        var wtads = document.createElement('script');
        wtads.async = true;
        wtads.type = 'text/javascript';
        wtads.src = url+profileVersionId+'/pwt.js';
        var node = document.getElementsByTagName('script')[0];
        node.parentNode.insertBefore(wtads, node);
    })();
    </script>
    <!--Wrapper Script ends here -->

<!-- A9 Script Start -->
    <script>
     !function(a9,a,p,s,t,A,g){if(a[a9])return;function q(c,r){a[a9]._Q.push([c,r])}a[a9]={init:function(){q
("i",arguments)},fetchBids:function(){q("f",arguments)},setDisplayBids:function(){},_Q:[]};A=p.createElement(s);
A.async=!0;A.src=t;g=p.getElementsByTagName(s)[0];g.parentNode.insertBefore(A,g)}("apstag",window,document,"
script","//c.amazon-adsystem.com/aax2/apstag.js");
     apstag.init({
         pubID: 'xxxx',
         adServer: 'googletag'
     });
     apstag.fetchBids({
         slots: [{
             slotID: 'Div2',
             sizes: [[300, 250], [300, 600]]
         },
         {
```

```
            slotID: 'Div1',
            sizes: [[728 ,90]]
        }],
        timeout: 2e3 // Make Sure this timeout is less than or equal to OpenWrap TimeOut
    }, function(bids) {
        // Your callback method, in this example it triggers the first DFP (GAM) request for googletag's
disableInitialLoad integration after bids have been set
        googletag.cmd.push(function(){
            apstag.setDisplayBids();
            window.OWT.notifyExternalBiddingComplete(notifyId); // This will tell OpenWrap that all the
external bidders have returned bid.
           // googletag.pubads().refresh(); No need to call refresh in callback as OpenWrap takes care of
display.
        });
    });
    </script>
    <!-- A9 Script End -->
    <script type="text/javascript">
        googletag.cmd.push(function() {
            var s1 = googletag.defineSlot('/15671365/DMDemo', [
                [728, 90]
            ], 'Div1').addService(googletag.pubads());
            s1.setTargeting('abc', 123);
            var s2 = googletag.defineSlot('/15671365/DMDemo1', [
                [300, 250],
                [300, 600]
            ], 'Div2').addService(googletag.pubads());
            s2.setTargeting('2abc', 123);

            notifyId = window.OWT.registerExternalBidders(['Div1', 'Div2']); // Notifies OpenWrap that there
are some external bidders for which it has to wait before calling GAM.

            // common targetings
            googletag.pubads().setTargeting('v', 'classifieds');
            googletag.pubads().setTargeting('article-id', '65207');
            googletag.pubads().enableSingleRequest();
            googletag.enableServices();

        });
    </script>
</head>
<body style='background-color:whitesmoke;'>
    <br><br>
    <!-- DMDemo -->
    Adslot : /15671365/DMDemo
    <br>
    <div id='Div1'>
    <script>
        googletag.cmd.push(function() {
           googletag.display("Div1");
        });
     </script>
    </div>
    <br><br>
    <!-- DMDemo1 -->
    Adslot : /15671365/DMDemo1
    <br>
    <div id='Div2'>
        <script>
        googletag.cmd.push(function() {
           googletag.display("Div2");
        });
     </script>
    </div>
</body>
</html>
```

Advanced GPT pages

Refer this section if you have advanced GPT implementation like periodic refresh of slots/on the fly define and display of slots for infinite scroll.

Example for Display

```
var notifyDiv1 = window.OWT.registerExternalBidders(["Div1"]);
googletag.display('Div1');
setTimeout(function () {
    window.OWT.notifyExternalBiddingComplete(notifyDiv1);
}, 1000);
```

Example for disableInitialload with Refresh

or each googletag.pubads().refresh([slot]) we need synchronization as follows:

```
<script>
var googletag = googletag || {};
googletag.cmd = googletag.cmd || [];
googletag.disableInitialload();

var notifyDiv1 = window.OWT.registerExternalBidders(["Div1"]);

googletag.pubads().refresh([slot]);
setTimeout(function () {
    window.OWT.notifyExternalBiddingComplete(notifyDiv1);
}, 1000);
</script>

<body>
    <div id="Div1">
        <script>
            googletag.cmd.push(function() {
                googletag.display("Div1");
            });
        </script>
    </div>
</body>
```