

Publisher Optimization

Implementation Guide

Last updated: 9/2020

Table of Contents

Overview	3
Use Cases	4
Forecasting	4
Trafficking	4
Technical Overview	4
Parameters and Keys	6
Configurations	6
Brand Safety	8
Context Control	9
Invalid Traffic	10
Viewability	10
Implementation Steps	11
Include pet.js	11
Gather Request Data	12
Fire Request	12
Google Ad Manager	12
AppNexus	20
Get Response Data	23
Ad Server	24
Google Ad Manager	24
AppNexus	25
Best Practices	26
Debugging	26
General	26

Overview

Integral Ad Science's (IAS) developed **publisher optimization** to allow your ad server to automatically deliver campaigns to goals defined by IAS measurement data. This solution allows you to:

- Sell direct campaigns to advertisers with guarantees based upon IAS data – for example, a RON (Run of Network) campaign that has 70% viewability, no brand safety risk, and no invalid traffic.
- Set up private marketplaces (PMP) with similar guarantees – for example, a PMP containing semi-transparent inventory that has 80% viewability, no brand safety risk, and no invalid traffic.

Your ad server's existing facilities for forecasting and automatic optimization can be fully leveraged. As a result, you can minimize waste from over buffering while minimizing risk of under delivery. This solution can minimize the amount of time ad ops needs to monitor campaigns with viewability or brand safety goals and unlock opportunities to sell segments of high quality inventory to advertisers at a premium.

Publisher optimization uses historical information via the publisher verification pixel to create a viewability prediction model. Once the model is released, you pass incoming real time data to publisher optimization for instantaneous decision making.

Publisher optimization works on the following channels for display and video formats for Desktop and mobile web.

Note: You must load ad slots asynchronously.

Publisher optimization supports **ad refresh** which uses the JavaScript library's viewability. Ad refresh shares all features and limitations with publisher optimization's viewability and contains a parameter to enable and configure. Publishers configure ad refresh via the IAS Platform; see the Publisher Ad Refresh Getting Started Guide for more information.

Optimization supports **Accelerated Mobile Pages (AMP)** based inventory via a parameter. However, **pet.js** can't be loaded directly on an AMP page without an AMP iframe.

See the **Configurations** section for more information on the parameters.

Use Cases

Publisher optimization allows you to forecast inventory and intelligently traffic segments.

Forecasting

You want forecasting available for viewable inventory, viewable inventory above a certain media quality bar. You are left to guess at how much "buffer" you need to add to a normal campaign's run to avoid under delivery, and even then closely watch the campaign so it delivers.

Publisher optimization solves those problems because it allows you to:

- Eliminate wasted ad spend trying to achieve viewability minimums.
- Meet commitments for direct and PMP inventory.
- Forecast and automate optimization on viewability, ad fraud, and brand safety.
- Automate 100% to avoid a need for any manual optimization.

Trafficking

You want to traffic segments off their highest value inventory (for example, most viewable, most brand safe, etc.) and offer it at a premium to advertisers.

Publisher optimization solves those problems because it allows you to:

- Sell direct campaigns and set up PMPs with guarantees from IAS data.
- Integrate directly into Google Ad Manager's optimization, forecasting, and delivery functionality.

Technical Overview

Publisher optimization uses a JavaScript library for Desktop and mobile web environments.

IAS designed publisher optimization to minimize the page latency and delay to make ad requests. Publisher optimization is fully cacheable by the browser and strategically located in servers in several regional IAS data centers across the US, Europe, Asia, and the South Pacific to minimize response time for your users.

Include the publisher optimization tag directly on your pages which is run before you make the ad request to your ad server (for example, Google Ad Manager) and the tag returns a set of key value pairs to include with your ad request. While IAS isn't a bidder, IAS does run before your ad request and it's easiest to include IAS as an entry in your header bidding wrapper.

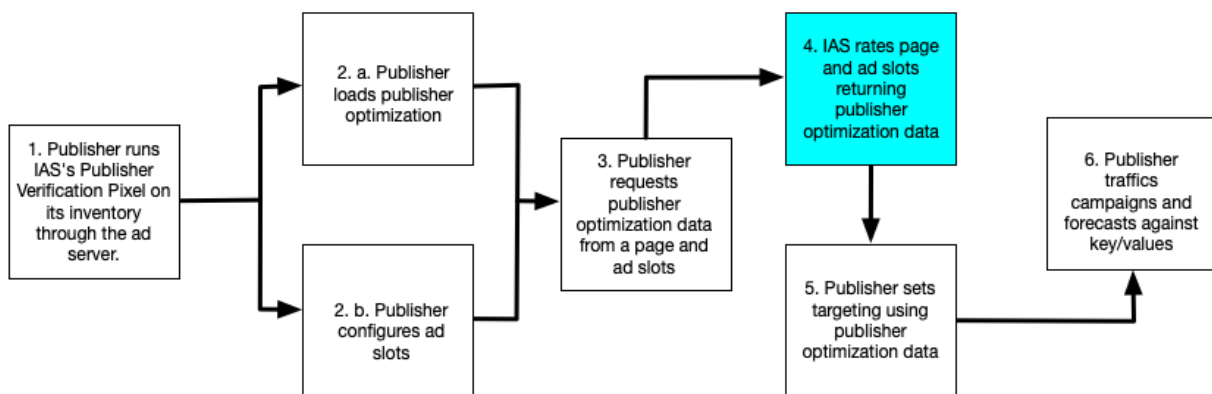
If you are using a header bidding wrapper, IAS does not add latency outside of what you budgeted there. If you are not using a header bidding wrapper, you can set a configurable maximum time to wait for publisher optimization results before continuing with the ad request.

Note: See the Publisher Optimization Prebid.js Adapter Implementation Guide if you are using header bidding.

The easiest way to use publisher optimization is to place it directly onto your page's header so publisher optimization can be retrieved and executed as soon as possible. When using publisher optimization, you need to feed information about the ad slots so IAS can make better viewability predictions about them, as explained later in this document.

Integration steps summary

1. Run a publisher verification pixel on every ad slot for 15 days allowing IAS to train a model for viewability prediction.
2. Integrate publisher optimization into your page.
3. Optimize your inventory using publisher optimization.



The steps below describe how to integrate publisher optimization:

1. Run IAS's pixel on its inventory through the ad server (for example, Google Ad Manager) which allows you to receive reporting from IAS and allows IAS to build the predictive models necessary for the solution.
2.
 - a. Load publisher optimization asynchronously to include the publisher optimization script directly on a page.
 - b. Configure the ad slots by passing information (HTML ID attribute, ad size, and ad unit path) to publisher optimization representing each of the ad placements on the page. Optionally, you can pass in a callback to handle IAS data, or a timeout value for an XMLHttpRequest (XHR) to IAS. You must not send random ad slot IDs to publisher optimization.
3. Publisher optimization fires requests to IAS once loaded.
4. IAS rates the page and all placement/slot IDs, returning results to your page before regular ad server requests are made.

5. Add the returned IAS JSON (JavaScript Object Notation) output to your existing ad requests as key/value targeting parameters.
6. Traffic campaigns and forecasts against the key/values. IAS provides key/values for:
 - predicted viewability for each ad slot (Desktop and mobile web only).
 - brand safety risk along the 7 brand safety dimensions IAS tracks (Desktop and mobile web only).
 - whether the current ad represents invalid traffic (IVT).

Parameters and Keys

Configurations

For better targeting line items, you need to feed information about the ad placements to a JavaScript library, **pet.js** via the following parameters:

Name	Field	Type	Required	Note:
adSlots				A variable that can contain either a single ad slot object or an array of ad slots.
	size	Array	Yes	Either one or multiple sizes for the ad. A single size is specified as an array (for example, [300, 250]), or an array of arrays for multiple sizes (for example, [[300, 250], [300, 600]]).
	adSlotId	String	Yes	"id" attribute of the ad slot's HTML element.
	adUnitPath	String	Yes	The unicode encoded ad unit path as defined in the ad server. Ensure each ad unit path corresponds to one placement location on the page.
dataHandler		Function	Yes	Function that handles returned IAS data.
pubId		String	Yes	A unique ID that IAS uses to identify publishers, provided by IAS.
timeout		Number	No	Timeout value for the AJAX request for IAS data. Default is 1000 milliseconds.
__iasAdRefreshConfig		Object	No	Object used for ad refresh. The __iasAdRefreshConfig object has the refreshTargeting object.

Name	Field	Type	Required	Note:
refreshTargeting		Object	No	<p>Object within the <code>__iasAdRefreshConfig</code> object which supports ad refresh. The <code>__iasAdRefreshConfig</code> variable on the window object exists at PET runtime. For example, here you set the targeting value of 'rfv' which you can get via <code>slot.getTargeting('rfv')</code> after the first refresh if you are pulling the value programmatically on your page:</p> <pre> window.__iasAdRefreshConfig.refreshTargeting = { enabled: true, targetingKey: 'rfv' }; </pre>
	enabled	boolean	No	Turns on or off targeting by ad count. By default, targeting by ad count is disabled.
	targetingKey	String	No	Set a targetingKey to a value so IAS can refresh. The targetingKey value must match the targeting value you set in Google Ad Manager.
effectiveURL		String	No	<p>Used for AMP pages. Set this to the AMP ad URL:</p> <pre> window.__iasPET.effectiveURL=<url> </pre> <p>Note: pet.js can't be loaded directly on an AMP page without an AMP iframe.</p>

Brand Safety

Publisher optimization supports seven positive targeting brand safety areas and custom segments which include Context Control.

As a best practice, you should not apply brand safety targeting to sponsorships or home page take overs.

Key	Description	Values	Notes
adt	IAS adult brand safety risk	veryLow, low, medium, high	Risk of the page containing adult content.
alc	IAS Alcohol Brand Safety Risk	veryLow, low, medium, high	Risk of the page containing content related to alcohol.
dln	IAS Illegal Download Brand Safety Risk	veryLow, low, medium, high	Risk of the page containing content related to illegal downloads.
drg	IAS Drug Brand Safety Risk	veryLow, low, medium, high	Risk of the page containing content related to drugs and drug use.
hat	IAS Hate Speech Brand Safety Risk	veryLow, low, medium, high	Risk of the page containing hate speech.
off	IAS Offensive Language Brand Safety Risk	veryLow, low, medium, high	Risk of the page containing offensive language.
vio	IAS Violence Brand Safety Risk	veryLow, low, medium, high	Risk the page contains violent content.

Key	Description	Values	Notes
ias-kw	Context Control: (Avoidance)	Custom per client	IAS_<unique ID>_PG For example, IAS_23_PG. Context Control (Avoidance) segments use negative targeting; the presence of the segment response object identifies when the page is in a context control segment.
ias-kw	Keyword segment	Custom per client	IAS_<unique ID>_KW For example, IAS_33_KW or IAS_133_33_KW for legacy lists. Keyword segments use negative targeting; the presence of the keyword segment response object identifies when the URL is in a segment.

Context Control

Context control is a solution that blocks or optimizes campaigns on web pages by scanning page contents against a set of custom criteria, to accord with a team's brand suitability thresholds. Context control lets you avoid pages which are not suitable for the brand's custom standards.

Publisher optimization returns a custom object in the JSON response when the page is in a segment in a profile using the key **ias-kw** for both context control and legacy keyword segments.

Context control uses the convention IAS_<unique ID>_PG and IAS_<unique ID>_KW for legacy keyword segments. Also, IAS returns **IAS_UNSCORED_PG** to denote pages which aren't scored for context control. If you activate an IAS_<unique ID>_PG, you should also activate IAS_UNSCORED_PG in your ad server.

Context control (Avoidance) uses negative targeting; the presence of the response object identifies when the page is not suitable for the brand's custom standards. Conversely, the absence of any context control (Avoidance) information means the page is brand suitable.

Seg Code	Targeting Methodology	Notes
IAS_<unique ID>_PG	Negative targeting	Not eligible for the campaign
IAS_UNSCORED_PG	Negative targeting	IAS has not scored the page.
IAS_<unique ID>_KW	Negative targeting	Not eligible for the campaign

Note: The presence of the segment code in the response indicates when the page is in violation of the advertiser's custom brand suitability standards.

IAS returns context control segments in the same response for standard IAS viewability, brand safety, and invalid traffic.

Invalid Traffic

Key	Description	Values	Notes
fr	IAS Invalid Traffic	true, false	Whether the ad represents invalid traffic (for example, the visitor is a bot rather than a human).

Viewability

As a best practice, you should not apply viewability targeting to sponsorships or home page take overs.

Key	Description	Values	Notes
vw	IAS viewability prediction	40, 50, 60, 70, 80	This meets the MRC standard. Probability that an ad slot is in view. Each value indicates that the ad is at least that likely to be in view (for example, "50" means "50% or more likely to be in view"). Choose a single bucket when trafficking a line item.
grm	GroupM Viewability Prediction	40, 50, 60, 70, 80	This meets the GroupM standard. Publisher uses the current GroupM/Publicis standard of 100% in view for 1 second.
pub	Publicis Viewability	40, 50, 60, 70, 80	This meets the Publicis standard.

Key	Description	Values	Notes
	Prediction		
vw05, vw10, vw15, vw30	IAS Viewability Time In View Prediction	40, 50, 60, 70, 80	Probability of time in view for display to be equal or exceed 5, 10, 15, and 30 second thresholds.
vw_vv	IAS Video Viewability Prediction	40, 50, 60, 70, 80	This meets the MRC video standard. Probability that an ad slot is in view. Each value indicates that the ad is at least that likely to be in view (for example, "50" means "50% or more likely to be in view"). Choose a single bucket when trafficking a line item.
grm_vv*	GroupM Video Viewability Prediction	40, 50, 60, 70, 80	This meets the GroupM video standard.
pub_vv*	Publicis Video Viewability Prediction	40, 50, 60, 70, 80	This meets the Publicis video standard.

* The GroupM and Publicis video viewability prediction targeting is currently in a beta period. Contact your IAS representative if you wish to be part of the beta program.

Implementation Steps

The Desktop and mobile web environment uses a JavaScript library. Publisher optimization returns JSON with the response data as key/value targeting parameters.

IAS hosts **pet.js** on a content delivery network (CDN). You need to include **pet.js** on your pages to make requests and respond to the request output. **pet.js** is the naming convention to refer to the latest **iasPet.<version>.js** library.

Note: Contact your IAS representative for the pet.js version to use.

Include pet.js

```
<html>
<head>
  <script src="//cdn.adsafeprotected.com/iasPET.1.js"/>
</head>
```

</html>

Gather Request Data

You need to send **pet.js** the following information:

- Slot information: ad size, path, and ID
- Publisher ID
- Data handler function

Fire Request

You might load ads differently from the sample code depending whether you use lazy loading.

Google Ad Manager

Note: This is sample code which you must adapt to fit your environment.

Display

```
<!-- Include the pet.js script in the header of your page or early in the body -->

<script async='async' src='//cdn.adsafeprotected.com/iasPET.1.js'></script>
<!-- Include this script after you've defined all GPT ad slots -->
<script>
  // this is an *example* function. in your implementation, point to
  // an existing function that you use to request ads from DFP
  function requestAds() {
    googletag.cmd.push(function() {
      // display the ads on your page. replace these
      // IDs with those on your page.
      googletag.display('div-gpt-ad-1411587747174-0');
      googletag.display('div-gpt-ad-1411587747175-0');
      googletag.display('div-gpt-ad-1411587747176-0');
    });
  }

  // Set up IAS pet.js
  var iasDataHandler, __iasPET = __iasPET || {};
  __iasPET.queue = __iasPET.queue || [];
  __iasPET.pubId = 'xxxx'; // your account manager provides this ID

  // this is the maximum amount of time in milliseconds to wait
```

```
// for a PET response before requesting ads without PET data.
// IAS recommends starting at 2 seconds
// when testing and adjusting downwards as appropriate.
// remember to replace 'requestAds' below with the
// function you use to request ads from DFP.
var IASPET_TIMEOUT = 2000;
var __iasPETTimeoutRequestAds = setTimeout(requestAds, IASPET_TIMEOUT);

// this function is called when a PET response is received. it
// sets the targeting data for DFP and request ads
// remember to replace requestAds() with the function you use for requesting
// ads from DFP
var iasDataHandler = function(adSlotData) {
    clearTimeout(__iasPETTimeoutRequestAds);
    __iasPET.setTargetingForGPT();
    requestAds();
};

// make the PET request
googletag.cmd.push(function() {
    // read the currently defined GPT ad slots for sending to the PET endpoint
    // defined all GPT ad slots before calling PET
    var gptSlots = googletag.pubads().getSlots();
    var iasPETSlots = [];
    for (var i = 0; i < gptSlots.length; i++) {
        var sizes = gptSlots[i].getSizes().map(function(size) {
            if (size.getWidth() && size.getHeight())
                return [size.getWidth(), size.getHeight()];
            else
                return [1, 1];
        });
        iasPETSlots.push({
            adSlotId: gptSlots[i].getSlotElementId(),
            //size: can either be a single size (for example, [728, 90])
            // or an array of sizes (for example, [[728, 90], [970, 90]])
            size: sizes,
            adUnitPath: gptSlots[i].getAdUnitPath()
        });
    }
});
```

```
    });  
  }  
  // make the request to PET. if your page makes multiple ad requests to DFP  
  // (for example, lazily loaded ads, infinite scrolling pages, etc.), make  
  // a request to PET before every request to DFP  
  __iasPET.queue.push({  
    adSlots: iasPETSlots,  
    dataHandler: iasDataHandler  
  });  
});  
</script>
```

Video

HTML:

```
<html>  
  <head>  
    <title>IMA HTML5 Demo</title>  
    <link rel="stylesheet" type="text/css" href="style.css">  
  </head>  
  <body>  
    <div id="mainContainer">  
      <div id="content">  
        <video id="contentElement">  
          <source src="./SampleVideo_1280x720_1mb.mp4"></source>  
        </video>  
      </div>  
      <div id="adContainer"></div>  
    </div>  
    <button id="playButton">Play</button>  
    <script type="text/javascript"  
src="//imasdk.googleapis.com/js/sdkloader/ima3.js"></script>  
    <script type="text/javascript" src="https://static.adsafeprotected.com/vans-  
adapter-google-ima.js"></script>  
    <script async='async'  
src='https://cdn.adsafeprotected.com/iasPET.1.js'></script>  
    <script type="text/javascript" src="ads.js"></script>  
  </body>  
</html>
```

ads.js:

```
var videoContent = document.getElementById('contentElement');
var adDisplayContainer =
  new google.ima.AdDisplayContainer(
    document.getElementById('adContainer'),
    videoContent);
// Must be done as the result of a user action on mobile
adDisplayContainer.initialize();
// Re-use this AdsLoader instance for the entire lifecycle of your page.
var adsLoader = new google.ima.AdsLoader(adDisplayContainer);
// Add event listeners
adsLoader.addEventListener(
  google.ima.AdsManagerLoadedEvent.Type.ADS_MANAGER_LOADED,
  onAdsManagerLoaded,
  false);
adsLoader.addEventListener(
  google.ima.AdErrorEvent.Type.AD_ERROR,
  onAdError,
  false);
function onAdError(adErrorEvent) {
  // Handle the error logging and destroy the AdsManager
  console.log(adErrorEvent.getError());
  adsManager.destroy();
}
// An event listener to tell the SDK that our content video
// is completed so the SDK can play any post-roll ads.
var contentEndedListener = function() {adsLoader.contentComplete();};
videoContent.onended = contentEndedListener;
// Request video ads.
var adsRequest = new google.ima.AdsRequest();
var myAdSize = '640x480';
var myAdUnit = '/1234567/myfolder/pub_opt/preroll'; //replace appropriately
var myAdReferrerUrl = 'https://integralads.com';
var myAdDescriptionUrl = 'https://integralads.com';
var myAdTimeStamp = Date.now();
var myVideoSrc = document.getElementById('contentElement').currentSrc;
var myVideoName = myVideoSrc.substr(myVideoSrc.lastIndexOf('/') + 1);
```

```
adsRequest.adTagUrl =
'https://pubads.g.doubleclick.net/gampad/ads?sz='+myAdSize
 +'&iu='+myAdUnit
 +'&impl=s&gdfp_req=1&env=vp&output=vast&unviewed_position_
start=1&url='+myAdReferrerUrl
 +'&description_url='+myAdDescriptionUrl
 +'&correlator='+myAdTimeStamp; //DFP
// Specify the linear and nonlinear slot sizes. This helps the SDK to
// select the correct creative if multiple are returned.
adsRequest.linearAdSlotWidth = 640;
adsRequest.linearAdSlotHeight = 400;
adsRequest.nonLinearAdSlotWidth = 640;
adsRequest.nonLinearAdSlotHeight = 150;
var playButton = document.getElementById('playButton');
playButton.addEventListener('click', requestAds);
/*****IAS PET START*****/
var globalAdRequestParams;
var slotAdRequestParamsMap;
var iasDFPUtilites = (function () {
  const equal = "%3D";
  const ampersand = "%26";
  const comma = "%2C";
  var adSlotDataObj;
  var parseIASResponseData = function (adSlotData) {
    adSlotDataObj = JSON.parse(adSlotData);
  }
  var getIASPageTargetingAsQueryString = function () {
    var iasPageTargeting = "";
    iasPageTargeting = appendRequestParam(
      iasPageTargeting, "fr", adSlotDataObj.fr);
    for (const k in adSlotDataObj.brandSafety) {
      iasPageTargeting = appendRequestParam(
        iasPageTargeting, k, adSlotDataObj.brandSafety[k]);
    }
    return iasPageTargeting;
  }
  var getIASSlotTargetingAsQueryString = function () {
```



```
var iasSlotTargeting = {};  
for (const slotId in adSlotDataObj.slots) {  
  const slotIdValues = adSlotDataObj.slots[slotId];  
  var slotParams = "";  
  for (const key in slotIdValues) {  
    var segments = slotIdValues[key];  
    if (Array.isArray(segments)) {  
      slotParams = slotParams.concat(ampersand, key, equal);  
      const last = segments.length - 1;  
      for (var i = 0, len = segments.length; i < len; i++) {  
        var separator = comma;  
        if(i === last) {  
          separator = "";  
        }  
        slotParams = slotParams.concat(segments[i], separator);  
      }  
    } else {  
      slotParams = appendRequestParam(slotParams, key, segments);  
    }  
    iasSlotTargeting[slotId] = slotParams;  
  }  
}  
return iasSlotTargeting;  
}  
  
var appendRequestParam = function (query, key, value) {  
  return query.concat(ampersand, key, equal, value);  
}  
  
return {  
  parseIASResponseData: parseIASResponseData,  
  getIASPageTargetingAsQueryString: getIASPageTargetingAsQueryString,  
  getIASSlotTargetingAsQueryString: getIASSlotTargetingAsQueryString  
};  
})();  
  
var iasDataHandler;  
var __iasPET = __iasPET || {};  
__iasPET.queue = __iasPET.queue || [];  
__iasPET.pubId = 'xxx'; //your account manager provides this ID
```

```
iasDataHandler = function (adSlotData) {
  console.log("Callback function");
  iasDFPUtilites.parseIASResponseData(adSlotData);
  globalAdRequestParams = iasDFPUtilites.getIASPageTargetingAsQueryString();
  slotAdRequestParamsMap = iasDFPUtilites.getIASSlotTargetingAsQueryString();
  console.log('&cust_params=' + globalAdRequestParams.substr(3)
    + slotAdRequestParamsMap['ad-1']);
  adsRequest.adTagUrl = adsRequest.adTagUrl + '&cust_params='
    + globalAdRequestParams.substr(3)
    + slotAdRequestParamsMap['ad-1'];
}
var iasPETSlots = [];
var adSlots = ['ad-1'];
for (var i = 0; i < adSlots.length; i++) {
  iasPETSlots.push({
    adSlotId: adSlots[i],
    size: [1, 1],
    adUnitPath: myAdUnit,
    type: "video"
  });
}
__iasPET.queue.push({
  adSlots: iasPETSlots,
  dataHandler: iasDataHandler
});
/*****IAS PET END*****/
function requestAds() {
  adsLoader.requestAds(adsRequest);
}
function onAdsManagerLoaded(adsManagerLoadedEvent) {
  var videoElement = document.getElementById('contentElement');
  var adsRenderingSettings = new google.ima.AdsRenderingSettings();
  // Get the ads manager.
  adsManager = adsManagerLoadedEvent.
    getAdsManager(videoContent,adsRenderingSettings);
  // Add listeners to the required events.
  adsManager.addEventListener(
```

```
    google.ima.AdErrorEvent.Type.AD_ERROR,
    onAdError);
adsManager.addEventListener(
    google.ima.AdEvent.Type.CONTENT_PAUSE_REQUESTED,
    onContentPauseRequested);
adsManager.addEventListener(
    google.ima.AdEvent.Type.CONTENT_RESUME_REQUESTED,
    onContentResumeRequested);
try {
    // Initialize the ads manager. Ad rules playlist will start at this time.
    adsManager.init(640, 360, google.ima.ViewMode.NORMAL);
    // Call start to show ads. Single video and overlay ads will
    // start at this time; this call will be ignored for ad rules, as ad rules
    // ads start when the adsManager is initialized.
    adsManager.start();
} catch (adError) {
    // An error may be thrown if there was a problem with the VAST response.
    // Play content here, because we won't be getting an ad.
    videoContent.play();
}
/***** Start IAS CODE *****/
/*
/vans-adapter-google-ima.js|jsvid\?videoId|mon\?videoId/
/vans-adapter-google-ima|videoId/
*/
var config = {
    anId: '123456', //replace appropriately
    campId: myAdSize,
    chanId: myAdUnit,
    placementId: 'Open Auction',
    pubOrder: 'Video',
    pubId: 'Direct', // when non-AdX creatives
    custom: window.location.hostname,
    custom2: myVideoName
};
googleImaVansAdapter.init(google, adsManager, videoElement, config);
/***** End IAS CODE *****/
```

```
}  
  
function onContentPauseRequested() {  
    // This function is where you should setup UI for showing ads (e.g.  
    // display ad timer countdown, disable seeking, etc.)  
    videoContent.removeEventListener('ended', contentEndedListener);  
    videoContent.pause();  
}  
  
function onContentResumeRequested() {  
    // This function is where you should ensure that your UI is ready  
    // to play content.  
    videoContent.addEventListener('ended', contentEndedListener);  
    videoContent.play();  
}
```

AppNexus

Note: This is sample code which you must adapt to fit your environment.

```
<html>  
    <head>  
        <script type="text/javascript">  
            var apntag = apntag || {};  
            apntag.anq = apntag.anq || [];  
            (function() {  
                var d = document, scr = d.createElement('script'), pro =  
                    d.location.protocol, tar=d.getElementsByTagName("head")[0];  
                scr.type = 'text/javascript'; scr.async = true;  
                scr.src = ((pro === 'https:') ? 'https' : 'http') +  
                    '://acdn.adnxs.com/ast/ast.js';  
                if(!apntag.l){  
                    apntag.l=true; tar.insertBefore(scr, tar.firstChild);  
                }  
            })();  
            apntag.anq.push(function() {  
                apntag.setPageOpts({  
                    member: XXXX,  
                    keywords: {  
                        "mots-cles": ["accueil", "actualite", "infos"],  
                        "rubrique": "accueil",
```

```
        "page-payant": "0"
    }
});
apntag.defineTag({
    invCode: "test_ias",
    sizes: [[728,90],[1000,300]],
    targetId: "test_ias"
});
//apntag.loadTags();
});
</script>

<!-- START OF IAS PUBLISHER OPTIMIZATION
Analyzis = "iasPET|pub\?anId|ib.adnxs.com/ut/v3"-->
<script async='async'
    src='https://cdn.adsafeprotected.com/iasPET.1.js'></script>
<script>
var IAS_PET_TIMEOUT = 2000;
var adserverRequestSent = false;
var iasDataHandler;
var __iasPET = __iasPET || {};
__iasPET.queue = __iasPET.queue || [];
__iasPET.pubId = 'XXXX';
setTargetingForAst = function(adSlotData) {
    var adSlotDataObj = JSON.parse(adSlotData);
    var apnSlots = apntag.requests.tags;
    var apnKeys = Object.keys(apnSlots);
    var iasTargeting = {};
    iasTargeting = adSlotDataObj.brandSafety;
    iasTargeting.fr = adSlotDataObj.fr;
    for (var i = 0; i < apnKeys.length; i++) {
        apntag.setKeywords(apnKeys[i],iasTargeting);
        apntag.setKeywords(apnKeys[i],adSlotDataObj.slots[apnKeys[i]]);
    }
}
requestAds = function() {
    console.log("Display Ads function");
```

```
        if (adserverRequestSent) return;
        clearTimeout(iasTimeOut);
        apntag.anq.push(function() {
            apntag.loadTags();
        });
        adserverRequestSent = true;
    }
    iasDataHandler = function (adSlotData) {
        console.log("Callback function");
        setTargetingForAst(adSlotData);
        requestAds();
    }
    var iasTimeOut = setTimeout(function() {
        requestAds();
    }, IAS_PET_TIMEOUT);

    apntag.anq.push(function() {
        var apnSlots = apntag.requests.tags;
        var apnKeys = Object.keys(apnSlots);
        var iasPETSlots = [];
        for (var i = 0; i < apnKeys.length; i++) {
            var sizes = apnSlots[apnKeys[i]].sizes;
            iasPETSlots.push({
                adSlotId: apnKeys[i],
                size: sizes,
                adUnitPath: '-'
            });
        }
        __iasPET.queue.push({
            adSlots: iasPETSlots,
            dataHandler: iasDataHandler
        });
    });
</script>
<!-- END OF IAS PUBLISHER OPTIMIZATION -->
</head>
<body>
```

```
<h1>Test IAS</h1>
  <div>
    <div id="test_ias">
      <script type="text/javascript">
        apntag.anq.push(function() {
          //signal to script that this DOM element has
          //been loaded and is ready to be populated with an ad
          apntag.showTag('test_ias');
        });
      </script>
    </div>
  </div>
</body>
</html>
```

Get Response Data

```
{
  "brandSafety": {
    "adt": "veryLow",
    "dlm": "low",
    "drg": "veryLow",
    "alc": "high",
    "hat": "veryLow",
    "vio": "veryLow",
    "off": "veryLow" },
  "custom": {
    "ias-kw": [
      "IAS_12345_123_KW",
      "IAS_23456_PG"
    ]
  },
  "fr": false,
  "slots": {
    "slot-09": {
      "vw": [ "40", "50", "60", "70" ],
      "id": "5848565c-4dd4-11e6-9f0b-0025904ea2be" },
    "slot-321": {
```

```

"vw": [ "40", "50", "60" ],
"id": "1248565c-4dd4-11e6-9f0b-0025904ea2bf" }
}

```

Ad Server

Once you set up the IAS publisher optimization scripts on your pages, you can use the IAS Google Ad Manager or AppNexus key values to target your campaigns.

Google Ad Manager

×
New targeting preset

Video position	✓ All video positions	▼
Inventory	✓ All inventory	▼

Custom targeting
^

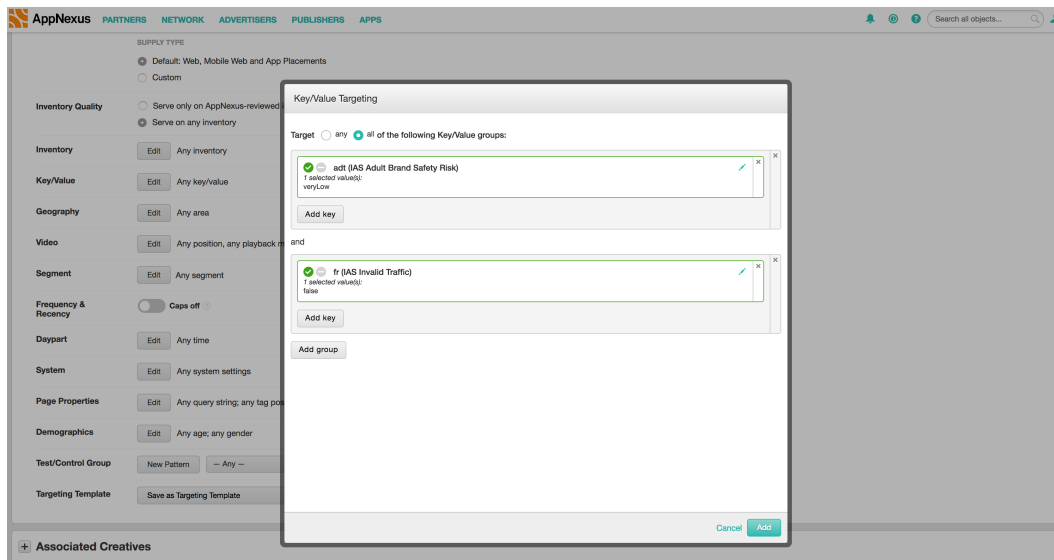
CLEAR

IAS Invalid Traffic (fr) ▼	is any of ▼	false × Search or paste a comma-sep...	⋮	×
IAS Violence Brand Saf... ▼	is any of ▼	veryLow × Search or paste a comm...	⋮	×
IAS Viewability-MRC (v... ▼	is any of ▼	80 × Search or paste a comma-sep...	⋮	×
IAS Keyword Blocking (... ▼	is none of ▼	IAS_123_PG and IAS_UNSCORED_PG ×	⋮	×
		Search or paste a comma-separated list	⋮	×

AND
×

IAS Keyword Blocking (ias-kw)

OR



Best Practices

As a best practice, you should not apply brand safety or viewability targeting to sponsorships or home page take overs.

Debugging

By default publisher optimization does not display any errors in the browser's dev tool console. Add query parameter **iasdebug=true** to the page's URL to enable the debug mode. The console shows errors that specify points of failure.

General

- Publisher optimization uses a Google Publisher Tag (GPT) command queue-like way to push custom configurations for asynchronous loading. Therefore, the order of step 2.a. and 2.b. in the flow chart in "Technical Overview" on page 4 does not matter.
- You do not need to wait for publisher optimization to load before you configure ad slots, or set up publisher optimization.
- Do not send random ad slot IDs to publisher optimization.
- Before setting keyword targeting, make sure the GPT service is available. If you need to set keyword targeting for ad slots, make sure that ad slot is already defined in GPT.
- IAS recommends you do not add publisher verification pixels to empty creatives that contain only a piece of JavaScript code making the parent container collapse. These empty creatives can affect publisher optimization's ability to accurately predict viewability on placements where they are used, since publisher verification can't distinguish between an empty creative and a "real" one. Instead, you should allow the ad server to return a "no fill" response when no creatives are available to fulfill an ad request.