NEW FEATURES HAVE BEEN RELEASED.

The latest version of the specification can be found in the 2.x GitHub repository located here: https://github.com/InteractiveAdvertisingBureau/openrtb2.x

iab. TECH LAB

OpenRTB
Version 2.6

Released April 2022

Drafted by the Programmatic Supply Chain Working Group

Please email support@iabtechlab.com with feedback or questions. This document is available online at https://iabtechlab.com/standards/openrtb

Introduction

The RTB Project, formerly known as the OpenRTB Consortium, assembled in November 2010 to develop a new API specification for companies interested in an open protocol for the automated trading of digital media across a broader range of platforms, devices, and advertising solutions. This document is OpenRTB version 2.6 released for public comment in December 2021; this is the culmination of the working group efforts and can be found at: http://www.iabtechlab.com/openrtb

About the IAB Technology Lab

The IAB Technology Laboratory is a nonprofit research and development consortium charged with producing and helping companies implement global industry technical standards and solutions. The goal of the Tech Lab is to reduce friction associated with the digital advertising and marketing supply chain while contributing to the safe growth of an industry. The IAB Tech Lab spearheads the development of technical standards, creates and maintains a code library to assist in rapid, cost-effective implementation of IAB standards, and establishes a test platform for companies to evaluate the compatibility of their technology solutions with IAB standards, which for 18 years have been the foundation for interoperability and profitable growth in the digital advertising supply chain. Further details about the IAB Technology Lab can be found at https://iabtechlab.com.

Significant Contributors to the 2.6 version specification

Stanislav Belov, Software Engineer, Google; Allen Dove, CTO, Magnite; Steven Katz, Senior Principal Architect, Yahoo!; Curt Larson, Chief Product Officer, Sharethrough; Dr. Neal Richter, Director of Science, Amazon Advertising; Jud Spencer, Principal Software Engineer, The Trade Desk; Ian Trider, VP, RTB Platform Operations, Basis Technologies; Rob Hazan, Sr. Director Product Management, Index Exchange; James Wilhite, VP Product, Publica; Jean-Luc Wasmer, VP Partnership Operations, Triton Digital; Sam Mansour, Principle Product Manager, Oracle; Scott Kay, Engineering Manager, Xandr; Emma Fenlon, Sr. Manager, Exchange Quality, Yahoo!; Liam Whiteside, Head of Ad Technology, Global; Don Marti, VP Ecosystem Innovation, Café Media; Aron Schatz, Director Product Management, Double Verify; Jake Jolly, Product Manager, Google; Amit Shetty, VP Product, Pixalate

IAB Tech Lab Lead:

Jill Wittkopp, Sr. Director of Product, IAB Tech Lab

License

Disclaimer

THE STANDARDS, THE SPECIFICATIONS, THE MEASUREMENT GUIDELINES, AND ANY OTHER MATERIALS OR SERVICES PROVIDED TO OR USED BY YOU HEREUNDER (THE "PRODUCTS AND SERVICES") ARE PROVIDED "AS IS" AND "AS AVAILABLE," AND IAB TECHNOLOGY LABORATORY, INC. ("TECH LAB") MAKES NO WARRANTY WITH RESPECT TO THE SAME AND HEREBY DISCLAIMS ANY AND ALL EXPRESS, IMPLIED, OR STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AVAILABILITY, ERROR-FREE OR UNINTERRUPTED OPERATION, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, COURSE OF PERFORMANCE, OR USAGE OF TRADE. TO THE EXTENT THAT TECH LAB MAY NOT AS A MATTER OF APPLICABLE LAW DISCLAIM ANY IMPLIED WARRANTY, THE SCOPE AND DURATION OF SUCH WARRANTY WILL BE THE MINIMUM PERMITTED UNDER SUCH LAW. THE PRODUCTS AND SERVICES DO NOT CONSTITUTE BUSINESS OR LEGAL ADVICE. TECH LAB DOES NOT WARRANT THAT THE PRODUCTS AND SERVICES PROVIDED TO OR USED BY YOU HEREUNDER SHALL CAUSE YOU AND/OR YOUR PRODUCTS OR SERVICES TO BE IN COMPLIANCE WITH ANY APPLICABLE LAWS, REGULATIONS, OR SELF-REGULATORY FRAMEWORKS, AND YOU ARE SOLELY RESPONSIBLE FOR COMPLIANCE WITH THE SAME, INCLUDING, BUT NOT LIMITED TO, DATA PROTECTION LAWS, SUCH AS THE PERSONAL INFORMATION PROTECTION AND ELECTRONIC DOCUMENTS ACT (CANADA), THE DATA PROTECTION DIRECTIVE (EU), THE E-PRIVACY DIRECTIVE (EU), THE GENERAL DATA PROTECTION REGULATION (EU), AND THE E-PRIVACY REGULATION (EU) AS AND WHEN THEY BECOME EFFECTIVE.

TABLE OF CONTENTS

Introduction *1*
About the IAB Technology Lab *1*
License *1*
Disclaimer *2*

GETTING STARTED **1**

1. INTRODUCTION **2**

1.1 MISSION / OVERVIEW 2
1.2 HISTORY OF OPENRTB 2
1.3 VERSION HISTORY 3
OpenRTB Real-Time Bidding API *3*
OpenRTB Display Block List Branch *3*
1.4 RESOURCES 3
1.5 TERMINOLOGY 3

2. OPENRTB BASICS **4**

2.1 TRANSPORT 5
2.2 SECURITY 5
2.3 DATA FORMAT 5
2.4 DATA ENCODING 6
2.5 OPENRTB VERSION HTTP HEADER 6
2.6 VERSIONING BEHAVIOR 7
2.7 PRIVACY 8
2.8 RELATIONSHIP TO BRAND SAFETY CERTIFIED 8
2.9 CUSTOMIZATION AND EXTENSIONS 8

3. BID REQUEST SPECIFICATION **9**

3.1 OBJECT MODEL 9
3.2 OBJECT SPECIFICATIONS 11
3.2.1 Object: BidRequest *11*
3.2.2 Object: Source *13*
3.2.3 Object: Regs *13*
3.2.4 Object: Imp *14*
3.2.5 Object: Metric *15*
3.2.6 Object: Banner *15*
3.2.7 Object: Video *17*
3.2.8 Object: Audio *19*
3.2.9 Object: Native *21*
3.2.10 Object: Format *22*
3.2.11 Object: Pmp *23*
3.2.12 Object: Deal *23*
3.2.13 Object: Site *24*
3.2.14 Object: App *25*
3.2.15 Object: Publisher *26*
3.2.16 Object: Content *26*
3.2.17 Object: Producer *28*
3.2.18 Object: Device *28*
3.2.19 Object: Geo *30*
3.2.20 Object: User *31*

3.2.21 Object: Data *32*
3.2.22 Object: Segment *32*
3.2.23 Object: Network *32*
3.2.24 Object: Channel *33*
3.2.25 Object: SupplyChain *33*
3.2.26 Object: SupplyChainNode *34*
3.2.27 Object: EID *35*
3.2.28 Object: UID *35*
3.2.29 Object: UserAgent *35*
3.2.30 Object: BrandVersion *36*

4 BID RESPONSE SPECIFICATION 36

4.2 OBJECT MODEL 36
4.3 OBJECT SPECIFICATIONS 38
4.3.1 Object: BidResponse *38*
4.3.2 Object: SeatBid *39*
4.3.3 Object: Bid *39*
4.4 AD SERVING OPTIONS 42
4.4.1 Markup Served on the Win Notice *42*
4.4.2 Markup Served in the Bid *42*
4.4.3 Comparison of Ad Serving Approaches *42*
Ad Served on the Win Notice *42*
Ad Served in the Bid *43*
4.5 SUBSTITUTION MACROS 43
4.5.1 Notes on the macro ${AUCTION_MIN_TO_WIN} *44*

5 ENUMERATED LISTS SPECIFICATION 45

6 BID REQUEST/RESPONSE SAMPLES 46

6.3 GITHUB REPOSITORY 46
6.4 BID REQUESTS 46
6.4.1 Example 1 – Simple Banner *46*
6.4.2 Example 2 – Expandable Creative *47*
6.4.3 Example 3 – Mobile *49*
6.4.4 Example 4 – Video *51*
6.4.5 Example 5 – PMP with Direct Deal *54*
6.4.6 Example 6 – Native Ad *55*
6.5 BID RESPONSES 56
6.5.1 Example 1 – Ad Served on Win Notice *56*
6.5.2 Example 2 – VAST XML Document Returned Inline *57*
6.5.3 Example 3 – Direct Deal Ad Served on Win Notice *58*
6.5.4 Example 4 – Native Markup Returned Inline *59*

7 IMPLEMENTATION NOTES 59

7.3 NO-BID SIGNALING 59
7.4 IMPRESSION EXPIRATION 60
7.5 PMP & DIRECT DEALS 62
Best Practice Bidding Logic *62*
Warning *62*
Policy Recommendations *62*
Anti-Patterns *62*
Subjecting Deal ID Bids to an internal auction on price *62*
Associating Deal ID to the wrong Object *63*

Improper Handling of the Private vs Open Market Flag *63*
Improper handling of Seat IDs *63*
Silently Applying Margin Discounts to Deal ID Bids *63*
Use cases *64*
7.6 SKIPPABILITY 65
Bid Request *66*
Bid Response *66*
7.5 REGS RESOURCES 68
7.6 POD BIDDING FOR VIDEO AND AUDIO 68
Terminology *68*
Recommendations *69*
Pod bidding example scenarios *69*
7.7 NETWORK VS CHANNEL EXAMPLE CASES 77
7.8 COUNTING BILLABLE EVENTS AND TRACKED ADS 78

OVERVIEW OF COUNTING METHODOLOGIES 78

PIXEL IN MARKUP (BANNER ADS) 78
VAST <IMPRESSION> EVENT (VIDEO/AUDIO) 79
BILLING NOTICE ("BURL") 79
NATIVE EVENTTRACKERS, IMPTRACKERS, JSTRACKERS 80
WIN NOTICE ("NURL") – NOT A BILLABLE OR TRACKED AD EVENT 80

BEST PRACTICES FOR SERVER-SIDE BILLING NOTIFICATIONS 80

APPENDIX A. ADDITIONAL INFORMATION 82

APPENDIX B. SPECIFICATION CHANGE LOG 83

Getting Started

This specification contains a detailed explanation of an RTB (Real-Time Bidding) interface. Not all objects are “required,” and each object may contain several optional parameters. As that term is used herein, “required” attributes are designated as such if their omission would technically break the protocol. This means that it is technically possible for two implementers to conduct a minimally viable transaction if all required attributes are included, and impossible without them. This does not, however, mean that including only required attributes will be satisfactory to implementers for business reasons. For example, a request with only required attributes does not supply any context about what is being bid on, such as whether it is for a site or app, or which site/app it is for. While the transaction is technically possible, in practice, market participants will likely find this is to be unsatisfactory.

Some optional attributes are denoted "recommended". As that term is used herein “recommended” means that their inclusion is customary and, thus, likely to be expected by most implementers. While they are not required for a minimally viable transaction to occur, the market norm is to include such attributes; implementers may be unwilling to transact unless these attributes are included. The designation as "recommended" is based on awareness of these market norms and should not be interpreted as a suggestion by IAB Tech Lab that an implementer propagate any specific attribute. Implementers should determine which attributes they will make available based on discussion with their business partners, and in a manner consistent with applicable law.

	Attribute	Type	Description
Examples of required attributes. Grouped at the tops of tables for convenience.	`id`	string; required	. . .
	`imp`	object array; required	. . .
Examples of recommended attributes. Grouped after required attributes.	`site`	object; recommended	. . .
	`app`	object; recommended	. . .
Examples of optional attributes, with and without defaults. Attributes are assumed optional unless explicitly qualified as required or recommended.	`test`	integer; default 0	. . .
	`at`	integer; default 2	. . .
	`tmax`	integer	. . .
	`wseat`	string array	. . .

Figure 1: Example of how Required, Recommended, and Optional attributes are presented.

IMPORTANT: Since **recommended** attributes are not required, they may not be available from all supply sources. It is suggested that all parties to OpenRTB transaction develop an integration checklist to identify which attributes the supply side supports in the bid request, and which attributes the demand side requires for ad decisioning.

1. Introduction

1.1 Mission / Overview

The mission of OpenRTB is to spur growth in Real-Time Bidding (RTB) marketplaces by providing open industry standards for communication between buyers of advertising and sellers of publisher inventory. There are several aspects to these standards including but not limited to the actual real-time bidding protocol, information taxonomies, offline configuration synchronization, and many more.

This document specifies a standard for the Real-Time Bidding Interface that grew out of previous OpenRTB collaboration on the "block list project" and the "OpenRTB Mobile" project. These protocol standards aim to simplify the connection between suppliers of publisher inventory (i.e., exchanges, networks working with publishers, and sell-side platforms) and competitive buyers of that inventory (i.e., bidders, demand side platforms, or networks working with advertisers).

The overall goal of OpenRTB is and has been to create a *lingua franca* for communicating between buyers and sellers. The intent is not to regulate exactly how each business operates. It aims to make integration between parties easier, so that innovation can happen at a deeper-level at each of the businesses in the ecosystem.

1.2 History of OpenRTB

OpenRTB was launched as a pilot project between three demand-side platforms (DataXu, MediaMath, and Turn) and three sell-side platforms (Admeld, PubMatic, and The Rubicon Project) in November 2010. The first goal was to standardize communication between parties for exchanging block lists. Version 1.0 of the OpenRTB block list specification was released in December 2010.

After a positive response from the industry, Nexage approached the OpenRTB project with a proposal to create an API specification for OpenRTB focusing on the actual real-time bid request/response protocol and specifically to support mobile advertising. The mobile subcommittee was formed between companies representing the buy-side (DataXu, Fiksu, and [X+1]) and companies representing the sell-side (Nexage, Pubmatic, Smaato, and Jumptap). This project resulted in the OpenRTB Mobile 1.0 specification, which was released in February 2011.

Following the release of the mobile specification, a video subcommittee was formed with video ad exchanges (BrightRoll and Adap.tv) collaborating with DataXu and ContextWeb to incorporate support for video. The goal was to incorporate support for display, video, and mobile in one document. This effort resulted in OpenRTB 2.0, which was released as a unified standard in June 2011.

Due to very widespread adoption by the industry, OpenRTB was adopted as an IAB Tech Lab standard in January 2012 with the release of version 2.1. Governance over the technical content of the specification remains with the OpenRTB community and its governance rules.

1.3 Version History

OpenRTB Real-Time Bidding API

2.6 Introduces Ad Pods for CTV transactions, a structured User-Agent object and other minor enhancements.
2.5 Support for header bidding, billing and loss notifications, Flex Ads, Payment ID, tactic ID, impression metrics, out-stream video, and many more minor enhancements.
2.4 Support for Audio ad units and the largest set of minor to moderate enhancements in v2.x history.
2.3 Support for Native ad units and multiple minor enhancements.
2.2 New enhancements for private marketplace direct deals, video, mobile, and regulatory signals.
2.1 Revisions for IQG compliance, minor enhancements, and corrections.
2.0 Combines display, mobile, and video standards into a unified specification.
1.0 Original Release of OpenRTB Mobile.

OpenRTB Display Block List Branch

1.2 Publisher Preferences API (proposed).
1.1 Minor edits to include real-time exchange of creative attributes.
1.0 Original Release of OpenRTB block list specifications.

1.4 Resources

OpenRTB GitHub Repository https://github.com/InteractiveAdvertisingBureau/openrtb

Updates to the specification are made through the IAB Tech Lab's Programmatic Supply Chain Working Group. Contact techlab@iabtechlab.com to join the working group and participate in Slack discussions.

1.5 Terminology

The following terms are used throughout this document specifically in the context of the OpenRTB Interface and this specification.

Term	Definition
RTB	Bidding for individual impressions in real-time (i.e., while a consumer is waiting).
Exchange	A service that conducts an auction among bidders per impression.
Bidder	An entity that competes in real-time auctions to acquire impressions.
Seat	An advertising entity (e.g., advertiser, agency) that wishes to obtain impressions and uses bidders to act on their behalf; a customer of a bidder and usually the owner of the advertising budget.
Publisher	An entity that operates one or more sites.
Site	Ad supported content including web and applications unless otherwise specified.
Deal	A pre-arranged agreement between a Publisher and a Seat to purchase impressions under certain terms.

2. OpenRTB Basics

The following figure illustrates the OpenRTB interactions between an exchange and its bidders. Ad requests originate at publisher sites or applications. For each inbound ad request, bid requests are broadcast to bidders, responses are evaluated under prevailing auction rules, and a winner is selected. The winning bidder is notified of the auction win via a win notice. Ad markup can either be included in the bid prospectively or in response to the win notice. A separate billing notice is also available to accommodate varying policies enacted by exchanges which are beyond the scope of the OpenRTB specification (e.g., billing on device delivery, viewability, etc.). The win notice informs the bidder's pricing algorithms of a success, whereas the billing notice indicates that spend should actually be applied. A loss notification is also available to inform the bidder of the reason their bid did not win.

The URLs for win, billing, and loss notices and the ad markup itself can contain any of several standard macros that enable the exchange to communicate critical data to the bidder (e.g., clearing price).

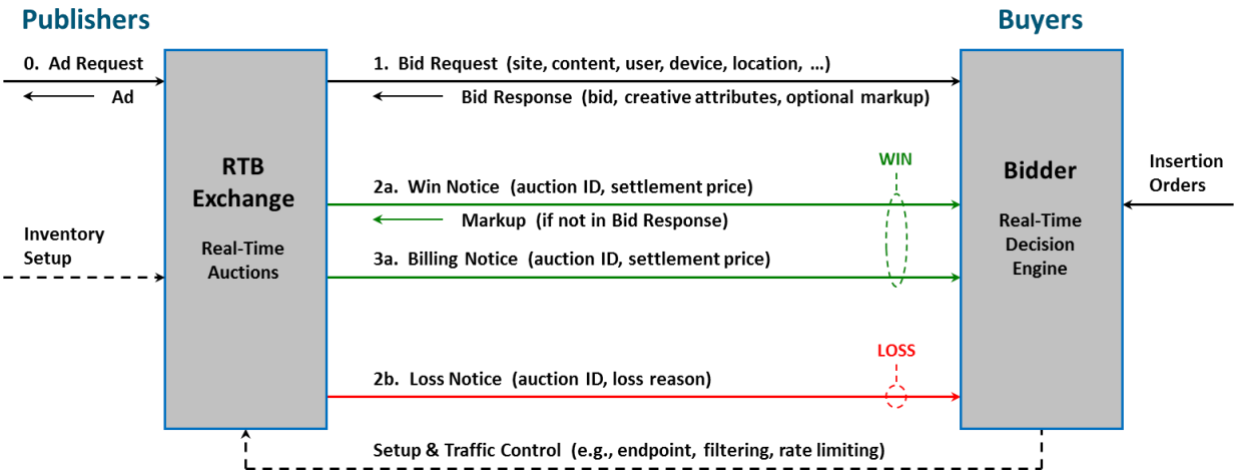


Figure 2: Reference Model - Request Sequence.

This specification focuses on the real-time interactions of bid request and response and the win, billing, and loss notices. Related specifications include:

- AdCOM: The Advertising Common Object Model, a companion specification that defines common objects and values expected to be used across many IAB Tech Lab spec. As of OpenRTB 2.6, all enumerated lists used by OpenRTB 2.6 are kept in AdCOM to enable more rapid iteration.
- OpenRTB Ad Management API: A companion specification for facilitating creative review between bidders and exchanges.

Other interactions (e.g., block list synchronization, traffic control, etc.) are candidates for future OpenRTB initiatives or alternate projects.

2.1 Transport

The base protocol between an exchange and its bidders is HTTP. Specifically, HTTP POST is required for bid requests to accommodate greater payloads than HTTP GET and facilitate the use of binary representations. Win notices may be either POST or GET at the discretion of the exchange.

Calls returning content (e.g., any bid response, a win notice that returns markup) should return HTTP code 200. Calls returning no content in response to valid requests (e.g., an empty bid response which is one option for indicating no-bid, a win notice that does not return markup) should return HTTP 204. Invalid calls (e.g., a bid request containing a malformed or corrupt payload) should return HTTP 400 with no content.

BEST PRACTICE: One of the simplest and most effective ways of improving connection performance is to enable HTTP Persistent Connections, also known as Keep-Alive. This has a profound impact on overall performance by reducing connection management overhead as well as CPU utilization on both sides of the interface.

2.2 Security

HTTPS (i.e., secure HTTP) is not required for OpenRTB compliance. However, there is a growing trend in the industry to use HTTPS for added security of exchange/bidder communications. It is recommended, therefore, that exchanges and bidders consider supporting both HTTP and HTTPS.

2.3 Data Format

JSON (JavaScript Object Notation) is the suggested format for bid request and bid response data payloads. JSON was chosen for its combination of human readability and compactness. The data payloads are described in Section 3 and Section 4.

Optionally, an exchange may also offer binary representations (e.g., compressed JSON, ProtoBuf, Avro, etc.), which can be more efficient in terms of transmission time and bandwidth. The IAB Tech Lab may offer reference implementations for these or other formats. When available, the use of these IAB reference implementations is highly recommended to reduce exchange-specific variations.

The bid request specifies the representation as a mime type using the Content-Type HTTP header. The mime type for the standard JSON representation is “application/json” as shown. The format of the bid response must be the same as the bid request.

```
Content-Type: application/json
```

If alternative binary representations are used, the exchange or SSP should specify the Content-Type appropriately. For example: “Content-Type: avro/binary” or “Content-Type: application/x-protobuf”. If the content-type is missing, the bidder should assume the type is application/json, unless a different default has been selected by an exchange.

As a convention, the absence of an attribute has a formal meaning. In most cases, this indicates that the value is unknown, unless otherwise specified.

2.4 Data Encoding

Compressing data sent between exchanges and bidders can be very beneficial. Compression greatly reduces the size of data transferred and thus saves network bandwidth for both exchanges and bidders. To realize these savings fully, compression should be enabled for both the bid request sent by the exchange and the bid response returned by the bidder.

Compression can be enabled on the bid response using standard HTTP 1.1 mechanisms. Most webservers already support gzip compression of response content and as such it is an ideal choice. For an exchange to signal they would like the response to be compressed, it should set the standard HTTP 1.1 Accept-Encoding header. The encoding value used should be “gzip”.

```
Accept-Encoding: gzip
```

This header represents to bidders an indication by the exchange that it is capable of accepting gzip encoding for the response. If the bidder server supports this and is correctly configured, it will automatically respond with content that is gzip encoded. This will be indicated using the standard HTTP 1.1 Content-Encoding header.

```
Content-Encoding: gzip
```

To enable compression on the bid request, it must first be agreed upon between the exchange and the bidder that this is supported. This is similar to when a custom data format is used since the exchange has to know both format and encoding before sending the bid request. If the bidder supports it, the exchange should indicate it is sending a gzip compressed bid request by setting the HTTP 1.1 Content-Encoding header. The encoding value used should be “gzip”.

```
Content-Encoding: gzip
```

If this header is not set then it is assumed that the request content isn’t encoded. In HTTP 1.1, the Content-Encoding header is usually only used for response content. However, by using this header for the request content as well we are able to indicate a request is compressed regardless of the data format used. This is useful since even binary data formats can benefit from being compressed.

2.5 OpenRTB Version HTTP Header

The OpenRTB Version should be passed in the header of a bid request with a custom header parameter. This will allow bidders to recognize the version of the message contained before attempting to parse the request.

Additionally, it is recommended albeit optional that bidders place an identically formatted message in the HTTP header of the response with the protocol version the bidder has implemented. The message may contain a different version number than the request header.

x-openrtb-version: 2.6

This version should be specified as <major>.<minor> (e.g., 2.6).

2.6 Versioning Behavior

Breaking changes are not permitted without a major version increment. The minor version number increments are reserved for non-breaking changes. This means that implementers need not specifically treat the minor version of OpenRTB 2.x as significant, in most circumstances. For example:

- Bidders and exchanges must tolerate receiving new or unexpected fields and enumerated list values gracefully, treating them as unknown or ignoring them
- Bidders and exchanges should freely transmit new fields or enumerated list values

For example, if a new field or object is introduced in a future version of OpenRTB 2.x, exchanges should immediately start transmitting it to bidders, if the exchange chooses to implement support for that field. Bidders should start consuming it at their discretion, if it is relevant to them. Neither party needs to explicitly negotiate an "upgrade" to the latest minor version number, but rather should discuss specific fields of interest as they become available.

New minor versions of OpenRTB 2.x may introduce additional optional ways of handling things. For example, *burl* field was introduced in OpenRTB 2.5. Use of *burl* (instead of including a pixel in markup in *adm* field) is a breaking change for a specific exchange <> bidder integration, but this is a result of a decision between these parties to switch impression counting methodology and not a result of OpenRTB 2.5 itself. Partners should discuss such situations before making breaking changes to their integrations.

2.7 Privacy

Without limiting the scope of the Disclaimer, the OpenRTB specification does include several features to support implementer's communication of information regarding consumer privacy, including, but not limited to:

- The status of do not track headers (Section 3.2.18)
- Presence of site/app privacy policies (Section 3.2.13 and Section 3.2.14)
- Regulations or laws that the transmitter of a bid request believes are applicable (Section 3.2.3)
- Consent and opt-out information strings for the user (US Privacy String and Transparency and Consent Framework consent strings) (Section 3.2.3 and Section 3.2.20)

As noted in the Disclaimer, each implementer is responsible to ensure that their implementation complies with any applicable laws, regulations, or self-regulatory frameworks.

2.8 Relationship to Brand Safety Certified

OpenRTB is fully compatible with the Brand Safety Certified available here: https://www.tagtoday.net/brand-safety .

2.9 Customization and Extensions

The OpenRTB spec allows for exchange specific customization and extensions of the specification. Any object may contain extensions. To keep extension fields consistent across platforms, they should consistently be named "`ext`".

In order to better support common and well known extensions, IAB Tech Lab is now hosting these extensions at our Github repository : https://github.com/InteractiveAdvertisingBureau/openrtb/tree/master/extensions.

3. Bid Request Specification

RTB transactions are initiated when an exchange or other supply source sends a bid request to a bidder. The bid request consists of the top-level bid request object, at least one impression object, and may optionally include additional objects providing impression context.

3.1 Object Model

Following is the object model for the bid request. The top-level object (i.e., in JSON the unnamed outer object) is denoted as `BidRequest` in the model. Of its direct subordinates, only `Imp` is technically required since it is fundamental to describing the impression being sold and it requires at least one of `Banner` (which may allow multiple formats), `Video`, `Audio`, and `Native` to define the type of impression (i.e., whichever one or more the publisher is willing to accept; although a bid will be for exactly one of those specified). An impression can optionally be subject to a private marketplace.

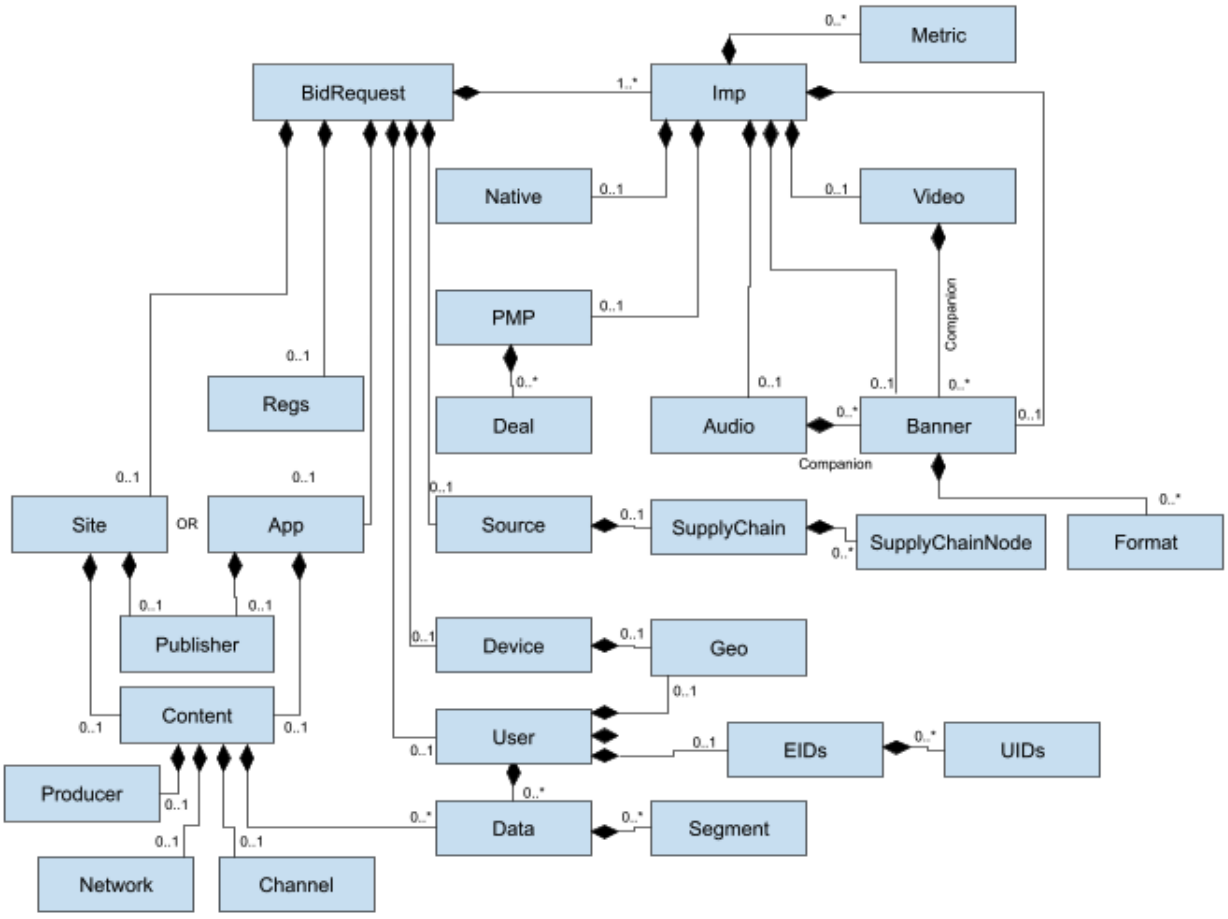


Figure 3: Bid Request object model.

Other subordinates to the `BidRequest` provide various forms of information to assist bidders in making targeting and pricing decisions.

Not shown in the model figure is an extensions object. This is an object of undefined structure that can be added to any other object to convey exchange-specific extensions to the standard. Exchanges using these objects are responsible for publishing their extensions to their bidders.

The following table summarizes the objects in the Bid Request model and serves as an index into the detailed definitions in the subsections that follow.

Object	Section	Description
`BidRequest`	3.2.1	Top-level object.
`Source`	3.2.2	Request source details on post-auction decisioning (e.g., header bidding).
`Regs`	3.2.3	Regulatory conditions in effect for all impressions in this bid request.
`Imp`	3.2.4	Container for the description of a specific impression; at least 1 per request.
`Metric`	3.2.5	A quantifiable often historical data point about an impression.
`Banner`	3.2.6	Details for a banner impression (incl. in-banner video) or video companion ad.
`Video`	3.2.7	Details for a video impression.
`Audio`	3.2.8	Container for an audio impression.
`Native`	3.2.9	Container for a native impression conforming to the Dynamic Native Ads API.
`Format`	3.2.10	An allowed size of a banner.
`Pmp`	3.2.11	Collection of private marketplace (PMP) deals applicable to this impression.
`Deal`	3.2.12	Deal terms pertaining to this impression between a seller and buyer.
`Site`	3.2.13	Details of the website calling for the impression.
`App`	3.2.14	Details of the application calling for the impression.
`Publisher`	3.2.15	Entity that controls the content of and distributes the site or app.
`Content`	3.2.16	Details about the published content itself, within which the ad will be shown.
`Producer`	3.2.17	Producer of the content; not necessarily the publisher (e.g., syndication).
`Device`	3.2.18	Details of the device on which the content and impressions are displayed.
`Geo`	3.2.19	Location of the device or user's home base depending on the parent object.
`User`	3.2.20	Human user of the device; audience for advertising.
`Data`	3.2.21	Collection of additional user targeting data from a specific data source.
`Segment`	3.2.22	Specific data point about a user from a specific data source.
`Network`	3.2.23	Network on which an ad will be displayed.
`Channel`	3.2.24	Channel on which an ad will be displayed.
`SupplyChain`	3.2.25	Chain of nodes from beginning to end representing the entities involved in the direct flow of payment for inventory.
`SupplyChainNode`	3.2.26	Nodes defining the identity of an entity participating in the supply chain of a bid request.
`EIDs`	3.2.27	Extended identifiers support
`UIDs`	3.2.28	Extended identifier user id support
`UserAgent`	3.2.29	Structured user agent information

BrandVersion	3.2.30	Name and version information for a browser or similar software component

3.2 Object Specifications

3.2.1 Object: BidRequest

The top-level bid request object contains a globally unique bid request or auction ID. This `id` attribute is required as is at least one impression object (Section 3.2.4). Other attributes in this top-level object establish rules and restrictions that apply to all impressions being offered.

There are also several subordinate objects that provide detailed data to potential buyers. Among these are the `Site` and `App` objects, which describe the type of published media in which the impression(s) appear. These objects are highly recommended, but only one applies to a given bid request depending on whether the media is browser-based web content or a non-browser application, respectively.

Attribute	Type	Description
id	string; required	Unique ID of the bid request, provided by the exchange.
imp	object array; required	Array of `Imp` objects (Section 3.2.4) representing the impressions offered. At least 1 `Imp` object is required.
site	object; recommended	Details via a `Site` object (Section 3.2.13) about the publisher's website. Only applicable and recommended for websites.
app	object; recommended	Details via an `App` object (Section 3.2.14) about the publisher's app (i.e., non-browser applications). Only applicable and recommended for apps.
device	object; recommended	Details via a `Device` object (Section 3.2.18) about the user's device to which the impression will be delivered.
user	object; recommended	Details via a `User` object (Section 3.2.20) about the human user of the device; the advertising audience.
test	integer; default 0	Indicator of test mode in which auctions are not billable, where 0 = live mode, 1 = test mode.
at	integer; default 2	Auction type, where 1 = First Price, 2 = Second Price Plus. Exchange-specific auction types can be defined using values 500 and greater.
tmax	integer	Maximum time in milliseconds the exchange allows for bids to be received including Internet latency to avoid timeout. This value supersedes any *a priori* guidance from the exchange.
wseat	string array	Allowed list of buyer seats (e.g., advertisers, agencies) allowed to bid on this impression. IDs of seats and knowledge of the buyer's customers to which they refer must be coordinated between bidders and the exchange *a priori*. At most, only one of `wseat` and `bseat` should be used in the same request. Omission of both implies no seat restrictions.

Attribute	Type	Description
bseat	string array	Block list of buyer seats (e.g., advertisers, agencies) restricted from bidding on this impression. IDs of seats and knowledge of the buyer's customers to which they refer must be coordinated between bidders and the exchange *a priori*. At most, only one of `wseat` and `bseat` should be used in the same request. Omission of both implies no seat restrictions.
allimps	integer; default 0	Flag to indicate if Exchange can verify that the impressions offered represent all of the impressions available in context (e.g., all on the web page, all video spots such as pre/mid/post roll) to support road-blocking. 0 = no or unknown, 1 = yes, the impressions offered represent all that are available.
cur	string array	Array of allowed currencies for bids on this bid request using ISO-4217 alpha codes. Recommended only if the exchange accepts multiple currencies.
wlang	string array	Allowed list of languages for creatives using ISO-639-1-alpha-2. Omission implies no specific restrictions, but buyers would be advised to consider `language` attribute in the `Device` and/or `Content` objects if available. Only one of wlang or wlangb should be present.
wlangb	string array	Allowed list of languages for creatives using IETF BCP 47I. Omission implies no specific restrictions, but buyers would be advised to consider `language` attribute in the `Device` and/or `Content` objects if available. Only one of wlang or wlangb should be present.
bcat	string array	Blocked advertiser categories using the specified category taxonomy. The taxonomy to be used is defined by the cattax field. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
cattax	integer; default 1	The taxonomy in use for bcat. Refer to the AdCOM 1.0 list List: Category Taxonomies for values
badv	string array	Block list of advertisers by their domains (e.g., "ford.com").
bapp	string array	Block list of applications by their app store IDs. See OTT/CTV Store Assigned App Identification Guidelines for more details about expected strings for CTV app stores. For mobile apps in Google Play Store, these should be bundle or package names (e.g. com.foo.mygame). For apps in Apple App Store, these should be a numeric ID.
source	object	A Source object (Section 3.2.2) that provides data about the inventory source and which entity makes the final decision.
regs	object	A Regs object (Section 3.2.3) that specifies any industry, legal, or governmental regulations in force for this request.

Attribute	Type	Description
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.2 Object: Source

This object describes the nature and behavior of the entity that is the source of the bid request upstream from the exchange. The primary purpose of this object is to define post-auction or upstream decisioning when the exchange itself does not control the final decision. A common example of this is header bidding, but it can also apply to upstream server entities such as another RTB exchange, a mediation platform, or an ad server combines direct campaigns with 3rd party demand in decisioning.

Attribute	Type	Description
fd	Integer; recommended	Entity responsible for the final impression sale decision, where 0 = exchange, 1 = upstream source.
tid	string; recommended	Transaction ID that must be common across all participants in this bid request (e.g., potentially multiple exchanges).
pchain	string; recommended	Payment ID chain string containing embedded syntax described in the TAG Payment ID Protocol v1.0.
schain	object; recommended	This object represents both the links in the supply chain as well as an indicator whether or not the supply chain is complete. Details via the `SupplyChain` object (section 3.2.25)
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.3 Object: Regs

This object contains any legal, governmental, or industry regulations that the sender deems applicable to the request. See Section 7.5 for more details on the flags supporting Coppa, GDPR and CCPA.

Attribute	Type	Description
coppa	integer	Flag indicating if this request is subject to the COPPA regulations established by the USA FTC, where 0 = no, 1 = yes. Refer to Section 7.5 for more information.
gdpr	integer	Flag that indicates whether or not the request is subject to GDPR regulations 0 = No, 1 = Yes, omission indicates Unknown. Refer to Section 7.5 for more information.
us_privacy	string	Communicates signals regarding consumer privacy under US privacy regulation. See US Privacy String specifications. Refer to Section 7.5 for more information.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.4 Object: Imp

This object describes an ad placement or impression being auctioned. A single bid request can include multiple `Imp` objects, a use case for which might be an exchange that supports selling all ad positions on a given page. Each `Imp` object has a required ID so that bids can reference them individually.

The presence of `Banner` (Section 3.2.6), `Video` (Section 3.2.7), and/or `Native` (Section 3.2.9) objects subordinate to the `Imp` object indicates the type of impression being offered. The publisher can choose one such type which is the typical case or mix them at their discretion. However, any given bid for the impression must conform to one of the offered type.

Attribute	Type	Description
`id`	string; required	A unique identifier for this impression within the context of the bid request (typically, starts with 1 and increments.
`metric`	object array	An array of `Metric` object (Section 3.2.5).
`banner`	object	A `Banner` object (Section 3.2.6); required if this impression is offered as a banner ad opportunity.
`video`	object	A `Video` object (Section 3.2.7); required if this impression is offered as a video ad opportunity.
`audio`	object	An `Audio` object (Section 3.2.8); required if this impression is offered as an audio ad opportunity.
`native`	object	A `Native` object (Section 3.2.9); required if this impression is offered as a native ad opportunity.
`pmp`	object	A `Pmp` object (Section 3.2.11) containing any private marketplace deals in effect for this impression.
`displaymanager`	string	Name of ad mediation partner, SDK technology, or player responsible for rendering ad (typically video or mobile). Used by some ad servers to customize ad code by partner. Recommended for video and/or apps.
`displaymanagerver`	string	Version of ad mediation partner, SDK technology, or player responsible for rendering ad (typically video or mobile). Used by some ad servers to customize ad code by partner. Recommended for video and/or apps.
`instl`	integer; default 0	1 = the ad is interstitial or full screen, 0 = not interstitial.
`tagid`	string	Identifier for specific ad placement or ad tag that was used to initiate the auction. This can be useful for debugging of any issues, or for optimization by the buyer
`bidfloor`	float; default 0	Minimum bid for this impression expressed in CPM.
`bidfloorcur`	string; default “USD”	Currency specified using ISO-4217 alpha codes. This may be different from bid currency returned by bidder if this is allowed by the exchange.

Attribute	Type	Description
clickbrowser	integer	Indicates the type of browser opened upon clicking the creative in an app, where 0 = embedded, 1 = native. Note that the Safari View Controller in iOS 9.x devices is considered a native browser for purposes of this attribute.
secure	integer	Flag to indicate if the impression requires secure HTTPS URL creative assets and markup, where 0 = non-secure, 1 = secure. If omitted, the secure state is unknown, but non-secure HTTP support can be assumed.
iframebuster	string array	Array of exchange-specific names of supported iframe busters.
rwdd	integer; default 0	Indicates whether the user receives a reward for viewing the ad, where 0 = no, 1 = yes. Typically video ad implementations allow users to read an additional news article for free, receive an extra life in a game, or get a sponsored ad-free music session. The reward is typically distributed after the video ad is completed.
ssai	integer; default 0	Indicates if server-side ad insertion (e.g., stitching an ad into an audio or video stream) is in use and the impact of this on asset and tracker retrieval, where 0 = status unknown, 1 = all client-side (i.e., not server-side), 2 = assets stitched server-side but tracking pixels fired client-side, 3 = all server-side.
exp	integer	Advisory as to the number of seconds that may elapse between the auction and the actual impression.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.5 Object: Metric

This object is associated with an impression as an array of metrics. These metrics can offer insight into the impression to assist with decisioning such as average recent viewability, click-through rate, etc. Each metric is identified by its type, reports the value of the metric, and optionally identifies the source or vendor measuring the value.

Attribute	Type	Description
type	string; required	Type of metric being presented using exchange curated string names which should be published to bidders *a priori*.
value	float; required	Number representing the value of the metric. Probabilities must be in the range 0.0 – 1.0.
vendor	string; recommended	Source of the value using exchange curated string names which should be published to bidders *a priori*. If the exchange itself is the source versus a third party, "EXCHANGE" is recommended.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.6 Object: Banner

This object represents the most general type of impression. Although the term "banner" may have very specific meaning in other contexts, here it can be many things including a simple static image, an

expandable ad unit, or even in-banner video (refer to the `Video` object in Section 3.2.7 for the more generalized and full featured video ad units). An array of `Banner` objects can also appear within the `Video` to describe optional companion ads defined in the VAST specification.

The presence of a `Banner` as a subordinate of the `Imp` object indicates that this impression is offered as a banner type impression. At the publisher's discretion, that same impression may also be offered as video, audio, and/or native by also including as `Imp` subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
`format`	object array; recommended	Array of format objects (Section 3.2.10) representing the banner sizes permitted. If none are specified, then use of the `h` and `w` attributes is highly recommended.
`w`	integer	Exact width in device-independent pixels (DIPS); recommended if no `format` objects are specified.
`h`	integer	Exact height in device-independent pixels (DIPS); recommended if no `format` objects are specified.
`btype`	integer array	Blocked banner ad types. Values: 1 = XHTML Text Ad, 2 = XHTML Banner Ad, 3 = JavaScript Ad, 4 = iframe.
`battr`	integer array	Blocked creative attributes. Refer to List: Creative Attributes in AdCOM 1.0.
`pos`	integer	Ad position on screen. Refer to List: Placement Positions in AdCOM 1.0.
`mimes`	string array	Content MIME types supported. Popular MIME types may include, "`image/jpeg`" and "`image/gif`".
`topframe`	integer	Indicates if the banner is in the top frame as opposed to an iframe, where 0 = no, 1 = yes.
`expdir`	integer array	Directions in which the banner may expand. Refer to List: Expandable Directions in AdCOM 1.0.
`api`	integer array	List of supported API frameworks for this impression. Refer to List: API Frameworks in AdCOM 1.0. If an API is not explicitly listed, it is assumed not to be supported.
`id`	string	Unique identifier for this banner object. Recommended when `Banner` objects are used with a Video object (Section 3.2.7) to represent an array of companion ads. Values usually start at 1 and increase with each object; should be unique within an impression.
`vcm`	integer	Relevant only for `Banner` objects used with a Video object (Section 3.2.7) in an array of companion ads. Indicates the companion banner rendering mode relative to the associated video, where 0 = concurrent, 1 = end-card.
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.7 Object: Video

This object represents an in-stream video impression. Many of the fields are non-essential for minimally viable transactions, but are included to offer fine control when needed. Video in OpenRTB generally assumes compliance with the VAST standard. As such, the notion of companion ads is supported by optionally including an array of `Banner` objects (refer to the `Banner` object in Section 3.2.6) that define these companion ads.

The presence of a `Video` as a subordinate of the `Imp` object indicates that this impression is offered as a video type impression. At the publisher's discretion, that same impression may also be offered as banner, audio, and/or native by also including as `Imp` subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
`mimes`	string array; required	Content MIME types supported (e.g., "`video/mp4`").
`minduration`	integer; default 0 recommended	Minimum video ad duration in seconds. This field is mutually exclusive with rqddurs; only one of minduration and rqddurs may be in a bid request.
`maxduration`	integer; recommended	Maximum video ad duration in seconds. This field is mutually exclusive with rqddurs; only one of maxduration and rqddurs may be in a bid request.
`startdelay`	integer; recommend ed	Indicates the start delay in seconds for pre-roll, mid-roll, or post-roll ad placements. Refer to List: Start Delay Modes in AdCOM 1.0.
`maxseq`	integer; recommended	Indicates the maximum number of ads that may be served into a "dynamic" video ad pod (where the precise number of ads is not predetermined by the seller). See Section 7.6 for more details.
`poddur`	integer; recommended	Indicates the total amount of time in seconds that advertisers may fill for a "dynamic" video ad pod (See Section 7.6 for more details), or the dynamic portion of a "hybrid" ad pod. This field is required only for the dynamic portion(s) of video ad pods. This field refers to the length of the entire ad break, whereas minduration/maxduration/rqddurs are constraints relating to the slots that make up the pod.
`protocols`	integer array; recommended	Array of supported video protocols. Refer to List: Creative Subtypes - Audio/Video in AdCOM 1.0.
`w`	integer; recommended	Width of the video player in device independent pixels (DIPS).
`h`	integer; recommended	Height of the video player in device independent pixels (DIPS).
`podid`	string	Unique identifier indicating that an impression opportunity belongs to a video ad pod. If multiple impression opportunities within a bid request share the same `podid`, this indicates that those impression opportunities belong to the same video ad pod.

Attribute	Type	Description
podseq	integer; default 0	The sequence (position) of the video ad pod within a content stream. Refer to List: Pod Sequence in AdCOM 1.0 for guidance on the use of this field.
rqddurs	integer array	Precise acceptable durations for video creatives in seconds. This field specifically targets the Live TV use case where non-exact ad durations would result in undesirable 'dead air'. This field is mutually exclusive with minduration and maxduration; if rqddurs is specified, minduration and maxduration must not be specified and vice versa.
placement	integer	Video placement type for the impression. Refer to List: Placement Subtypes - Video in AdCOM 1.0.
linearity	integer	Indicates if the impression must be linear, nonlinear, etc. If none specified, assume all are allowed. Refer to List: Linearity Modes in AdCOM 1.0. Note that this field describes the expected VAST response and not whether a placement is in-stream, out-stream, etc. For that, see placement.
skip	integer	Indicates if the player will allow the video to be skipped, where 0 = no, 1 = yes. If a bidder sends markup/creative that is itself skippable, the Bid object should include the attr array with an element of 16 indicating skippable video. Refer to List: Creative Attributes in AdCOM 1.0.
skipmin	integer; default 0	Videos of total duration greater than this number of seconds can be skippable; only applicable if the ad is skippable.
skipafter	integer; default 0	Number of seconds a video must play before skipping is enabled; only applicable if the ad is skippable.
sequence	integer; default 0 DEPRECATED	If multiple ad impressions are offered in the same bid request, the sequence number will allow for the coordinated delivery of multiple creatives.
slotinpod	integer; default 0	For video ad pods, this value indicates that the seller can guarantee delivery against the indicated slot position in the pod. Refer to List: Slot Position in Pod in AdCOM 1.0 guidance on the use of this field.
mincpmpersec	float	Minimum CPM per second. This is a price floor for the "dynamic" portion of a video ad pod, relative to the duration of bids an advertiser may submit.
battr	integer array	Blocked creative attributes. Refer to List: Creative Attributes in AdCOM 1.0.
maxextended	integer	Maximum extended ad duration if extension is allowed. If blank or 0, extension is not allowed. If -1, extension is allowed, and there is no time limit imposed. If greater than 0, then the value represents the number of seconds of extended play supported beyond the maxduration value.
minbitrate	integer	Minimum bit rate in Kbps.
maxbitrate	integer	Maximum bit rate in Kbps.

Attribute	Type	Description
`boxingallowed`	integer; default 1	Indicates if letter-boxing of 4:3 content into a 16:9 window is allowed, where 0 = no, 1 = yes.
`playbackmethod`	integer array	Playback methods that may be in use. If none are specified, any method may be used. Refer to List: Playback Methods in AdCOM 1.0. Only one method is typically used in practice. As a result, this array may be converted to an integer in a future version of the specification. It is strongly advised to use only the first element of this array in preparation for this change.
`playbackend`	integer	The event that causes playback to end. Refer to List: Playback Cessation Modes in AdCOM 1.0.
`delivery`	integer array	Supported delivery methods (e.g., streaming, progressive). If none specified, assume all are supported. Refer to List: Delivery Methods in AdCOM 1.0.
`pos`	integer	Ad position on screen. Refer to List: Placement Positions in AdCOM 1.0.
`companionad`	object array	Array of `Banner` objects (Section 3.2.6) if companion ads are available**.**
`api`	integer array	List of supported API frameworks for this impression. Refer to List: API Frameworks in AdCOM 1.0. If an API is not explicitly listed, it is assumed not to be supported.
`companiontype`	integer array	Supported VAST companion ad types. Refer to List: Companion Types in AdCOM 1.0. Recommended if companion `Banner` objects are included via the `companionad` array. If one of these banners will be rendered as an end-card, this can be specified using the `vcm` attribute with the particular banner (Section 3.2.6).
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.8 Object: Audio

This object represents an audio type impression. Many of the fields are non-essential for minimally viable transactions, but are included to offer fine control when needed. Audio in OpenRTB generally assumes compliance with the VAST standard. As such, the notion of companion ads is supported by optionally including an array of `Banner` objects (refer to the `Banner` object in Section 3.2.6) that define these companion ads.

The presence of a `Audio` as a subordinate of the `Imp` object indicates that this impression is offered as an audio type impression. At the publisher's discretion, that same impression may also be offered as banner, video, and/or native by also including as `Imp` subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
`mimes`	string array; required	Content MIME types supported (e.g., "`audio/mp4`").
`minduration`	integer; default 0 recommended	Minimum audio ad duration in seconds. This field is mutually exclusive with rqddurs; only one of minduration and rqddurs may be in a bid request.
`maxduration`	integer; recommended	Maximum audio ad duration in seconds. This field is mutually exclusive with rqddurs; only one of maxduration and rqddurs may be in a bid request.
`poddur`	integer; recommended	Indicates the total amount of time that advertisers may fill for a "dynamic" audio ad pod, or the dynamic portion of a "hybrid" ad pod. This field is required only for the dynamic portion(s) of audio ad pods. This field refers to the length of the entire ad break, whereas minduration/maxduration/rqddurs are constraints relating to the slots that make up the pod.
`protocols`	integer array; recommended	Array of supported audio protocols. Refer to List: Creative Subtypes - Audio/Video in AdCOM 1.0.
`startdelay`	integer; recommended	Indicates the start delay in seconds for pre-roll, mid-roll, or post-roll ad placements. Refer to List: Start Delay Modes in AdCOM 1.0.
`rqddurs`	integer array	Precise acceptable durations for audio creatives in seconds. This field specifically targets the live audio/radio use case where non-exact ad durations would result in undesirable 'dead air'. This field is mutually exclusive with minduration and maxduration; if rqddurs is specified, minduration and maxduration must not be specified and vice versa.
`podid`	string	Unique identifier indicating that an impression opportunity belongs to an audioad pod. If multiple impression opportunities within a bid request share the same `podid`, this indicates that those impression opportunities belong to the same audio ad pod.
`podseq`	integer; default 0	The sequence (position) of the audio ad pod within a content stream. Refer to List: Pod Sequence in AdCOM 1.0 for guidance on the use of this field.
`sequence`	integer; default 0 DEPRECATED	If multiple ad impressions are offered in the same bid request, the sequence number will allow for the coordinated delivery of multiple creatives.
`slotinpod`	integer; default 0	For audio ad pods, this value indicates that the seller can guarantee delivery against the indicated sequence. Refer to List: Slot Position in Pod in AdCOM 1.0 for guidance on the use of this field.
`mincpmpersec`	float	Minimum CPM per second. This is a price floor for the "dynamic" portion of an audio ad pod, relative to the duration of bids an advertiser may submit.
`battr`	integer array	Blocked creative attributes. Refer to List: Creative Attributes in AdCOM 1.0.

Attribute	Type	Description
maxextended	integer	Maximum extended ad duration if extension is allowed. If blank or 0, extension is not allowed. If -1, extension is allowed, and there is no time limit imposed. If greater than 0, then the value represents the number of seconds of extended play supported beyond the maxduration value.
minbitrate	integer	Minimum bit rate in Kbps.
maxbitrate	integer	Maximum bit rate in Kbps.
delivery	integer array	Supported delivery methods (e.g., streaming, progressive). If none specified, assume all are supported. Refer to List: Delivery Methods in AdCOM 1.0.
companionad	object array	Array of Banner objects (Section 3.2.6) if companion ads are available.
api	integer array	List of supported API frameworks for this impression. Refer to List: API Frameworks in AdCOM 1.0. If an API is not explicitly listed, it is assumed not to be supported.
companiontype	integer array	Supported companion ad types. Refer to List: Companion Types in AdCOM 1.0. Recommended if companion Banner objects are included via the companionad array.
maxseq	integer	The maximum number of ads that can be played in an ad pod.
feed	integer	Type of audio feed. Refer to List: Feed Types in AdCOM 1.0.
stitched	integer	Indicates if the ad is stitched with audio content or delivered independently, where 0 = no, 1 = yes.
nvol	integer	Volume normalization mode. Refer to List: Volume Normalization Modes in AdCOM 1.0.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.9 Object: Native

This object represents a native type impression. Native ad units are intended to blend seamlessly into the surrounding content (e.g., a sponsored Twitter or Facebook post). As such, the response must be well-structured to afford the publisher fine-grained control over rendering.

The Native Subcommittee has developed a companion specification to OpenRTB called the Dynamic Native Ads API. It defines the request parameters and response markup structure of native ad units. This object provides the means of transporting request parameters as an opaque string so that the specific parameters can evolve separately under the auspices of the Dynamic Native Ads API. Similarly, the ad markup served will be structured according to that specification.

The presence of a Native as a subordinate of the Imp object indicates that this impression is offered as a native type impression. At the publisher's discretion, that same impression may also be offered as banner, video, and/or audio by also including as Imp subordinates objects of those types. However, any given bid for the impression must conform to one of the offered types.

Attribute	Type	Description
request	string; required	Request payload complying with the Native Ad Specification. The root node of the payload, "native", was dropped in the Native Ad Specification 1.1. For Native 1.0, this is a JSON-encoded string consisting of a unnamed root object with a single subordinate object named 'native', which is the Native Markup Request object, section 4.1 of OpenRTB Native 1.0 specification. For Native 1.1 and higher, this is a JSON-encoded string consisting of an unnamed root object which is itself the Native Markup Request Object, section 4.1 of OpenRTB Native 1.1+.
ver	string; recommended	Version of the Dynamic Native Ads API to which request complies; highly recommended for efficient parsing.
api	integer array	List of supported API frameworks for this impression. Refer to List: API Frameworks in AdCOM. If an API is not explicitly listed, it is assumed not to be supported.
battr	integer array	Blocked creative attributes. Refer to List: Creative Attributes in AdCOM.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.10 Object: Format

This object represents an allowed size (i.e., height and width combination) or Flex Ad parameters for a banner impression. These are typically used in an array where multiple sizes are permitted. It is recommended that either the w/h pair or the wratio/hratio/wmin set (i.e., for Flex Ads) be specified.

Attribute	Type	Description
w	integer	Width in device independent pixels (DIPS).
h	integer	Height in device independent pixels (DIPS).
wratio	integer	Relative width when expressing size as a ratio.
hratio	integer	Relative height when expressing size as a ratio.
wmin	integer	The minimum width in device independent pixels (DIPS) at which the ad will be displayed the size is expressed as a ratio.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.11 Object: Pmp

This object is the private marketplace container for direct deals between buyers and sellers that may pertain to this impression. The actual deals are represented as a collection of `Deal` objects. Refer to Section 7.3 for more details.

Attribute	Type	Description
`private_auction`	integer; default 0	Indicator of auction eligibility to seats named in the Direct Deals object, where 0 = all bids are accepted, 1 = bids are restricted to the deals specified and the terms thereof.
`deals`	object array	Array of `Deal` (Section 3.2.12) objects that convey the specific deals applicable to this impression.
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.12 Object: Deal

This object constitutes a specific deal that was struck *a priori* between a buyer and a seller. Its presence with the `Pmp` collection indicates that this impression is available under the terms of that deal. Refer to Section 7.3 for more details.

Attribute	Type	Description
`id`	string; required	A unique identifier for the direct deal.
`bidfloor`	float; default 0	Minimum bid for this impression expressed in CPM.
`bidfloorcur`	string; default "USD"	Currency specified using ISO-4217 alpha codes. This may be different from bid currency returned by bidder if this is allowed by the exchange.
`at`	integer	Optional override of the overall auction type of the bid request, where 1 = First Price, 2 = Second Price Plus, 3 = the value passed in `bidfloor` is the agreed upon deal price. Additional auction types can be defined by the exchange.
`wseat`	string array	Allowed list of buyer seats (e.g., advertisers, agencies) allowed to bid on this deal. IDs of seats and the buyer's customers to which they refer must be coordinated between bidders and the exchange *a priori*. Omission implies no seat restrictions.
`wadomain`	string array	Array of advertiser domains (e.g., advertiser.com) allowed to bid on this deal. Omission implies no advertiser restrictions.
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.13 Object: Site

This object should be included if the ad supported content is a website as opposed to a non-browser application. A bid request must not contain both a `Site` and an `App` object. At a minimum, it is useful to provide a site ID or page URL, but this is not strictly required.

Attribute	Type	Description
`id`	string; recommended	Exchange-specific site ID.
`name`	string	Site name (may be aliased at the publisher's request).
`domain`	string	Domain of the site (e.g., "mysite.foo.com").
`cattax`	integer	The taxonomy in use. Refer to the AdCOM list List: Category Taxonomies for values. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
`cat`	string array	Array of IABTL content categories of the site. The taxonomy to be used is defined by the cattax field.
`sectioncat`	string array	Array of IABTL content categories that describe the current section of the site. The taxonomy to be used is defined by the cattax field.
`pagecat`	string array	Array of IABTL content categories that describe the current page or view of the site. The taxonomy to be used is defined by the cattax field
`page`	string	URL of the page where the impression will be shown.
`ref`	string	Referrer URL that caused navigation to the current page.
`search`	string	Search string that caused navigation to the current page.
`mobile`	integer	Indicates if the site has been programmed to optimize layout when viewed on mobile devices, where 0 = no, 1 = yes.
`privacypolicy`	integer	Indicates if the site has a privacy policy, where 0 = no, 1 = yes.
`publisher`	object	Details about the Publisher (Section 3.2.15) of the site.
`content`	object	Details about the Content (Section 3.2.16) within the site.
`keywords`	string	Comma separated list of keywords about the site. Only one of 'keywords' or 'kwarray' may be present.
`kwarray`	string array	Array of keywords about the site. Only one of 'keywords' or 'kwarray' may be present.
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.14 Object: App

This object should be included if the ad supported content is a non-browser application (typically in mobile) as opposed to a website. A bid request must not contain both an `App` and a `Site` object. At a minimum, it is useful to provide an App ID or bundle, but this is not strictly required.

Attribute	Type	Description
`id`	string; recommended	Exchange-specific app ID.
`name`	string	App name (may be aliased at the publisher's request).
`bundle`	string	The store ID of the app in an app store. See OTT/CTV Store Assigned App Identification Guidelines for more details about expected strings for CTV app stores. For mobile apps in Google Play Store, these should be bundle or package names (e.g. com.foo.mygame). For apps in Apple App Store, these should be a numeric ID.
`domain`	string	Domain of the app (e.g., "mygame.foo.com").
`storeurl`	string	App store URL for an installed app; for IQG 2.1 compliance.
`cattax`	integer; default 1	The taxonomy in use. Refer to the AdCOM list List: Category Taxonomies for values.
`cat`	string array	Array of IAB content categories of the app. The taxonomy to be used is defined by the cattax field. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
`sectioncat`	string array	Array of IAB content categories that describe the current section of the app. The taxonomy to be used is defined by the cattax field
`pagecat`	string array	Array of IAB content categories that describe the current page or view of the app. The taxonomy to be used is defined by the cattax field
`ver`	string	Application version.
`privacypolicy`	integer	Indicates if the app has a privacy policy, where 0 = no, 1 = yes.
`paid`	integer	0 = app is free, 1 = the app is a paid version.
`publisher`	object	Details about the Publisher (Section 3.2.15) of the app.
`content`	object	Details about the Content (Section 3.2.16) within the app.
`keywords`	string	Comma separated list of keywords about the app. Only one of 'keywords' or 'kwarray' may be present.
`kwarray`	string array	Array of keywords about the site. Only one of 'keywords' or 'kwarray' may be present.
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.15 Object: Publisher

This object describes the entity who directly supplies inventory to and is paid by the exchange. This may be a publisher, intermediary exchange, ad network, etc.

Attribute	Type	Description
id	string	Exchange-specific seller ID. Every ID must map to only a single entity that is paid for inventory transacted via that ID. Corresponds to a seller_id of a seller in the exchange's sellers.json file.
name	string	Seller name (may be aliased at the seller's request).
cattax	integer; default 1	The taxonomy in use. Refer to the AdCOM list List: Category Taxonomies for values.
cat	string array	Array of IAB content categories that describe the publisher. The taxonomy to be used is defined by the cattax field. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
domain	string	Highest level domain of the seller (e.g., "seller.com").
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.16 Object: Content

This object describes the content in which the impression will appear, which may be syndicated or non-syndicated content. This object may be useful when syndicated content contains impressions and does not necessarily match the publisher's general content. The exchange might or might not have knowledge of the page where the content is running, because of the syndication method. For example, might be a video impression embedded in an iframe on an unknown web property or device.

Attribute	Type	Description
id	string	ID uniquely identifying the content.
episode	integer	Episode number.
title	string	Content title. *Video Examples:* "Search Committee" (television), "A New Hope" (movie), or "Endgame" (made for web). *Non-Video Example:* "Why an Antarctic Glacier Is Melting So Quickly" (Time magazine article).
series	string	Content series. *Video Examples:* "The Office" (television), "Star Wars" (movie), or "Arby 'N' The Chief" (made for web). *Non-Video Example:* "Ecocentric" (Time Magazine blog).
season	string	Content season (e.g., "Season 3").
artist	string	Artist credited with the content.
genre	string	Genre that best describes the content (e.g., rock, pop, etc).
album	string	Album to which the content belongs; typically for audio.

Attribute	Type	Description
isrc	string	International Standard Recording Code conforming to ISO-3901.
producer	object	Details about the content Producer (Section 3.2.17).
url	string	URL of the content, for buy-side contextualization or review.
cattax	integer; default 1	The taxonomy in use. Refer to list List: Category Taxonomies in AdCOM 1.0 for values.
cat	string array	Array of IAB content categories that describe the content. The taxonomy to be used is defined by the cattax field. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
prodq	integer	Production quality. Refer to List: Production Qualities in AdCOM 1.0 .
context	integer	Type of content (game, video, text, etc.). Refer to List: Content Contexts in AdCOM 1.0.
contentrating	string	Content rating (e.g., MPAA).
userrating	string	User rating of the content (e.g., number of stars, likes, etc.).
qagmediarating	integer	Media rating per IQG guidelines. Refer to List: Media Ratings in AdCOM 1.0.
keywords	string	Comma separated list of keywords describing the content. Only one of 'keywords' or 'kwarray' may be present.
kwarray	string array	Array of keywords about the site. Only one of 'keywords' or 'kwarray' may be present.
livestream	integer	0 = not live, 1 = content is live (e.g., stream, live blog).
sourcerelationship	integer	0 = indirect, 1 = direct.
len	integer	Length of content in seconds; appropriate for video or audio.
language	string	Content language using ISO-639-1-alpha-2. Only one of language or langb should be present.
langb	string	Content language using IETF BCP 47. Only one of language or langb should be present.
embeddable	integer	Indicator of whether the content is embeddable (e.g., an embeddable video player), where 0 = no, 1 = yes.
data	object array	Additional content data. Each Data object (Section 3.2.21) represents a different data source.
network	object	Details about the network (Section 3.2.23) the content is on.
channel	object	Details about the channel (Section 3.2.24) the content is on.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.17 Object: Producer

This object defines the producer of the content in which the ad will be shown. This is particularly useful when the content is syndicated and may be distributed through different publishers and thus when the producer and publisher are not necessarily the same entity.

Attribute	Type	Description
`id`	string	Content producer or originator ID. Useful if content is syndicated and may be posted on a site using embed tags.
`name`	string	Content producer or originator name (e.g., "Warner Bros").
`cattax`	integer: default 1	The taxonomy in use. Refer to the AdCOM 1.0 list List: Category Taxonomies for values.
`cat`	string array	Array of IAB content categories that describe the content producer. The taxonomy to be used is defined by the cattax field. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
`domain`	string	Highest level domain of the content producer (e.g., "producer.com").
`ext`	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.18 Object: Device

This object provides information pertaining to the device through which the user is interacting. Device information includes its hardware, platform, location, and carrier data. The device can refer to a mobile handset, a desktop computer, set top box, or other digital device.

Attribute	Type	Description
`geo`	object; recommended	Location of the device assumed to be the user's current location defined by a `Geo` object (Section 3.2.19).
`dnt`	integer; recommended	Standard "Do Not Track" flag as set in the header by the browser, where 0 = tracking is unrestricted, 1 = do not track.
`lmt`	integer; recommended	"Limit Ad Tracking" signal commercially endorsed (e.g., iOS, Android), where 0 = tracking is unrestricted, 1 = tracking must be limited per commercial guidelines.

Attribute	Type	Description
ua	string	Browser user agent string. This field represents a raw user agent string from the browser. For backwards compatibility, exchanges are recommended to always populate 'ua' with the User-Agent string, when available from the end user's device, even if an alternative representation, such as the User-Agent Client-Hints, is available and is used to populate 'sua'. No inferred or approximated user agents are expected in this field. If a client supports User-Agent Client Hints, and 'sua' field is present, bidders are recommended to rely on 'sua' for detecting device type, browser type and version and other purposes that rely on the user agent information, and ignore 'ua' field. This is because the 'ua' may contain a frozen or reduced user agent string.
sua	UserAgent object	Structured user agent information defined by a UserAgent object (see Section 3.2.29). If both 'ua' and 'sua' are present in the bid request, 'sua' should be considered the more accurate representation of the device attributes. This is because the 'ua' may contain a frozen or reduced user agent string.
ip	string	IPv4 address closest to device.
ipv6	string	IP address closest to device as IPv6.
devicetype	integer	The general type of device. Refer to List: Device Types in AdCOM 1.0.
make	string	Device make (e.g., "Apple").
model	string	Device model (e.g., "iPhone").
os	string	Device operating system (e.g., "iOS").
osv	string	Device operating system version (e.g., "3.1.2").
hwv	string	Hardware version of the device (e.g., "5S" for iPhone 5S).
h	integer	Physical height of the screen in pixels.
w	integer	Physical width of the screen in pixels.
ppi	integer	Screen size as pixels per linear inch.
pxratio	float	The ratio of physical pixels to device independent pixels.
js	integer	Support for JavaScript, where 0 = no, 1 = yes.
geofetch	integer	Indicates if the geolocation API will be available to JavaScript code running in the banner, where 0 = no, 1 = yes.
flashver	string	Version of Flash supported by the browser.
language	string	Browser language using ISO-639-1-alpha-2. Only one of language or langb should be present.
langb	string	Browser language using IETF BCP 47. Only one of language or langb should be present.
carrier	string	Carrier or ISP (e.g., "VERIZON") using exchange curated string names which should be published to bidders *a priori*.

Attribute	Type	Description
mccmnc	string	Mobile carrier as the concatenated MCC-MNC code (e.g., "310-005" identifies Verizon Wireless CDMA in the USA). Refer to https://en.wikipedia.org/wiki/Mobile_country_code for further examples. Note that the dash between the MCC and MNC parts is required to remove parsing ambiguity. The MCC-MNC values represent the SIM installed on the device and do not change when a device is roaming. Roaming may be inferred by a combination of the MCC-MNC, geo, IP and other data signals.
connectiontype	integer	Network connection type. Refer to List: Connection Types in AdCOM 1.0.
ifa	string	ID sanctioned for advertiser use in the clear (i.e., not hashed).
didsha1	string; DEPRECATED	Hardware device ID (e.g., IMEI); hashed via SHA1.
didmd5	string; DEPRECATED	Hardware device ID (e.g., IMEI); hashed via MD5.
dpidsha1	string; DEPRECATED	Platform device ID (e.g., Android ID); hashed via SHA1.
dpidmd5	string; DEPRECATED	Platform device ID (e.g., Android ID); hashed via MD5.
macsha1	string; DEPRECATED	MAC address of the device; hashed via SHA1.
macmd5	string; DEPRECATED	MAC address of the device; hashed via MD5.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.19 Object: Geo

This object encapsulates various methods for specifying a geographic location. When subordinate to a `Device` object, it indicates the location of the device which can also be interpreted as the user's current location. When subordinate to a `User` object, it indicates the location of the user's home base (i.e., not necessarily their current location).

The `lat`/`lon` attributes should only be passed if they conform to the accuracy depicted in the `type` attribute. For example, the centroid of a geographic region such as postal code should not be passed.

Attribute	Type	Description
lat	float	Latitude from -90.0 to +90.0, where negative is south.
lon	float	Longitude from -180.0 to +180.0, where negative is west.
type	integer	Source of location data; recommended when passing `lat`/`lon`. Refer to List: Location Types in AdCOM 1.0.

accuracy	integer	Estimated location accuracy in meters; recommended when lat/lon are specified and derived from a device's location services (i.e., type = 1). Note that this is the accuracy as reported from the device. Consult OS specific documentation (e.g., Android, iOS) for exact interpretation.
lastfix	integer	Number of seconds since this geolocation fix was established. Note that devices may cache location data across multiple fetches. Ideally, this value should be from the time the actual fix was taken.
ipservice	integer	Service or provider used to determine geolocation from IP address if applicable (i.e., type = 2). Refer to List: IP Location Services in AdCOM 1.0.
country	string	Country code using ISO-3166-1-alpha-3.
region	string	Region code using ISO-3166-2; 2-letter state code if USA.
regionfips104	string	Region of a country using FIPS 10-4 notation. While OpenRTB supports this attribute, it was withdrawn by NIST in 2008.
metro	string	Google metro code; similar to but not exactly Nielsen DMAs. See Appendix A for a link to the codes.
city	string	City using United Nations Code for Trade & Transport Locations. See Appendix A for a link to the codes.
zip	string	ZIP or postal code.
utcoffset	integer	Local time as the number +/- of minutes from UTC.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.20 Object: User

This object contains information known or derived about the human user of the device (i.e., the audience for advertising). The user `id` is an exchange artifact and may be subject to rotation or other privacy policies. However, when present, this user ID should be stable long enough to serve reasonably as the basis for frequency capping and retargeting.

Attribute	Type	Description
id	string	Exchange-specific ID for the user.
buyeruid	string	Buyer-specific ID for the user as mapped by the exchange for the buyer.
yob	integer; DEPRECATED	Year of birth as a 4-digit integer.
gender	string; DEPRECATED	Gender, where "M" = male, "F" = female, "O" = known to be other (i.e., omitted is unknown).
keywords	string	Comma separated list of keywords, interests, or intent. Only one of 'keywords' or 'kwarray' may be present.
kwarray	string array	Array of keywords about the user. Only one of 'keywords' or 'kwarray' may be present.

customdata	string	Optional feature to pass bidder data that was set in the exchange's cookie. The string must be in base85 cookie safe characters and be in any format. Proper JSON encoding must be used to include "escaped" quotation marks.
geo	object	Location of the user's home base defined by a Geo object (Section 3.2.19). This is not necessarily their current location.
data	object array	Additional user data. Each Data object (Section 3.2.21) represents a different data source.
consent	string	When GDPR regulations are in effect this attribute contains the Transparency and Consent Framework's Consent String data structure.
eids	object array	Details for support of a standard protocol for multiple third party identity providers (Section 3.2.27)
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.21 Object: Data

The data and segment objects together allow additional data about the related object (e.g., user, content) to be specified. This data may be from multiple sources whether from the exchange itself or third parties as specified by the id field. A bid request can mix data objects from multiple providers. The specific data providers in use should be published by the exchange *a priori* to its bidders.

Attribute	Type	Description
id	string	Exchange-specific ID for the data provider.
name	string	Exchange-specific name for the data provider.
segment	object array	Array of Segment (Section 3.2.22) objects that contain the actual data values.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.22 Object: Segment

Segment objects are essentially key-value pairs that convey specific units of data. The parent Data object is a collection of such values from a given data provider. The specific segment names and value options must be published by the exchange *a priori* to its bidders.

Attribute	Type	Description
id	string	ID of the data segment specific to the data provider.
name	string	Name of the data segment specific to the data provider.
value	string	String representation of the data segment value.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.23 Object: Network

This object describes the network an ad will be displayed on. A Network is defined as the parent entity of the Channel object's entity for the purposes of organizing Channels. Examples are companies that own and/or license a collection of content channels (Viacom, Discovery, CBS, WarnerMedia, Turner and others),

or studio that creates such content and self-distributes content. Name is a human-readable field while domain and id can be used for reporting and targeting purposes. See 7.6 for further examples.

Attribute	Type	Description
id	string	A unique identifier assigned by the publisher. This may not be a unique identifier across all supply sources.
name	string	Network the content is on (e.g., a TV network like “ABC")
domain	string	The primary domain of the network (e.g. “abc.com” in the case of the network ABC). It is recommended to include the top private domain (PSL+1) for DSP targeting normalization purposes.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.24 Object: Channel

This object describes the channel an ad will be displayed on. A Channel is defined as the entity that curates a content library, or stream within a brand name for viewers. Examples are specific view selectable ‘channels’ within linear and streaming television (MTV, HGTV, CNN, BBC One, etc) or a specific stream of audio content commonly called ‘stations.’ Name is a human-readable field while domain and id can be used for reporting and targeting purposes. See 7.6 for further examples.

Attribute	Type	Description
id	string	A unique identifier assigned by the publisher. This may not be a unique identifier across all supply sources.
name	string	Channel the content is on (e.g., a local channel like “WABC-TV")
domain	string	The primary domain of the channel (e.g. “abc7ny.com” in the case of the local channel WABC-TV). It is recommended to include the top private domain (PSL+1) for DSP targeting normalization purposes.
ext	object	Placeholder for exchange-specific extensions to OpenRTB.

3.2.25 Object: SupplyChain

This object is composed of a set of nodes where each node represents a specific entity that participates in the transacting of inventory. The entire chain of nodes from beginning to end represents all entities who are involved in the direct flow of payment for inventory. Detailed implementation examples can be found here: https://github.com/InteractiveAdvertisingBureau/openrtb/blob/master/supplychainobject.md

Attribute	Type	Description
complete	integer; required	Flag indicating whether the chain contains all nodes involved in the transaction leading back to the owner of the site, app or other medium of the inventory, where 0 = no, 1 = yes.
nodes	object array; required	Array of SupplyChainNode objects in the order of the chain. In a complete supply chain, the first node represents the initial

		advertising system and seller ID involved in the transaction, i.e. the owner of the site, app, or other medium. In an incomplete supply chain, it represents the first known node. The last node represents the entity sending this bid request.
ver	string; required	Version of the supply chain specification in use, in the format of "major.minor". For example, for version 1.0 of the spec, use the string "1.0".
ext	object	Placeholder for advertising-system specific extensions to this object

3.2.26 Object: SupplyChainNode

This object is associated with a SupplyChain object as an array of nodes. These nodes define the identity of an entity participating in the supply chain of a bid request. Detailed implementation examples can be found here: https://github.com/InteractiveAdvertisingBureau/openrtb/blob/master/supplychainobject.md. The SupplyChainNode object contains the following attributes:

Attribute	Type	Description
asi	string; required	The canonical domain name of the SSP, Exchange, Header Wrapper, etc system that bidders connect to. This may be the operational domain of the system, if that is different than the parent corporate domain, to facilitate WHOIS and reverse IP lookups to establish clear ownership of the delegate system. This should be the same value as used to identify sellers in an ads.txt file if one exists.
sid	string; required	The identifier associated with the seller or reseller account within the advertising system. This must contain the same value used in transactions (i.e. OpenRTB bid requests) in the field specified by the SSP/exchange. Typically, in OpenRTB, this is publisher.id. For OpenDirect it is typically the publisher's organization ID.Should be limited to 64 characters in length.
rid	string	The OpenRTB RequestId of the request as issued by this seller.
name	string	The name of the company (the legal entity) that is paid for inventory transacted under the given seller_ID. This value is optional and should NOT be included if it exists in the advertising system's sellers.json file.
domain	string	The business domain name of the entity represented by this node. This value is optional and should NOT be included if it exists in the advertising system's sellers.json file.
hp	integer	Indicates whether this node will be involved in the flow of payment for the inventory. When set to 1, the advertising system in the asi field pays the seller in the sid field, who is responsible for paying the previous node in the chain. When set to 0, this node is not involved in the flow of payment for the inventory. For version 1.0 of SupplyChain, this property should always be 1. Implementers should ensure that they propagate this field onwards when constructing SupplyChain objects in bid requests sent to a downstream advertising system.

Attribute	Type	Description
ext	object	Placeholder for advertising-system specific extensions to this object

3.2.27 Object: EID

Extended identifiers support in the OpenRTB specification allows buyers to use audience data in real-time bidding. This object can contain one or more UIDs from a single source or a technology provider. The exchange should ensure that business agreements allow for the sending of this data.

Attribute	Type	Description
source	string	Source or technology provider responsible for the set of included IDs. Expressed as a top-level domain.
uids	object array;	Array of extended ID UID objects from the given source. Refer to 3.2.28 Extended Identifier UIDs
ext	object	Placeholder for vendor specific extensions to this object

3.2.28 Object: UID

This object contains a single user identifier provided as part of extended identifiers. The exchange should ensure that business agreements allow for the sending of this data.

Attribute	Type	Description
id	string	The identifier for the user.
atype	integer	Type of user agent the ID is from. It is highly recommended to set this, as many DSPs separate app-native IDs from browser-based IDs and require a type value for ID resolution. Refer to List: Agent Types in AdCOM 1.0
ext	object	Placeholder for vendor specific extensions to this object

3.2.29 Object: UserAgent

Structured user agent information, which can be used when a client supports User-Agent Client Hints. If both device.ua and device.sua are present in the bid request, device.sua should be considered the more accurate representation of the device attributes. This is because the device.ua may contain a frozen or reduced user agent string due to deprecation of user agent strings by browsers.

Attribute	Type	Description
browsers	array of BrandVersion objects; recommended	Each BrandVersion object (see Section 3.2.30) identifies a browser or similar software component. Implementers should send brands and versions derived from the Sec-CH-UA-Full-Version-List header*.
platform	BrandVersion object; recommended	A BrandVersion object (see Section 3.2.30) that identifies the user agent's execution platform / OS. Implementers should send a brand derived from the Sec-CH-UA-Platform header, and version derived from the Sec-CH-UA-Platform-Version header *.
mobile	integer	1 if the agent prefers a "mobile" version of the content, if available, i.e. optimized for small screens or touch input. 0 if the agent prefers the "desktop"

		or “full” content. Implementers should derive this value from the Sec-CH-UA-Mobile header *.
architecture	string	Device’s major binary architecture, e.g. “x86” or “arm”. Implementers should retrieve this value from the Sec-CH-UA-Arch header*.
bitness	string	Device’s bitness, e.g. “64” for 64-bit architecture. Implementers should retrieve this value from the Sec-CH-UA-Bitness header*.
model	string	Device model. Implementers should retrieve this value from the Sec-CH-UA-Model header*.
source	Integer; default 0	The source of data used to create this object, List: User-Agent Source in AdCOM 1.0
ext	object	Placeholder for vendor specific extensions to this object

*or an equivalent JavaScript accessor from NavigatorUAData interface. This header or accessor are only available for browsers that support User-Agent Client Hints.

3.2.30 Object: BrandVersion

Further identification based on User-Agent Client Hints, the BrandVersion object is used to identify a device’s browser or similar software component, and the user agent’s execution platform or operating system.

Attribute	Type	Description
brand	string; required	A brand identifier, for example, “Chrome” or “Windows”. The value may be sourced from the User-Agent Client Hints headers, representing either the user agent brand (from the Sec-CH-UA-Full-Version header) or the platform brand (from the Sec-CH-UA-Platform header).
version	array of string	A sequence of version components, in descending hierarchical order (major, minor, micro, …)
ext	object	Placeholder for vendor specific extensions to this object

4 Bid Response Specification

RTB responses contain bids that reference specific impressions within a bid request. Bids are in essence an offer to buy. The bid response consists of the top-level bid response object and optional objects that depict the specific bids. An empty HTTP response constitutes a no-bid and is in fact the most bandwidth friendly form of this signal although returning a response with a “no-bid reason” is encouraged. A malformed response or a response that contains no actual bids will also be interpreted as no-bid.

4.2 Object Model

Following is the object model for the bid response. The top-level object (i.e., in JSON the unnamed outer object) is denoted as `BidResponse` in the model. A bid response may contain bids from multiple

"seats" (i.e., the buying entity upstream from the actual bidder). In fact, a response may contain multiple bids from the same seat; typically but not necessarily from different campaigns. This can improve the seat's chances of winning since most exchanges enforce various block lists on behalf of their publishers.

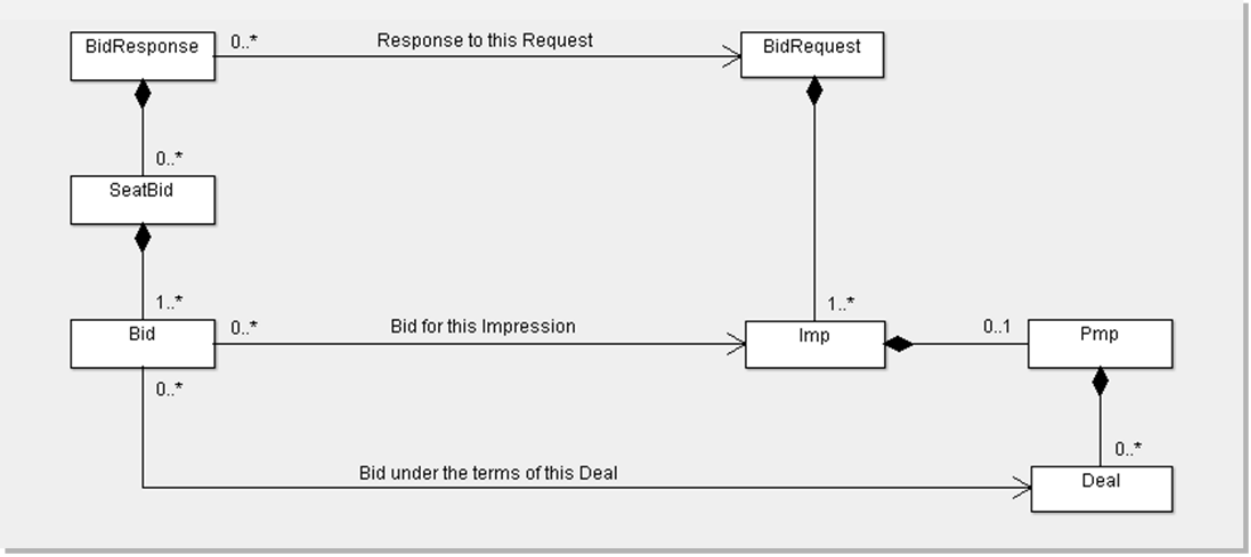


Figure 4: Bid Response object model.

Referring to the figure, the actual response objects are shown on the left, specifically the `BidResponse` top level object the seat specific `SeatBid` collections of `Bid` objects. The other objects shown are those objects from the bid request to which response objects related. Specifically, `BidResponse` includes the `BidRequest` ID for positive tracking purposes, and since a request can include multiple impressions `Bid` includes the ID of the `Imp` for which the bid is an offer to purchase. If a bid is made under the terms of a private marketplace deal, the `Bid` also includes the ID of the specific `Deal` object.

Not shown in the model figure is an extensions object. This is an object of undefined structure that can be added to any other object to convey bidder-specific extensions to the standard. Bidders using these objects are responsible for publishing their extensions to their exchanges.

The following table summarizes the objects in the Bid Response model and serves as an index into the detailed definitions in the subsections that follow.

Object	Section	Description
`BidResponse`	4.2.1	Top-level object.
`SeatBid`	4.2.2	Collection of bids made by the bidder on behalf of a specific seat.
`Bid`	4.2.3	An offer to buy a specific impression under certain business terms.

4.3 Object Specifications

The subsections that follow define each of the objects in the bid response model. Several conventions are used throughout:

- Attributes are "required" if their omission would technically break the protocol.
- Some optional attributes are denoted "recommended" due to their elevated business importance.
- Unless a default value is explicitly specified, an omitted attribute is interpreted as "unknown".

4.3.1 Object: BidResponse

This object is the top-level bid response object (i.e., the unnamed outer JSON object). The `id` attribute reflects the bid request ID for logging purposes. Similarly, `bidid` is an optional response tracking ID for bidders. If specified, it can be included in the subsequent win notice call if the bidder wins. At least one `seatbid` object is required, which contains at least one bid for an impression. Other attributes are optional.

To express a "no-bid", the options are to return an empty response with HTTP 204. Alternately if the bidder wishes to convey to the exchange a reason for not bidding, just a `BidResponse` object is returned with a reason code in the `nbr` attribute.

Attribute	Type	Description
`id`	string; required	ID of the bid request to which this is a response.
`seatbid`	object array	Array of seatbid objects; 1+ required if a bid is to be made.
`bidid`	string	Bidder generated response ID to assist with logging/tracking.
`cur`	string; default "USD"	Bid currency using ISO-4217 alpha codes.
`customdata`	string	Optional feature to allow a bidder to set data in the exchange's cookie. The string must be in base85 cookie safe characters and be in any format. Proper JSON encoding must be used to include "escaped" quotation marks.
`nbr`	integer	Reason for not bidding. Refer to List: No-Bid Reason Codes in OpenRTB 3.0.
`ext`	object	Placeholder for bidder-specific extensions to OpenRTB.

4.3.2 Object: SeatBid

A bid response can contain multiple `SeatBid` objects, each on behalf of a different bidder seat and each containing one or more individual bids. If multiple impressions are presented in the request, the `group` attribute can be used to specify if a seat is willing to accept any impressions that it can win (default) or if it is only interested in winning any if it can win them all as a group.

Attribute	Type	Description
`bid`	object array; required	Array of 1+ `Bid` objects (Section 4.2.3) each related to an impression. Multiple bids can relate to the same impression.
`seat`	string	ID of the buyer seat (e.g., advertiser, agency) on whose behalf this bid is made.
`group`	integer; default 0	0 = impressions can be won individually; 1 = impressions must be won or lost as a group.
`ext`	object	Placeholder for bidder-specific extensions to OpenRTB.

4.3.3 Object: Bid

A `SeatBid` object contains one or more Bid objects, each of which relates to a specific impression in the bid request via the `impid` attribute and constitutes an offer to buy that impression for a given `price`.

Attribute	Type	Description
`id`	string; required	Bidder generated bid ID to assist with logging/tracking.
`impid`	string; required	ID of the `Imp` object in the related bid request.
`price`	float; required	Bid price expressed as CPM although the actual transaction is for a unit impression only. Note that while the type indicates float, integer math is highly recommended when handling currencies (e.g., BigDecimal in Java).
`nurl`	string	Win notice URL called by the exchange if the bid wins (not necessarily indicative of a delivered, viewed, or billable ad); optional means of serving ad markup. Substitution macros (Section 4.4) may be included in both the URL and optionally returned markup.
`burl`	string	Billing notice URL called by the exchange when a winning bid becomes billable based on exchange-specific business policy (e.g., typically delivered, viewed, etc.). Substitution macros (Section 4.4) may be included.
`lurl`	string	Loss notice URL called by the exchange when a bid is known to have been lost. Substitution macros (Section 4.4) may be included. Exchange-specific policy may preclude support for loss notices or the disclosure of winning clearing prices resulting in ${AUCTION_PRICE} macros being removed (i.e., replaced with a zero-length string).
`adm`	string	Optional means of conveying ad markup in case the bid wins; supersedes the win notice if markup is included in both. Substitution macros (Section 4.4) may be included.

Attribute	Type	Description
adid	string	ID of a preloaded ad to be served if the bid wins.
adomain	string array	Advertiser domain for block list checking (e.g., "ford.com"). This can be an array of for the case of rotating creatives. Exchanges can mandate that only one domain is allowed.
bundle	string	The store ID of the app in an app store (e.g., Apple App Store, Google Play). See OTT/CTV Store Assigned App Identification Guidelines for more details about expected strings for CTV app stores. For mobile apps in Google Play Store, these should be bundle or package names (e.g. com.foo.mygame). For apps in Apple App Store, these should be a numeric ID.
iurl	string	URL without cache-busting to an image that is representative of the content of the campaign for ad quality/safety checking.
cid	string	Campaign ID to assist with ad quality checking; the collection of creatives for which `iurl` should be representative.
crid	string	Creative ID to assist with ad quality checking.
tactic	string	Tactic ID to enable buyers to label bids for reporting to the exchange the tactic through which their bid was submitted. The specific usage and meaning of the tactic ID should be communicated between buyer and exchanges *a priori*.
cattax	integer: default 1	The taxonomy in use. Refer to the AdCOM 1.0 list List: Category Taxonomies for values.
cat	string array	IAB content categories of the creative. The taxonomy to be used is defined by the cattax field. If no cattax field is supplied IAB Content Category Taxonomy 1.0 is assumed.
attr	integer array	Set of attributes describing the creative. Refer to List: Creative Attributes in AdCOM 1.0.
apis	integer array	List of supported APIs for the markup. If an API is not explicitly listed, it is assumed to be unsupported. Refer to List: API Frameworks in AdCOM 1.0.
api	integer; DEPRECATED	*NOTE: Deprecated in favor of the `apis` integer array.* API required by the markup if applicable. Refer to List: API Frameworks in AdCOM 1.0.
protocol	integer	Video response protocol of the markup if applicable. Refer to List: Creative Subtypes - Audio/Video in AdCOM 1.0.
qagmediarating	integer	Creative media rating per IQG guidelines. Refer to List: Media Ratings in AdCOM 1.0.
language	string	Language of the creative using ISO-639-1-alpha-2. The non-standard code "xx" may also be used if the creative has no linguistic content (e.g., a banner with just a company logo). Only one of language or langb should be present.
langb	string	Language of the creative using IETF BCP 47. Only one of language or langb should be present.
dealid	string	Reference to the `deal.id` from the bid request if this bid pertains to a private marketplace direct deal.
w	integer	Width of the creative in device independent pixels (DIPS).

Attribute	Type	Description
h	integer	Height of the creative in device independent pixels (DIPS).
wratio	integer	Relative width of the creative when expressing size as a ratio. Required for Flex Ads.
hratio	integer	Relative height of the creative when expressing size as a ratio. Required for Flex Ads.
exp	integer	Advisory as to the number of seconds the bidder is willing to wait between the auction and the actual impression.
dur	integer	Duration of the video or audio creative in seconds.
mtype	integer	Type of the creative markup so that it can properly be associated with the right sub-object of the BidRequest.Imp. Values: 1 = Banner 2 = Video, 3 = Audio 4 = Native
slotinpod	integer; default 0	Indicates that the bid response is only eligible for a specific position within a video or audio ad pod (e.g. first position, last position, or any). Refer to List: Slot Position in Pod in AdCOM 1.0 for guidance on the use of this field.
ext	object	Placeholder for bidder-specific extensions to OpenRTB.

For each bid, the `nurl` attribute can contain the win notice URL. If the bidder wins the impression, the exchange calls this notice URL to inform the bidder of the win and to convey certain information using substitution macros (see Section 4.5) such as the clearing price. The win notice return or the `adm` attribute can be used to serve markup (see Section 4.4). In either case, the exchange will also apply the aforementioned substitution to any macros found in the markup.

BEST PRACTICE: The essential function of the win notice is to inform a bidder that they won an auction. It does not necessarily imply ad delivery, creative viewability, or billability. Exchanges are highly encouraged to publish to their bidders their event triggers, billing policies, and any other meaning they attach to the win notice. Also, please refer to Section 7.2 for additional guidance on expirations.

BEST PRACTICE: Firing of the billing notice should be server-side and as "close" as possible to where the exchange books revenue in order to minimize discrepancies between exchange and bidder.

BEST PRACTICE: For VAST Video, the IAB prescribes that the VAST impression event is the official signal that the impression is billable. If the `burl` attribute is specified, it too should be fired at the same time if the exchange is adhering to this policy. However, subtle technical issues may lead to additional discrepancies and bidders are cautioned to avoid this scenario.

Several other attributes are used for ad quality checks or enforcing publisher restrictions. These include the advertiser domain via `adomain`, a non-cache-busted URL to an image representative of the content of the campaign via `iurl`, an ID of the campaign and of the creative within the campaign via `cid` and `crid` respectively, an array of creative attribute via `attr`, and the dimensions via `h` and `w`. If the bid pertains to a private marketplace deal, the `dealid` attribute is used to reference that agreement from the bid request.

4.4 Ad Serving Options

The fulfilment of an RTB transaction within the scope of this OpenRTB specification lies in the delivery of markup. Depending on the impression and other ad type constraints, this markup can be XHTML, HTML5, XHTML or HTML5 with embedded JavaScript, a VAST document for video or audio, a Native ad unit structure, and potentially other formats in the future.

The OpenRTB specification does not require any processing of the ad markup by the exchange other than macro substitution (refer to Section 4.5) and delivery to the supply-side. There are, however, multiple standard methods for transferring markup from the bidder to the exchange. The method used is at the discretion of the bidder, subject to support by the exchange. Bidders and exchanges should discuss the method to be used during integration.

4.4.1 Markup Served on the Win Notice

In this method, ad markup is returned to the exchange is via the win notice. In this case, the response body of the win notice call (i.e., invoking the `bid.nurl` attribute) contains the ad markup and only the ad markup; there must be no other structured data in the response body. Using this method, the `bid.adm` attribute must be omitted.

4.4.2 Markup Served in the Bid

In this method, ad markup is returned directly in the bid itself. This is accomplished via the `bid.adm` attribute. If both the `adm` attribute and win notice return data, the `adm` contents will take precedence.

4.4.3 Comparison of Ad Serving Approaches

Each of the ad serving methods has its own advantages that may be of varying importance to either the exchange or the bidder.

Ad Served on the Win Notice

- *Reduced Bandwidth Costs:* Serving ad markup only upon winning can save large amounts of bandwidth usage, the costs for which can be large at high volumes or when sending multiple bids per bid response.
- *Additional Bidder Flexibility:* Bidders may typically know the ad they will serve at the time of bid, but this provides an additional optional decision point after the clearing price has been established.

Ad Served in the Bid

- *Reduced Risk of Forfeiture:* A forfeit is the scenario in which a bidder wins, but forfeits due to failure to serve the ad markup. The risk of an additional HTTP failure (e.g., calling the win notice) is mitigated by this method.
- *Potential Concurrency:* The exchange can choose to return that ad markup and call the win notice concurrently, thereby improving user experience.

4.5 Substitution Macros

The win notice and billing notice URLs and their format are defined by the bidder. For the exchange to convey certain information to the bidder (e.g., the clearing price), several substitution macros can be inserted into these URLs. Prior to calling a win or billing notice URL, the exchange will search the specified URL for any of the defined macros and replace them with the appropriate data.
Note that the substitution is simple in the sense that wherever a legal macro is found, it will be replaced without regard for syntax correctness. Furthermore, if the source value is an optional parameter that was not specified, the macro will simply be removed (i.e., replaced with a zero-length string).

These same substitution macros can also be placed in the ad markup. The exchange will perform the same data substitutions as in the aforementioned notice URLs. This occurs irrespective of whether the markup is returned on the win notice or passed in the `bid.adm` attribute of the bid response. A use case for macros in the ad markup might be when a bidder prefers to receive its win notification from the device itself. To accomplish this, the bidder would include a tracking pixel in the ad markup, the URL for which would include any of the available macros.

Macro	Description
`${AUCTION_ID}`	ID of the bid request; from `BidRequest.id` attribute.
`${AUCTION_BID_ID}`	ID of the bid; from `BidResponse.bidid` attribute.
`${AUCTION_IMP_ID}`	ID of the impression just won; from `imp.id` attribute.
`${AUCTION_SEAT_ID}`	ID of the bidder seat for whom the bid was made.
`${AUCTION_AD_ID}`	ID of the ad markup the bidder wishes to serve; from `bid.adid` attribute.
`${AUCTION_PRICE}`	Clearing price using the same currency and units as the bid.
`${AUCTION_CURRENCY}`	The currency used in the bid (explicit or implied); for confirmation only.
`${AUCTION_MBR}`	Market Bid Ratio defined as: clearance price / bid price.
`${AUCTION_LOSS}`	Loss reason codes. Refer to List: Loss Reason Codes in OpenRTB 3.0.
`${AUCTION_MIN_TO_WIN}`	Minimum bid to win the exchange's auction, using the same currency and units as the bid.

Note that OpenRTB compliance exchanges must support all macros for which data is available and support substitution in both markup and URLs for win and billing notification.

> ***BEST PRACTICE:*** When rendering markup for test or ad quality purposes, some macro values (e.g., clearing price) may not be known. In these cases, substitute "AUDIT" as the macro value.

> ***BEST PRACTICE:*** Encoding of macro data should be used sparingly due to the additional processing overhead. For communications strictly between exchange and bidder (e.g., a win notice called from the exchange), encoding is generally considered unnecessary.

Prior to substitution, macro data values can be encoded for security purposes using various obfuscation or encryption algorithms. This may be of particular interest for use cases where price information is carried beyond the exchange, through the publisher, and into the device browser via a tracking pixel in the markup.

To specify that a particular macro is to be encoded, the suffix "`:X`" should be appended to the macro name, where X is a string that indicates the algorithm to be used. Algorithms choices are not defined by this specification and must be mutually agreed upon between parties. As an example, suppose that the price macro is to be encoded using Base64 and that its code is "`B64`". The macro would then be written as follows:

```
${AUCTION_PRICE:B64}
```

4.5.1 Notes on the macro ${AUCTION_MIN_TO_WIN}

Except where the value "AUDIT" applies, as above, the macro is replaced by:

- The empty string, if the exchange chooses not to provide the value, for example because:
  - The bid was not allowed into the auction, e.g., because of blocks on the seat, advertiser, etc.
  - Per the loss code, price was not the deciding factor in the loss.
  - As dictated by exchange privacy controls, e.g., if price data is not shared to bidders that did not meet the desired floor, or if the exchange supports a publisher or winning bidder election not to share price data with the other bidders.
- The minimum price required to tie with the winning bid, when your bid lost to another in the auction.
- The minimum price required to tie with the next-closest bid, or the floor if there was only one bid, when your bid won the auction

Note, the exchange in question may not be the final decision-maker. It may propagate two or more bids into a second auction. In that case, still only one auction bid is the "winner" and the macro is replaced for all bids as above.

In the following examples, assume an auction with a floor price of $0.85, and that each row is a competing bid. Note the exchange may also choose to replace the macros with the empty string for the reasons above.

For a first-price auction:

Bid price	${AUCTION_PRICE}	${AUCTION_MIN_TO_WIN}
$1.00	$1.00	$0.90
$0.90	Empty string	$1.00
$0.80	Empty string	$1.00
Invalid	n/a	Empty string

For a second-price auction (where, for the sake of the illustration, the clearing price is the second-best price plus $0.01):

Bid price	${AUCTION_PRICE}	${AUCTION_MIN_TO_WIN}
$1.00	$0.91	$0.90
$0.90	Empty string	$0.91
$0.80	Empty string	$0.91
Invalid	n/a	Empty string

5 Enumerated Lists Specification

OpenRTB utilizes the Advertising Common Object Model (AdCOM 1.0) for enumerations. In previous versions, any additional enumerations required not just a documentation update, but a version update. Using AdCOM 1.0, items will be able to be added to enumerations without requiring a new OpenRTB version. Implementers will need to ensure that they are utilizing the most current enumerations as possible as new standards come out.

6 Bid Request/Response Samples

6.3 GitHub Repository

The official OpenRTB Github repository now contains a set of validated example requests. This repository should be considered the canonical examples for implementers.

https://github.com/InteractiveAdvertisingBureau/openrtb

6.4 Bid Requests

6.4.1 Example 1 – Simple Banner

Following is a basic example of a bid request for a banner ad. Some optional parameters are included in this example.


```
{
 "id": "80ce30c53c16e6ede735f123ef6e32361bfc7b22",
  "at": 1,
   "cur": [
    "USD"
  ],
    "imp": [
     {
      "id": "1",
      "bidfloor": 0.03,
      "banner": {
       "h": 250,
       "w": 300,
       "pos": 0
      }
     }
    ],
     "site": {
  "id": "102855",
   "cat": [
    "IAB3-1"
  ],
    "domain": "www.foobar.com",
     "page": "http://www.foobar.com/1234.html ",
      "publisher": {
   "id": "8953",
    "name": "foobar.com",
     "cat": [
      "IAB3-1"
```

```
      ],
       "domain": "foobar.com"
   },
   "user": {
    "id": "55816b39711f9b5acf3b90e313ed29e51665623f"
   }
  }
}
```

6.4.2 Example 2 – Expandable Creative

This example builds the first and adds parameters to describe support for an expandable creative, and passes data about the user from "Data Provider 1".

```
{
  "id": "123456789316e6ede735f123ef6e32361bfc7b22",
   "at": 2,
    "cur": [
     "USD"
    ],
      "imp": [
       {
        "id": "1",
        "bidfloor": 0.03,
        "iframebuster": [
         "vendor1.com",
         "vendor2.com"
        ],
        "banner": {
         "h": 250,
         "w": 300,
         "pos": 0,
         "battr": [
          13
         ],
         "expdir": [
          2,
          4
         ]
        }
       }
      ],
       "site": {
   "id": "102855",
    "cat": [
     "IAB3-1"
    ],
```

```
    "domain": "www.foobar.com",
     "page": "http://www.foobar.com/1234.html",
      "publisher": {
   "id": "8953",
    "name": "foobar.com",
     "cat": [
      "IAB3-1"
     ],
      "domain": "foobar.com"
  }
 },
 "user": {
  "id": "55816b39711f9b5acf3b90e313ed29e51665623f",
   "buyeruid": "5456787654678765678987656789876 54",
    "data": [
     {
      "id": "12341318394918",
      "name": "auto intenders"
     },
     {
      "id": "1234131839491234",
      "name": "auto enthusiasts"
     },
     {
      "id": "23423424",
      "name": "data-provider1-age",
      "value": "30-40"
     }
    ]
 }
}
```

6.4.3 Example 3 – Mobile

This example uses a device object to reflect a mobile device, and an app object to reflect a request from a mobile application .


```
{
 "id": "IxexyLDIIk",
  "at": 2,
   "bcat": [
    "IAB25",
    "IAB7-39",
    "IAB8-18",
    "IAB8-5",
    "IAB9-9"
   ],
    "badv": [
     "apple.com",
     "go-text.me",
     "heywire.com"
    ],
     "imp": [
      {
       "id": "1",
       "bidfloor": 0.5,
       "instl": 0,
       "tagid": "agltb3B1Yi1pbmNyDQsSBFNpdGUY7fD0FAw",
       "banner": {
        "w": 728,
        "h": 90,
        "pos": 1,
        "btype": [
         4
        ],
        "battr": [
         14
        ],
        "api": [
         3
        ]
       }
      }
     ],
      "app": {
  "id": "agltb3B1Yi1pbmNyDAsSA0FwcBiJkfIUDA",
   "name": "Yahoo Weather",
    "cat": [
     "IAB15",
     "IAB15-10"
    ],
```

```
      "ver": "1.0.2",
       "bundle": "12345",
        "storeurl": "https://itunes.apple.com/id628677149",
         "publisher": {
    "id": "agltb3B1Yi1pbmNyDAsSA0FwcBiJkfTUCV",
     "name": "yahoo",
      "domain": "www.yahoo.com"
   }
  },
  "device": {
   "dnt": 0,
    "ua": "Mozilla/5.0 (iPhone; CPU iPhone OS 6_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gec
ko) Version/5.1 Mobile/9A334 Safari/7534.48.3",
      "ifa": "AA000DFE74168477C70D291f574D344790E0BB11",
       "carrier": "VERIZON",
        "language": "en",
         "make": "Apple",
          "model": "iPhone",
           "os": "iOS",
            "osv": "6.1",
             "js": 1,
              "connectiontype": 3,
               "devicetype": 1
  },
  "user": {
   "id": "fffffd5135596709273b3a1a07e466ea2bf4fff"
  }
}
```

6.4.4 Example 4 – Video

The following example illustrates a bid request for a video impression with two companion ad slots (1 expandable). Additionally, the video content itself is described in the "content" object. A few notes about specific fields in the example:

- `battr`: User interactive and alert type ads (value “13” and “14”, respectively) are explicitly being blocked for both the video and its companions.
- `pos`: Indicates this opportunity is “above the fold”.
- `companiontype`: Indicates only static or HTML resources are allowed.


```
{
  "id": "1234567893",
   "at": 2,
    "tmax": 120,
     "imp": [
      {
       "id": "1",
       "bidfloor": 0.03,
       "video": {
        "w": 640,
        "h": 480,
        "pos": 1,
        "startdelay": 0,
        "minduration": 5,
        "maxduration": 30,
        "maxextended": 30,
        "minbitrate": 300,
        "maxbitrate": 1500,
        "apis": [
         1,
         2
        ],
        "protocols": [
         2,
         3
        ],
        "mimes": [
         "video/x-flv",
         "video/mp4",
         "application/javascript"
        ],
        "linearity": 1,
        "boxingallowed": 1,
        "playbackmethod": [
         1,
         3
        ],
        "delivery": [
         2
```

```
        ],
        "battr": [
          13,
          14
        ],
        "companionad": [
          {
            "id": "1234567893-1",
            "w": 300,
            "h": 250,
            "pos": 1,
            "battr": [
              13,
              14
            ],
            "expdir": [
              2,
              4
            ]
          },
          {
            "id": "1234567893-2",
            "w": 728,
            "h": 90,
            "pos": 1,
            "battr": [
              13,
              14
            ]
          }
        ],
        "companiontype": [
          1,
          2
        ]
      }
    }
  ],
    "site": {
"id": "1345135123",
  "name": "Site ABCD",
    "domain": "siteabcd.com",
      "cat": [
        "IAB2-1",
        "IAB2-2"
      ],
        "page": "http://siteabcd.com/page.htm",
          "ref": "http://referringsite.com/referringpage.htm",
            "privacypolicy": 1,
              "publisher": {
  "id": "pub12345",
```

```
      "name": "Publisher A"
    },
    "content": {
      "id": "1234567",
      "series": "All About Cars",
       "season": "2",
        "episode": 23,
          "title": "Car Show",
           "cat": [
            "IAB2-2"
           ],
            "keywords": "keyword-a,keyword-b,keyword-c"
    }
  },
  "device": {
    "ua": "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-
US; rv:1.9.2.16) Gecko/20110319 Firefox/3.6.16",
      "os": "OS X",
       "js": 1
  },
  "user": {
    "id": "456789876567897654678987656789",
     "buyeruid": "5456787654678765678987656789876 54",
      "data": [
       {
        "id": "6",
        "name": "Data Provider 1",
        "segment": [
         {
          "id": "12341318394918",
          "name": "auto intenders"
         },
         {
          "id": "1234131839491234",
          "name": "auto enthusiasts"
         }
        ]
       }
      ]
  }
}
```

6.4.5 Example 5 – PMP with Direct Deal

Following is a basic example of a bid request for a banner ad with a direct deal. Some optional parameters are included in this example.


```
{
 "id": "80ce30c53c16e6ede735f123ef6e32361bfc7b22",
  "at": 1,
   "cur": [
    "USD"
   ],
    "imp": [
     {
      "id": "1",
      "bidfloor": 0.03,
      "banner": {
       "h": 250,
       "w": 300,
       "pos": 0
      },
      "pmp": {
       "private_auction": 1,
       "deals": [
        {
         "id": "AB-Agency1-0001",
         "at": 1,
         "bidfloor": 2.5,
         "wseat": [
          "Agency1"
         ]
        },
        {
         "id": "XY-Agency2-0001",
         "at": 2,
         "bidfloor": 2,
         "wseat": [
          "Agency2"
         ]
        }
       ]
      }
     }
    ],
     "site": {
  "id": "102855",
   "domain": "www.foobar.com",
    "cat": [
     "IAB3-1"
    ],
     "page": "http://www.foobar.com/1234.html",
```

```
        "publisher": {
     "id": "8953",
      "name": "foobar.com",
       "cat": [
        "IAB3-1"
       ],
        "domain": "foobar.com"
   }
  },
  "device": {
   "ua": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/537.13 (KHTML, like Gecko) Versi
on/5.1.7 Safari/534.57.2"
  },
  "user": {
   "id": "55816b39711f9b5acf3b90e313ed29e51665623f"
  }
}
```


6.4.6 Example 6 – Native Ad

For details on Native Ad implementation including examples, please review the OpenRTB Native Ads API Specification: https://iabtechlab.com/standards/openrtb-native/

6.5 Bid Responses

6.5.1 Example 1 – Ad Served on Win Notice

Following is an example of a bid response with the ad served on win notice. The bid for this impression is a $9.43 CPM.

```
{
 "id": "1234567890",
  "bidid": "abc1123",
   "cur": "USD",
    "seatbid": [
     {
      "seat": "512",
      "bid": [
       {
        "id": "1",
        "impid": "102",
        "price": 9.43,
        "nurl": "http://adserver.com/winnotice?impid=102",
        "iurl": "http://adserver.com/pathtosampleimage",
        "adomain": [
         "advertiserdomain.com"
        ],
        "cid": "campaign111",
        "crid": "creative112",
        "attr": [
         1,
         2,
         3,
         4,
         5,
         6,
         7,
         12
        ]
       }
      ]
     }
    ]
}
```

6.5.2 Example 2 – VAST XML Document Returned Inline

Following is an example of a bid response that returns the VAST document inline to be served. A few notes about specific fields in the example:

- The bid for this impression is a $3.00 CPM.
- Note that since there both a win notice URL and an inline VAST document in the `adm` attribute, which constitutes the ad markup. The win notice is still called, but if it were to return markup it would be ignored in favor of the contents of the `adm` attribute.


```
{
  "id": "123",
   "seatbid": [
    {
      "bid": [
       {
        "id": "12345", "impid": "2", "price": 3.00, "nurl": "http://example.com/winnoticeurl",
        "adm": "<?xml version=\"1.0\" encoding=\"utf-
8\"?>\n<VAST version=\"2.0\">\n<Ad id=\"12345\">\n<InLine>\n<AdSystem version=\"1.0\">SpotX
change</AdSystem>\n<AdTitle>\n<![CDATA[Sample VAST]]>\n</AdTitle>\n<Impression>http://sam
ple.com</Impression>\n<Description>\n<![C DATA[A sample VAST feed]]>\n</Description>\n <Creati
ves>\n<Creative sequence=\"1\" id=\"1\">\n<Linear>\n<Duration>00:00:30</Duration>\n <Tracking
Events>< /TrackingEvents>\n<VideoClicks>\n<ClickThrough>\n<![CDATA[http://sample.com/openrt
b test]]>\n</ClickThrough >\n</ VideoClicks >\n< MediaFiles >\n < MediaFile delivery =\"progressive
\" bitrate=\"256\" width=\"640\" height=\"480\" type=\"video/mp4\">\n<![CDATA[http://sample.co
m/video.mp4]]>\n< /MediaFile>\n</MediaFiles >\n < /Linear>\n< /Creative>\n</Creatives >\n < /In
Line>\n</Ad >\n < /VAST>"
}
   ]
}
]
}
```

6.5.3 Example 3 – Direct Deal Ad Served on Win Notice

Following is an example of a bid response with the ad served on win notice. The bid for this impression is a $5.00 CPM against a direct deal.

```
{
  "id": "1234567890",
   "bidid": "abc1123",
    "cur": "USD",
     "seatbid": [
      {
       "seat": "512",
       "bid": [
        {
         "id": "1",
         "impid": "102",
         "price": 5.00,
         "dealid": "ABC-1234-6789",
         "nurl": "http: //adserver.com/winnotice?impid=102",
         "adomain": [
          "advertiserdomain.com"
         ],
         "iurl": "http: //adserver.com/pathtosampleimage",
         "cid": "campaign111",
         "crid": "creative112",
         "adid": "314",
         "attr": [
          1,
          2,
          3,
          4
         ]
        }
       ]
      }
     ]
}
```

6.5.4 Example 4 – Native Markup Returned Inline

Following is an example of a bid response that returns a native ad inline to be served. The `adm` attribute contains an encoded string of a native ad request that conforms to the Dynamic Native Ads API and specifically the same version as that used for the request string. Alternatively, the `adm` attribute could have been omitted in favor of returning the native ad markup in the response to the win notice `nurl`.

```
{
 "id": "123",
  "seatbid": [
   {
    "bid": [
     {
      "id": "12345", "impid": "2", "price": 3.00, "nurl": "http://example.com/winnoticeurl",
      "adm": "{\"native\":{\"ver\":\"1.0\",\"link\":{ ... },
      \"imptrackers\":[ ... ],\"assets\":[ ... ]}}"
}
    ]
   }
  ]
}
```

7 Implementation Notes

The following section will provide brief notes on how certain objects and fields are to be interpreted and implemented.

7.1 No-Bid Signaling

This section covers best practices for using the optional no-bid signaling. See the List: No-Bid Reason Codes in OpenRTB 3.0 for the enumerated list of no-bid reason codes.

Many exchanges support multiple response types as a no-bid:

- HTTP 204 “No Content” from the bidder *(most economical in terms of bandwidth)*.
- An empty JSON object:
  `{}`
- A well-formed no bid response:
  `{"id": "1234567890", "seatbid": []}`
- A well-formed no bid response with a reason code:
  `{"id": "1234567890", "seatbid": [], "nbr": 2}`

An important issue in RTB is when impressions are triggered by software robots mimicking web browsers. Such robots may be implicitly or explicitly driving these false transactions. The following represents a set of symmetric best practices for exchanges and bidders to help recognize and reject these events.

Responsibility of the exchange

Make best effort to classify and reject "non-human traffic" requests for ads to the exchange via the following best practices:

- (Recommended) Filter impressions from known spiders via user-agent classification.
- (Recommended) Filter impressions from suspected NHT via a "detector".

Responsibility of the bidder

- (Recommended) no-bid impressions from known spiders via user-agent classification.
- (Recommended) no-bid impressions from suspected NHT via a "detector".
- Specify a no-bid reason code in either case.

Where:

- For exchanges, filtering the impression means that the exchange should respond to the "ad call" with either a blank HTTP 204 response or an unpaid ad (PSA) and not offered to any bidders.
- For bidders, filtering the impression means that the bidder should respond with a no-bid.
- For both exchanges and bidders, the impression transaction records should be clearly marked in any logging systems and be removed from contributing to any event counts associated with planning, forecasting, and reporting systems.

7.2Impression Expiration

Recapping the typical impression flow through RTB, an ad will be requested by a client (e.g., web browser, mobile app or an SDK therein) possibly through other server intermediaries, and ultimately to the RTB exchange. The exchange conducts an auction among buyers who bid with a proposed price, possibly markup for use if the bid wins (markup can also be delivered on the win notice itself), and other metadata about the bid. The exchange then selects a winner, issues a win notice to the winning bidder, and passes the markup back to the client.

Winning the auction, however, does not guarantee that the ad will be successfully delivered to the client or that it will meet viewability expectations. Furthermore, policies vary among exchanges as to the criteria for billing. Most consider an ad billable upon some form of delivery or rendering vs. the auction win alone. This aligns better with the buyer's obvious goal of ensuring that the impressions they pay for are actually displayed.

Some exchanges attempt to facilitate this alignment by placing the win notice in the winning ad markup so that it can serve as both a win notice and rendering notice. This is neither endorsed nor prohibited by OpenRTB except that it precludes the exchange from accepting markup on the win notice return as described in Section 4.3.1. Similarly, many buyers use their own tracking URL placed within their ad markup to signal rendering independent of the OpenRTB auction win notice. In video specifically, VAST supports an impression tracking URL that is often used for billing and is always distinct from the auction win notice.

To abstract the concept, let us refer to "*billing notice*" as the firing of some notification URL at the time when the clearing price of the impression will be booked as spend. This is irrespective of whether the actual OpenRTB win notice URL is delegated to the client for firing or some other tracking URL is used.

For buyers, this billing notice is used to book progress toward spend goals and frequency caps and drive pacing algorithms. When the billing notice is delayed significantly, these critical functions can be seriously impaired. There are legitimate reasons for some delays such as caching. A common scenario is a video interstitial impression in a mobile app. Refining the example, consider a game where the video is prefetched during game play so that it can be shown after the current game level ends. This is important for the user experience but can delay the rendering of the ad for many minutes.

Bidders are strongly advised to track the time between the auction and the win and/or billing notices to ensure reasonable delays. If unreasonable delays are encountered frequently, bidders may elect to ignore such events and bring them to the attention of the exchange for resolution. Unfortunately, the sequence from ad request through the auction and finally to rendering and billing is fundamentally not transactional. There are simply too many parties, policies, and technologies involved and thus a good support relationship between exchange and buyer is still important.

The OpenRTB protocol does provide some real-time assistance, however. The `imp.exp` attribute (Section 3.2.4) in the bid request allows an exchange to provide guidance to bidders of the number of seconds that may elapse between the auction and the billing event. As usual, omitted means unknown. Bidders can then decide if they want to bid understanding the likely delay. Bidders are advised, however, to interpret this as guidance as opposed to a contract unless the exchange expresses otherwise since exchanges are not always in a position to make hard guarantees (e.g., the SDK within the client app may not be under the exchange's control).

Similarly, the `bid.exp` attribute (Section 4.2.3) in the bid response allows the bidder to express the maximum number of seconds they are willing to tolerate between auction and billing notice. This allow the exchange to drop bids with expiration constraints it believes are likely to be violated. Bidders should not assume that a delayed billing notice greater than their specified bid expirations will not be billable. That is a policy and contract discussion between bidder and exchange and not imposed by OpenRTB.

The following expiration times are offered as examples of reasonable delays based on the nature of the impression. These are only provided as rules of thumb. A more data-driven method of determining these times in specific situations is highly recommended.

- Desktop and mobile web browsers: 1 Minute
- Mobile app banner ads that may be cached: 5 Minutes
- Mobile app native ads that may be cached: 10 Minutes
- Mobile and video interstitials: 30 Minutes *(or even longer)*
- Audio or video with server-side stitching: Very Long or Unknown

7.3 PMP & Direct Deals

Best Practice Bidding Logic

```
Receive request and parse;
Create empty bid list for response;

 If request contains the impression[].pmp object; match bids against each pmp.deals[];
enforce targeting for dealID and seatID; append best M matching bids to response;

If pmp.private_auction = False;
match open auction bids against the request; append top N bids by price to response;
Return response list to exchange;
```

Recommendations

- M >= 1, preferably one per matching Deal ID.
- N >= 2 to assist with blocking rate issues.
- Minimum viable is “1+1” bidding.
- Ideal is “M+N” bidding.

Warning

Returning only one bid when both Deal ID and open auction bids are valid creates problems. The exchange side may be configured by a publisher to prioritize all Deal ID bids above open auction bids, or to force a price auction between them with different floors by class of bid. There are multiple common practices that depend on how the publisher prefers to sell inventory with Deal ID.

Policy Recommendations

- A Deal ID should be utilized for any situation where the auction may be awarded to a bid not on the basis of price alone. Any prioritization of bids other than by price should have a Deal ID.
- A Deal ID is recommended for all situations where a preferential floor may be assigned to a seat entity.

Anti-Patterns

The below is a set of anti-patterns that OpenRTB supporting platforms have observed in various attempts to implement Deal ID bidding logic.

Subjecting Deal ID Bids to an internal auction on price

The ideal bidding logic describes a process of being liberal about sending bids. Deal ID bids may not be subject to a classic price auction. There may be an expectation that the buyer and seller want

prioritization to achieve a larger objective: complete delivery of the Deal represented by the Deal ID. Thus any bidding logic that sorts Deal ID bids by price (with or without open marketplace bids) and truncates the list too aggressively can endanger the fulfillment of the Deal.

Associating Deal ID to the wrong Object

A Deal ID should be treated as a "targeting token" associated to orders, line-items or campaigns. If the Deal ID is associated to a Seat/Buyer it may create an undesired application of the Deal ID too many active campaigns. Alternatively if it is associated to the Advertiser it may limit that entity to only a single Deal ID.

Improper Handling of the Private vs Open Market Flag

The `pmp.private_auction` flag indicates that the seller is willing or not willing to accept open market bids (i.e., "all bidders are welcome"). If this flag is not read and interpreted correctly, bid responses may be invalid. Open market bids sent to a private impression auction may be rejected and should not have been exposed to all bidders.

Improper handling of Seat IDs

If Seat IDs are treated as a filter of eligible demand partners on an open market impression, this defeats the "all bidders are welcome" intention.

Silently Applying Margin Discounts to Deal ID Bids

With Deal ID buyers are sellers are communicating directly. The Exchange and Bidder become third-party automation platforms. If there are any automatic or silent discounts of bid prices (based upon margins or fees) set by either the exchange or the bidder, then the Deal may fail to function correctly.

Use cases

Case-1: Open Trading Agreement with Buyer

- Between publisher and buying entity.
- Publisher sets an access rule defining the price floor for a specific buyer.
- Locked to the buyer.
- Broadcast price floor.
- Public/open inventory.
- No Deal ID needed (Deal ID is optional).
- No named advertiser(s).
- No prioritization of bids.
- Daily total or frequency caps optional on publisher/exchange side.
- All placements or limited to specific placements.
- Targeting is up to the buyer/bidder.

Case-2: Open Trading Agreement with Buyer with Named Advertisers

- As Case-1 with a list of named advertisers.

Case-3: Open Bidding with Deal ID as Value-added Markers

- Between publisher and buying entity.
- Publisher sets a price floor for URL masked inventory.
- Public/open inventory (i.e., all buyers welcome).
- Deal ID represents “Package Tokens”.
- Each Deal ID signals that the impression falls into various content and placement categories.
- Floor is associated to each Deal ID to signal cost for usage of that token.
- Winner is decided by bid price.
- Execution of targeting is up to the buyer/bidder.

Case-4: First Look Trading Agreement

- Between publisher and buying entity.
- Publisher sets an access rule defining the price floor for the buyer.
- Locked to the buyer.
- Known price floor.
- Deal ID needed.
- Optional named advertiser list.
- Prioritization of bids expected.
- Daily total or frequency caps optional on publisher/exchange side.
- All placements or limited to specific placements.
- Targeting is up to the buyer/bidder.

Case-5: Direct Option Deal with Advertiser via RTB

- Between Publisher and Advertiser or their representative.
- Publisher sets a rule defining a price floor and prioritization for specific advertiser(s).
- Fill rate is expected to be greater than or equal to X%.
- Locked to the buyer.
- Private/exclusive inventory.
- Limited to a set list of advertiser names (generally variants of one name).
- Known price floor.
- Deal ID needed.
- Prioritization of bids expected.
- Daily total or frequency caps will apply on bidder side; optional on Exchange side.
- Limited to specific placements.
- Targeting is mostly enforced by buyer/bidder.

Case-6: Direct Option Deal with Advertiser via RTB with Private Data

- Same as Case-4.
- Deal ID represents some combination of private first-party data from the Publisher.

Case-7: Full-Fill Direct Deal with Advertiser via RTB

- Same as Case-4.
- Fill rate is expected to be 100% or nearly so.

Case-8: Full-Fill Direct Deal with Advertiser via RTB with Private Data

- Same as Case-6.
- Deal ID represents some combination of private first-party data from the Publisher.

7.4 Skippability

This section clarifies the common use cases related to declaring skippability of video creatives.

Under most circumstances for RTB transactions, publishers and exchanges prefer to control the ability to skip the ad. OpenRTB therefore assumes by default that a standard linear video ad can be used as the response to a skippable request and the ability to skip the ad will be provided by the supplier's player automatically.

The presence of the `video.skip` attribute in the bid request with a value of "1" should be assumed to mean that the publisher will impose a skip button on the ad. The absence of the `video.skip` attribute should be assumed to mean that it is unknown whether the publisher will impose a skip button.

DSPs should confirm with publishers whether it is permissible to respond with ads that provide their own skip functionality (e.g., using VPAID to render a skip button). If bidding with such an ad and only if doing so, the bid must indicate creative attribute "16" using the `attr` array in the bid response.

Note: VAST 4.0 separates VPAID interactivity from the media file so this is deprecated and only applies to earlier versions of VAST.

Some examples of these concepts follow:

Bid Request

Case-1: Skippable after N Seconds for All Creatives

In this case, the publisher will impose skippability. All ads will be skippable, but only after 5 seconds of play. Creatives with a total duration of 5 seconds or less would not be skippable since they would never reach this threshold.

```
"video": {
..., "skip": 1, "skipafter": 5, ...
}
```

Case-2: Skippable after N Seconds for Minimum Duration Creatives

In this case, the publisher will impose skippability. However, only creatives with a total duration greater than 15 seconds will be skippable. For ads that satisfy this minimum total duration, skippability is enabled after 5 seconds of play. Note that although these values are integers, they will compare as precise values with actual video durations. For example, a video with duration 15.1 seconds does satisfy a `skipmin` value of 15 (i.e., think of the `skipmin` value as being 15.0).

```
"video": {
..., "skip": 1, "skipmin": 15, "skipafter": 5, ...
}
```

Case-3: Non-Skippable unless Requested by the Ad Markup

In this case, the publisher will not impose skippability. Ads will only be skippable if requested by the ad markup. This is supported by VPAID and VAST 3.0, for example.

```
"video": {
..., "skip": 0, ...
}
```

Case-4: Unknown Skippability

In this case, the `skip` attribute is omitted which indicates that exchange does not know if skippability will be imposed by the publisher. This may be the case, for example, when the exchange is not an SSP and thus may not have control or full knowledge of the publisher's intentions.

Bid Response

Consider Case-3 above, where the publisher does not impose skippability. If the ad markup itself will request skippability (e.g., via VPAID or VAST 3.0), then the bid must signal this intention. This is accomplished by including creative attribute 16 (i.e., Skippable) in the bid as shown below. If the markup is not going to request skippability, then this creative attribute should not be indicated.

When responding to Case-3 with this skippable attribute specified in the bid, the publisher should provide skippability either by instructing the VAST 3.0 player to activate skippability (refer to the VAST 3.0 “skipoffset” attribute) or by allowing the ad to render its own skip button using VPAID.

```
"bid": {
..., "attr": [16], ...
}
```

In Case-1 and Case-2 where the publisher may impose its own skippability, creative attribute 16 should not be specified. Furthermore, publishers are advised to filter responses containing attribute 16 since this could conflict with the skip button rendered by the publisher. When using a VAST 3.0 response, publishers may choose to implement support for VAST 3.0 “skipoffset” at their discretion and ads should be assumed to play non-skippable if the player does not support it.

7.5 Regs Resources

The regs object contains any legal, governmental, or industry regulations that the sender deem applicable to the request.

Please see the below resources for more details and framework specifications should you choose to implement them:

GDPR (General Data Protection Regulation)

https://github.com/InteractiveAdvertisingBureau/GDPR-Transparency-and-Consent-Framework

CCPA (California Consumer Privacy Act)

https://github.com/InteractiveAdvertisingBureau/USPrivacy

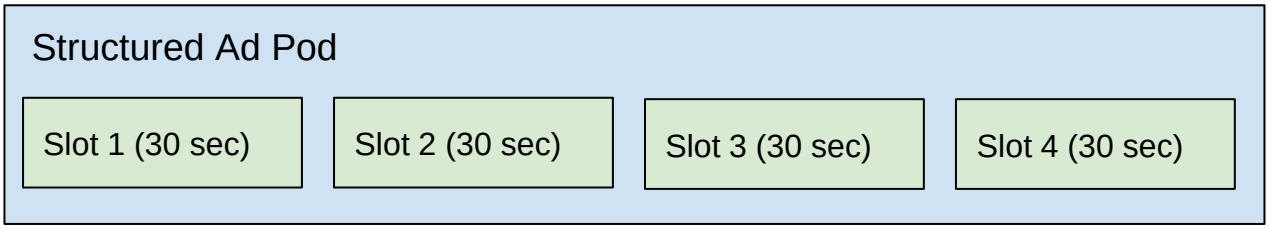
7.6 Pod Bidding for Video and Audio

Starting in version 2.6, OpenRTB now supports 'pod bidding' for video and audio content streams. An ad pod is the term describing an ad break of the type you'd see in a TV-like viewing experience or hear on a radio stream. An ad pod typically contains one or more in-stream creative assets that play out contiguously within a stream of video or audio content. Ad podding features in OpenRTB 2.6 build on capabilities in previous versions for including multiple ad requests within a single bid request object to indicate those ad requests are in some way related. Pod bidding signals communicate additional information about the pod & impression opportunities within the pod such as the sequence of the ad impressions, total pod length, maximum # of ads within a pod, multiple pod associations, and more.

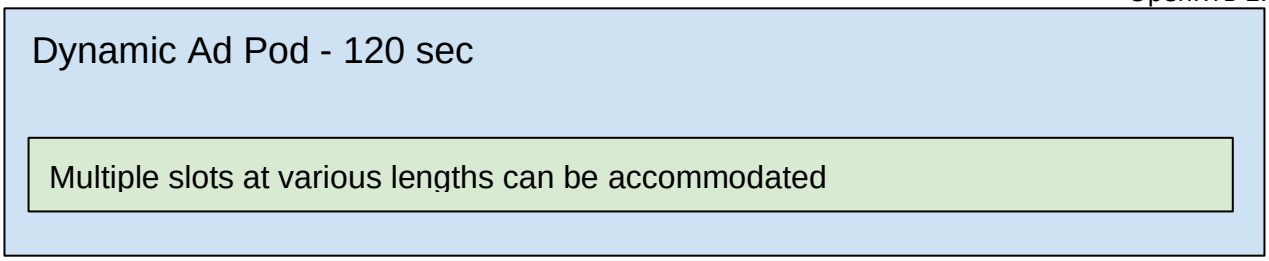
Terminology

Ad Slot: Space for an individual ad impression within a pod.

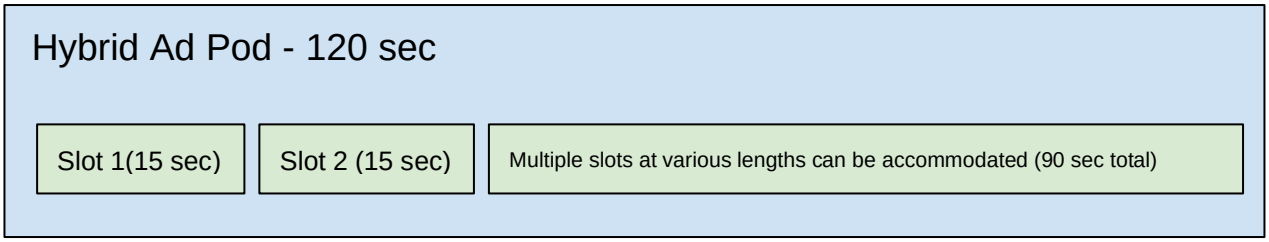
Structured Pod: The seller offers a fully defined pod structure; the number of ad slots, their slot in the ad pod, and duration is pre-defined and static.



Dynamic Pod: The seller offers a pod structure where the number of ads & the duration of each ad in the break is indeterminate, but the total duration and maximum number of ads are constrained. In other words, the total duration of the pod is known, but the number and durations of the individual ads within the break may not be defined ahead of time. This allows bidders more flexibility to optimize their selection of ads across the demand on their platform.



Hybrid Pod: The seller offers a pod structure containing BOTH structured and dynamic components. In other words, the ad pod is composed of some combination of ad slots with predetermined durations, and ad slots constrained by a total duration & maximum number of ads.



Recommendations

- Sellers should only indicate a *slotinpod* of 1, 2, or -1 if they can absolutely guarantee placement of an ad within the first or last slot of an ad pod.
- Buyers should only indicate a *slotinpod* in response to a dynamic pod segment, including a dynamic component of a hybrid pod, if they only want to buy the first or last ad slot specifically
  - Note that buyers should *only* return a *slotinpod* value in response to a Dynamic portion of a pod.
- Buyers should avoid using the *slotinpod* field in bid responses for structured pods, or the structured components of hybrid pods, because the *impid* field already uniquely identifies the ad slot.
- Buyers should look for mincpmpersec when available, otherwise fall back to bidfloor
- Sellers should include either the required durations field (communicating exact durations) OR the max & min duration fields, but not both.
- Sellers are encouraged to include the maxseq field when offering a dynamic pod with a pod duration
- Sellers are encouraged to offer dynamic pods when possible to allow bidders to source the most optimal demand from their platforms
- Buyers should expect that final pod construction is done by the seller. Buyers who submit N bids for a particular pod may find that the seller selects anywhere between 0 to N of those bids to construct the pod that is shown to the user. Furthermore, the seller may co-mingle bids from other buyers in that pod.

Pod bidding example scenarios

"Structured" Ad Pod Request/Response

This scenario illustrates an example where the bid request contains 2 structured ad pods, and the response corresponds to the first positions in each of the 2 signaled pods.

BidRequest

```
{
        "imp": [{
                        "id": "1",
                        "video": {
                                "podid": "pod_1",
                                "podseq”: 1,
                                "slotinpod": 1,
                                "mimes": [
                                        "video/mp4",
                                        "video/ogg",
                                        "video/webm"
                                ],
                                "linearity": 1,
                                "maxduration": 60,
                                "minduration": 0,
                                ...
                        },
                        "exp": 7200,
                        "bidfloor": 8,
                        "bidfloorcur": "USD",
                },
                {
                        "id": "2",
                        "video": {
                                "podid": "pod_1",
                                "podseq": 1,
                                "slotinpod": 0,
                                "mimes": [
                                        "video/mp4",
                                        "video/ogg",
                                        "video/webm"
                                ],
                                "linearity": 1,
                                "maxduration": 30,
                                "minduration": 0,
                                ...
                        },
                        "exp": 7200,
                        "bidfloor": 8,
                        "bidfloorcur": "USD",
                        ...
                },
                {
                        "id": "3",
                        "video": {
                                "podid": "pod_2",
                                "podseq": 0,
                                "slotinpod": 1,
```

```
                        "mimes": [
                                "video/mp4",
                                "video/ogg",
                                "video/webm"
                        ],
                        "linearity": 1,
                        "maxduration": 30,
                        "minduration": 0,
                        ...
                },
                "exp": 7200,
                "bidfloor": 8,
                "bidfloorcur": "USD",
                ...
        },
        {
                "id": "4",
                "video": {
                        "podid": "pod_2",
                        "podseq": 0,
                        "slotinpod": 0,
                        "mimes": [
                                "video/mp4",
                                "video/ogg",
                                "video/webm"
                        ],
                        "linearity": 1,
                        "maxduration": 60,
                        "minduration": 0,
                        ...
                },
                "exp": 7200,
                "bidfloor": 8,
                "bidfloorcur": "USD",
                ...
        }
    ],
    ...
}
```

BidResponse

```
{
    "id": "9b9ee818a85d948d5231ffe839a9729a",
    "seatbid": [{
                "bid": [{
```

```
                                "id": "1",
                                "impid": "1",
                                "price": 0.27,
                                "adid": "123456",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserA.com"
                                ],
                                "cid": "456789",
                                "crid": "123456",
                                "dur": 30
                        }],
                        "seat": "1"
                },
                {
                        "bid": [{
                                "id": "2",
                                "impid": "3",
                                "price": 0.27,
                                "adid": "234567",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserB.com"
                                ],
                                "cid": "567890",
                                "crid": "234567",
                                "dur": 15
                        }],
                        "seat": "2"
                }
        ],
        "cur": "USD"
}
```


"Dynamic" Ad Pod Request/Response

This scenario illustrates an example where the bid request contains 1 dynamic pod, the publisher can guarantee delivery against the first or last slot, and the response contains 3 bids from 3 different advertisers, for the signalled pod. The first bid in the response is only eligible for the first position in the pod.

BidRequest
{

```
        "imp": [{
                "id": "1",
                "video": {
                        "podid": "preroll_pod",
                        "mimes": [
                                "video/mp4",
                                "video/ogg",
                                "video/webm"
                        ],
                        "linearity": 1,
                        "maxduration": 60,
                        "minduration": 0,
                        "maxseq": 4,
                        "poddur": 60,
                        "slotinpod": 2,
                        ...
                },
                "exp": 7200,
                "bidfloor": 8,
                "bidfloorcur": "USD",
                ...
        }],
        ...
}
```

BidResponse

```
{
        "id": "9b9ee818a85d948d5231ffe839a9729a",
        "seatbid": [{
                        "bid": [{
                                "id": "1",
                                "impid": "1",
                                "price": 0.27,
                                "adid": "123456",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserA.com"
                                ],
                                "cid": "456789",
                                "crid": "123456",
                                "slotinpod": 1,
                                "dur": 30
                        }],
                        "seat": "1"
```

```
                },
                {
                        "bid": [{
                                "id": "2",
                                "impid": "1",
                                "price": 0.27,
                                "adid": "234567",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserB.com"
                                ],
                                "cid": "567890",
                                "crid": "234567",
                                "dur": 15
                        }],
                        "seat": "1"
                },
                {
                        "bid": [{
                                "id": "3",
                                "impid": "1",
                                "price": 0.27,
                                "adid": "345678",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserC.com"
                                ],
                                "cid": "678901",
                                "crid": "345678",
                                "dur": 15
                        }],
                        "seat": "2"
                }
        ],
        "cur": "USD"
}
```


"Hybrid" Ad Pod Request/Response

This scenario illustrates an example where the bid request contains slot 1 in the first impression object and a dynamic pod to fill the duration. The response contains 3 bids from 3 different advertisers, for the signalled pod. The first bid in the response is only eligible for the first position in the pod and the second two responses would be to fill the remainder of the pod.

BidRequest

```
{
        "imp": [{
                        "id": "1",
                        "video": {
                                "podid": "pod_1",
                                "slotinpod": 1,
                                "mimes": [
                                        "video/mp4",
                                        "video/ogg",
                                        "video/webm"
                                ],
                                "linearity": 1,
                                "maxduration": 15,
                                "minduration": 15,
                                ...
                        },
                        "exp": 7200,
                        "bidfloor": 15,
                        "bidfloorcur": "USD",
                        ...
                },
                {
                        "id": "2",
                        "video": {
                                "podid": "pod_1",
                                "slotinpod": 0,
                                "mimes": [
                                        "video/mp4",
                                        "video/ogg",
                                        "video/webm"
                                ],
                                "linearity": 1,
                                "maxduration": 60,
                                "minduration": 0,
                                "maxseq": 4,
                                "poddur": 60,
                                ...
                        },
                        "exp": 7200,
                        "bidfloor": 8,
                        "bidfloorcur": "USD",
                        ...
                }
        }],
        ...
}
```

BidResponse


```
{
        "id": "9b9ee818a85d948d5231ffe839a9729a",
        "seatbid": [{
                        "bid": [{
                                "id": "1",
                                "impid": "1",
                                "price": 20.27,
                                "adid": "123456",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserA.com"
                                ],
                                "cid": "456789",
                                "crid": "123456",
                                "dur": 15
                        }],
                        "seat": "1"
                },
                {
                        "bid": [{
                                "id": "2",
                                "podid": "pod_1",
                                "impid": "2",
                                "price": 9.27,
                                "adid": "234567",
                                "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                "adomain": [
                                        "advertiserB.com"
                                ],
                                "cid": "567890",
                                "crid": "234567",
                                "dur": 15
                        }],
                        "seat": "1"
                },
                {
                        "bid": [{
                                "id": "3",
                                "podid": "pod_1",
                                "impid": "2",
                                "price": 10.27,
                                "adid": "345678",
```

```
                                    "adm": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><VAST
version=\"2.0\"> ...",
                                    "adomain": [
                                            "advertiserC.com"
                                    ],
                                    "cid": "678901",
                                    "crid": "345678",
                                    "dur": 15
                            }],
                            "seat": "2"
                    }
            ],
            "cur": "USD"
}
```


7.7 Network vs Channel Example Cases

Starting in version 2.6, OpenRTB now supports Network and Channel objects. See 3.2.23 and 3.2.24 for details).While these examples are straight forward for traditional linear television, the options for CTV consumption warrant a few examples.

Example 1: A user viewing content on an internet connected device, on an app with multiple channel options (e.g. Discovery+ App > HGTV Channel/show)

- Discovery is the Network
- HGTV is the Channel

Example 2: A user viewing content on an internet connected device, on an app that streams content directly (Roku > Hulu > Hulu Original show)

- Hulu is the network (also identified by bundleID)
- Hulu is the channel

Example 3: A user viewing content on an internet connected device, on a device offered channel (Roku > Fubo > Comedy Central show)

- Roku is the device
- FuboTV is the network (also identified by bundleID)
- Comedy Central is the channel

Example 4: A user is viewing content on an internet connected device, on a device offered channel that licenses content (Samsung TV > Pluto > Pluto TV Spotlight)

- Samsung TV is the device
- Pluto is the network (also identified by bundleID)
- PlutoTV Spotlight is the channel

7.8 Counting Billable Events and Tracked Ads

There are multiple conventions for how to count billable events or tracked ads via OpenRTB, typically an impression or other such common metric. This section outlines the common ones, addresses common mistakes, and offers a comparison of the approaches.

This section addresses technical methods available for implementers to consume these events. These events have specific business definitions and criteria for counting eligibility set by the Media Rating Council, and implementers should also consult the Media Rating Council's guidelines.

Implementers should discuss the definition of the billable event and the technical basis for counting it with their counterparties to determine a mutually acceptable approach.

Overview of counting methodologies

Method	Remarks
Pixel in markup	• Widely supported • Normally fired from client-side browser • Prone to discrepancies • May overcount in some circumstances (i.e. mobile app) • Only applicable for display
VAST <Impression> event	• Recommended for audio/video • Only applicable for audio/video
Billing notice ("burl")	• Best alignment between DSPs and exchanges to count tracked ads • Minimal discrepancy • Not recommended for audio/video, otherwise applicable to all creative types • Usually (and recommended to be) fired server-to-server, but based on an initial client-side event
Native eventtrackers/imptrackers/jstrackers	• Only applicable to native ads

Pixel in markup (banner ads)

This is the original convention for counting impressions/tracked ads; in this method, the OpenRTB specification itself does not address how to receive the events. The bidder self-embeds a tracking pixel in their HTML markup (i.e. an <img> tag which makes a request to the bidder's servers). When the client device loads the markup, it fires the pixel.

For banner ads on web, this is a widely adopted approach to counting billable events/tracked ads, however some circumstances arise in which there may be discrepancies. Differences in timing between when an exchange's and a DSP's pixel load may result in discrepancies, and the noisy nature of the public Internet and variable connectivity quality of client devices may result in one pixel firing but not the other. Additionally, this method does not address certain scenarios well, namely:

- **Mobile apps** – a billable event may only be counted when the ad is displayed. In mobile apps, the markup is often fetched well in advance of being displayed to buffer against slow and unreliable connections. The markup may never be displayed if the user abandons the app before the ad is displayed – especially for interstitials.
- **Creative auditing** – markup may be loaded to scan for malvertising, etc., which may generate spurious extra billable/tracked ad events, including for unwon auctions.

BEST PRACTICE: When it is possible to do so, exchanges should avoid using adm-based notifications as the determinant for billing events in the mobile app context, and instead use burl or an independent measurement approach (e.g. OMID), that is predicated upon an ad actually being displayed to the user.

VAST <Impression> event (video/audio)

The VAST specification includes a provision for <Impression> objects, which demand chain participants can use to request notifications when a billable event has occurred. The IAB prescribes that for video, the VAST <Impression> event is the official signal that the billable event has occurred.

Demand chain participants are discouraged from using billing notice URLs (burl) for video/audio transactions.

Billing notice ("burl")

Billing notice support was introduced in OpenRTB 2.5. In this scenario, **outside** of the ad markup itself, a "billing notice URL" is included in the bid response. A billing event is when a transaction results in a monetary charge from the publisher to an exchange, and subsequently from the exchange or other intermediary to one of their demand-side partners. This event is subject to publisher and exchange-specific business policies that should be conveyed clearly to their partners. For a DSP, this event signals that they can increment spend and deduct the remaining budget against the related campaign. The exchange conveys this event by invoking the URL provided by the demand source in the bid.burl attribute.

BEST PRACTICE: Firing the billing notice URL represents the fulfillment of a business transaction between a publisher and an exchange, or between the exchange and its demand partner. This should not be delegated to another party including a client-side pixel, although a pixel may be the initiating signal for billing to the exchange.

BEST PRACTICE: Exchanges, upon determining that a billable event has occurred (e.g., receipt of client-initiated billable event), and in order to minimize discrepancies between themselves and their demand sources, should invoke the billing notice from the server-side. This should be done as "close" as possible to the moment when the exchange books revenue. See the below section regarding best practices for server-side billing notifications.

BEST PRACTICE: Exchanges are highly encouraged to standardize on a client-initiated render or viewability event as the basis for the billing event. This is generally the most consistent approach in a complex supply chain scenario composed of multiple auction decision points.

BEST PRACTICE: Publishers should generally refer to the Media Rating Council's guidelines to determine when the criteria have been met to consider a transaction billable.

BEST PRACTICE: The public internet is noisy and this event is financial in nature. If an entity calling a billing notice receives a response other than HTTP 200 or 204, it should consider a retry scheme (e.g., every 10

seconds for the next minute). Conversely, an entity receiving billing notices should endeavor to make their endpoint idempotent to avoid double counting.

BEST PRACTICE: When it is possible to do so, exchanges should avoid using adm-based notifications as the determinant for billing events in the mobile app context, and instead use burl or an independent measurement approach (e.g. OMID), that is predicated upon an ad actually being displayed to the user.

For VAST video/audio, if the bid.burl attribute is specified, it should be fired at the same time as the VAST <Impression> event. However, subtle technical issues may lead to additional discrepancies and bidders are cautioned to avoid this scenario. One option is for exchanges nearest a video supply source to use the VAST <Impression> event as their billing signal and then use the billing notice URL (burl) as described.

Native eventtrackers, imptrackers, jstrackers

For native ads specifically, the OpenRTB Native specification offers options for including an impression tracking URL or script to be loaded at impression time. See the OpenRTB Native spec for more information. For native video, most platforms utilize the impression events within VAST for billing and other event notifications, rather than the structured tracking options available within the native spec.

Win notice ("nurl") – not a billable or tracked ad event

At first glance, an auction "win" and the associated win notice ("nurl") field appears suitable as a proxy for billable/tracked ad counting. However, winning an auction does not guarantee that an impression will indeed be served, and in fact in many cases only a small percentage of won auctions will become impressions. This occurs because of downstream auctions (i.e. client-side header bidding), inability to play back media (in video and audio), etc.

Win notice URLs should never be used to count impressions or tracked ads.

Best Practices for server-side billing notifications

In some cases, publishers or their vendors may choose to fire impression notifications from a server. This is very common in long-form video, which uses server-side ad insertion to coordinate the delivery and measurement of ads to a "thin" client on the user's device. It is also common in mobile app, where the monetization SDK uses a server-side service to fire burl notifications.

The following best practice is derived from the VAST 4.2 spec (page 17), but recommended for any impression notification (for all formats, regardless of protocol or version).

BEST PRACTICE: When possible, exchanges are encouraged to send billing notice URL (burl) notifications from the server-side, to minimize discrepancies with demand partners. The billable event itself should originate from a client-side event per MRC guidelines.

BEST PRACTICE: When firing impression notifications via HTTP request from the server-side, the notifier should:

- Make use of the X-Forwarded-For or X-Device-IP HTTP header to indicate the IP address of the client device on behalf of which the notification is being sent.

- Make use of the X-Device-User-Agent HTTP header to indicate the UserAgent of the client device on behalf of which the notification is being sent.

These HTTP headers allow recipients of impression notifications to run anti-IVT checks using metadata about the end user device, rather than the server itself.

BEST PRACTICE: When firing impression notifications via HTTP request from the server-side, the notifier should establish an ads.cert Call Sign and make use of the ads.cert Authenticated Connections protocol to cryptographically sign notifications. This allows recipients of impression notifications, who've established ads.cert Call Signs of their own, to authenticate the sender for anti-fraud purposes.

Appendix A. Additional Information

- Creative Commons / Attribution License
  creativecommons.org/licenses/by/3.0
- IAB (Interactive Advertising Bureau)
  www.iab.com
- IAB Quality Assurance Guidelines (QAG):
  www.iab.com/guidelines/iab-quality-assurance-guidelines-qag-taxonomy/
- JavaScript Object Notation (JSON)
  www.json.org
- MMA (Mobile Marketing Association)
  mmaglobal.com
- OpenRTB Project on Github
  github.com/openrtb/OpenRTB/
- Apache Avro
  avro.apache.org

  Protocol Buffers (Protobuf)
  code.google.com/p/protobuf
- Google Metro Codes
  code.google.com/apis/adwords/docs/appendix/metrocodes.html
- U.N. Code for Trade and Transport Locations:
  www.unece.org/cefact/locode/service/location.htm

Appendix B. Specification Change Log

This appendix serves as an index of specification changes across 2.x versions. These changes pertain only to the substance of the specification and not routine document formatting, organization, or content without technical impact.

Version 2.5 to 2.6:

Section	Description
3.2.1 **3.2.16** **4.2.3** **3.2.18**	*Added new language field to support IETF BCP 47* IETF BCP 47 offers additional layers of granularity, for example, differentiating written language versions of the same spoken language (e.g. Traditional and Simplified Chinese) https://en.wikipedia.org/wiki/IETF_language_tag
5	*Removed section (Enumerated Lists)* All references now point to AdCOM 1.0 / OpenRTB 3.0 Lists
3.2.1 **3.2.13** **3.2.14** **3.2.15** **3.2.16** **3.2.17** **4.2.3**	*Objects: BidRequest, Site, App, Publisher, Content, Producer, Bid,* Use of cattax for all taxonomy references
3.2.7 **3.2.8**	*Objects: Video, Audio* Added rqddurs
3.2.7 **3.2.8**	*Objects: Video, Audio* Added maxseq, poddur, podid, podseq, mincpmpersec, slotinpod for pod bidding support Updated July 2022 podid is a string, there was a conflict between the examples and the specification, the specification was incorrectly labeled integer
4.4 **4.4.1**	*Substitution Macros* Added `AUCTION_MIN_TO_WIN`
4.2.3	*Object: Bid* Added apis
3.2.4	*Object: Imp* Added rwdd
3.2.1 **3.2.14** **4.2.3**	*Object: App, Bid, BidRequest* Clarified language around use of storeid vs bundle
3.2.4	*Object Imp* Added ssai

3.2.18	*Object: Device* Clarified language around mccmnc and roaming
3.2.23 **3.2.24**	*Added Objects: Network, Channel, SupplyChain, SupplyChainNode, EIDs and UIDs*
4.2.3	*Object Bid* Added mtype
3.2.6 **3.2.7** **3.2.16**	*Removed previously deprecated attributes* Object: Banner, wmax, hmax, wmin, hmin Object: Video, protocol Object: Content, videoquality
7.6	*Pod Bidding for Video and Audio implementers guide*
7.7	*Network and Channel object examples*
3.2.7 **3.2.8** **3.2.18** **3.2.20** **4.2.3**	*Deprecated attributes* Object: Video, sequence Object: Audio, sequence Object: Device, didsha1, didmd5, dpidsha1, dpidmd5, macsha1, macmd5 Object: User, yob, gender Object: Bid, api
7.8	*Added Counting Billable events and tracked ads*
3.2.29 **3.2.30**	*Object: UserAgent & Object Brand Version added*

Version 2.4 to 2.5:

Section	Description
2.4	*Section: Data Encoding* New section added.
3.1	*Object Model: Bid Request* Updated to include `Source` and `Metric` objects.
3.2.1	*Object: BidRequest* Attributes `bseat`, `wlang`, and `source` have been added.
3.2.2	*Object: Source* New `Source` object has been added including the Payment ID `pchain` attribute.
3.2.4	*Object: Imp* Attribute `metric` has been added.

3.2.5	*Object: Metric* New `Metric` object has been added.
3.2.6	*Object: Banner* Attribute `vcm` has been added.
3.2.7	*Object: Video* Attributes `placement` and `playbackend` have been added. Guidance added to use only the first element of attribute **`playbackmethod`** in preparation for future conversion to an integer.
3.2.10	*Object: Format* Attributes `wratio`, `hratio`, and `wmin` have been added.
3.2.13	*Object: Device* Attribute `mccmnc` has been added. Attribute `carrier` has been clarified to eliminate a reference to using "WIFI" as a carrier.
4.2.3	*Object: Bid* Attributes `burl`, `lurl`, `tactic`, `language`, `wratio`, and `hratio` have been added.
4.4	*Substitution Macros:* Macros ${AUCTION_MBR} and ${AUCTION_LOSS} have been added. A best practice has been added to use "AUDIT" for unknown values when rendering for test or quality purposes.
5.6	*List: API Frameworks* Item 6 has been added.
5.9	*List: Video Placement Types* New list has been added.
5.10	*List: Playback Methods* Items 5-6 have been added.
5.11	*List: Playback Cessation Modes* New list has been added.
5.24	*List: No-Bid Reason Codes* Items 9-10 have been added.
5.25	*List: Loss Reason Codes* New list has been added.