How Machine-Checkable Is OpenRTB? Classifying the Normative Content of the Protocol That Clears Real-Time Advertising

Aleksander Sekowski

Independent Researcher

rtblint.org . github.com/aleksUIX/rtblint

Preprint - July 2026

Abstract

OpenRTB is the wire protocol of open programmatic advertising, the exchange-traded layer of a United States internet advertising market that reached $294.6 billion in 2025. Unlike protocols governed by IETF or ISO process, OpenRTB has no reference validator, no conformance test suite, and no formal normative language conventions: neither OpenRTB 2.6 nor OpenRTB 3.0 references RFC 2119, and outside the boilerplate legal disclaimer neither contains a single uppercase normative keyword. This paper asks a question that has not been answered quantitatively before: how much of the OpenRTB specification can any validator, of any design, actually enforce?

We extract every sentence carrying normative keywords from the canonical OpenRTB 2.6 (release 2.6-202606) and OpenRTB 3.0 specification texts: 417 sentences, of which 288 survive screening as genuine conformance statements. We hand-classify each into four enforceability classes: (A) expressible in JSON Schema on a single document, (B) checkable by a stateless lint rule on a single message, (C) requiring cross-message state, a second artifact, or runtime interaction, and (D) carrying no machine-decidable conformance criterion at all. A blind recoding of a 60-statement sample agrees with the author's labels at Cohen's kappa 0.77 to 0.79, and the headline conclusions survive the recoders' stricter boundary readings. In OpenRTB 2.6, 53.5 percent of conformance statements are statically checkable (14.5 percent schema, 39 percent lint), 20.5 percent require runtime context, and 26 percent are not machine-decidable; deduplicating clauses the specification repeats verbatim across objects shifts these to 45.7 percent statically checkable and 31.8 percent undecidable, so between a quarter and a third of the specification's conformance content has no machine test under either counting. OpenRTB 3.0 inverts the profile: 50 percent of its conformance content is runtime behavior and only 27.3 percent (24.1 percent deduplicated) is statically checkable. Recommendation-strength statements decay fastest: 33 percent of "should" statements in 2.6 have no decidable criterion, against 16 percent of "must" statements. A longitudinal analysis of the object catalogs across 16 releases (2012 to 2026) shows the protocol surface grew from 167 to 417 field definitions while the required core grew only from 16 to 22, meaning 94.7 percent of the modern protocol is optional surface whose usage is settled bilaterally, off the wire. The absence of a conformance loop is measurable inside the specification itself: two of the nine example payloads embedded in the 2.6 text violate the document's own object tables, and have done so in every release since October 2022. We catalog the recurring ambiguity patterns that force

implementations to diverge, relate the findings to the robustness-principle critique and to supply-chain transparency studies, and publish the statement dataset, the classification codebook, and a 96-rule open-source validator (rtblint) as artifacts.

1. Introduction

Real-time bidding conducts a sealed-bid auction for an advertising impression in the time a web page takes to load. The protocol that carries these auctions, OpenRTB, is maintained by the IAB Tech Lab and underpins open programmatic trading, the exchange-traded segment of a US internet advertising market that reached $294.6 billion in 2025 [1]. A bid request serialized as OpenRTB JSON leaves a supply-side platform, crosses several intermediaries, and returns as a bid response carrying a price and ad markup, within a latency budget on the order of 100 milliseconds [13].

For a protocol of this economic weight, OpenRTB's conformance infrastructure is remarkably thin. The IAB Tech Lab publishes no reference validator and no conformance test suite; its compliance program is a paid audit, not public tooling [30]. The specification itself is prose with field tables. It does not adopt the RFC 2119 normative vocabulary that IETF specifications have used since 1997 [5, 6]: we find zero occurrences of uppercase MUST, SHOULD, or MAY in either the 2.6 or 3.0 specification body. Requirement strength is conveyed by lowercase prose ("must", "should", "may"), by table annotations ("required", "recommended"), and by definitions that are themselves idiosyncratic: OpenRTB 2.6 defines "required" as attributes "designated as such if their omission would technically break the protocol," and explicitly disclaims normative intent for "recommended" attributes, which "should not be interpreted as a suggestion by IAB Tech Lab that an implementer propagate any specific attribute" [2].

The practical consequence is that every exchange, bidder, and intermediary ships its own interpretation. The industry's own transparency studies measure the downstream cost of an unauditable supply chain: ISBA and PwC could match only 12 percent of 267 million impressions end to end and left an average 15 percent of advertiser spend in an unattributable "unknown delta" [9]; the ANA estimated a $22 billion efficiency opportunity in 2023 [10]. Auditing a supply chain presupposes that the messages crossing it conform to a common contract. This paper measures how much of that contract exists in machine-checkable form.

We make five contributions:

1. A statement-level enforceability dataset. We extract 417 normative-keyword sentences from the canonical OpenRTB 2.6-202606 and OpenRTB 3.0 texts, screen them to 288 conformance statements, and hand-classify each into a four-class enforceability taxonomy with a published codebook.
2. A quantified enforceability profile of both protocol generations. OpenRTB 2.6 is 53.5 percent statically checkable; OpenRTB 3.0 is 27.3 percent, because its normative weight sits in transport and event behavior. In both, between a quarter and a third of the conformance content has no machine-decidable criterion.
3. A longitudinal catalog analysis. Across 16 releases from 2012 to 2026, the field surface

grew 2.5-fold while the required core stayed nearly flat, quantifying how the protocol accumulates optional, bilaterally-interpreted surface.

4. An internal conformance finding. Two of the nine example payloads embedded in the current 2.6 specification violate the specification's own object tables, and have done so in every release since October 2022; the defects sit in the mechanically checkable layer.
5. Open artifacts. The statement dataset, the extraction and classification scripts, the per-version object catalogs, and rtblint, a 96-rule OpenRTB validator with JSON Schema exports for 17 spec releases, are available open source.

The remainder of the paper is organized as follows. Section 2 provides background and related work. Section 3 describes the corpus and classification method. Section 4 presents the quantitative results. Section 5 catalogs the recurring ambiguity patterns behind class D. Section 6 discusses implications for specification authors, implementers, and the auditability agenda. Sections 7 through 9 cover limitations, future work, and conclusions.

2. Background and Related Work

2.1 OpenRTB

OpenRTB 2.0 was published in 2012 and the 2.x line has been the industry's transaction protocol since. Version 2.6, finalized in April 2022, moved to dated continuous releases (2.6-YYYYMM) in the InteractiveAdvertisingBureau/openrtb2.x repository; the latest at the time of writing is 2.6-202606 (June 2026), and no OpenRTB 2.7 exists [2]. OpenRTB 3.0, finalized in November 2018, is a structural rewrite that layers transport objects over a separate domain object model, AdCOM 1.0 [3, 4]. Industry adoption of 3.0 has remained minimal; the market standardized on the actively maintained 2.6 line. We analyze both, because they represent two philosophies of the same protocol and, as Section 4 shows, two different enforceability profiles.

An OpenRTB transaction minimally involves a bid request (an id and one or more imp impression objects, with device, user, site or app context), a bid response (seatbid arrays of bid objects carrying price and markup), and a set of post-auction interactions: win, billing, and loss notices carried by URL fields (nurl, burl, lurl) with substitution macros such as ${AUCTION_PRICE}. This structure matters for enforceability: the payloads are checkable objects, but much of the protocol's meaning lives in the interaction sequence around them.

2.2 Normative language and machine-checkable specifications

RFC 2119 established a reserved vocabulary (MUST, SHOULD, MAY) for expressing requirement levels [5], and RFC 8174 clarified that only the uppercase forms carry normative force [6]. The IETF ecosystem increasingly pairs prose with machine-readable formalisms: JSON payloads are constrained by JSON Schema [8] or CDDL [24], and profiles such as I-JSON exist precisely because "valid JSON" underdetermines interoperable behavior [23, 17]. The conformance-testing discipline itself was standardized for OSI protocols in ISO/IEC 9646 [20], which distinguishes static conformance review from dynamic test campaigns; the distinction reappears in our class taxonomy.

A specification written in prose can be studied as an artifact. Yen et al. built SAGE, which found five ambiguities and six under-specifications in the ICMP RFC and showed they block automatic code generation [14]. Pacheco et al. extracted finite state machines from RFC prose and synthesized protocol attacks from the extraction gaps [15]. The LangSec literature argues that input languages must be formally recognizable or parsing differentials become security boundaries [16], and Seriot demonstrated empirically that real JSON parsers disagree on edge cases [17]. Murata et al. gave the formal-language taxonomy for XML schema expressiveness [18]; the same question, what a schema language can and cannot state, drives our class A/B boundary. Newcombe et al. documented the industrial payoff of formal specification at AWS [19], and RFC 9413 collects the IETF's own critique of the robustness principle [21, 22], which is directly relevant because OpenRTB mandates tolerant reading as a compliance requirement (Section 5.5).

2.3 Ad tech measurement and validation

Measurement studies have quantified the RTB ecosystem's behavior (header bidding [12], exchange fraud [11], the RTB survey of Wang et al. [13]) and its financial opacity [9, 10]. Tooling exists in fragments: community validators cover OpenRTB up to 2.5, Google publishes protocol buffer bindings, and the IAB Tech Lab offers a paid compliance audit [30]. We are not aware of prior published work that quantifies the enforceability of the OpenRTB specification itself. The nearest antecedent is our earlier preprint on VAST validation at bid-time scale [31], which addressed the creative payload carried inside bid.adm; this paper addresses the protocol that carries it.

3. Method

3.1 Corpus

We analyze the canonical specification texts as published in the IAB Tech Lab repositories: the OpenRTB 2.6-202606 markdown (2,269 lines) and the OpenRTB 3.0 final markdown (1,461 lines). Front matter, licenses, tables of contents, and the legal disclaimer are excluded. Field-definition tables (markdown tables in 2.6, HTML tables in 3.0) are parsed structurally into (object, attribute, type annotation, description) tuples; all remaining text is treated as body prose.

3.2 Statement extraction

We segment prose and field-description cells into sentences and retain every sentence containing at least one normative marker: must, must not, shall, should, should not, may, required, recommended, optional, cannot, prohibited, not permitted, not allowed (case-insensitive). This yields 417 keyword sentences: 280 from 2.6 (121 body prose, 159 field descriptions) and 137 from 3.0 (96 prose, 41 field descriptions). Each sentence is also tagged with an obligation strength: obligation (must/shall/required/cannot), recommendation (should/recommended), or permission (may/optional), by its strongest marker. One tagging artifact is disclosed rather than corrected: fourteen statements of the form "only one of X and Y may be present" are grammatically permissive but semantically prohibitive; they are tagged by their surface marker and therefore counted under

permissions.

Keyword recall has a known blind spot: the field tables encode type and requiredness constraints positionally ("integer; required") without normative sentences. We therefore maintain a second, exhaustive dataset of field-level constraints parsed from every object table (Section 3.4), and treat the sentence dataset as covering the specification's prose normativity.

3.3 Classification codebook

Each keyword sentence was screened and classified by the author. Sentences that state no conformance constraint (definitions of terms, explanatory rationale, worked examples, changelog and errata text) are labeled X and excluded from enforceability denominators; 129 of 417 sentences fall in this class. The remaining 288 conformance statements receive exactly one of four classes:

- Class A - JSON Schema. The constraint is expressible in JSON Schema draft 2020-12 against a single document: types, unconditional presence, enumerated values, ranges, lengths, simple patterns. Example: "At least 1 Imp object is required."
- Class B - stateless lint. The constraint is decidable from a single message but exceeds practical schema expressiveness: cross-field conditionals and exclusivities, external registry membership (ISO 4217, BCP 47, content taxonomies), conditional requiredness, markup coherence of an embedded creative. Example: "A bid request with a DOOH object must not contain a site or app object."
- Class C - runtime, cross-message, or cross-artifact. Deciding conformance requires more than one message in isolation: request/response pairing, win/billing/loss notice behavior, macro substitution at event time, HTTP transport and header requirements, immutability across intermediaries, or fetching a second artifact such as ads.txt or sellers.json. Example: "ID of the bid request to which this is a response; must match the request.id attribute."
- Class D - not machine-decidable. No observable criterion exists at any layer: values "coordinated between bidders and the exchange a priori," semantics settled by bilateral business agreement, provenance claims that cannot be verified from the data, and quality guidance with no threshold. Example: "IDs of seats and knowledge of the buyer's customers to which they refer must be coordinated between bidders and the exchange a priori."

Two boundary policies deserve note. First, presence recommendations ("device object is recommended") are class B: a linter can warn deterministically on absence, and this is precisely what production linters do. Recommendations conditioned on facts outside the message ("recommended only if the exchange accepts multiple currencies") are class D. Second, permissions are classified by the constraint they impose on the message grammar: "any object may contain extensions" is class A because a schema must encode the allowance, while permissions granting behavioral discretion ("win notices may be either POST or GET") follow the layer of the behavior. Sentences bundling several constraints are coded by their dominant clause. Heuristic pre-labeling seeded the pass, but every one of the 417 labels was assigned or confirmed manually; the labeled dataset ships with the paper so the coding can be audited row by row.

As a reliability check, a random sample of 60 of the 417 sentences (fixed seed, published) was independently recoded by two runs of a large language model given only the codebook above, blind to the author's labels and to each other. Agreement with the author's labels is 81.7 and 83.3 percent (Cohen's kappa 0.77 and 0.79), substantial agreement on the Landis and Koch scale [32]. The two runs agree with each other on 59 of 60 items; since both are runs of the same model, this measures the stability of the codebook under a consistent reader, not human independence, and we disclose it as such. Disagreements concentrate on two boundaries already flagged above as judgment calls: whether recommendations conditioned on facts outside the message ("should be included if the content is a DOOH screen," store-dependent bundle formats) are lint-checkable presence warnings (author) or undecidable (recoder), and where the schema/lint line sits for constant-value and cardinality constraints. Adopting the recoders' stricter reading of the first boundary would move 12 statement instances from class B to class D and lower the 2.6 statically checkable share from 53.5 to 47.5 percent; no conclusion in Section 4 changes under that reading. The sample, both recodings, and the computation ship with the artifacts.

3.4 Field-constraint and catalog datasets

Independently of the sentence dataset, we parse every object-definition table in both specifications and, for the longitudinal analysis, use the machine-extracted object catalogs that ship with rtblint: one catalog per specification release, 2.0 through 2.6-202606 plus 3.0, each recording every object, field, type annotation, and requiredness marker with a line-level citation into the archived source text. The sole exception is 2.6-202204, whose PDF-only source has no extracted catalog; it is omitted from the longitudinal analysis. The 2.6-202606 catalog records 38 objects and 417 field definitions; the 3.0 transport catalog records 10 objects and 80 field definitions (its domain objects live in AdCOM and are out of scope here).

3.5 The validator instrument

As a lower bound on what static validation covers in practice, we use rtblint, our open-source OpenRTB linter: 96 stable rule identifiers spanning shape checks against the per-version catalogs, documented-value checks against 43 referenced AdCOM value lists, semantic single-message rules, and 8 request/response pair rules; it exports JSON Schemas for 17 specification releases. The instrument matters to the argument in one direction only: everything it enforces is demonstrably enforceable, so its coverage certifies our A and B labels are non-empty in practice rather than in principle.

4. Results

4.1 The enforceability profile

Table 1 and Figure 1 give the headline distribution over the 288 conformance statements, counted two ways: per statement instance, and deduplicated by normalized text, because the specification repeats some clauses verbatim across objects (the keywords/kwarray exclusivity boilerplate alone appears eight times).

Spec	n	A: schema	B: lint	C: runtime	D: undecidable	A+B
OpenRTB 2.6-202606, instances	200	14.5%	39.0%	20.5%	26.0%	53.5%
OpenRTB 2.6-202606, deduplicated	151	15.2%	30.5%	22.5%	31.8%	45.7%
OpenRTB 3.0, instances	88	17.0%	10.2%	50.0%	22.7%	27.3%
OpenRTB 3.0, deduplicated	83	14.5%	9.6%	51.8%	24.1%	24.1%
Combined, instances	288	15.3%	30.2%	29.5%	25.0%	45.5%

Deduplication lowers the statically checkable share of 2.6 by eight points, because the lint class contains most of the repeated boilerplate, and raises the undecidable share to just under a third. Every conclusion below holds under either counting; where the text quotes a single figure, it is the instance-weighted one, and Figure 1 plots instances.

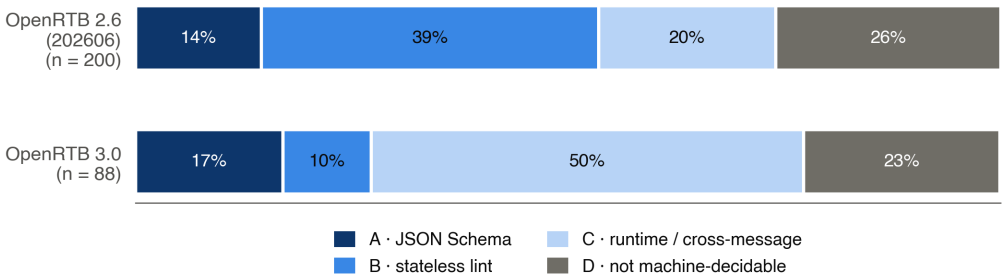

Figure 1. Enforceability class shares of conformance statements in OpenRTB 2.6-202606 and OpenRTB 3.0.

Three observations. First, JSON Schema, the formalism most often proposed as "the fix" for loose JSON protocols, can express only about one in seven of OpenRTB 2.6's conformance statements. The largest enforceable tranche (39 percent) needs a rule engine: conditional requiredness, mutual exclusions, registry lookups, embedded-markup coherence. A schema is necessary but not close to sufficient.

Second, between a quarter and a third of each specification is not machine-decidable by construction. This is not a parsing limitation; the statements delegate their meaning to bilateral agreements or to facts no validator can observe. Section 5 catalogs the patterns.

Third, the two protocol generations fail differently. OpenRTB 3.0 moved its domain vocabulary out to AdCOM and spent its own normative budget on transport, security, and eventing: HTTPS and TLS 1.2+ mandates, version headers, compression negotiation, billing and loss notice semantics, and the ads.cert signing chain. Half of 3.0's conformance content is class C, checkable only by observing live interactions. Its statically checkable share (27.3 percent) is half that of 2.6. A protocol can become more rigorous and simultaneously less lintable.

4.2 Obligation strength predicts enforceability

Figure 2 splits the 2.6 statements by obligation strength.

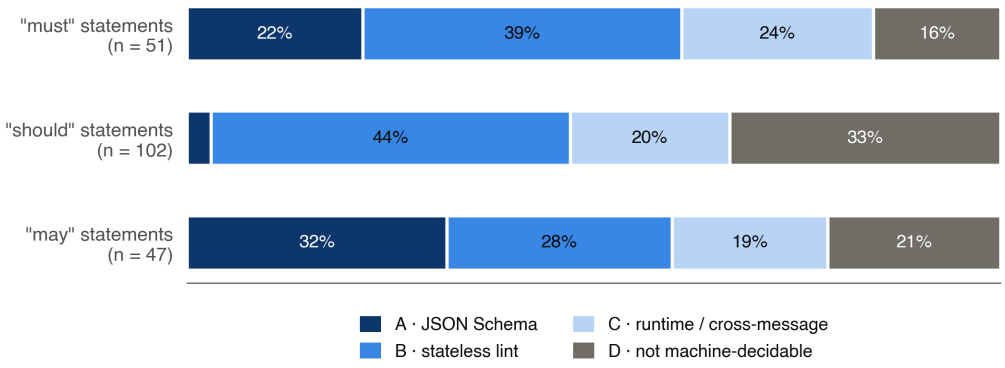

Figure 2. Enforceability class by obligation strength, OpenRTB 2.6-202606 conformance statements.

"Must" statements are the most enforceable: 61 percent statically checkable, 16 percent undecidable. "Should" statements are the least: 33 percent undecidable, and they are also the largest cohort (102 of 200). The specification's preferred way to express expectations, the lowercase "should," is the construction most likely to have no conformance test. This matches the intuition behind RFC 8174's uppercase-only rule [6]: requirement strength that lives in ordinary prose diction drifts into ordinary prose vagueness.

The permission profile is bimodal. Permissions that shape the message grammar ("substitution macros may be included," "any object may contain extensions") are class A allowances a validator must encode or it will reject legal traffic; permissions granting operational discretion are undecidable by design. Notably, a validator that ignores permissive statements will be wrong in the strict direction, producing false rejections, which practitioners experience as validators that "don't work on real traffic." Twelve of the 47 "may" statements are the exclusivity prohibitions noted in Section 3.2 (the other two of the fourteen are in 3.0); moving them to the obligation column would shift the "may" bar further toward class B without changing the ordering of the three rows.

4.3 The field tables are the checkable core, and they are 95 percent optional

The 2.6-202606 object catalog defines 417 fields across 38 objects. Of these, 22 (5.3 percent) are marked required and 37 recommended; 34 carry inline enumerations and 35 reference external registries or taxonomies. The type layer of all 417 is schema-expressible, which is why the field tables dominate classes A and B, and why validators feel most solid exactly there.

But requiredness itself is conditional more often than it is absolute. "Required if this impression is offered as a video ad opportunity" (each of the four media objects), "1+ required if a bid is to be made" (seatbid), "required only for the dynamic portion(s) of video ad pods" (poddur), "required for Flex Ads" (wratio): these are class B conditionals, not class A presence checks, and several hinge on intent ("if a bid is to be made") that the message only reveals implicitly.

4.4 Sixteen releases of growth in the optional surface

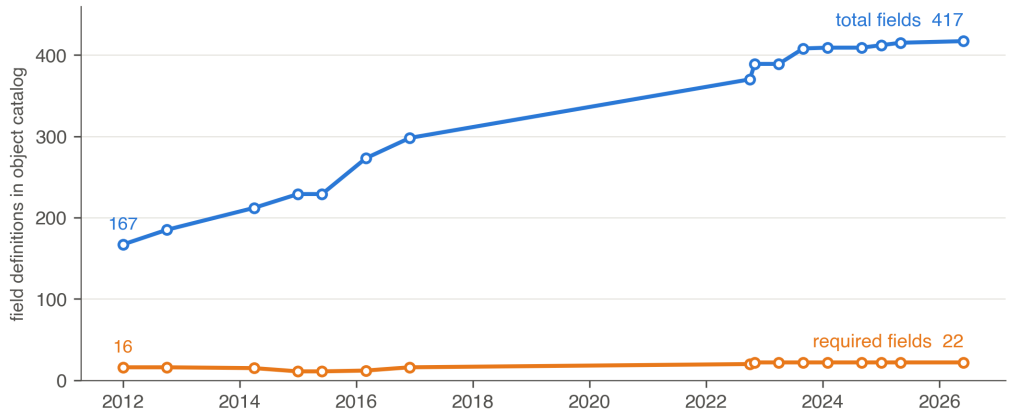

Figure 3. OpenRTB 2.x object catalog growth, 2012 to 2026: total field definitions versus required field definitions per release.

Figure 3 tracks the catalogs across the 2.x line. The field surface grew from 167 definitions in 2.0 (2012) to 417 in 2.6-202606 (2026); the required core moved from 16 to 22. The releases since 2022 added fields for GPP consent, structured user agents, DOOH, ad pod deduplication, EID provenance, content genre taxonomies, extended content identifiers, and programming liveness; seven of the nine monthly releases with extractable catalogs grew the field surface, the other two changed prose and macros only. Almost all of the additions are optional, and many carry their own bilateral-interpretation clauses. The protocol's growth mode is accretion of optional vocabulary: each addition expands the space of legal-but-divergent implementations while leaving the machine-enforceable contract nearly unchanged. In ISO 9646 terms [20], the static conformance requirements have been flat for a decade while the dynamic and bilateral surface compounds.

4.5 What a validator covers today

Mapped against this profile, rtblint's 96 rules land as follows: 14 operate at the schema layer (payload shape, types, presence, documented values, per-version field existence), 74 are single-message semantic rules including version-delta diagnostics (deprecated, moved, removed fields across the 17 supported releases), and 8 are request/response pair rules that reach into class C (bid.impid must reference an offered impression, bid mtype against offered media types, deal references, seat and currency restrictions, response id matching request id). Cross-artifact class C checks (ads.txt, sellers.json consistency) are implementable but require fetching infrastructure outside a linter's trust boundary. Class D is irreducible: no engineering effort recovers a conformance test the specification never defined.

4.6 The specification's own examples do not conform

If the enforceable half of the specification were actually being enforced anywhere, the first payloads to pass through the loop would be the examples the specification itself ships. We extracted every fenced JSON payload from the 2.6-202606 text that parses as a bid request or response (9 payloads) and validated each against the version's own object tables. Two fail:

- The user-data example places a value attribute directly on a Data object. The

specification's own Data table (Section 3.2.21) defines no such field; values belong on the Segment objects nested inside data.segment.

- The video example populates imp[].video.apis. The Video table (Section 3.2.7) defines only api; apis exists solely on the response-side Bid object, where it deprecates api.

We verified both against the object tables by hand to rule out extraction artifacts. Both defects are present in every archived monthly release from 2.6-202210 through 2.6-202606, a span of nearly four years and nine releases. Both sit squarely in the mechanical layer this paper labels class A: a schema generated from the field tables and run over the document's own code blocks in continuous integration would have flagged them on the day they were committed. The finding is small in isolation and large as evidence: the specification has no conformance loop even internally, so the drift the taxonomy predicts for implementations begins in the normative document itself. The extraction and validation script ships with the paper's artifacts.

5. The Ambiguity Catalog

The 72 class D statements (68 unique after deduplication) are not random noise; they cluster into five patterns. We quote representative statements verbatim.

5.1 A priori bilateral coordination

The most common pattern delegates semantics to out-of-band agreement: seat IDs ("must be coordinated between bidders and the exchange a priori"), metric vendors and types ("exchange curated string names which should be published to bidders a priori"), data segment taxonomies ("must be published by the exchange a priori"), carrier names, tactic IDs, and even the macro encoding algorithm ("not defined by this specification and must be mutually agreed upon between parties"). Each such clause makes the wire format a projection of a private contract. Two conformant integrations can exchange identical bytes with different meanings, and no observer, including an auditor, can distinguish them from the traffic.

5.2 Conditions on unobservable facts

A second cluster conditions conformance on facts outside the message: cur is "recommended only if the exchange accepts multiple currencies"; app bundle formats depend on which app store the inventory lives in; the structured user agent fields "should" be derived from specific Sec-CH-UA headers, a provenance claim nothing in the payload can verify; lat/lon "should only be passed if they conform to the accuracy depicted in the type attribute," where the accuracy of a coordinate is precisely what a validator cannot know.

5.3 Receiver-interpretation rules

Some statements govern how the receiver must think: sua "should be considered the more accurate representation" when both user agent forms are present; a user ID "should be stable long enough to serve reasonably as the basis for frequency capping"; "integer math is highly recommended when handling currencies." These are real interoperability constraints, yet none has

an observable conformance criterion at the message layer.

5.4 Policy-contingent behavior

The notification layer is normative and simultaneously waivable: loss notice support "is not required for OpenRTB compliance"; exchanges "may" redact ${AUCTION_PRICE} to an empty string by policy; billing best practice is a "should" with explicitly tolerated deviations. The revenue-bearing events of the protocol are the least specified part of it, which is worth holding next to the industry's measured settlement discrepancies [9, 10].

5.5 Mandated tolerance

Finally, the specification mandates the robustness principle as a compliance requirement: implementers "must tolerate receiving new or unexpected fields and enumerated list values gracefully" and "should freely transmit new fields." Tolerant reading is what RFC 9413 warns ossifies protocols and hides errors [21]: combined with a 26 percent undecidable share and no conformance suite, it guarantees that malformed or misinterpreted traffic circulates without backpressure. LangSec predicts the consequence: when the input language is not decidable, every implementation's parser becomes its own de facto specification [16], and Seriot's JSON study shows how much parsers diverge even on a fully formalized grammar [17]. The Prebid project's ortbConverter, a dedicated translation layer between the dominant open-source stack's internal model and OpenRTB [29], is the ecosystem quietly pricing this in. So is the fact that when Google moved its exchange to first-price auctions in 2019, the meaning of price-related fields shifted ecosystem-wide by announcement rather than by protocol change [28].

6. Discussion

6.1 For specification authors

Three low-cost changes would move mass out of class D. First, adopt RFC 2119/8174 conventions; the finding in Section 4.2 that "should" statements are twice as likely as "must" statements to be undecidable is an argument for making requirement strength lexically auditable. Second, ship the field tables as machine-readable artifacts (JSON Schema for class A, a documented rule list for class B conditionals) alongside the prose; the object catalogs used in this paper demonstrate the extraction is mechanical, and an official version would eliminate an entire category of divergence. Third, for every "a priori" delegation, either define a discovery mechanism (as ads.txt and sellers.json did for seller identity [25, 26, 27]) or mark the field explicitly as bilateral so tooling can treat it as opaque rather than guessing.

6.2 For implementers and auditors

The static half of OpenRTB 2.6 is cheaply enforceable today: rtblint's committed benchmark suite (100,000 validations, Apple M4) validates a typical small bid request in about 10 microseconds and mid-size payloads in under 0.4 milliseconds in batch mode, and the companion VAST study reached the same conclusion for the creative layer [31]. The uncomfortable half is class C: the

request/response, notification, and settlement behavior where the money moves. ISO 9646-style dynamic conformance testing [20] has no ad tech equivalent, and the transparency studies suggest what its absence costs. An industry that funded a $22 billion waste estimate [10] has not funded a public conformance harness for its own transaction protocol; the asymmetry is the finding.

6.3 The 3.0 lesson

OpenRTB 3.0 is the more rigorous document (TLS mandated, signing specified, eventing formalized) and the less adopted one. Its enforceability profile shows the two facts are related: rigor concentrated in class C raises the cost of every integration while remaining invisible to static tooling, so no individual adopter can demonstrate conformance cheaply. A specification that wants adoption and rigor needs its rigor where validators can reach it.

7. Limitations

Classification was performed by a single human coder (the author). We mitigate by publishing every labeled sentence with its location so any label can be contested, by seeding with deterministic heuristics whose rationale is recorded per row, and by the independent recoding check reported in Section 3.3. The author also maintains rtblint, and a validator author has an interest in the checkable share appearing large; the mitigation, beyond the row-level dataset, is that the central finding runs against that interest: roughly half the specification cannot be validated by any tool, including ours. Sentence-level keyword extraction misses normative content expressed without keywords and occasionally splits or bundles constraints; bundled sentences were coded by dominant clause. Counting is instance-weighted by default, which up-weights clauses the specification repeats across objects; Section 4.1 reports the deduplicated distribution alongside it and the conclusions survive both. The obligation taxonomy treats lowercase modal verbs as normative even though the specification never formally grants them that status, which is itself one of the paper's findings. The catalogs derive from rtblint's extraction pipeline; its markdown table parser disagrees with the shipped catalog by two rows on 2.6-202606 (415 versus 417 field rows) due to line-wrapped cells, a discrepancy we report rather than hide. Finally, we analyze what the specification says, not what traffic does; the enforceability ceiling measured here is an upper bound on what any wire-level study can check, not a measurement of violation rates.

8. Future Work

The natural sequel is an in-the-wild conformance study: capturing real OpenRTB traffic (for example, at a browser vantage point where client-side auctions are observable) and measuring violation rates against the enforceable half of the profile, per endpoint and per version. Extending the catalog analysis into AdCOM would complete the 3.0 picture. A second coder over the published dataset would firm up the taxonomy. Cross-layer validation, checking the OpenRTB contract against the VAST creative it carries (duration versus maxduration, media types versus mimes), connects this paper to the companion creative-layer work [31].

9. Conclusion

Measured at the sentence level, the OpenRTB 2.6 specification is roughly half machine-checkable (53.5 percent by statement instance, 45.7 percent deduplicated), and the largest checkable share needs lint rules, not schemas. A quarter of it cannot be validated by any tool because the text delegates meaning to private agreements and unobservable facts. The protocol has grown for fourteen years almost entirely in its optional surface, its requirement language has never been formalized, and its mandated tolerance for unexpected input removes the feedback that would otherwise surface divergence. The cost of the missing conformance loop is visible in the normative document itself: the specification's own example payloads have violated its own object tables in every release for nearly four years. None of this is unfixable: the checkable core is mechanically extractable (we publish it), the conditional rules are enumerable (we implement 96 of them), and the precedent for converting bilateral folklore into discoverable artifacts exists inside ad tech itself in ads.txt and sellers.json. What has been missing is a quantified account of where the specification stands. This paper supplies one: 288 classified conformance statements, four enforceability classes, two protocol generations, and a validator that certifies the enforceable half is enforceable in practice.

Data and Artifact Availability

The statement dataset (417 labeled sentences with locations and rationales), the field-constraint and per-version catalog datasets, the reliability sample with both recodings, the extraction, classification, reliability, and spec-example validation scripts, and the figures are archived in the rtblint repository under paper/openrtb-checkability/. rtblint itself (Rust core, CLI, WebAssembly and MCP interfaces, JSON Schema exports for 17 OpenRTB releases, committed benchmark suite) is open source at github.com/aleksUIX/rtblint with a live playground at rtblint.org.

Suggested Citation

Sekowski, A. (2026). How Machine-Checkable Is OpenRTB? Classifying the Normative Content of the Protocol That Clears Real-Time Advertising. Preprint, ResearchGate. doi:10.13140/RG.2.2.27937.57448.

References

[1] IAB and PwC. Internet Advertising Revenue Report: Full Year 2025. IAB, April 2026. iab.com/insights/internet-advertising-revenue-report-full-year-2025. US internet advertising revenue of $294.6 billion in 2025.

[2] IAB Tech Lab. OpenRTB 2.6 Specification, releases 2.6-202204 through 2.6-202606. github.com/InteractiveAdvertisingBureau/openrtb2.x. Finalized April 2022; dated continuous releases thereafter.

[3] IAB Tech Lab. OpenRTB Specification v3.0. Finalized November 2018.

github.com/InteractiveAdvertisingBureau/openrtb.

[4] IAB Tech Lab. AdCOM 1.0 (Advertising Common Object Model). November 2018. github.com/InteractiveAdvertisingBureau/AdCOM.

[5] Bradner, S. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, BCP 14, IETF, March 1997. doi:10.17487/RFC2119.

[6] Leiba, B. Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. RFC 8174, BCP 14, IETF, May 2017. doi:10.17487/RFC8174.

[7] Bray, T. (ed.). The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, STD 90, IETF, December 2017. doi:10.17487/RFC8259.

[8] Wright, A., Andrews, H., Hutton, B., and Dennis, G. JSON Schema: A Media Type for Describing JSON Documents. Draft 2020-12, json-schema.org.

[9] ISBA and PwC. Programmatic Supply Chain Transparency Study. ISBA, May 2020. 15 percent of advertiser spend unattributable; 12 percent of 267 million impressions matched end to end.

[10] Association of National Advertisers. Programmatic Media Supply Chain Transparency Study: Complete Report. ANA, December 2023. $22 billion efficiency opportunity; MFA sites at 21 percent of impressions.

[11] Stone-Gross, B., Stevens, R., Zarras, A., Kemmerer, R., Kruegel, C., and Vigna, G. Understanding Fraudulent Activities in Online Ad Exchanges. In Proceedings of ACM IMC 2011, pp. 279-294. doi:10.1145/2068816.2068843.

[12] Pachilakis, M., Papadopoulos, P., Markatos, E. P., and Kourtellis, N. No More Chasing Waterfalls: A Measurement Study of the Header Bidding Ad-Ecosystem. In Proceedings of ACM IMC 2019, pp. 280-293. doi:10.1145/3355369.3355582.

[13] Wang, J., Zhang, W., and Yuan, S. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting. Foundations and Trends in Information Retrieval, 11(4-5):297-435, 2017. doi:10.1561/1500000049.

[14] Yen, J., Lévai, T., Ye, Q., Ren, X., Govindan, R., and Raghavan, B. Semi-Automated Protocol Disambiguation and Code Generation. In Proceedings of ACM SIGCOMM 2021. doi:10.1145/3452296.3472910.

[15] Pacheco, M. L., von Hippel, M., Weintraub, B., Goldwasser, D., and Nita-Rotaru, C. Automated Attack Synthesis by Extracting Finite State Machines from Protocol Specification Documents. In Proceedings of IEEE Symposium on Security and Privacy 2022, pp. 51-68. doi:10.1109/SP46214.2022.9833673.

[16] Sassaman, L., Patterson, M. L., Bratus, S., and Shubina, A. The Halting Problems of Network Stack Insecurity. ;login: (USENIX), 36(6):22-32, December 2011.

[17] Seriot, N. Parsing JSON is a Minefield. seriot.ch/projects/parsing_json.html, 2016 (updated 2018).

[18] Murata, M., Lee, D., Mani, M., and Kawaguchi, K. Taxonomy of XML Schema Languages Using Formal Language Theory. ACM Transactions on Internet Technology, 5(4):660-704, 2005.

doi:10.1145/1111627.1111631.

[19] Newcombe, C., Rath, T., Zhang, F., Munteanu, B., Brooker, M., and Deardeuff, M. How Amazon Web Services Uses Formal Methods. Communications of the ACM, 58(4):66-73, April 2015. doi:10.1145/2699417.

[20] ISO/IEC. Information technology - Open Systems Interconnection - Conformance testing methodology and framework. ISO/IEC 9646 (seven parts), 1994-1998.

[21] Thomson, M., and Schinazi, D. Maintaining Robust Protocols. RFC 9413, IETF, June 2023. doi:10.17487/RFC9413.

[22] Postel, J. (ed.). DoD Standard Transmission Control Protocol. RFC 761, January 1980, Section 2.10 ("be conservative in what you do, be liberal in what you accept from others"); reiterated in RFC 793, September 1981.

[23] Bray, T. The I-JSON Message Format. RFC 7493, IETF, March 2015. doi:10.17487/RFC7493.

[24] Birkholz, H., Vigano, C., and Bormann, C. Concise Data Definition Language (CDDL). RFC 8610, IETF, June 2019. doi:10.17487/RFC8610.

[25] IAB Tech Lab. ads.txt: Authorized Digital Sellers, v1.0 June 2017, v1.1 2022. iabtechlab.com/ads-txt.

[26] IAB Tech Lab. sellers.json, v1.0, July 2019. iabtechlab.com/sellers-json.

[27] IAB Tech Lab. OpenRTB SupplyChain Object, v1.0, July 2019. github.com/InteractiveAdvertisingBureau/openrtb/blob/master/supplychainobject.md.

[28] Cox, S. Simplifying programmatic: first price auctions for Google Ad Manager. Google Ad Manager blog, March 2019.

[29] Prebid.org. ortbConverter: the Prebid.js OpenRTB conversion library. github.com/prebid/Prebid.js, libraries/ortbConverter.

[30] IAB Tech Lab. Technology Compliance Programs. iabtechlab.com/compliance-programs. Paid audit and certification covering OpenRTB, VAST, and related standards.

[31] Sekowski, A. VAST XML Validation at Bid-Time Scale: Latency Analysis and Integration Patterns for Programmatic Video Pipelines. Preprint, April 2026. doi:10.13140/RG.2.2.11404.27520.

[32] Landis, J. R., and Koch, G. G. The Measurement of Observer Agreement for Categorical Data. Biometrics, 33(1):159-174, 1977. doi:10.2307/2529310.

Specification texts analyzed: OpenRTB 2.6-202606 (June 2026) and OpenRTB 3.0 FINAL (November 2018), as archived from the IAB Tech Lab repositories. Dataset, codebook, and scripts accompany this preprint. Analysis performed July 2026.