iab. TECH LAB

DemandChain Object Specification

Version 1.0

Released March 2021

Please email openMedia@iabtechlab.com or support@iabtechlab.com with feedback or questions. This document is available online at https://iabtechlab.com/buyers-json-demand-chain

About IAB Tech 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.

License

Disclaimer

Special thanks to John Clyman, VP Engineering, Magnite for his leadership

Other Significant Contributors Include:

Paul Bannister, Chief Strategy Officer, Cafe Media; Per Bjorke, Sr. Product manager, Ad Traffic Quality, Google; Eric Bozinny, Sr. Director, Marketplace Quality, Pubmatic; Julien Delhommeau, Sr. Solutions Consultant, Xandr; Emma Fenlon, Sr. Manager, Exchange Quality, Verizon Media; Rahul Gupta, VP Client Solutions, Pulsepoint; Aaron Herman, Product Manager, Ads Integrity, Google; Curtis Light, Staff Software Engineer, Google; John Murphy, Chief Strategy Officer, Confiant; Alexandre Nderagakura, Product Specialist - Programmatic, Smart ad server; Angie Pennington, Sales & Operations Strategy Lead, Verizon Media; Amit Shetty, VP Programmatic & Partnerships, IAB Tech Lab, Lindsay Superczynski-Matthies, Sr P&E Optimization Specialist, Exchange Quality, Verizon Media; Maddy Want, Director of Product, Index Exchange.

IAB Tech Lab Lead:

Amit Shetty, VP Programmatic & Partnerships, IAB Tech Lab

TABLE OF CONTENTS

1 Abstract 1

2 Introduction 1

3 Implementation 1

4 Node Definition 1

5 OpenRTB Object: DemandChain 2

6 OpenRTB Object: DemandChainNode 3

7 Implementation Details 4

8 Validation & Enforcement Guidance 5

9 Serializing and Exposing Creative Source Data 5

Embedding of Serialized Data in Ad Markup 7

10 Examples 9

1 Abstract

As part of a comprehensive effort to promote transparency in ad tech and specifically to protect publishers and end users from malvertising and other objectionable ads, the DemandChain object enables sellers to see all parties who are involved in buying the creative embedded in a given bid response. This extension object can be used with OpenRTB 2.x or OpenRTB 3.0.

DemandChain, in conjunction with buyers.json, provides a buy-side complement to the SupplyChain and sellers.json specifications that enhance supply-side transparency.

2 Introduction

Ads.txt, and later SupplyChain and sellers.json, have been extremely successful in allowing DSPs and other buyers to determine the full chain of entities supplying inventory offered in bid requests. Similar transparency is essential for the sell side to provide visibility into buyers so they can: identify and block bad actors, make informed judgments about the provenance of creatives, and enforce limits on reselling and similar behaviors. This implementation should be as transparent as possible to sellers as well as to trusted third parties such as anti-malware scanning providers. It should enable them, including their representatives with only modest levels of technical sophistication, to easily understand who is participating in the delivery of any creative that is served on their inventory.

While there is benefit to considering additional transparency efforts to prevent spoofing of agency identity, and to exploring cryptographic techniques to prevent manipulation of demand by untrustworthy intermediaries, both are beyond the scope of this initial effort.

3 Implementation

The DemandChain (or dchain) object is composed primarily of a set of nodes where each node represents a specific entity that participates in the purchasing of an impression. The entire chain of nodes from beginning to end represents all entities who are involved in the direct flow of payment for the impression and the creative that runs within it. Future versions of the specification may also include entities who are involved in the transaction but are not involved in payment.

Specifically, a complete dchain specifies every intermediary between the ultimate payor (typically a brand or a self-serve advertiser) and the ultimate publisher where the impression is served. It is possible that not all participants in this chain are integrated programmatically, a consideration which is further described in the Node Definition below.

4 Node Definition

A node contains two recommended properties; the advertising system identifier (asi) and the buyer seat ID (bsid). These properties MUST be present for any programmatic advertising system. However, the asi and bsid are marked as "recommended" because they may be omitted when, and only when, representing non-programmatic participants in the payment chain.

The asi is the domain name of the advertising system. The bsid is used to identify the buyer of the inventory; that is, who is paying the advertising system to serve this creative. The domain name identified in the asi should match the domain that hosts the buyers.json file, and the bsid should be a value that appears as a buyer_id in that file.

The bsid cannot represent multiple entities even if multiple entities are paying the advertising system. Every bsid must map to only a single entity represented as the business that will pay for that impression. This bsid must be used in all inventory transactions. A buying entity, however, may have multiple buyer seat IDs within an advertising system.

5 OpenRTB Object: DemandChain

This object represents both the links in the demand chain as well as an indicator for whether or not the demand chain is complete.

The DemandChain object should be included in the BidResponse.seatbid.bid.ext.dchain attribute in OpenRTB 2.5 or later. For OpenRTB 2.4 or prior, the BidResponse.ext.dchain attribute should be used.

The DemandChain object includes the following attributes:

Attribute	Type	Description
complete	integer; required	Flag indicating whether the chain contains all nodes involved in the transaction leading back to the originator and ultimate source of payment for the creative, where 0 = no, 1 = yes.
nodes	object array; required	Array of DemandChainNode objects in the order of the chain. In a complete demand chain, the first node represents the initial advertising system and buyer ID involved in the transaction, i.e. the originator and ultimate source of payment for the creative. In an incomplete demand chain, it represents the first known node. The last node represents the entity sending this bid response.
ver	string; required	Version of the DemandChain specification in use, in the format of “major.minor”. For example, for version 1.0 of the spec, use the string “1.0” (numeric values are invalid)
ext	object; optional	Placeholder for advertising-system specific extensions to this object.

6 OpenRTB Object: DemandChainNode

This object is associated with a DemandChain object as an array of nodes. These nodes define the identity of an entity participating in the supply chain of a bid request. The DemandChainNode object contains the following attributes:

Attribute	Type	Description
asi	string; recommended	The canonical domain name of the DSP or other buyer system that is generating the bid response. This should be the same location that hosts a buyers.json file. This field is required for any ASI that is involved in the programmatic demand chain, but may be omitted for buy-side entities involved in the payment flow prior to reaching the first DSP. If present, must be a hostname or domain, not full URL. Correct: domain.com. Incorrect: https://domain.com.
bsid	string; recommended	The identifier associated with the buyer seat within the advertising system. This must contain the same value, if any, used in transactions (i.e. BidResponse.SeatBid.Seat in OpenRTB bid responses), and must be a value that appears as a buyer_id in the buyers.json file. Should be limited to 64 characters in length. This field is required for any ASI that is involved in the programmatic demand chain, but may be omitted when the asi itself is omitted, that is for buy-side entities involved in the payment flow prior to reaching the first DSP.
rid	string; optional	The OpenRTB bid request or auction ID (i.e. BidRequest.id) of the request as issued by this seller.
name	string; recommended	The name of the company (the legal entity) that is paying under the given bsid. This value is recommended but should NOT be included if it exists in the advertising system's buyers.json file (and is not marked confidential there). It MUST be included if the asi is absent or null..
domain	string; recommended	The business domain name of the entity represented by this node. This value is recommended but should NOT be included if it exists in the advertising system's buyers.json file (and is not marked confidential there). It MUST be included if the asi is absent or null, unless the buyer has literally no web

		presence.
ext	object; optional	Placeholder for advertising-system specific extensions to this object.

While none of the fields above are designated as required, an inclusive-or relationship is implied between the (asi, bsid) fields and the (name, domain) fields, Any programmatic participant in the demand chain MUST include an "asi" and "bsid" value. Any non-programmatic participant (who is thus unable to include an "asi" value) must be represented by a "name" value as well as a "domain" value unless the buyer has literally no web presence.

This spec recognizes that "name" and "domain" values for non-programmatic participants cannot be tied back to a buyers.json file and may not be consistently represented by different dchain initiators. Nevertheless, even imperfectly normalized information is still directionally valuable in helping isolate serious ad quality violations.

Note that if some party has listed confidential entries in its buyers.json, it can also use the "name" and "domain" fields in the dchain node to privately share information about these entities at the level of individual bid responses.

7 Implementation Details

If the advertising system is originating the bid request for this inventory, it should create the DemandChain object with a node for itself, showing the identity of the buyer on the system. If this buyer is the ultimate payor, the DemandChain object should include a "complete" attribute of 1, and this should be the only node in the "nodes" array.

If that buyer is not the ultimate payor, and the advertising system knows of additional payors in the chain, it should include as many of these intermediary payors as it is able. If the list is known complete, the DemandChain object should include a "complete" attribute of 1. Otherwise, if there are known to be payors beyond the ones indicated in the DemandChain object, the "complete" attribute must be set to 0.

It is invalid for a reseller to copy the DemandChain object from the previous seller to their request for that inventory without also inserting their node into the chain. If a reseller doesn't insert itself in the chain, their bid request must not include the unmodified DemandChain object it received.

If an advertising system is passing along a bid response that didn't previously contain a DemandChain object, it should create the DemandChain object itself, mark the "complete" attribute with a value of 0 and insert its node into the "nodes" array.

If an advertising system is passing along a bid response that has a DemandChain object, the advertising system should copy the existing object, keeping the original value of the "complete" flag, and append its node to the end of the "nodes" array.

8 Validation & Enforcement Guidance

As a solution based on collective participation, buyers.json and DemandChain require a common understanding of how each party should validate entries and enforce specification compliance.

An advertising system who appends a node to the DemandChain object should validate the syntactic format of the preceding node in real-time.

- If the preceding node passes syntactic validation successfully, the advertising system can append its new node and continue.
- If the preceding node fails syntactic validation, the advertising system can choose how to handle it, based on their own policy and preference.

Broken syntax often indicates a systemic problem, so the errors should at minimum be logged and investigated - advertising systems could also choose to take stricter action such as disposing of the bid response altogether. If the advertising system does process a bid response containing a syntactically invalid DemandChain object, it should not pass along that invalid object, but should instead initiate a new DemandChain object with complete set to 0.

Advertising systems may also choose to log DemandChain nodes for later asynchronous analysis against business rules: for example, confirming the buyer information in the preceding DemandChain nodes against the listed information in the buyer's buyers.json file, logging the first appearance of new seats, or any other more complex logical observations. While not necessary to perform in real-time, logging, analysis, and monitoring this information will lead to better quality data for the whole supply chain, and is expected of those systems who can do so.

9 Serializing and Exposing Creative Source Data

It is useful to expose limited data about ad creatives in the ad markup that is delivered to a client. Doing so allows problematic ads to be readily identified by participants who may not have access to bid-stream data, including publisher ad ops personnel and third-party anti-malware vendors.

This is helpful as an addition to, but not instead of, passing the dchain object in the BidResponse.seatbid.bid.ext.dchain attribute in OpenRTB 2.5 or later.

It is therefore recommended that the DSP, or other party first generating ad markup that will appear in an OpenRTB or similar programmatic bid response, serialize the following creative source data into the ad markup in a predictable manner:

- DSP domain
- buyer seat ID
- creative ID

To support versioning and platform-specific extensions, these attributes should be serialized into a comma-delimited string as follows:

```
ver,dsp_domain,buyer_seat_id,creative_id,ext
```

- `ver` should always be 1.0 for creative source data serialized compliant with this DemandChain specification.
- `dsp_domain` should be the domain where the DSP's buyers.json file can be found.
- `buyer_seat_id` should be the buyer seat ID on the DSP's platform. Normally this is the value used in the `BidResponse.SeatBid.Bid.seat` field in the bid response, and also a `bsid` value present in the DSP's buyers.json file.
- `creative_id` should be the unique identifier that the DSP uses to identify a creative, typically the `BidResponse.SeatBid.Bid.crid` value in the bid response.
- `ext` is a placeholder to allow platform-specific extensions. Format of this data, if present, is implementation-defined.

Any field whose value may contain a comma, percent sign, or other potentially unsafe character must be URL encoded per RFC 3986. Any field whose value is absent or not applicable should be represented by an empty string. Trailing commas may optionally be omitted.

Examples:

```
1.0,mydsp.example,ac1f9e,89708971
```

This example represents the buyer seat `(ac1f9e)` and creative ID `(89708971)` for `mydsp.example`.

```
1.0,opaquedsp.example,,1472%2C20%25%20off%20300x250,
```

This example represents opaquedsp.example, which is missing a buyer seat (third value empty), creative ID `(1472)` with description `(%2C20%25%20off%20300x250)` indicating (20% off 300x250), and an empty extension object (presence of a comma with nothing that follows).

Embedding of Serialized Data in Ad Markup

To simplify identification and parsing by observers of the creative source data, it is recommended to expose the serialized creative source data in ad markup as part of a standardized identifier, like this:

```
data-ad-creative-source="serialized_data"
```

The location of this identifier in ad markup will vary based on creative type and implementation details.

For HTML ad markup, the preferred location is to appear as an attribute on the outermost HTML tag. `serialized_data` should be HTML-escaped to ensure it can be safely embedded in an attribute value if it contains special characters including double quotes, backslashes, and so on.

While the outermost HTML tag is an implementation-specific detail, it may look something like this:

```
<div name="adm" data-ad-creative-source="1.0,mydsp.example,ac1f9e,89708971">
...
</div>
```

or like this:

```
<iframe name="adm" src="https://assets.mydsp.example/c/89708971" data-ad-
creative-source="1.0,mydsp.example,ac1f9e,89708971">
...
</iframe>
```

Alternately, the identifier may be embedded as an HTML comment:

```
<!-- data-ad-creative-source="1.0,mydsp.example,ac1f9e,89708971" -->
```

or as a JavaScript comment:

```
// data-ad-creative-source="1.0,mydsp.example,ac1f9e,89708971"
```

Any of these embeddings allow easy identification via a simple string match or human inspection.

For XML ad markup, such as used for VAST creatives, the preferred embedding is to create an `AdCreativeSource` tag containing the data-`ad-creative-source` attribute within the standard extension object defined for the format. `serialized_data` should be XML-escaped (effectively the same as HTML-escaped) to ensure it can be safely embedded in an attribute value if it contains special characters including double quotes, backslashes, and so on. This embedding serves the dual purpose of providing a standard, structured way of embedding the creative source data, while maintaining the ability to perform a string match in the familiar manner.

For example, in VAST:

```
<Extensions>
  <Extension>
    <AdCreativeSource data-ad-creative-
source="1.0,mydsp.example,ac1f9e,89708971">
  </Extension>
</Extensions>
```

10 Examples

A complete DemandChain includes every intermediary between the ultimate payor (typically a brand or self-serve advertiser) and the ultimate publisher. In a comparatively simple case this might be represented as:

> *Advertiser -> DSP -> SSP -> Publisher*

In a particularly complex case it could be:

> *Brand -> Agency/Holdco -> DSP of record -> additional reselling DSP -> SSP/agg1 -> SSP/agg2 -> SDK provider -> Publisher*

Some of these intermediaries, particularly the ones before the DSP of record, may not be integrated programmatically. Thus as a practical matter the first DSP to initiate a bid response should both initiate the dchain and identify any non-programmatic payment relationships downstream of the DSP, to the extent it is able. Each entity that handles the bid response must, in turn, add its link to the dchain.

Example:
An app developer uses the HypotheticalMobileMoney SDK to send a bid request to SSP1, who fans it out to various DSPs. PopularDSP responds with a bid from RespectedAdAgency, the AOR (agency of record) representing FamousBrand.

PopularDSP initially responds with:

```
"dchain" : {
  "ver": "1.0",
  "complete" : 1,
  "nodes" : [
    {
      "asi": null,
      "name": "FamousBrand",
      "domain": "famousbrand.example"
    },
    {
      "asi": null,
      "name": "RespectedAdAgency",
      "domain": "respectedadagency.example"
    },
    {
      "asi": "populardsp.example",
      "bsid": "12345",
    }
  ]
}
```

A consumer of dchain can consult the buyers.json file for PopularDSP at populardsp.example to determine that bsid 12345 indeed maps to the entity RespectedAdAgency.

SSP1 then extends the chain as it processes the bid response and passes onward to HypotheticalMobileMoney:

```
"dchain" : {
  "ver": "1.0",
  "complete" : 1,
  "nodes" : [
    {
      "asi": null,
      "name": "FamousBrand",
      "domain": "famousbrand.example"
    },
    {
      "asi": null,
      "name": "RespectedAdAgency",
      "domain": "respectedadagency.example"
    },
    {
      "asi": "populardsp.example",
      "bsid": "12345",
    },
    {
      "asi": "ssp1.example",
      "bsid": "202049",
    }
  ]
}
```

A consumer of dchain can consult the buyers.json file for SSP1 at populardsp.example to determine that bsid 202049 indeed maps to the entity PopularDSP (populardsp.example). (Even though SSP1 is an SSP, it is not always the final SSP that touches a bid response before it reaches the ultimate publisher.)

In the case of a self-service advertiser through a self-service-oriented DSP:

```
"dchain" : {
  "ver": "1.0",
  "complete" : 1,
  "nodes" : [
    {
      "asi": null,
      "name": "John's Groovy E-Commerce Business",
      "domain": "johnsgroovy.example"
    },
    {
      "asi": "selfservicedsp.example",
      "bsid": "af67ca81928ad90e",
    }
  ]
}
```

If SSP1 receives a bid response with no dchain, it will initiate one:

```
"dchain" : {
  "ver": "1.0",
  "complete" : 0,
  "nodes" : [
    {
      "asi": "ssp1.example",
      "bsid": "123456789",
    }
  ]
}
```

If a DSP does not know or is not able to share the relationship beyond its immediate contracted partner:

```
"dchain" : {
  "ver": "1.0",
  "complete" : 0,
  "nodes" : [
    {
      "asi": null,
      "name": "An Agency",
      "domain": "anagency.example"
    },
    {
      "asi": "majordsp.example",
      "bsid": "12345",
    }
  ]
}
```