OpenRTB Ad Management API Specification

OpenRTB 3.0 Framework Companion Specification

DRAFT FOR PUBLIC COMMENT
September 2017

Executive Summary

The new IAB Tech Lab Ad Management API Specification is part of the OpenRTB 3.0 Framework release. Comments and feedback should be submitted using this OpenRTB 3.0 Framework Public Comment Submission Form. Public comments are open until December 15, 2017.

Ad management occurs when a buyer (or a representative party) submits creatives for creative approval, and supply platforms approve or disapprove of those creatives. Before the publication of this technical standard, supply platforms have relied on proprietary methods and tools for ad management, or none at all. Creative approval process is necessary for a few reasons; to ensure creatives comply with publisher content guidelines, adhere to brand safety standards, detect malware, and prevent delivery of non-functional ads.

In a quality review process, analysts or automated processes check the ad's landing page, tracking tags, text, the creative itself, and more. The results of these checks are made available for buying platforms to consume and modify bidding behavior accordingly.

The OpenRTB Working Group has identified a need to standardize creative approval process, and to reduce pain points for buyers and sellers in the digital advertising industry. Using a standard approach allows for greater scale for buyers to submit creatives for approval. Supply platforms using the standardized Ad Management API gain control over the creatives that serve on their webpage. Both supply platforms and demand platforms benefit from increased bidding efficiency, as demand platforms can avoid invalid bids and notify customers of defects. The Ad Management API Specification will support multiple scenarios that exist for exchanges with restrictive or permissive bidding, and different markup delivery methods.

About IAB Technology Laboratory

The IAB Technology Laboratory is an independent, international, nonprofit research and development consortium charged with producing and helping companies implement global industry technical standards. Comprised of digital publishers and ad technology firms, as well as marketers, agencies, and other companies with interests in the interactive marketing arena, the IAB Tech Lab's goal is to reduce friction associated with the digital advertising and marketing supply chain, while contributing to the safe and secure growth of the industry. The organization's governing member companies include AppNexus, Extreme Reach, Google, GroupM, Hearst Magazines Digital Media, Integral Ad Science, LinkedIn, Moat, Pandora, PubMatic, Sonobi, Tremor Video, and Yahoo! JAPAN. Established in 2014, the IAB Tech Lab is headquartered in New York City with an office in San Francisco.

Original Author of the Ad Management API Specification Proposal

Ian Trider, Director, RTB Platforms, Centro and IAB Tech Lab Commit Group Member

IAB Tech Lab OpenRTB Commit Group Members

Jim Butler, AOL; Allen Dove, SpotX; Haskell Garon, Google; Curt Larson, Sharethrough; Pierre Nicolas, Criteo; Neal Richter, Rakuten Marketing; Bill Simmons, DataXu; Ian Trider, Centro

IAB Tech Lab OpenRTB Working Group Members
https://iabtechlab.com/working-groups/openrtb-working-group/

IAB Tech Lab Contact
Jennifer Derke, Director of Product, Programmatic & Data, IAB Tech Lab
openrtb@iabtechlab.com

Table of Contents

Introduction and Background

The purpose of this specification is to provide a standardized means for demand and supply partners to communicate with each other regarding ads that will be used in bidding. The specification provides for a RESTful API to be implemented by supply partners.

Suppliers have vastly different policies with regards to creatives, and this specification makes no attempt to enforce any set of business rules. Rather, such rules are a matter for bidders and suppliers to communicate with each other as a part of integration. The terms “required”, “recommended”, and “optional” in this specification refer only to technical compliance, not business policy. Likewise, “must” and “will” refer to requirements for a technically compliant implementation.

Wherever possible, this specification makes reference to OpenRTB as it is assumed this is the standard “common language” of bidding that will be used between suppliers and bidders. However, this specification is not inherently constrained to partners who use OpenRTB during bidding.

Out of convenience, this specification will refer to some shorthand terms for expected common scenarios. These are outlined in the following two sections.

Bidding options

Permissive bidding

In a permissive bidding scenario, the supplier allows a new ad to win impressions until (and if) a point in time occurs at which the ad is deemed to be disapproved. Bidders are expected to stop bidding with disapproved ads, and suppliers may discard bid responses using such a creative.

Restrictive bidding

In a restrictive bidding scenario, the supplier does not allow new ads to win impressions until (and if) a point in time occurs at which the ad is deemed to be approved. Suppliers may discard bids made using such an ad until approval.

A supplier using restrictive bidding may choose to allow new ads to enter its audit queue by simply having the bidder start bidding with such an ad (“submission via bidding”), however bids may be discarded by the supplier until it is approved. Submission via bidding allows bidders to participate in auctions even if they have not implemented support for this API, however optimal performance is ensured by doing so.

Ad serving options

Ad markup in bid response

In this scenario, the bidder is expected to include the markup for an ad (along with its ID) in each bid response.

The ad submitted by the bidder using this API is expected to be a faithful example of creative markup that behaves in a way that is representative of markup that will be used during bidding. It is expected that the exact markup will vary (for example, there will be varying components for an impression tracking URL, cachebusters, click URL, product IDs and images in dynamic creative, etc.). It is up to suppliers to communicate which deviations used during bidding will be deemed to constitute a material change to the creative (and thus may trigger a change in audit status).

Supplier-hosted ad markup

In this scenario, the exchange holds the ad markup. During bidding, the bidder makes reference to the markup on file using the “dadid” or “sadid” field in the bid response (OpenRTB v3.x) or similar (non-OpenRTB) and does not include the actual markup. The supplier provides a means for the bidder to specify custom macros in the markup for which values will be provided at bid time, so that variable components of the markup (impression tracking URLs, cachebusters, click URLs, etc.) may be substituted.

Rationale for this specification

There are a number of reasons why this specification is proposed. Consider the following sets of scenarios that exist for exchanges:

Restrictive bidding, supplier-hosted ad markup

In this scenario, there is an absolute need for an ad management API. This need is currently being filled by proprietary APIs, and DSPs must implement multiple proprietary specs to do business.

Restrictive bidding, ad markup in bid response

In this scenario, when suppliers use “submission via bidding” there is an artificial delay before ads can start to spend -- they are often quarantined and bids for a given ad ID are disallowed from bidding until they have been reviewed by the exchange. The result is that money is “left on the table” while waiting for the audit to clear, campaigns do not begin on time, and (if the exchange and/or DSP is unable or unwilling to implement multiple bids in bid responses) bids are submitted that will be discarded instead of the DSP sending bids for ads that are known to be approved. As a result, the rate of discarded bids can grow to be extremely high resulting in significant revenue loss.

The presence of a proprietary ad management API avoids the pitfalls of submission via bidding, but requires costly engineering resources, requiring demand platforms to maintain multiple implementations for each supply platform.

Permissive bidding, ad markup in bid response

This pattern is the most common among exchanges today. However, there are still problems that an ad management API helps rectify. It is not unusual for an exchange to block certain ad IDs platform-wide as a violation of their ad quality policy or due to technical reasons. This information is not communicated to the DSP, which continues to bid with an ad that will not be accepted. If the exchange and/or DSP is unable or unwilling to implement multiple bids in bid responses, there is a great deal of potential revenue that is lost due to DSP bids being filtered. This is particularly noteworthy with video, where technical problems with ads occur commonly, and as a result, exchanges implement blocks on defective ads. An ad management API makes it possible for the exchange to provide feedback to the DSP which it can incorporate to modify its bidding (e.g. discontinue bidding with ad IDs that the exchange has deemed to be unacceptable to serve).

Emerging formats

New formats are emerging that are beginning to be transactable via real-time bidding. Particularly, digital out-of-home and TV are here or on the near horizon. These new formats often have stringent ad approval requirements that necessitate an API.

API conventions

This API adheres to many of the conventions of RESTful APIs. The base protocol used for communication is HTTPS, and JSON is used to represent the body of requests and responses. Requests should be made with an HTTP header of “Accept: application/json” and “Content-Type: application/json” to indicate that the body of the request will be JSON and that JSON is expected in return.

Almost all fields are optional at a technical level, however suppliers may mandate the presence of certain fields as a business requirement. Both suppliers and bidders must gracefully deal with the presence of unexpected or unknown fields. Breaking changes are restricted to major versions of this specification (1.x, 2.x, etc.).

HTTP status codes are used by the exchange to express the status of requests made by the bidder:

Code	**Name**	**Description**
200	OK	The request was successful.

400	Bad Request	The request could not be interpreted successfully.
401	Unauthorized	The request did not contain correct authentication information.
404	Not Found	The resource does not exist.
429	Too Many Requests	The bidder has exceeded the rate limit set by the supplier and must wait before trying again.
500	Internal Service Error	The supplier has encountered technical difficulties.

Endpoints

Bidders will interact with the Ad Management API by making HTTP calls to specific endpoints. Suppliers will specify a **base URL** (denoted using the {baseUrl} placeholder in this document) and a **bidder ID** representing a given bidder/DSP (denoted using the {bidderId} placeholder in this document). All endpoints are relative to this base URL (indicated with the {baseUrl} placeholder throughout this document), proceeded with the version of the specification in the form of “v#.#”. For example, a supplier may define its base URL as “https://api.exchange.com/management”, in which case, assuming version 1.0 of this specification, and a bidder ID of 492, the ads endpoint will be reached at “https://api.exchange.com/management/v1.0/bidder/492/ads”.

Endpoint	**Methods supported**	**Description**
/bidder/{bidderId}/webhook	GET, POST	GET: returns the webhook registration object for a given bidder. POST: replaces the webhook registration object for a given bidder. PATCH: replaces only the specified fields in the webhook registration object for a given bidder.
/bidder/{bidderId}/ads	GET, POST	GET: returns a collection of ads for a given bidder. POST: submits a collection of ads for a given bidder. Optionally, the ads returned can be filtered to those with audit information last modified at or later than a date/time by including, on the query string, a key-value pair of **lastmod** and a date/time in the format of ISO 8601 (using the profile defined by W3C). For example: **/ads?lastmod=2016-08-14T17:51:54Z**

/bidder/{bidderId}/ads/{id}	GET, PUT, PATCH	A single ad. GET: returns the ad object. PUT: replaces the ad object, and returns the new object (including any fields or child objects provided by the exchange). PATCH: replaces only the specified fields in the ad object, and returns the new object (including any fields or child objects provided by the exchange).

Authentication

HTTP Basic authentication is used. Suppliers will provide bidders with a username and password. This username and password are combined with a colon (username:password) and base64 encoded. The result is used in the Authorization header in all calls the bidder makes to the API, e.g:

`Authorization: Basic` ***<base64 encoded value of "username:password">***

Resource representations

Resources are represented using JSON.

Webhook registration

A webhook registration resource is an object containing details of a bidder webhook which a supplier may use to notify the bidder upon changes to ads.

Attribute	**Type**	**Description**
hookurl	string	The bidder's webhook URL.
username	string	The username that should be used in Authorization headers.
password	string	The password that should be used in Authorization headers.

Collection of ads

A collection of ads is an object containing one or more ads with additional metadata.

Attribute	**Type**	**Description**
ts	string	The current date/time at the supplier (ISO 8601). (Only available on GET)
count	integer	The number of ads in this collection. (Only required on GET)
ads	object array	An array of ad resources.

Ad

An ad resource is an object representing each unique ad that will be used by the bidder. It is a AdCOM 1.0 ad object with relevant child objects. See the AdCOM specification for details.

Only the bidder may modify the ad object, except as noted in AdCOM.

The following AdCOM fields in the Ad object are relevant to this specification, with a list of the fields required, recommended or optional for any given ad.

Type	**Fields**
Required	creative One of: dadid, sadid
Recommended	secure
Optional	cat, lang, events audit (populated by supplier) One of: adomain, bundle

The following AdCOM fields in the Creative object are relevant to this specification:

Type	**Fields**
Required	One of: display, video, audio
Recommended	attr
Optional	iurl

The following AdCOM fields in the Display, Video, and Audio objects are relevant to this specification:

Type	**Fields**
Required	subtype
Recommended	api, mimes One of: adm, curl For display: one set of w and h, or wratio and hratio
Optional	qagmediarating For video and audio: companionad

Audit

Subordinate to the Ad object is a AdCOM 1.0 audit object. See the AdCOM specification for details. Among the many objects that may be present subordinate to the Ad object, it is specifically noted in this specification because its behaviour in the context of ad management is worth elaborating on.

Only the supplier may modify the audit object.

The following AdCOM fields in the audit object are relevant to this specification, with a list of the fields required, recommended or optional for any given audit object.

Type	**Fields**
Required	status, lastmod One of: adm, curl, native
Recommended	corr (if any corrections were made)
Optional	feedback

Webhooks

Suppliers may choose to support sending webhook calls to bidders to notify them of changes to ad audit status, and bidders may choose to receive such calls. Bidders register a webhook URL by manipulating the webhook registration object.

It is recommended to use webhooks as a means of reducing the required polling frequency substantially. Bidders should still poll occasionally to ensure that all changes are collected (to account for failures during webhook calls, etc.), but at a much lower frequency (such as once per day).

Authentication

Authentication is performed in a similar fashion to that described above on the supplier side, with the registered username and password" being used as a part of the Basic auth header when making calls to the bidder, e.g.:

`Authorization: Basic` ***`<base64 encoded value of "username:password">`***

Webhook calls

If a supplier supports webhooks and a bidder has registered a webhook registration object, upon status change for an ad or ad(s), the supplier will POST a collection of ads to the bidder's "hookurl", with an authorization header as described above.The JSON in the POST may be sparse, containing only changes. The bidder will respond 204 No Content if the webhook is successfully processed. In the event of failures such as timeouts, non-2xx status codes, etc., it is recommended the supplier make 3 retries over the span of a minute before abandoning the attempt.

Substitution macros

In the "exchange-hosted ad markup", since bidders do not include their ad markup in the bid response, the exchange must provide the facility for custom macros so that the bidder can provide any variable details (such as cachebusters, impression ID, product IDs, etc.) needed to serve the ad. These custom macros are substituted by the exchange when serving.
All substitution macros in OpenRTB section 4.4 are supported. In addition, bidders may use custom macros in the form of ${NAME} so long as they do not conflict with any reserved values.

The values for these custom macros are supplied by the bidder during bidding. Custom macros will be added to the OpenRTB 3.0 final specification to support. **(FIXME: Needs a proposal to the OpenRTB Working Group to add support for this.)**

Typical synchronization flow

Authentication is not shown here for brevity.

During initial integration, the bidder registers a webhook URL by making an HTTP POST call to:

{baseUrl}/bidder/{bidderId}/webhook

… with a webhook object.

As ads are created, the bidder places the ads into a queue for submission. The bidder makes one or more HTTP POST calls to:

{baseUrl}/{version}/bidder/{bidderId}/ads

… with a body containing a collection of ad resources. Suppliers will periodically update the audit object inside the ad with modifications to the **status** (e.g. to set ads as approved or denied), and any other fields as a result of the supplier's assessment of the ad. Bidders should expect that that fields in the audit object could change at any time.

As ads change audit status, the supply platform makes HTTP POST calls to the hookurl that is registered, containing a collection of Ad objects.

Bidders may also choose to periodically poll for updates (thereby preventing missed information if webhook calls fail) by making HTTP GET calls to:

{baseUrl}/{version}/bidder/{bidderId}/ads?lastmod={timestamp}

… to fetch any changes since the last time they polled for updates. Bidders should record the date/time of their most recent successful poll (based on the **timestamp** returned by the supplier, not local server time to avoid issues of slight differences in system time). This allows for subsequent calls to only be made for any changes since the last call.

Alternatively, bidders may query for the status of a particular ad by making an HTTP GET call to:

{baseUrl}/{version}/bidder/{bidderId}/ads/{id}

Bidders should make use of the information they receive to change their bidding behaviour appropriately.

If the bidder has made local changes to an ad, the bidder makes an HTTP PATCH or PUT call to:

{baseUrl}/{version}/bidder/{bidderId}/ads/{id}

… to update the ad record at the supplier. On a supplier with restrictive bidding, typically, this would result in the ad returning to status 1, "pending audit", but this is a matter of supplier business rules. For example, a supplier might deem only a subset of changes to be significant enough to require a re-audit (such as a change in landing page domain). It is up to suppliers to communicate what deviations constitute a material change to the ad (and thus may trigger a change in audit status).

Appendix A: Integration checklist

To facilitate integration, exchanges should provide a document similar to the below to inform bidders about the specifics of an exchange's implementation of this spec.

Exchange	
Base URL	
Bidder ID	
Version Used	
Rate Limit	
Policy	
Attributes required[1]	

Notes

[1] Here, this refers to a business requirement of the supply platform. It is assumed that all technically required fields will be present.

Appendix B: Examples

Minimal implementation

Exchange	SuperAds
Base URL	https://api.superads.com/management
Bidder ID	496
Version Used	v0.1
Rate Limit	100 requests per minute
Policy	Permissive bidding, ad markup in bid response
Attributes	Required in Ad object: adomain Required in Creative object: iurl, display Required in Display object: w, h All others will be ignored

Bidder ad submission

POST https://api.superads.com/management/v0.1/bidder/496/ads

```
{
  "count": 1,
  "ads": [
    {
      "dadid": "557391",
      "adomain": "advertiser.com",
      "creative": {
        "iurl": "http://cdn.dsp.com/iurls/557391.gif",
        "display": {
          "w": 300,
          "h": 250
        }
      }
    }
  ]
}
```

Response:

```
{
  "count": 1,
  "ads": [
    {
      "dadid": "557391",
      "adomain": "advertiser.com",
      "creative": {
        "iurl": "http://cdn.dsp.com/iurls/557391.gif",
        "display": {
          "w": 300,
          "h": 250
        }
      },
      "audit": {
        "status": 2,
        "lastmod": "YYYY-MM-DDTHH:MM:SSZ"
      }
    }
  ]
}
```

Bidder receives a webhook call from supplier

In this case, notifying it that an ad has been disapproved.

POST <hookurl>

```
{
  "count": 1,
  "ads": [
    {
      "dadid": "557391",
      "audit": {
        "status": 4,
        "feedback": "Content disallowed by exchange policy.",
        "lastmod": "YYYY-MM-DDTHH:MM:SSZ"
      }
    }
  ]
}
```

Bidder polls for updates

GET https://api.superads.com/management/v0.1/bidder/496/ads?lastmod=YYYY-MM-DDTHH:MM:SSZ

```
{
  "ts": "YYYY-MM-DDTHH:MM:SSZ",
  "count": 27,
  "ads": [
    {
      "dadid": "557391",
      "adomain": "advertiser.com",
      "creative": {
        "iurl": "http://cdn.dsp.com/iurls/557391.gif",
        "display": {
          "w": 300,
          "h": 250
        }
      },
      "audit": {
        "status": 3,
        "lastmod": "YYYY-MM-DDTHH:MM:SSZ"
      }
    },
    { ... }
  ]
}
```

OR

```
{
  "ts": "YYYY-MM-DDTHH:MM:SSZ",
  "count": 27,
  "ads": [
    {
      "dadid": "557391",
      "adomain": "advertiser.com",
      "creative": {
        "iurl": "http://cdn.dsp.com/iurls/557391.gif",
        "display": {
          "w": 300,
          "h": 250
        }
      },
      "audit": {
        "status": 4,
        "feedback": "Content disallowed by exchange policy."
        "lastmod": "YYYY-MM-DDTHH:MM:SSZ"
      }
    },
    { ... }
  ]
}
```

Moderately elaborate implementation

Exchange	AdvancedAds
Base URL	https://api.advancedads.com/management
Bidder ID	496
Version Used	v0.1
Rate Limit	100 requests per minute
Policy	Restrictive bidding, ad markup in bid response
Attributes	Required in Ad object: adomain Required in Creative object: one of display, video Required in Display: adm, one of set w and h or wratio and hratio Required in Video: adm or curl, api, mimes

Bidder ad submission

POST https://api.advancedads.com/management/v0.1/bidder/496/ads

```
{
  "count": 1,
  "ads": [
    {
      "dadid": "557391",
      "cat": [
          "IAB17"
      ],
      "adomain": "advertiser.com",
      "creative": {
        "display": {
          "w": 300,
          "h": 250,
          "adm": "<script src=\"...\"></script>"
        }
      }
    }
  ]
}
```

Response:

```
{
  "count": 1,
  "ads": [
    {
      "dadid": "557391",
      "cat": [
        "IAB17"
      ],
      "adomain": "advertiser.com",
      "creative": {
        "display": {
          "w": 300,
          "h": 250,
          "adm": "<script src=\"...\"></script>"
        }
      },
      "audit": {
        "status": 1,
        "lastmod": "YYYY-MM-DDTHH:MM:SSZ"
      }
    }
  ]
}
```

Bidder polls for updates

GET https://api.advancedads.com/management/v0.1/bidder/496/ads?lastmod=YYYY-MM-DDTHH:MM:SSZ

```
{
  "timestamp": "YYYY-MM-DDTHH:MM:SSZ",
  "count": 27,
  "ads": [
    {
      "dadid": "557391",
      "cat": [
        "IAB17"
      ],
      "adomain": "advertiser.com",
      "creative": {
        "display": {
          "w": 300,
          "h": 250,
          "adm": "<script src=\"...\"></script>"
        }
      },
      "audit": {
        "status": 3,
        "feedback": "Corrected category. Added missing attribute \"In-
Banner Video (Automatic)\".",
        "lastmod": "YYYY-MM-DDTHH:MM:SSZ",
        "corr": {
          "ad": {
            "cat": [
              "IAB20"
            ],
            "creative": {
              "attr": [
                6
              ]
            }
          }
        }
      }
    },
    { ... }
  ]
}
```

In the above scenario, the supplier has approved the ad but has updated it to reflect that, in the its opinion, the ad should be classified as category Travel, and has auto-play in-banner video. The equivalent webhook call would be similar (see above example).

Appendix A. Resources
OpenRTB 3.0 Framework
AdCOM