OpenWrap Integration Guide

This documentation describes the process of integrating the wrapper tag in OpenWrap. For more information about generating a wrapper tag, see the Open Wrap User Guide

Why use OpenWrap?

Publishers typically set lower priorities for programmatic demand in their ad servers, thus limiting it to remnant impressions via trafficked tags. The programmatic marketplace can only propose bids for that limited volume of inventory it's exposed to. As a result, publishers lose visibility and monetization of high-value programmatic demand on a large share of their traffic.

OpenWrap provides a transparent enterprise wrapper solution for Prebid and lets publishers evaluate programmatic bids from multiple sources concurrently for every impression. OpenWrap also lets ad servers begin utilizing real-time bids returned instead of the static bids that are used in case of yield tags.

How does OpenWrap work?

OpenWrap provides the pricing from programmatic open marketplaces from multiple SSP partners, which can be consumed by passing key-value pairs to ad servers in real time.

An easy-to-use UI provides the option of controlling partners and inventory mapping allowing for quick application and testing.

Ad call flow

1. PubMatic's wrapper tag JavaScript code is placed on the publisher's site, which makes an ad call to all the configured partners before calling the ad server.
2. OpenWrap runs an auction across all the bids from all partners.
3. The winning bid is sent to the publisher's ad server by dynamically adding this as a key value to the ad server call. PubMatic Wrapper line items are created in the publisher's ad server at various priorities targeting the bid signal in ad server call. The relevant line item is activated.
4. Publisher's ad server makes the final decision on which campaign/line item to serve considering Wrapper bid as an additional parameter, and the ad is served.

Integration process

This table provides an overview of the integration process.

Regarding integration timing:

- Most integration activities are dependent on the completion of prior activities. If a dependent activity is not completed in the time required, it will delay subsequent activities and potentially increase the overall time it takes to complete the integration.
- The sample timeline below is for a simple integration (one or two properties, a few profiles, 3-5 bidders). Larger integrations take longer because of the repetition and/or volume of information to collect and implement.

If you have questions or concerns, feel free to reach out to PubMatic.

Phase 1: Solution design (1 day)		
Activity	**Ownership**	**Details**
Information Gathering	PubMatic Publisher	1. Publisher completes the OpenWrap Integration Questionnaire 2. Publisher reviews PubMatic's OpenWrap documentation and sends follow-up questions to PubMatic. 3. Publisher reviews the demo page and identifies the appropriate API and tag for their production environment. - JS API: OpenWrap JavaScript API - JS hooks: OpenWrap Hooks for JS 4. PubMatic reviews the updated questionnaire, Wrapper functionality, and set up details with the publisher and then outlines the project plan.
Phase 2: Implement and test (2-3 days)		
Inventory setup (sites & tags)	PubMatic Publisher	1. Publisher provides a list of ad unit sizes for bidder setup 2. PubMatic sets up bidder per these guidelines: Account Setup Guide for Publishers (Sites & Ad Tags)

Create Partner Profiles & Provide Partner Mapping Details	Publisher	1. PubMatic shares bidder-specific configuration template with the publisher 2. Publisher connects with each bidder and obtains bid parameter details for respective bidder configuration 3. Publisher shares the updated bidder-specific template with PubMatic for partner configuration, and PubMatic review the details to ensure no mismatches.
Wrapper Profile Setup	PubMatic	1. PubMatic reviews the profile configuration requirements with the publisher 2. Publisher shares profile configuration values with PubMatic 3. PubMatic prepares for profile setup based on Publisher's input
Ad server setup (order, line item, creative)	PubMatic Publisher	1. PubMatic shares the ad server configuration template with the publisher 2. Publisher updates the ad server configuration template and sends it back to PubMatic 3. Publisher (or PubMatic) creates the orders and line items in the ad server
On-page setup & Integration Testing	PubMatic Publisher	1. Copy the wrapper tag to the test web page & provide test page to PubMatic. 2. Ensure requests/responses are generated for each partner who is included across a given profile. 3. Verify that key/value pairs are passed to the ad server call. 4. Verify that ad server is responding with the winning creative and that the creative is rendered.
Phase 3: Launch and ramp (2-3 days)		
Account ramp	PubMatic Publisher	1. Copy wrapper tag to a live page and repeat the on-page setup & integration testing (from phase 2) 2. Once verified, send test traffic of 100k impressions per day from a live web page 3. Perform discrepancy checks between OpenWrap analytics and ad server data 4. Repeat these steps for each property set up and launch

Integration options

OpenWrap supports two integration options:

- JavaScript API
- On-page integrations for GAM

The JavaScript API is the most flexible method, we recommend using it if you have multiple wrappers on the page. For example, if you're running OpenWrap & Amazon TAM together, you should use the JavaScript API. If OpenWrap is the only wrapper on the page, you can use either method, but GPT patching is only available if you're using GAM as your ad server.

> While GPT Patching continues to be supported, new implementations should the JavaScript API.

Wrapper partner JS inclusions and parameters

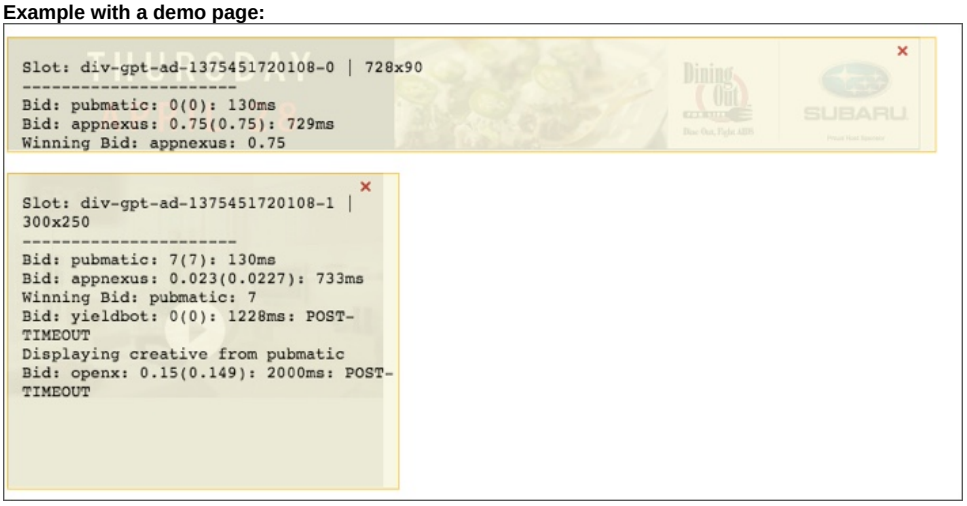
Use the information in the tables provided in the Partner Integrations Available in OpenWrap document.

Integration testing

Complete the following steps to ensure that each header tag in the Wrapper is working successfully:

1. All the calls for partners are firing for the included/mapped ad units.
    - This can be verified in the network panel of browser console. An ad call should be made to the respective partner end point listed in previous section.
2. All partners are bidding.
    - This can be verified on the debug console. The debug console is automatically loaded when debugging a version using the debug URL generated while in push to staging mode.

Example with a demo page:



- If a partner is not bidding after multiple tries over a period of several minutes, you may have to take this up with the respective partner.

3. Winning keys are passed to the ad server.
    - The winning partner can be checked in the debug console. However, to check if the key values are being passed to the ad server you can use the browser console.
4. Winning creative rendering for each partner.
    - Once any partner wins and the key values are sent to GAM, GAM should deliver the Wrapper tag campaign (which was already configured).
    - If a partner is not winning because of low bids, the publisher can increase the rev-share parameter of other partners in order to make that particular partner win.
    - After rendering the creative, last step is to check that the partner's tracker is fired.
5. Reporting - Please ensure partners are counting impressions in their system.
6. Check wrapper analytics.

Ad server configuration

Refer to instructions provided in documents mentioned below according to your ad server.

GAM

Ad Server Configuration

`If using SafeFrames, see SafeFrames.`

PMP & PMP-G support

Please refer to PMP-Guaranteed Implementation Guide for Publishers to understand how PubMatic supports PMP and PMP-G.

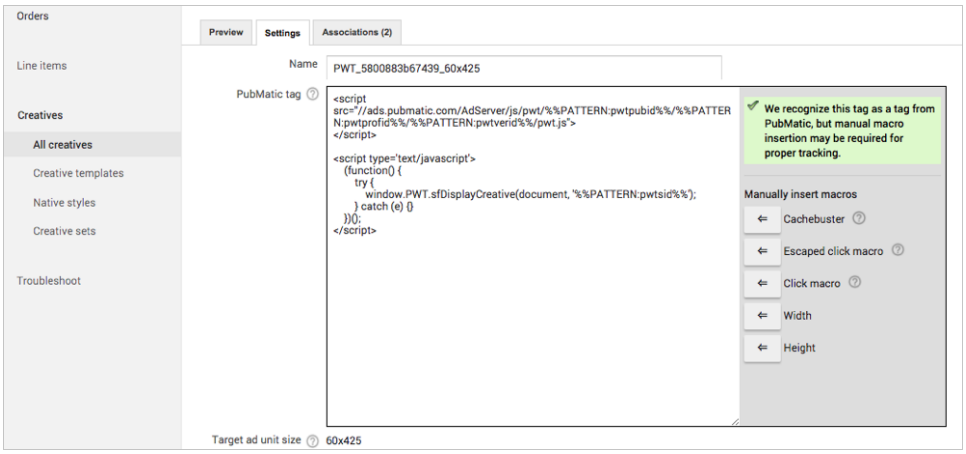
SafeFrames (GAM only)

Publishers using PubMatic's OpenWrap can place the creative script in a Friendly iframe or a SafeFrame. Using a Friendly iframe, the content from the ad server renders in a frame on the same server as the host content. SafeFrame is a managed, API-enabled iframe that opens a line of communication between the publisher page content and the iframe-contained external contend (for example, ads). This provides a line of communication that affords data collection and rich interaction (for example, ad expansion), unavailable within a Friendly iframe.

Creating SafeFrame line items in GAM

1. Navigate to the **Creative Settings** for a line item and enter the appropriate creative script.

    There are different methods for rendering creatives for PMP bids and Open Market bids. Use the appropriate related script below



SafeFrame for rendering PMP bids

```
<script src="//ads.pubmatic.com/AdServer/js/pwt/%%PATTERN:pwtpubid%%/%%PATTERN:pwtprofid%%/%%PATTERN:
pwtverid%%/pwt.js">
</script>

<script type='text/javascript'>
         (function() {
                  try {
                           window.PWT.sfDisplayPMPCreative(document, '%%PATTERN:pwtdeal_pubmatic%%', []);
          } catch (e) {}
          })();
</script>
```

SafeFrame for rendering Open Auction bids

```
<script src="//ads.pubmatic.com/AdServer/js/pwt/%%PATTERN:pwtpubid%%/%%PATTERN:pwtprofid%%/%%PATTERN:
pwtverid%%/pwt.js">
</script>

<script type='text/javascript'>
    (function() {
        try {
            window.PWT.sfDisplayCreative(document, '%%PATTERN:pwtsid%%');
        } catch (e) {}
    })();
</script>
```

2. Check the box for **Serve into a SafeFrame.**
3. Complete any other required fields and choose **Save**.

Wrapper keys sent to GAM

Key Name	Explanation	Sample Value
`pwtecp`	The eCPM of bid in USD up to 4 decimal places.	`1.4356`
`pwtbst`	Bid Status Flag: will be 1 for positive bids.	`1`
`pwtdid`	Deal Id: in cases when a bid has an associated deal.	`XYZ-DEAL 123`
`pwtsid`	The slot/bid id of the highest or winning bid.	`e7424477d70315b8a4bae5cff1887edf`

`pwtpid`	All partner data organized by partner name. Add a filter to retrieve data for a specific partner. Examples: `pwtpid` - to retrieve all partner data by partner name. `pwtpid=pubmatic` - to retrieve data only for PubMatic. For partner names, see bidder codes in PreBid bidders documentation	`pubmatic`
`pwtdeal_<BidderCode>`	Partner-specific `dealID` and `dealChannel` based targeting can be done using this key. Format of the value is: `<DealChannel>_-_<DealId>_-_<BidId>`.	`Key: pwtdeal_pubmatic` `Value: PMPG_-_dealxyz_-_123456`
`pwtplt`	Value of the ad format impression. Possible values: display, native, amp, inapp, video	`{{ display}}`
`pwtsz`	The size of the highest/winning bid.	`WxH: 300x250`

Additional keys sent for Send All Bid

When **Send All Bids** is enable, the following additional keys will be sent for every bidder with a non-zero bid on an ad unit:

Key name	Explanation	Sample value
pwtsid_<bidder_code>	The slot/id of the that bidder's bid.	pwtsid_pubmatic
pwtbst_<bidder_code>	Bid Status Flag: will be 1 for positive bids.	pwtbst_pubmatic
pwtecp_<bidder_code>	The eCPM of bid in USD up to 4 decimal places.	pwtecp_pubmatic
pwtsz_<bidder_code>	The ad size of that bidder's bid	pwtsz_pubmatic

SafeFrame scenario and results

Scenario	Result
Inside GAM, SafeFrame creative code added but SafeFrame option is not selected.	Creative rendered correctly if it does not use SafeFrame feature. For example, a static image creative renders correctly but an expandable creative that relies on SafeFrame will not work correctly.
Inside GAM, legacy/regular pwt creative code added, SafeFrame option selected and SafeFrame creative rendered.	Error seen on browser console. Creative is rendered but if it is rich media creative/expandable then it will not work as expected.
Display non-compatible expandable creative inside SafeFrame	Creative does not render correctly or post creative expansion results into page error or unknown behavior.
Display SafeFrame creative inside a non-SafeFrame	Error seen on the browser console. Creative is rendered but if it is rich media creative /expandable then it will not work as expected.

Creative to be rendered inside SafeFrame must be SafeFrame compatible and needs to be tested on the page.

References

To learn more about SafeFrames see:

- SafeFrames_v1.1_final.pdf
- IAB SafeFrame Guidelines

Known limitations

The following sections describe known limitations for OpenWrap.

Code generation performance

The table below shows how performance of API code generation with gulp execution varies according to the number of adapter partners.

Number of Partners	Time to Execute

1	18-20 seconds
10	25-30 seconds

This impacts flow, user flows, and API calls when:

- Copying profile.
- Pushing wrapper code to staging.
- Mapping upload.

SafeFrame

- Expandable creative will not work correctly after rendering inside of a SafeFrame on Safari, IE11/Edge.

Support for previous code build

Upon a stable code release of OpenWrap (with Prebid), prior code builds are unsupported. All subsequent OpenWrap profile versions will use the new stable code build.

Rubicon legacy adapter support

OpenWrap no longer supports Rubicon legacy adapter in any future code builds. Rubicon Legacy adapter works only in releases prior to OpenWrap (with Prebid).

> **To use Rubicon Fastlane…**
>
> Publishers must migrate Rubicon accounts to Rubicon Fastlane and get new IDs to map in OpenWrap.

IndexExchange adapter

OpenWrap ignores bids from Index Exchange adapter in a non-SRA setup. Perform the SRA GPT setup process to make the IndexExchange adapter work in OpenWrap.