OpenRTB 3.0 Framework

Launching Secure Supply Chain Standards

Draft for Public Comment

September 2017
Introduction

OpenRTB 3.0 is the largest overhaul of the OpenRTB protocol since its inception in 2010. The purpose of this major update is to meet the market’s demand for security, transparency, authentication, and trust in programmatic advertising. Specifically, three major changes (along with many others) are being proposed:

- A signed supply chain to enable the demand side to validate many of the fields in the bid request and to have a clear chain of custody
- New ways to transmit display demand to enable safer creative controls for publishers
- An updated and more flexible object model in the specification

Along with the updates for OpenRTB, the IAB Tech Lab is also updating its take on cross-specification standards development. OpenMedia, announced earlier this summer, is an approach to specification development that allows for effective scalable iteration on programmatic specifications including OpenRTB.

This document is not a full specification. The timeline for the release of a full specification for OpenRTB 3.0 is detailed later in this document. We have released these major proposed updates for an extended public comment period. We encourage the industry to provide feedback on workflows and technical implementation of 3.0.

This document, OpenRTB 3.0 Framework, has been released in early September for a 3-month public comment period. Readers will find accompanying resources and some companion specifications that are available for public comment at this time. Comments and feedback may be submitted using this form. Public comments are open until December 15, 2017.

About the IAB Technology Laboratory

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

It was a simpler world when RTB was created, with shorter supply chains and fewer opportunities for fraud. With today’s complex and variable supply chains, there are many opportunities for fraud as an impression request makes its way to the buy-side, including:

- Domain fraud (to change the domain to a more valuable publisher)
- Location fraud (to change the IP address/location to a more valuable location)
- User ID fraud (to change the device ID or buyer id to an ID that has historically monetized well)
- Device fraud (to change from a type of device that doesn’t tend to monetize well (such as a smartphone) to one that does (such as a desktop))

Unfortunately, it’s difficult to detect these types of fraud in the current standard. Such detection is imprecise and only really works forensically, which is rather too late.

These and other types of fraud diminish the value that advertisers receive and the money that quality publishers receive, eroding overall trust in the ecosystem.

Conversely, publishers and publisher platforms have little ability to control the types of creatives they run on their site due to the opaque way in which traditional display ads are transmitted. Many types of undesirable creatives sometimes make their way to publisher pages, such as:

- Heavy payloads
- Non-brand-safe ads
- Heavy or malware JavaScript
- Excessive pixel fires

These behaviors result in bad user experiences and the installation of ad blockers, diminishing user trust and further eroding publisher monetization. Further, new legislation such as the General Data Protection Regulation (GDPR) in the EU may prove difficult to implement in the world of traditional display.

OpenRTB 3.0 attempts to extend prior solutions (such as payment chain and most recently ads.txt, Authorized Digital Sellers) to address many of these issues and improve the programmatic ecosystem for all parties (users, publishers, and advertisers).

As we move forward with OpenRTB 3.0, we encourage buyers to begin demanding authenticated supply from their partners and publishers to demand safe performant ads from their demand partners.
# Table of Contents

**OpenRTB 3.0 Framework**

- Introduction
  - About the IAB Technology Laboratory

**Table of Contents**

1. **A. OpenMedia Specification Landscape**
2. **B. Updates for OpenRTB 3.0**
3. **C. Complex Digital Advertising Supply Chain**
4. **D. OpenRTB Protocol Layers**
   - Layer-1: Transport
     - Communications
     - Version Headers
     - Security
   - Layer-2: Format
     - Representation
     - Encoding
   - Layer-3: Transaction
   - Layer-4: Domain
5. **E. OpenRTB Transaction Layer**
   - Object Model
     - Object: Openrtb
   - Bid Request Payload
     - Object: Request
     - Object: Source
     - Object: Offer
     - Object: Item
     - Object: Pmp
     - Object: Deal
   - Bid Response Payload
     - Object: Response
     - Object: Seatbid

---

[https://iabtechlab.com/openrtb](https://iabtechlab.com/openrtb)
Object: Bid

F. Updated Billing Notifications

Event Notification

Event: Billing

Event: Loss

Substitution Macros

Additional Implementation Guidance

Idempotency

Failure handling

Timing

Avoid holding state

Open Items for Consideration before OpenRTB 3.0 Final Specification

G. Domain Layer: Advertising Common Object Model

Principles of AdCOM

H. Authentication: Signed Bid Requests

I. Addressing Data Leakage

J. Ad Management API Specification

K. Consumer Identifier support

Best Practices

L. Additional OpenRTB Companion Specifications

M. Plan for Finalizing OpenRTB 3.0

Appendix A: List of Companion Specifications

Appendix B: Community Resources

Appendix C: Change Log
A. OpenMedia Specification Landscape

OpenMedia is the perspective and methodology for the specification landscape within which the IAB Tech Lab programmatic standards live. The approach is both conceptual (improving understanding of the specification landscape) and pragmatic (functioning as a tool to indicate where new standards will be most effective). OpenMedia describes a layered approach (similar to OSI network model) to organizing transactional specifications and domain specifications - or simply; a layer for how to transact, and a layer to describe what is transacted.

Open Media Specification Landscape

Layer 4  
Advertising Common Object Model (Ad-COM)

Layer 3  
Transaction
- OpenRTB
- OpenDirect
- Ad Management

Layer 2  
JSON, binary

Layer 1  
HTTP

Related Resources: 
- Ads.txt
- Validators

By using OpenMedia as the baseline for all programmatic standards, we will unify object models, create functionality between transactional specifications (unlocking sales opportunities to be built between RTB pipes and automated guaranteed pipes), and allow for rapid scalability of inclusion of new media objects (like digital out of home, etc.).

The names of specifications in layers 3 and 4 all refer to standards operated by IAB Tech Lab. OpenRTB, OpenDirect have been longstanding protocols. There are two new specifications in the landscape diagram: the Ad Management Specification and the Advertising Common Object Model (AdCOM). Ad Management Specification describes a standardized way to communicate information about ads (including ad approvals). AdCOM describes the ad, the creative, and the media channels that are traded over OpenRTB transactions, and, in future, OpenDirect.

Deconstructing the domain layer object model can give publishers control over the creative to be served on their page. More detail is available in section D. Protocol layers, Section G Domain Layer, and Section J. Ad Management.
B. Updates for OpenRTB 3.0

The goal of this document is to highlight important updates to the OpenRTB protocol, in order to better support transparent supply chain and innovative programmatic markets. See below for an overview of the proposed updates for OpenRTB 3.0, and why these updates are important for the digital advertising industry.

<table>
<thead>
<tr>
<th>Proposed Update</th>
<th>Why</th>
<th>What’s new</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated Object Model Approach: Layers</td>
<td>Scalability of new media channels, Effective across transactional specifications</td>
<td>Common Object Model for ad domain, and transaction layer objects for each programmatic spec</td>
</tr>
<tr>
<td>Transaction Layer</td>
<td>Utility of an open ad request, better maintenance and clearer implementation of RTB transactions</td>
<td>Updated object model including: request object, response object</td>
</tr>
<tr>
<td>Updated billing notifications</td>
<td>Aligning expected billing notification function in observed industry needs. Clarity in reconciliation of</td>
<td>Bid object and Item object, and new implementation guidelines</td>
</tr>
<tr>
<td>Domain Layer: Advertising Common Object Model (AdCOM)</td>
<td>Deconstructing creative object to support transparency of javascript trackers running on creatives (Aiding in publisher control of user experience) alignment with AMP, native, dynamic content and new ad portfolio standards, allows for better control of creatives</td>
<td>New companion AdCOM spec, highlighting the updates to ad object and creative object</td>
</tr>
<tr>
<td>Signed bid requests</td>
<td>Security and Authorization, allowing verification of an un-tampered inventory supply</td>
<td>Updates to the bid request and new message signing function</td>
</tr>
<tr>
<td>Trusted data providers</td>
<td>Solving for data leakage</td>
<td>Tdp element in bid request</td>
</tr>
<tr>
<td>Ad Management Spec</td>
<td>Provides a standardized means for demand and supply partners to communicate with each other regarding creatives that will be used in bidding</td>
<td>New companion specification</td>
</tr>
<tr>
<td>Consumer Identifier support</td>
<td>Cross device ID support (including Digitrust, etc)</td>
<td>New object array</td>
</tr>
</tbody>
</table>
C. Complex Digital Advertising Supply Chain

OpenRTB is generally thought of as the interaction between an exchange and its demand sources or sometimes as the auctioneer and its bidders. This was the underlying model through the v2.x versions. This can best be seen in the following figure if one focuses on the Demand Platform and Exchange-N; pretending that this is the only exchange in the picture and that the publisher feeds its supply directly.

With that focus, OpenRTB defines a method of selling micro-commodities (e.g., advertising impressions) by broadcasting bid requests from the exchange to its demand partners, collecting bids in response, and determining a winner based on some form of auction rules. Markup may be requested if it was not already included in the bid. When the sale becomes billable based on exchange business policy (e.g., rendered markup), the exchange notifies the winning buyer with critical data such as the clearing price. Others may be notified that they lost the auction.

Today’s ecosystem is much more complicated, however. Header bidding has become extremely common, which results in a new decision point at or near the publisher or user agent rendering the exchange as no longer the sole decider. It has also become common to have more than one supply chain intermediary; perhaps a mediator or exchange which uses another exchange as a demand source in addition to its own directly integrated demand.

These structures are depicted in the full reference model diagram, where each intermediary may use other intermediaries (to their right in the figure) as demand sources and each may either have the direct relationship with a publisher or bid into an upstream (to the left in the figure) intermediary. Note that although the figure shows the general case of N exchanges, practical considerations (e.g., latency) will tend to impose natural limits.

OpenRTB bid requests, bid responses, and events can be implemented between any pair of these entities although the publisher integrations, depicted here as publisher to Exchange-1, are usually less standardized. A given entity may or may not be aware of the larger supply chain, but consider that this chain may vary for different publishers. In any case, OpenRTB assumes
that a given entity only has a business relationship, at least within the context of a given transaction, with its directly implemented neighbors to the left (supply side) and the right (demand side).

The implications of this can be illustrated in event propagation and private marketplace deals. Speaking in the first person from the point of view of an entity in this reference model:

- I assume that all billing or loss events I receive are either from or with the full authority of my supply side partner (i.e., the entity to my immediate left).
- I am responsible for sending billing or loss events only to my demand-side partners (i.e., the directly integrated entities to my immediate right).
- I broker and execute deals among my supply and demand partners (i.e., the entities to my immediate left and right). No other entity has any obligation to extend or propagate my deals further up or down the chain.

D. OpenRTB Protocol Layers

To assist in reuse of objects across different specifications and to enable various aspects of the specification to evolve at different paces, a layered approach is being adopted as of OpenRTB v3.0. The following illustrates this model. Expressed informally, Layer-1 moves bytes between parties, Layer-2 expresses the language of these bytes, Layer-3 specifies a commerce transaction using this language, and Layer-4 describes the goods being transacted.

<table>
<thead>
<tr>
<th>Layer - 4</th>
<th>Domain</th>
<th>Objects Describing the Subject of the Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer - 3</td>
<td>Transaction</td>
<td>Objects, Events, Macros, etc. Used to Perform Functions with Domain Objects</td>
</tr>
<tr>
<td>Layer - 2</td>
<td>Format</td>
<td>Method of Encoding Data Defined in Higher Layers</td>
</tr>
<tr>
<td>Layer - 1</td>
<td>Transport</td>
<td>Message Protocol for Communicating Payloads Defined by Higher Layers</td>
</tr>
</tbody>
</table>

The following subsections specify these layers as they pertain to the OpenRTB specification. Unless explicitly specified otherwise, annotated as optional, or called out as a best practice, all material aspects of these subsections are required for OpenRTB compliance.

Layer-1: Transport

*Communications*

The base protocol between an exchange and its demand sources is HTTP. Specifically, HTTP POST is required for bid requests to accommodate greater payloads than HTTP GET and facilitate the use of binary representations. Notification events may be either POST or GET at the discretion of the exchange.
Calls returning content (e.g., a bid response) 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 event notification) should return HTTP 204. Invalid calls (e.g., a bid request containing a malformed or corrupt payload) should return HTTP 400 with no content.

**Version Headers**

The OpenRTB version must 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. This version should be specified as `<major>.<minor>`.

```plaintext
x-openrtb-version: 3.0
```

Additionally, it is recommended albeit optional that responses include an identically formatted HTTP header with the protocol version implemented by the responder. It is assumed, however, that any response will be compatible with the version of the request and that version support is discussed *a priori* between the parties.

**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.

**Security**

In OpenRTB 3.0, HTTPS (i.e., secure HTTP) is required for OpenRTB compliance. For added security of exchange/bidder communications, it is therefore required that exchanges and bidders support HTTPS, and encrypt in transit all exchange/bidder communications.

**Layer-2: Format**

**Representation**

JSON (JavaScript Object Notation) is the default format for bid request and bid response data payloads. JSON was chosen for its combination of human readability and compactness.

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 integration 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.

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

If alternative binary representations are used, the exchange or SSP should specify the Content-Type appropriately. For example: “avro/binary” or “application/x-protobuf”. If the

[https://iabtechlab.com/openrtb](https://iabtechlab.com/openrtb)
content-type is missing, the bidder should assume the type is “application/json”, unless a different default has been selected by an exchange.

**Encoding**

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

Compression can be enabled on the bid response using standard HTTP 1.1 mechanisms. Most web servers 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 demand sources an indication by the exchange that it is capable of accepting gzip encoding for the response. If a demand source 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 demand source 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 demand source 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.

**Layer-3: Transaction**

The Transaction Layer is the heart of the real-time bidding protocol that comprises OpenRTB. It defines the commerce protocol between an exchange and its bidders, or more generally a supply chain intermediary and its integrated demand sources.

Refer to the Specification section of this document for full details.

**Layer-4: Domain**
The Domain Layer defines the objects on which the Transaction Layer operates; the media exchange being transacted. In a typical advertising auction, the bid request would contain domain objects in two places. First, the overall request would contain domain objects that describe the context for the sale such as the site or app, the device, and the user. Second, each item being offered for sale would contain domain objects that define the item such as impression details, specifications, and restrictions. Each bid in a response would include domain objects that define the creative to be delivered to the user if the auction is won.

Since the version of the Domain Layer specification can vary independent of the Transaction Layer, the root object in the Transaction Layer includes domain specification and version information. This is also critical since support for different versions will vary over time by exchange and/or demand source.

E. OpenRTB Transaction Layer

As discussed earlier, the OpenRTB protocol, as a part of the OpenMedia umbrella of specifications, will define the transaction layer of real time bidding. The following describes the updated object model and the proposed updated object definitions and fields. The intention of this document is for public comment. When the full specification for OpenRTB 3.0 is developed, the working group will then provide transition guidance.

Object Model

The UML class diagram that follows illustrates the overall payload structure including both request and response objects. Payloads are rooted in named objects; “Openrtb” as a common root and “Request” and “Response” as subordinate roots to identify the payload type.
Throughout the object model subsections, attributes may be indicated as “Required” or “Recommended”. Attributes are deemed required only if their omission would break the protocol and is not necessarily an indicator of business value otherwise. Attributes are recommended when their omission would not break the protocol but could dramatically diminish business value.

From a specification compliance perspective, any attribute not denoted required is optional, whether recommended or not. An optional attribute may have a default value to be assumed if omitted. If no default is indicated, then by convention its absence should be interpreted as unknown, unless otherwise specified.

**BEST PRACTICE:** Exchanges and demand sources are encouraged to publish to their partners, or even the Internet at large, the set of optional objects and attributes they support along with any extensions to the specification.

**Object: Openrtb**

This top-level object is the root for both request and response payloads. It includes versioning information and references to the Layer-4 domain model (e.g., AdCOM) on which transactions are based.

*Note:* As a convention in this document, objects being defined are denoted with uppercase first letter in deference to the common convention for class names in programming languages such as Java, whereas actual instances of objects and references thereto in payloads are lowercase.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ver</strong></td>
<td>string</td>
<td>Version of the Layer-3 OpenRTB specification (e.g., “3.0”).</td>
</tr>
<tr>
<td><strong>domainspec</strong></td>
<td>string; recommended</td>
<td>Identifier of the Layer-4 domain model contained within “domain” objects in the Advertising Common Object Model, “AdCOM” (e.g., “1.0”).</td>
</tr>
<tr>
<td><strong>domainver</strong></td>
<td>string; recommended</td>
<td>Specification version of the Layer-4 domain model referenced in the “domainspec” attribute.</td>
</tr>
<tr>
<td><strong>request</strong></td>
<td>object; required *</td>
<td>Bid request container. * Required only for request payloads.</td>
</tr>
<tr>
<td><strong>response</strong></td>
<td>object; required *</td>
<td>Bid response container. * Required only for response payloads.</td>
</tr>
<tr>
<td><strong>ext</strong></td>
<td>object</td>
<td>Optional exchange or demand source specific extensions.</td>
</tr>
</tbody>
</table>

Some of these attributes are optional. The “ver” attribute, for example, indicates the OpenRTB specification version to which this payload conforms. This is also conveyed in Layer-1 via an HTTP header. Its utility here is more to assist in diagnostics by making the payload more self-documenting outside the context of a runtime transaction. Similarly, the type of Layer-4 model being operated on would almost certainly be known and assumed at runtime and thus the “domainspec” attribute is also a form of diagnostic documentation.

The “domainver” attribute, however, does have runtime utility since the structures found within “domain” objects may vary in terms of their Layer-4 specification versions over time. This attribute can assist in invoking the correct domain object parser or unmarshalling code.

**Bid Request Payload**

The request object contains minimal high-level attributes (e.g., its ID, test mode, auction type, maximum auction time, buyer restrictions, etc.) and subordinate objects that cover the source of the request and the actual offer of sale. The latter includes the item(s) being offered and any applicable deals.

There are two points in this model that interface to Layer-4 domain objects: the “Request” object and the “Item” object. Domain objects included under “Request” would include those that provide context for the overall offer. These would include objects that describe the site or app, the device, the user, and others. Domain objects included in an “Item” object would specify details about the item being offered (e.g., the impression opportunity) and specifications and restrictions on the media that can be associated with acceptable bids.

**Object: Request**

The “Request” object contains a globally unique bid request ID. This “id” attribute is required as is an “Offer” with at least one “Item” object. Other attributes establish rules and restrictions that apply to all items being offered. This object also interfaces to Layer-4 domain objects for context such as the user, device, site or app, etc.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string; required</td>
<td>Unique ID of the bid request; provided by the exchange.</td>
</tr>
<tr>
<td>test</td>
<td>integer; default 0</td>
<td>Indicator of test mode in which auctions are not billable, where 0 = live mode, 1 = test mode.</td>
</tr>
<tr>
<td>tmax</td>
<td>integer</td>
<td>Maximum time in milliseconds the exchange allows for bids to be received including Internet latency to avoid timeout. This value supersedes any \textit{a priori} guidance from the exchange.</td>
</tr>
<tr>
<td>at</td>
<td>integer; default 2</td>
<td>Auction type, where 1 = First Price, 2 = Second Price Plus. Values greater than 500 can be used for exchange-specific auction types.</td>
</tr>
<tr>
<td>curs</td>
<td>string array, default [&quot;USD&quot;]</td>
<td>Array of currencies for bids on this bid request using ISO-4217 alpha codes. Recommended if the exchange accepts multiple currencies. If omitted, the single currency of “USD” is assumed.</td>
</tr>
<tr>
<td>wcurs</td>
<td>integer; default 0</td>
<td>Flag that determines the restriction interpretation of the “curs” array, where 0 = block list, 1 = whitelist.</td>
</tr>
<tr>
<td>seats</td>
<td>string array</td>
<td>Restriction list of buyer seats for bidding on this item. Knowledge of buyer’s customers and their seat IDs must be coordinated between parties \textit{a priori}. Omission implies no restrictions.</td>
</tr>
<tr>
<td>wseats</td>
<td>integer; default 0</td>
<td>Flag that determines the restriction interpretation of the “seats” array, where 0 = block list, 1 = whitelist.</td>
</tr>
<tr>
<td>source</td>
<td>object</td>
<td>A &quot;Source&quot; object that provides data about the inventory source and which entity makes the final decision.</td>
</tr>
<tr>
<td>offer</td>
<td>object; required</td>
<td>An “Offer” object that conveys the item(s) being offered for sale.</td>
</tr>
<tr>
<td>domain</td>
<td>object; recommended</td>
<td>Layer-4 domain object structure that provides context for the items being offered (e.g., user, device, site or app, etc.) conforming to the specification and version referenced in “openrtb.domainspec” and “openrtb.domainver”.</td>
</tr>
<tr>
<td>ext</td>
<td>object</td>
<td>Optional exchange-specific extensions.</td>
</tr>
</tbody>
</table>

**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 that combines direct campaigns with third party demand in decisioning.

[https://iabtechlab.com/openrtb](https://iabtechlab.com/openrtb)
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fd</td>
<td>integer;</td>
<td>Entity responsible for the final sale decision, where 0 = exchange, 1 =</td>
</tr>
<tr>
<td></td>
<td>recommended</td>
<td>upstream source.</td>
</tr>
<tr>
<td>tid</td>
<td>string;</td>
<td>Transaction ID that must be common across all participants in this bid</td>
</tr>
<tr>
<td></td>
<td>recommended</td>
<td>request (e.g., potentially multiple exchanges).</td>
</tr>
<tr>
<td>pchain</td>
<td>string;</td>
<td>Payment ID chain string containing embedded syntax described in the TAG</td>
</tr>
<tr>
<td></td>
<td>recommended</td>
<td>Payment ID Protocol.</td>
</tr>
<tr>
<td>ext</td>
<td>object</td>
<td>Optional exchange-specific extensions.</td>
</tr>
</tbody>
</table>

Object: Offer

This object is a collection of one or more “Item” objects representing the goods being sold. In addition to the item array, this collection also includes an indicator as to whether this collection of items constitutes all items in a given context (e.g., all impressions on a web page).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>object array;</td>
<td>Array of “Item” objects (at least one) that constitute the set of goods</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td>being offered for sale.</td>
</tr>
<tr>
<td>package</td>
<td>integer;</td>
<td>Flag to indicate if the Exchange can verify that the items offered</td>
</tr>
<tr>
<td></td>
<td>default 0</td>
<td>represent all of the items available in context (e.g., all impressions on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a web page, all video spots such as pre/mid/post roll) to support road-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blocking, where 0 = no or unknown, 1 = yes.</td>
</tr>
<tr>
<td>dburl</td>
<td>string</td>
<td>Billing notice URL called by the demand partner when a winning bid becomes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>billable based on partner-specific business policy. Substitution macros</td>
</tr>
<tr>
<td></td>
<td></td>
<td>may be included. One of burl in response or dburl in request is required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(exception for VAST).</td>
</tr>
<tr>
<td>ext</td>
<td>object</td>
<td>Optional exchange-specific extensions.</td>
</tr>
</tbody>
</table>

Object: Item

This object represents a unit of goods being offered for sale either on the open market or in relation to a private marketplace deal. The “id” attribute is required since there may be multiple items being offered in the same bid request and bids must reference the specific item of interest. This object interfaces to Layer-4 domain objects for deeper specification of the item being offered (e.g., an impression).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string;</td>
<td>A unique identifier for this item within the context of the offer (typically</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td>starts with 1 and increments).</td>
</tr>
</tbody>
</table>
### Object: Pmp

This object is the private marketplace container for direct deals between sellers and buyers that may pertain to this item. The actual deals are represented as a collection of “Deal” objects.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>integer; default 0</td>
<td>Indicator of auction eligibility to seats named in “Deal” objects, where 0 = all bids are accepted, 1 = bids are restricted to the deals specified and the terms thereof.</td>
</tr>
<tr>
<td>deal</td>
<td>object array</td>
<td>Array of “Deal” objects that convey special terms applicable to this item.</td>
</tr>
<tr>
<td>ext</td>
<td>object</td>
<td>Optional exchange-specific extensions.</td>
</tr>
</tbody>
</table>

### Object: Deal

This object constitutes a specific deal that was struck *a priori* between a seller and a buyer. Its presence within the “Pmp” collection indicates that this impression is available under the terms of that deal.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string; required</td>
<td>A unique identifier for the deal.</td>
</tr>
<tr>
<td>qty</td>
<td>integer</td>
<td>Number of instances of the item to which the deal applies. Default is the full quantity specified in the “item.qty” attribute.</td>
</tr>
<tr>
<td>flr</td>
<td>float</td>
<td>Minimum deal price for this item expressed in CPM.</td>
</tr>
</tbody>
</table>
flrcur | string; default “USD” | Currency of the “flr” attribute specified using ISO-4217 alpha codes.
--- | --- | ---
at | integer | Optional override of the overall auction type of the request, where 1 = First Price, 2 = Second Price Plus, 3 = the value passed in “flr” is the agreed upon deal price. Additional auction types can be defined by the exchange.
seat | string array | Whitelist of buyer seats 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 restrictions.
wadomain | string array | Array of advertiser domains (e.g., advertiser.com) allowed to bid on this deal. Omission implies no restrictions.
ext | object | Optional exchange-specific extensions.

Bid Response Payload

The response object contains minimal high-level attributes (e.g., reference to the request ID, bid currency, etc.) and an array of seat bids, each of which is a set of bids on behalf of a buyer seat.

The individual bid references the item in the request to which it pertains and buying information such as the price, a deal ID if applicable, and notification URLs. The media related to a bid is conveyed via Layer-4 domain objects (i.e., ad creative, markup) included in each bid.

Object: Response

This object is the bid response object under the “Openrtb” root. Its “id” attribute is a reflection of the bid request ID. The “bidid” attribute is an optional response tracking ID for bidders. If specified, it will be available for use in substitution macros placed in markup and notification URLs. At least one “Seatbid” object is required, which contains at least one “Bid” for an item. Other attributes are optional.

To express a “no-bid”, the most compact option is simply to return an empty response with HTTP 204. However, if the bidder wishes to convey a reason for not bidding, a “Response” object can be returned with just a reason code in the “nbr” attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string; required</td>
<td>ID of the bid request to which this is a response; must match the “request.id” attribute.</td>
</tr>
<tr>
<td>bidid</td>
<td>string</td>
<td>Bidder generated response ID to assist with logging/tracking.</td>
</tr>
<tr>
<td>nbr</td>
<td>integer</td>
<td>Reason for not bidding if applicable (seeEnumerations).</td>
</tr>
<tr>
<td>cur</td>
<td>string</td>
<td>Bid currency using ISO-4217 alpha codes.</td>
</tr>
</tbody>
</table>
**cdata**  |  string  |  Allows bidder to set data in the exchange’s cookie if supported by the exchange. The string must be in base65 cookie-safe characters. JSON encoding must be used to include “escaped” quotation marks.

**seatbid**  |  object array  |  Array of “Seatbid” objects; 1+ required if a bid is to be made.

**ext**  |  object  |  Optional demand source specific extensions.

---

**Object: Seatbid**

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>seat</td>
<td>string</td>
<td>ID of the buyer seat on whose behalf this bid is made.</td>
</tr>
<tr>
<td>package</td>
<td>integer; default 0</td>
<td>For offers with multiple items, this flag indicates if the bidder is willing to accept wins on a subset of bids or requires the full group as a package, where 0 = individual wins accepted; 1 = package win or loss only.</td>
</tr>
<tr>
<td>bid</td>
<td>object array; required</td>
<td>Array of 1+ “Bid” objects each related to an item. Multiple bids can relate to the same item.</td>
</tr>
<tr>
<td>ext</td>
<td>object</td>
<td>Optional demand source specific extensions.</td>
</tr>
</tbody>
</table>

**Object: Bid**

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string; recommended</td>
<td>Bidder generated bid ID to assist with logging/tracking.</td>
</tr>
<tr>
<td>item</td>
<td>string; required</td>
<td>ID of the item object in the related bid request; specifically “item.id”.</td>
</tr>
<tr>
<td>deal</td>
<td>string</td>
<td>Reference to a deal from the bid request if this bid pertains to a private marketplace deal; specifically “deal.id”.</td>
</tr>
<tr>
<td>price</td>
<td>float; required</td>
<td>Bid price expressed as CPM although the actual transaction is for a unit item only. Note that while the type indicates float, integer math is highly recommended when handling currencies (e.g., BigDecimal in Java).</td>
</tr>
</tbody>
</table>

[https://iabtechlab.com/openrtb](https://iabtechlab.com/openrtb)
cid | string | ID of a campaign of other logical grouping to which the media to be presented if the bid is won belongs.

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.

burl | string | Billing notice URL called by the supply partner when a winning bid becomes billable based on exchange-specific business policy (e.g., markup rendered). Substitution macros may be included. One of burl in the response or dburl in the request must be present (exception for VAST).

lurl | string | Loss notice URL called by the supply partner when a bid is known to have been lost. Substitution macros may be included. Exchange-specific policy may preclude support for loss notices or the disclosure of winning clearing prices resulting in \${OPENRTB\_PRICE} macros being removed (i.e., replaced with a zero-length string).

domain | object; required | Layer-4 domain object structure that specifies the media to be presented if the bid is won (e.g., creative) conforming to the specification and version referenced in “openrtb.domainspec” and “openrtb.domainver”.

ext | object | Optional demand source specific extensions.

F. Updated Billing Notifications

The following proposed updated billing notifications are written to support the new reference model in section C. Complex Digital Advertising Supply Chain. With multiple hops in the supply chain and multiple notifications, there are distinct needs for a supply billing URL (burl) and demand billing URL (dburl).

This new model allows for flexibility in what event is considered billable, and who counts it. Ordinarily, the event is an impression, though it can be otherwise (viewable impression, click, etc.). Both this and the question of whose count is authoritative is a matter to be addressed contractually between partners, but the protocol can any combination of:

- Billing based on supply partner’s count (burl)
- Billing based on demand partner’s count (dburl)
- Billing based on mutual confirmation

https://iabtechlab.com/openrtb
The OpenRTB working group has also developed best practices about the use of billing notifications in order to support accurate real time communication between partners. Business practices are ultimately decided by each partner.

The following changes are proposed for OpenRTB 3.0:

a. One of burl in response or dburl in request is required (exception for VAST)
b. Updates to loss notification
c. Maximum lengths for URLs

Event Notification

The following events are defined within this specification. Additional events may be defined in Layer-4 specific to domain objects.

In the complex supply chain scenario, an intermediary can choose to participate in these events (i.e., receive them and pass them on) or allow them to bypass. The former is generally expected, but there are intermediaries such as bid request traffic aggregators/shapers that may elect not to intercept these events if, for example, they charge purely on traffic volume versus revenue-share. An intermediary can accomplish interception by encoding event URLs on bids received onto their own URLs and providing these in bids returned upstream to form an event chain. Upon receipt of their event URLs, the original URLs are decoded and called. In the bypass scenario, the intermediary would simply preserve the original URLs in bids returned upstream.

Any entity that invokes an event URL must first resolve all standard OpenRTB substitution macros.

Event: Billing

A billing event is when a transaction results in a monetary charge from the exchange or other intermediary to one of their demand-side partners. This event is subject to exchange-specific business policies agreed upon between the exchange and their demand partners. For a DSP, this event signals that they can decrement spend against the related campaign. Depending on the agreed upon practice, either the exchange conveys this event by invoking the billing notice URL provided by the demand source in the “bid.burl” attribute, the demand partner conveys this event by invoking the demand billing notice URL provided by the supply partner in the “offer.dburl” attribute, or both are mutually fired.

**BEST PRACTICE:** Exchanges are highly encouraged to standardize on a client-initiated render event or equivalent as the basis for the billing event as the most consistent approach in a complex supply chain scenario with potentially multiple auction decision points.

**BEST PRACTICE:** Upon determining a billable event has occurred (e.g., receipt of client-initiated render signal), an exchange should invoke the billing notice from server-side and as “close” as possible to where the exchange books revenue in order to minimize discrepancies between itself and its demand sources.
**BEST PRACTICE:** Firing the billing URL represents the fulfillment of a business transaction between the exchange and its demand partner. This should not be delegated to another party including a client-side pixel.

For VAST Video, the IAB prescribes that the VAST impression event is the official signal that the impression is billable. If the “bid.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.

**Event: Loss**

A loss event is when an exchange determines that a bid has no chance of winning. At this point, an exchange may invoke the loss notification URL provided by the demand source in the “bid.lurl” attribute. Note that absence of a loss notice does not imply a win.

Exchange support for firing loss notifications is not required for OpenRTB compliance. Demand partners are encouraged to check with their supply-side partners. Furthermore, exchange-specific policy may preclude support for loss notices or the disclosure of winning clearing prices resulting in ${OPENRTB_PRICE} macros being removed (i.e., replaced with a zero-length string).

**Substitution Macros**

Notification URLs (e.g., billing, loss) and their format are defined by the demand source. In order for the exchange to convey certain information (e.g., the clearing price), a number of macros can be inserted into these URLs which the exchange is responsible for resolving prior to invoking. Substitution is assumed to be 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 how the markup is obtained. An example use case is for injecting data such as clearing price into tracking URLs that are embedded in the markup.

The following table defines the standard substitution macros. Note that OpenRTB compliant exchanges must support all macros for which data is available and support substitution in both markup and notification URLs.

<table>
<thead>
<tr>
<th>Macro</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>${OPENRTB_ID}</td>
<td>ID of the bid request; from “request.id” attribute.</td>
</tr>
<tr>
<td>${OPENRTB_BID_ID}</td>
<td>ID of the bid; from “response.bidid” attribute.</td>
</tr>
<tr>
<td>${OPENRTB_ITEM_ID}</td>
<td>ID of the item just won; from “item.id” attribute.</td>
</tr>
<tr>
<td>${OPENRTB_SEAT_ID}</td>
<td>ID of the bidder seat; from “seatbid.seat” attribute.</td>
</tr>
</tbody>
</table>
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.

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 agreed to be “B64”. The macro would then be written as follows:

${OPENRTB_PRICE:B64}$

Additional Implementation Guidance

The above “best practices” and the following implementation guidance are suggested to become an appendix in final spec.

Idempotency

A sburl/dburl endpoint should be idempotent to ensure that failure retries and other failure scenarios do not result in multiple impression counts on the DSP side and a discrepancy between exchange and DSP. Partners should encode an auction or BID ID into the URL to facilitate this deduplication.

Failure handling

As these calls generally occur over the public Internet, failures can be expected some portion of the time, however small. In the event that the call to an event notification URL does not succeed (that is, HTTP status code 200 or 204 is received), it is recommended that the upstream party retries calling of the URL once every 10 seconds (for no more than 1 minute) between each attempt. Further, it is recommended that the calling party records incidences of permanent failure for the purpose of identification of discrepancies resulting from this.

https://iabtechlab.com/openrtb
Timing
The event notification URLs should be called as close in time as possible to when the events are determined to have occurred, ideally within a matter of milliseconds (excepting failures and subsequent retries). Delayed event notification may interfere with some partners ability to count downstream events (e.g. clicks, conversions) if, for example, a billing event is delayed until after the arrival of these additional events.

The “exp” field in the bid request should be used by the upstream party to communicate the period of time after the auction at which a resulting billed event should no longer occur. For example, if an upstream entity has a policy that impressions must occur within 10 minutes of an auction, they should communicate this with the “exp” field, which allows the downstream entity to minimize risk of replay attacks and minimize the amount of state information it must hold (if, for example, it holds state for auctions that it has bid on).

In auctions for types of inventory where the billing event may occur a substantial (>1 minute, as long as 45-60 minutes) period of time after the auction win (for example, digital out-of-home or programmatic TV), bidders can use nurl, if support is provided, for the purposes of throttling pacing in the interests of avoiding overspend. Nurl should always be fired immediately upon auction win.

Avoid holding state
Both burl and dburl are potentially called only after certain conditions are met, for example if the ad has been determined to be seen. As it would be costly to hold burl/dburl and fire them only when a certain event occurs, it is recommended to encode these URLs (encrypted or not) in the mechanism that checks the condition. The handler that is notified that the condition is met can then forward the information to the partner.

For example, if the demand partner provides a bid and a sburl equal to https://demand.com/sburl/impct.aspx?actionid=123&price=${AUCTION_PRICE}. The supplier determines the bid has won for a price of 2, and sends to the client its impression tracking pixel: https://supply.com/impct.aspx?actionid=123&price=2&sburl=https%3A%2F%2Fdemand.com%2Fsurl%2Fimpct.aspx%3Fauctionid%3D123%26price%3D2
When the client calls the impression tracking pixel, the supplier’s handler retrieves the burl and calls it server-to-server.

A similar method can be used by the demand side to carry the dburl into its creative.

https://iabtechlab.com/openrtb
Open Items for Consideration before OpenRTB 3.0 Final Specification

- Video: Clarifying burl. When the exchange knows it's going to book revenue, that's when burl should be called. VAST calls could then be reconciled against burl.
- Open area for implementation around server-side notifications best practices.
- Further consideration of the requirement of both burl and dburl

G. Domain Layer: Advertising Common Object Model

In online advertising, there are a number of objects common to multiple transactional specifications. For example, across OpenRTB and OpenDirect, there is a common concept of a “site”, an “ad”, and so on. These objects describe the subject of a transaction. These objects can be carried on a transactional protocol. This can be illustrated using a layer concept, discussed above in section D.

The Advertising Common Object Model (AdCOM) describes the business objects that are common across the transactional specification. AdCOM is a new companion specification to OpenRTB.

AdCOM principles and initial specification contents are available [here](https://iabtechlab.com/openrtb). AdCOM is open for public comments during this OpenRTB 3.0 Framework public comment period until December 15, 2017.

After public comments, and with additional working group attention, a finalized version of AdCOM will be produced with full specification contents.

Principles of AdCOM

- When an object is useful across multiple transactional specifications, it should graduate upstream to AdCOM.
- AdCOM is backwards compatible within minor versions (e.g. 1.x). No breaking changes may be made within a minor version. Existing fields may not change in meaning, type, etc.
- Implementers of AdCOM must gracefully accept unexpected fields so to allow for introduction of new fields in minor versions. That is, while an implementation may not know what to do with an unexpected field, its presence must not break the implementation.
- AdCOM may be extended as needed for specific applications. These extensions must always be placed in a subordinate `ext` object, and all objects may have subordinate `ext`
objects. An exception is additions to enumerated lists. These are permitted in the standard fields when the AdCOM description of the list provides a means for reserving values for this. For example, a list of creative attributes may define values 500+ can be used for implementation-specific purposes.

- AdCOM describes the objects, not a particular way of expressing them. JSON is used for illustration in this document, but nothing precludes implementations that use a different means; for example, XML, Avro, Protobuf, etc.
- Only a subset of fields in any given object will be relevant for a given transactional specification. It is expected that the transactional specification indicates which fields are required, recommended, etc.
- Wherever possible, fields have been shortened as they may be transmitted in plain text very frequently, for example in OpenRTB.

The core object model in AdCOM is represented in the following diagram. More details are available in the companion specification document (link provided above).

**AdCOM Domain Layer Object Model**

<table>
<thead>
<tr>
<th>Item (Layer 3 - OpenRTB Transactional Spec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad Specification</td>
</tr>
<tr>
<td>Creative Specification</td>
</tr>
<tr>
<td>Display Specification</td>
</tr>
<tr>
<td>Video Specification</td>
</tr>
<tr>
<td>Audio Specification</td>
</tr>
<tr>
<td>Events Specification</td>
</tr>
<tr>
<td>Site, App, Publisher, and other context-related objects</td>
</tr>
</tbody>
</table>

[Link to OpenRTB](https://iabtechlab.com/openrtb)
H. Authentication: Signed Bid Requests

A major component of OpenRTB 3.0 is the shift to an authenticated supply chain. This move to standardize cryptographically signed bid requests is the next step needed in OpenRTB to ensure security and trust in the supply path hops in real time bidding.

At a high level, the principle of signed bid requests is that Publishers and Exchanges should sign messages within the supply chain of the real-time bidding transaction. This provides a traceable path to verify critical data of the inventory such as domain, publisher id, and a timestamp of the bid request.

Publishers will benefit from this anti-fraud measure in knowing that their inventory is securely passed for sale.

Advertisers and buyers will benefit from this by reviewing the supply authentication to gain confidence in where the inventory is coming from in the real-time bidding transaction.

Signed bid requests complements the recent ads.txt protocol. Ads.txt and the data within it should be used to validate authorized sellers their platforms for a source of inventory. Publisher signing of bid requests allow a buyer to validate the bid request and know that it’s trusted from the publisher and key elements of the bid request are unmodified. Together, these technologies are a powerful combination in fighting fraud to allow buyers to check for authenticity and authorization of the sales channel.

What does authentication look like?

1. Signed ad inventory contains a signature from the inventory owner (or their chosen tech delegate).
2. The end result is a signed message that will show the authentic path of the bid request. An untampered message is evidence of a secure supply source.
3. Buyers can run forensics comparing the details within the signed authentic message of the inventory and the actual details within the bid request as well as run forensics of the record of what the path of the bid request should be (ads.txt authorized sellers).

How to learn more or comment on authentication proposal:
“OpenRTB 3.0 Authentication: Signed Bid Requests” is currently available here. This external proposal document contains the entirety of specification requirements and implementation guidance relative to signing bid requests. In the creation of the full specification for 3.0, this proposal will be fully integrated into the specification.

“OpenRTB 3.0 Authentication: Signed Bid Requests” is open for public comment concurrently with the OpenRTB 3.0 Framework. Comments can be submitted until December 15, 2017.
I. Addressing Data Leakage

A step towards self-regulation in a safe data supply chain, understanding data leakage and where data is going and who is using it, OpenRTB 3.0 introduces the Trusted Data Providers element.

As a first step to solving the data leakage problems publishers face is for publishers to whitelist the companies with whom they are willing to share certain of their data, and require every company they work with to honor this whitelist. Similar to how ads.txt allows publishers to declare what companies sell their inventory in exchanges, this proposed protocol extends to pubs declaring any third party that may access their data. The TDP manifest described below is effectively the pub’s whitelist of third parties that may access their data directly or indirectly. Methods through which a third party may collect pub information include through tags called on page, server-to-server connections with other third parties, or network calls and pixel drops made by a 3rd party creative tag. For the TDP manifest and protocol, we define data as user ids, IP addresses, and lat/longs of users visiting a publisher’s domain(s). So, any third party that can serve JavaScript client side on the pub’s websites has access to this subset of the pub’s data.

More context can be seen here: http://bokonads.com/request-for-comments-trusted-data-partners/

Use cases in addressing data leakage:

Publishers concerned about data leakage could produce a JSON file that’s publicly accessible on their domains (such as example.com/TDP.json), and that file lists all 3rd parties which may be allowed access their data. The pub would then notify any partners that are called directly from page, such as their primary ad server or header bidding partner, and hold them accountable for adhering to that whitelist when serving advertising.

Adservers, SSPs, & header bidding wrappers would read the TDP manifest for the publisher and apply these rules to their monetization of the impression. This would include only sending bid requests to DSPs or data providers listed, sending the full TDP manifest to approved partners in the openRTB bid request so that DSPs can adhere to the list as well, screening out creatives that load assets that are not on the manifest, not dropping user synch pixels for companies not on the TDP manifest.

DSPs would read the TDP manifest and on bid request and also apply these filters to their logic. This includes screening any bids with creatives that contain assets not listed in the TDP manifest, not dropping user synch pixels for companies not on the TDP manifest, and not providing log level reporting including the publisher's data to any 3rd party or customer. A DSP getting a bid request could assume they are whitelisted; however, they may want the entire TDP list to control what creative they return to try and win the auction. For example, if doubleclick.net is not allowed, the DSP shouldn’t respond with a doubleclick creative because it will get blocked from winning the auction by the SSP.

Agencies and marketers would take note of the technologies they’re using for creative ad serving, impression tracking, brand safety, etc. and make decisions on how much they should add to a creative's payload knowing excessive use of many 3rd parties will reduce their reach with some publishers.

https://iabtechlab.com/openrtb
Applications to GDPR and ePR

GDPR and ePR will likely fundamentally change how our ecosystem operates and both ensure that consumers of publisher content have transparency and additional control over who accesses and uses their data. In simplistic terms, any user of a website in the EU has the right to know where their data is going (including pseudonymized user ids). If/when our ecosystem evolves to consumers having control on TDPs when they visit a publisher, the TDP protocol proposed here will still function properly. TDP manifest will now be for individual users in addition to publisher based.

The proposed specification update for OpenRTB 3.0 is to create a TDP manifest - effectively the whitelist for trusted data partners - the publisher will create a JSON file that lists trusted data partners:

```
{"trusted_data_partners": [
  {"domain": "adnxs.com", "name": "AppNexus", "comments": "programmatic partner"},
  {"domain": "doubleclick.net", "comments": "trusted ad server"},
  ...
]}
```

A data partner can be any third party that the publisher works with directly or indirectly. For companies that use multiple domains to deliver content, each domain should be listed in a separate line in the manifest:

```
{"domain": "atdmt.com", "name": "Atlas ad serving"},
{"domain": "atlasbx.com", "name": "Atlas analytics"}
```

The manifest should be published on a public URL (by default, /tdp.json) so that partners can retrieve it and so that it is easily available to the general public.

The TDP manifest should be sent on every programmatic ad request through a simple extension to the OpenRTB protocol, adding a tdp element to BidRequest with the public TDP manifest URL as the value. (Could also send the full list of domains on every request, but that seems heavyweight as a protocol).

Public Comments:
The TDP manifest proposal is open for public comment as a feature of OpenRTB 3.0. With public review considerations and further review from the working group, this proposal will be integrated into the development of a full specification for OpenRTB 3.0 later this year.
J. Ad Management API Specification

Ad management occurs when a buyer (or a representative party) submits ads for approval, and supply partners (or a representative party) approve or disapprove of those ads. Before the publication of this technical standard, there have been proprietary methods and tools for ad management. Ad approval process is desired and implemented for a few reasons; to ensure creatives comply with the laws of various regions, to respect publisher content guidelines, scan for malicious activity, and to gather creative assets where the markup for an ad is not delivered as a part of each transaction.

The OpenRTB Working Group has identified a need to standardize creative approval process, to reduce pain points for buyers and sellers in the digital advertising industry. Using a standard approach allows for greater scale for buyers to submit creatives for approval. The Ad Management API Specification will support multiple scenarios that exist for exchanges with restrictive or permissive bidding, and different markup delivery methods. It makes no attempt to impose any particular business practice (or even the implementation of any ad approval process at all), but rather provides a standardized means of operating according to common business practices on this topic.

Some of the benefits of the introduction of a standardized specification:

- Eases implementation of ad approval workflows for exchanges who wish to implement such policies, but have not done so yet
- Reduces demand partner integration effort by avoiding the need for multiple proprietary implementations per supply partner
- Helps enable supply partners to provide confidence to publishers that legal requirements on ads enforced (i.e. GDPR in Europe, Language in Québec, Labelling in China)
- Enables RTB transactions of new formats where real-time creative delivery is not possible and ad pre-registration is a technical necessity (e.g. Digital Out-of-Home)

The Ad Management API Specification v1 is available for review here.

When finalized, this specification will be hosted as a companion to OpenRTB 3.0. As a part of the OpenRTB 3.0 Framework release, the Ad Management API Specification is also open for public comment until December 15, 2017.

K. Consumer Identifier support

Consumer identifiers support in the OpenRTB specification allows buyers to use audience data in real time bidding.

https://iabtechlab.com/openrtb
The intention here is to solve for diverse use cases around identity, including

- Support for cross-device models in bid-stream
- Allow for the ecosystem of "single IDs"
- People-based identifiers of all types

**If adopted, changes to AdCOM spec:**

**Added field to user object**

<table>
<thead>
<tr>
<th>Field</th>
<th>Scope</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eids</td>
<td>optional</td>
<td>array</td>
<td>-</td>
<td>An array of Extended ID objects</td>
</tr>
</tbody>
</table>

**Object: Extended IDs**

<table>
<thead>
<tr>
<th>Field</th>
<th>Scope</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>required</td>
<td>string</td>
<td>-</td>
<td>Source or technology provider responsible for the set of included IDs. Expressed as a top-level domain.</td>
</tr>
<tr>
<td>uids</td>
<td>required</td>
<td>array</td>
<td>-</td>
<td>Array of extended ID UID objects</td>
</tr>
</tbody>
</table>

**Extended ID UIDs object**

<table>
<thead>
<tr>
<th>Field</th>
<th>Scope</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>required</td>
<td>string</td>
<td>-</td>
<td>Cookie or platform native identifier</td>
</tr>
<tr>
<td>atype</td>
<td>recommen ded</td>
<td>integer</td>
<td>-</td>
<td>Type of user agent the match is from. See agent types table.</td>
</tr>
<tr>
<td>ext</td>
<td>optional</td>
<td>object</td>
<td>-</td>
<td>Exchange-specific extension object</td>
</tr>
</tbody>
</table>

**Agent Types**

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Web</td>
<td>An ID intended which is tied to a specific browser or device (cookie-based, probabilistic, or other).</td>
</tr>
<tr>
<td>2</td>
<td>IDFA</td>
<td>Apple IDFA</td>
</tr>
<tr>
<td>3</td>
<td>AAID</td>
<td>Android Ad ID</td>
</tr>
<tr>
<td>4</td>
<td>Windows Advertising ID</td>
<td>Windows mobile advertising ID</td>
</tr>
<tr>
<td>5</td>
<td>Other mobile ID</td>
<td>Another mobile advertising identifier, such as a SHA1 or MD5 version, or other legacy native mobile identifier.</td>
</tr>
<tr>
<td>500+</td>
<td>Exchange specific</td>
<td></td>
</tr>
</tbody>
</table>

**Best Practices**

- The exchange should ensure that business agreements allow for the sending of this data.
- The atype field is highly recommended as many DSPs separate app native IDs from browser-based IDs and require a type value for ID resolution.
- Handling of opt out and privacy signals is out of the scope of this document. It is assumed that exchanges and DSPs will collaborate with the appropriate regulatory agencies and ID vendor(s) to ensure compliance.

**Example**

```json
"user": {
  "id": "aaa",
  "buyerid": "xxx",
  "eids": [
    {
      "source": "x-device-vendor-x.com",
      "uids": [
        { "id": "yyy", "atype": 1 },
        { "id": "zzz", "atype": 1 }
      ]
    }
  ]
}
```

[https://iabtechlab.com/openrtb](https://iabtechlab.com/openrtb)
L. Additional OpenRTB Companion Specifications

Recent companion updates in the OpenMedia specification landscape that have been finalized recently and are noteworthy for the OpenRTB working group.

- **Ads.txt Specification version 1.0** is a mechanism for publishers to declare their authorized resellers and fight fraud. The ads.txt specification mentions the OpenRTB publisher id field.
M. Plan for Finalizing OpenRTB 3.0

Public Comment
This OpenRTB 3.0 Framework document and companion specifications will be open for public comment for 90 days until December 15, 2017. Comments and feedback should be submitted using this OpenRTB 3.0 Framework Public Comment Submission Form.

The OpenRTB working group will review feedback. IAB Tech Lab members are welcome to participate and join this working group.

If your company would like to test proof of concept and implement any of these features, please do so with the involvement of the OpenRTB working group, so that we can provide expected transition help materials and get direct feedback from your efforts.

Additional resources for OpenRTB working group development:
- Formal discussion (including decisions for specification development) in Google Group Forum: https://groups.google.com/forum/#!forum/openrtb-dev
- Casual chat and implementation discussion on Slack, available to working group members (IAB Tech Lab Membership required).

Roadmap/Timeline
- September 12, 2017: OpenRTB 3.0 Framework and companion specifications published for public comment
- December 15, 2017: 90-day Public Comments closes
- During the 90-day public comment period, OpenRTB Working Group will review comments and feedback, and develop a full detailed OpenRTB 3.0 specification
- Expected Jan 2018: fully detailed specification released (With a 30-day public review period, then finalized document).
- Expected Feb 2018: Earliest possible complete implementation of OpenRTB 3.0.

Compliance
Forward facing compliance opportunities:
- OpenRTB is a protocol that does not define business policy.
- Potentially updating stratification of the “required”, “recommended”, “optional”, and new “critical” field types.
- Innovation is welcome in the industry. Compliance describes business function compatibility.
Future Directions

- The following is a list of unconfirmed potential future directions for additional feature update: GDPR compliance, signed bid responses, additional features for in-app support, etc.
  - To advance innovation in the RTB space, the working group may support formalizing extensions on the OpenRTB wiki - avoiding bloat by hosting a standardized place for companies to share specs.
  - An option is for incremental feature updates to be included in 3.1, as to encourage rapid adoption of 3.0 as soon as the full specification is finalized.

- The OpenRTB working group will review feedback and release a full specification in early 2018, with another 30-day review period to finalize the full specification.

Appendix A: List of Companion Specifications

External resources referenced in this OpenRTB 3.0 Framework Document:
- OpenMedia
- AdCOM
- OpenRTB 3.0 Authentication: Signed Bid Requests
- Ad Management Specification

Appendix B: Community Resources

Interactive Advertising Bureau (IAB) Tech Lab
https://iabtechlab.com

OpenRTB Website and Blog
https://iabtechlab.com/openrtb

OpenMedia Information
https://iabtechlab.com/specifications-guidelines/openmedia/

Development Community Mailing List
https://groups.google.com/forum/#!forum/openrtb-dev

User Community Mailing List
https://groups.google.com/forum/#!forum/openrtb-user
Appendix C: Change Log

This appendix serves as an index of specification changes from the current version to the previous. These changes pertain only to the substance of the specification and not routine document formatting, information organization, or content without technical impact. However, since OpenRTB v3.0 is a major revision, the change log will be omitted.