Common Ad Transport Standard (CATS)

Version 1.0

Released June 2020

Drafted by the Digital Video Technical Standards Working Group
Please email support@iabtechlab.com with any questions

Visit https://www.iabtechlab.com/standards/cats for the latest version

The Digital Video Technical Standards Working Group consists of members from the following companies:

- A Million Ads
- A+E Networks
- ABC TV Network
- Activision Blizzard Media
- Ad-iD
- AdColony
- Adform
- AdGear
- ADLOOX
- Admixer EU Gmbh
- Adobe
- AdPushup
- Adswizz
- AdTheorent
- Anyclip
- Aotter
- AppNexus
- Audit Bureau of Circulations UK
- Axel Springer SE
- BARC India
- Bidstack
- Blue 449
- Bonzai
- BroadSign
- Browsi
- Cadent
- Canvas Worldwide
- Centro
- Collectcent Digital Media Pvt. Ltd.
- Comcast
- Comcast Spotlight
- Connatix Native Exchange
- Conversant Media
- Cox Media Group
- Cyber Communications Inc.
- Cyberagent, Inc.
- Dailymotion
- Dentsu Aegis
- Digital Advertising Consortium Inc.
- Digitas LBI
- Disney Interactive
- Disney Streaming Services
- Display.io
- District M
- DoubleVerify
- EMX Digital
- ESPN.com
- Experian Marketing Services
- Extreme Reach
- Flashtalking
- Flipboard
- Forensiq
- FreeWheel
- Fyber
- Google
- GroupM
- GumGum
- Hearst
- Hulu
- HyperTV
- IAB Europe
- IAB Germany
- IAB Russia
- IAB Ukraine
- Index Exchange
- InMobi
- Innovid
- Integral Ad Science
- Intowow
- Invisibly
- Jivox
- Jukin Media
- JW Player
- Kargo
- KERV Interactive
- Konduit
- Line
- MediaMath
- Meetrics
- Microsoft Advertising
- Mintegral
- NASCAR Digital Media
- Nativo
- NBCUniversal
- NEXD
- Nielsen
- Oracle Data Cloud
- Pagescience
- Pandora
- PGA TOUR
- Philo
- Pixalate
- Powerinbox
- Premion
- Protected Media
- Publicis Media
- PubMatic
- PulsePoint
- Quantcast
- Roku
- Rubicon Project
- Simulmedia
- Sizmek
- Smaato
- Smart AdServer
- Sony Pictures Television
- Sovrn
- Spark Foundry
- Spectrum Reach
- SpotX
- StackAdapt
- Starcom Worldwide
- Taboola
- Teads
- Telaria
- TenMax
- Terragon Group
- The Media Trust Company
- The New York Times Company
- The Trade Desk
- TikTok Inc.
- Timehop
- TripleLift
- Triton Digital
- true[X]
- Twitch
- Twitter
- Unruly
- Verizon Media
- Verve
- ViacomCBS
- Videonow
- ViralGains
- WWE
- Xandr
- XUMO
- Yahoo Japan Corporation
- Yomedia Network
- YOSPACE
- ZEFR

The IAB Tech Lab lead on this initiative is Amit Shetty

About IAB Technology Laboratory

License:

Table of Contents:

1. Background/Introduction

The IAB Tech Lab Common Ad Transport Standard (CATS) standardizes communication between any two parties in the advertising technology ecosystem. While OpenRTB handles communication for real time bidding (RTB) transactions, it focusses only on real-time bidding and leaves a number of use cases (examples listed below) out of scope. A complementary standard was needed for requesting ads both in a bidding context as well as outside of it. CATS has been created to service this need.

CATS is based on other industry standards to promote interoperability. For example, AdCOM (Advertising Common Object Model) is used to facilitate the packaging of information about the impression being requested while OpenRTB is used as the model for the communication protocol. CATS aims to be lightweight and simple to use to spur adoption. It allows for extensibility in the form of extension objects - similar to OpenRTB - to allow for future needs to be addressed.

CATS is another tool for bridging the gap when one system needs to request an ad – or an ad component like verification code - from another system. With CATS, this communication is standardized to promote greater interoperability.

2. Use Cases

Common expected use cases for CATS include:

- A publisher making an ad request to a publisher ad server endpoint
- A publisher making an ad request for a creative resulting from a programmatic auction
- An SSAI server making an ad request on behalf of a client device
- A verification vendor proxying an ad request to insert measurement
- A publisher ad server making an ad request
- A creative auditing system making an ad request to validate the ad creatives against policies

3. Versioning and Future Proofing

CATS will use semantic versioning as outlined at https://semver.org/

Parsers must handle CATS versioning as follows:

- If the CATS request body has a major version that the parser does not understand, it must ignore that CATS request body and return a parsing error.
- If the CATS request body has a different minor version number than the parser understands, it should attempt to parse the CATS request body and only return an error when that is not successful.

- Parsers should not use patch versions to decide whether or not to make a parse attempt.

If the CATS version is not supported and the implementer decides not to parse the request body based on the version number, it must return an HTTP 400 error and should reply with an error response.

Any CATS implementation must be able to handle extensions without breaking, even if the extensions are not used. Additionally, the CATS standard requires parsers to not break even if new objects are seen in the request. The parser should expect to see updates over time and must not break if new fields are present, even if they are not used.

CATS 1.0.0 does not define a response protocol, but this is expected in a future version of CATS. A future response protocol will most likely be JSON with a mime type of application/json+cats. However, this is subject to change until defined. As an initial use case, for video VAST (Video Ad Serving Template) is expected to be the response, but CATS is intended to support all ad types and formats.

4. Error Responses

A server responding to a CATS ad request with an error may simply return the applicable HTTP status code.

To facilitate logging and debugging of issues, a server may decide to respond with an HTTP error and a JSON body that contains the following data structure:

```
{
      "message": "some explanation of what went wrong",
      "ext": {} // optional extension object with additional info
}
```

5. Communication

HTTP POST is required for all CATS requests. TLS 1.2+ is required for any CATS compliant communication.

CATS Version Header

The x-iabtl-cats-version header indicates to the server what the CATS version of this request is. This header is required for server-to-server requests and recommended for client-to-server requests.
Ex: x-iabtl-cats-version: 1.0.0

Content Type

JSON is the standardized mechanism for data serialization and all CATS compliant systems must support it. This requirement doesn't preclude two systems from communicating in their preferred serialization language between them. The default content type is application/json.

content-type: application/json

Compression

To cut down on data transmission where possible:

- Servers responding to CATS requests must accept gzip encoding on the incoming request body.
- Servers making CATS requests must use gzip encoding on the request body.
- Clients making CATS requests should use gzip encoding on the request body.

content-encoding: gzip

6. Extensions

Extensions are allowed on every object. Even if not explicitly stated in the reference model for the request, every object has the ability to have an optional “ext” object to be included for use for proprietary data transmissions between parties. Namespaces within extension objects should be used, where possible, by adding a child object with a specific, unique name. Any implementation must be able to handle extensions without breaking, even if the extensions are not used.

Example:

```
{
      “ext”: {
            “my-custom--namespace”: {
                  “some-key”: “value”
            }
      }
}
```

7. Payload

Unless otherwise explicitly noted, all objects and variables names are lowercase. Values are not constrained by any case. Section headers are uppercased for readability.

Field Requirements

The following tables describe the CATS objects. The column “R” defines what is required or recommended. The use of “Y” in this field indicates that it is a required (must use) field. The use of “S” indicates a recommended (should use) field.

All fields must be forwarded to the next link in the supply chain without modification unless explicitly mentioned in the definition. An originating CATS request should attempt to pass as much information as possible to the first hop in the supply chain.

If a non-required field is missing, unless it has an explicitly defined default, it must be interpreted as follows:

- Fields with type `string` must be interpreted as an empty string
- Fields with type `object` must be interpreted as null
- Fields with type `array` must be interpreted as an empty array
- Fields with type `integer` or `float` must be interpreted as 0

Object: Cats

Attribute	**Type**	**R**	**Definition**
ver	string	Y	Version of the CATS being used
dataspec	object		Information regarding the data model and version used to provide contextual information in this request.
request	object	Y	Contextual request level information.
ext	object		Optional request specific extensions

Object: Dataspec

Attribute	**Type**	**R**	**Definition**
model	string, default "adcom"		Identifier of the domain model used to define items within this request
ver	string, default "1.0" for "adcom" model		Version of the domain specification being used within this request.
ext	object		Optional request specific extensions

Object: Request

Attribute	**Type**	**R**	**Definition**
id	string	Y	Unique id of the request generated by

			the system making the current request.
test	integer, default 0		Indicates if the request is a test request. Any value other than 0 indicates a GIVT-like request and should be treated accordingly.
tmax	integer	S	Maximum time in milliseconds that the requestor allows the server to respond with a response. This includes round trip time. Each link in the supply chain is able to deduct time from this field before passing to the next link.
source	object	Y	A Source object that provides data about the origin of the request.
item	Array of objects	Y	Array of Item objects, at least one must be present, that represent the available impressions being offered for the ad request.
context	object	Y	Contextual information regarding the placement available for this request. For AdCOM v1.x, the objects allowed here all of which are optional are one of the Context DistributionChannel subtypes (i.e., Site, App, or Dooh), User, Device, Regs, Restrictions, and any objects subordinate to these as specified by AdCOM.
ext	object		Optional request specific extensions

Object: Source

Attribute	**Type**	**R**	**Definition**
tid	string	S	UUIDv4 format. Transaction id generated from the initial client request such as the initial ad request from systems such as a publisher adserver, header bidding wrapper, etc. This must be generated at the origin and must be copied throughout the entire chain of subsequent requests. Any transaction id must be generated before being split into multiple requests

			and the same id must be passed on any request originating from the same source/impression.
ts	integer	Y	Unix timestamp in milliseconds since the epoch at that request was originally made. This must be copied throughout the entire chain of subsequent requests.
ext	object		Optional request specific extensions

Object: Item

Attribute	**Type**	**R**	**Definition**
id	string	Y	A unique identifier for this item within the context of the request. It is usually an incremental count for multiple items (1, 2, 3).
qty	float, default 1.0		The number of individual impressions for an item for DOOH.
seq	integer		If multiple items are offered in the same ad request, the sequence number allows for the coordinated delivery of the items.
exp	integer		Number of seconds that may elapse between the original request (see Item.ts) and impression.
dt	integer		Timestamp when the item is expected to be fulfilled (e.g. when a DOOH impression will be displayed) in Unix format (i.e., milliseconds since the epoch).
spec	object	Y	Domain-specific information regarding the impression being requested. For AdCOM v1.x, the objects allowed here are Placement and any objects subordinate to these as specified by AdCOM.
ext	object		Optional request specific extensions

8. References

This specification is built from concepts or components of other industry standards. The two relevant standards that CATS references are OpenRTB and AdCOM with links to the current specifications as of this document publication date.
Additionally, RFC 2119 terminology is used for requirement levels in this document.

OpenRTB 3.0: [https://github.com/InteractiveAdvertisingBureau/openrtb/blob/master/OpenRTB v3.0 FINAL.md](https://github.com/InteractiveAdvertisingBureau/openrtb/blob/master/OpenRTB%20v3.0%20FINAL.md)

AdCOM 1.0: [https://github.com/InteractiveAdvertisingBureau/AdCOM/blob/master/AdCOM v1.0 FINAL.md](https://github.com/InteractiveAdvertisingBureau/AdCOM/blob/master/AdCOM%20v1.0%20FINAL.md)

RFC 2119: https://tools.ietf.org/html/rfc2119

9. Appendix: CATS & VAST

This initial release of CATS was driven by the Digital Video Working Group because Video has a need to standardize video requests. With VAST4, a macro based mechanism was introduced to standardize ad requests, but that has a size limitation (2048 characters including URL) since it works over HTTP GET requests. As consent data (for GDPR/CCPA support), URLs for brand safety and other useful data starts getting sent on requests, the macro approach will quickly meet its limits. CATS addresses these problems.

In addition, using CATS has the additional advantage of implementation efficiencies by sharing AdCOM with OpenRTB, also ensuring consistency across both implementations.

This initial version of CATS does not define a response protocol, but VAST is the expected response protocol in the case of video, until CATS defines a standardized response protocol (in the future, and not currently scheduled).

VAST Response Accepted

The Accept header is used to indicate the allowed response types. For CATS requests, this information must be appended to the original Accept header. This is especially important for client requests. This information must be passed in the Accept header in all cases. For CATS 1.0.0, VAST (application/vast+xml) is used as the response protocol. Future versions of CATS will likely implement a JSON response protocol.
Ex: Accept: application/vast+xml

10. Appendix : Examples

Generic

```
{
    "cats": {
        "ver": "1.0.0",
        "dataspec": {
            "model": "adcom",
            "ver": "1.0"
        },
        "request": {
            "id": "fe4b2056-69e8-46db-be1c-10f80297c80e",
            "tmax": 300,
            "source": {
                "tid": "237d9456-f13e-43bf-b673-16c9b2a15447",
                "ts": 1576811241942
            },
            "item": [
                {
                    "id": 1,
                    "qty": 1,
                    "exp": 60,
                    "spec": {} //Refer to the AdCOM Specification
                }
            ],
            "context": {
                "site": {}, //Refer to the AdCOM Specification
                "user": {},  //Refer to the AdCOM Specification
                "device": {},  //Refer to the AdCOM Specification
                "regs": {},  //Refer to the AdCOM Specification
                "restrictions": {}  //Refer to the AdCOM Specification
            }
        }
    }
}
```

VAST 4.2 With SIMID 1.0 and OMID 1.0

```
{
    "cats": {
        "ver": "1.0.0",
        "dataspec": {
            "model": "adcom",
            "ver": "1.0"
      },
      "request": {
        "id": "fe4b2056-69e8-46db-be1c-10f80297c80e",
        "tmax": 300,
        "source": {
            "tid": "237d9456-f13e-43bf-b673-16c9b2a15447",
            "ts": 1576811241942
        },
        "item": [
            {
                "id": 1,
                "qty": 1,
                "exp": 60,
                "spec": {
                    "placement": {
                        "tagid": "plc-ftr-123abc",
                        "ssai": 1,
                        "sdk": "video player plugin",
                        "secure": 1,
                        "admx": 1,
                        "video": {
                            "ptype": 1,
                            "pos": 7,
                            "delay": 600,
                            "skip": 0,
                            "playmethod": 1,
                            "playend": 1,
                            "clktype": 2,
                            "mime": "video/mp4",
                            "api": [
                                7,
                                8
                            ],
                            "ctype": [
                                13
                            ],
                            "maxdur": 30,
                            "maxext": 0,
                            "delivery": 1,
                            "linear": 1
                        }
                    }
                }
            }
        ],
        "context": {
            "site": {
                "id": 555,
                "name": "Really Good Website",
                "domain": "page.reallygoodwebsite.com",
                "cat": [
                    74,
                    106
                ],
                "cattax": 2,
                "privpolicy": 1,
                "page": "https://page.reallygoodwebsite.com/page.php",
                "ref": "https://searchengine.com",
                "search": "really good website",
                "mobile": 1,
                "amp": 0,
                "pub": {
                    "id": "A5555",
                    "name": "Really Good Publisher",
                    "domain": "reallygoodwebsite.com",
                    "cat": [
                        74,
                        106
                    ],
                    "cattax": 2
                },
                "user": {
                    "id": "a0af45c77890045deec100acb8443baff57c",
                    "buyeruid": "fcd428245623825603 4abcdef220d9aa5892",
                    "yob": 1983,
                    "gender": "M"
                },
                "device": {
                    "type": 4,
                    "ifa": "8846d6fa10008bceaaf322908dfcb221",
                    "ip": "1.2.3.4",
                    "ua": "Mozilla/Firefox",
                    "make": "OnePlus",
                    "model": "6",
                    "hwv": "6",
                    "os": 2,
                    "osv": "10.0",
                    "mccmnc": "310-005",
                    "geo": {
                        "type": 1,
                        "lat": 42.3601,
                        "lon": 71.0581,
                        "country": "USA",
                        "utcoffset": -500
                    }
                }
            }
        }
    }
}
```

11. Appendix: Adoption Guidance

It is expected that CATS will gradually be adopted by the industry over time. This means that there will be a prolonged period of partial support from both clients and servers.

To facilitate this transition, this section includes guidance around interoperability, compatibility and adoption of CATS in a non-CATS world.

Offline Negotiated Discovery

Two parties may agree through offline channels that their systems may exchange CATS requests.

Domain Discovery

Servers that support CATS can announce that they do so by hosting a well-known document at
https://my-server/.well-known/iabtl-cats.json

The response from this URL should be a JSON file with the following contents:
{“version”:”1.0.0”}

When this file is present on the server and correctly returns the above response, requests to any endpoint on that domain may be made using CATS, see also “Optimistic Discovery” below.

Connection Discovery

Servers that support CATS and are responding to any ad request (even non-CATS) are encouraged to add the following header to their HTTP responses:
x-iabtl-cats-support: 1.0.0

This indicates to the requestor that the server supports CATS for future ad requests. Clients may detect the presence of this header and decide to use CATS on future ad requests on the same HTTP connection.

Optimistic Discovery

Clients may also optimistically send a CATS request and fall back to a non-CATS request if the initial attempt fails or results in an unexpected response.

If the server is known to support CATS, or responds with a CATS error response, the client should not re-attempt the request, to prevent flooding the server with traffic and overloading it.