

Conformance of open-source Verifactu implementations: a census of the invoicing records published on GitHub

EasyxLab · EasyByte Hub S. Coop. Mad. · Study S1

 Working draft Published 2026-10-02 · Updated 2026-10-03

Abstract

Spain's Verifactu regime (RD 1007/2023, Orden HAC/1177/2024) requires invoicing software to emit hash-chained XML records. We searched GitHub systematically through its code and repository search APIs (1,008 repositories, 29,491 XML files downloaded), classified every file that contained a Verifactu record marker *before* linting it, and audited the corpus with `verifactu-lint` 0.4.0, maintained by the authors. Only 63 third-party repositories publish such XML, and most of it is not a record that claims to be valid: 81 of 191 unique files are templates, most of them copies of the AEAT's own documentation samples, whose hash fields literally read `Huella` or `AAAA`. Of the 35 files that purport to be valid records, **10 (28.6%, 95% CI 16.3–45.1) contain at least one ERROR**; at repository level, **6 of 19 (31.6%, 95% CI 15.4–54.0)** publish at least one such file. The most frequent defect, in 5 of the 19 repositories, is a declared hash that does not match the one computed from the record's fields. All 22 ERROR findings were reviewed one by one by the LLM-based study agent, and the hashes were recomputed with an independent implementation; no false positive was found (0/22). The 22 findings are not independent (10 come from two files of one generator): over the 12 distinct repository–rule situations the upper 95% bound is 24.3%. Public examples almost never contain a chain — 34 of 35 hold a single record — and outside the instrument's own repository nobody publishes a deliberately invalid record.

1. Introduction

Since 29 July 2025, producers of invoicing software ("SIF") must be able to declare that their systems comply with RD 1007/2023 and Orden HAC/1177/2024; users follow in

1. The core of the regime is technical: each invoicing record carries a SHA-256

fingerprint of eight of its fields (five in a cancellation record), including the fingerprint of the previous record (Orden HAC/1177/2024, art. 13.1), so that the sequence of records becomes tamper-evident.

Much of the Verifactu software being written today is small: plugins for open-source ERPs, libraries in PHP, Python, Go, C#, TypeScript. They share examples. A fixture copied from one repository, or from the AEAT's documentation, becomes the reference against which another project tests its generator. If the reference is wrong, the error propagates silently: a schema validator accepts a record whose hash is wrong, because a schema checks form, not arithmetic.

- **RQ1.** Do the Verifactu invoicing records that open-source projects publish on GitHub conform to the regulation and to the AEAT validation rules?
- **RQ2.** Which rules are broken most often?

The contribution is a census, not a sample: we looked at everything those APIs expose (limits in §7). We also report what turned out to be more informative than the conformance rate — *what kind of XML* the ecosystem publishes.

2. Related work and instrument

We are not aware of a published measurement of conformance for Verifactu records, nor for TicketBAI or Portugal's SAF-T hash chain. The AEAT test environment validates submissions, but it requires a certificate, validates one submission at a time and answers only to the submitter. The open-source libraries we found generate records; none audits them.

The instrument is `verifactu-lint` 0.4.0 (Apache-2.0), maintained by the authors. It implements 35 rules: 26 over invoicing records (`RRSIF001–014` and `RRSIF030–049`) and 9 over event records (`RRSIF020–028`). Each cites the article or AEAT error code it mirrors and reports one of three severities: **ERROR** (contradicts the cited norm), **AVISO** (very probably non-conformant), **INCOMPLETO** (cannot be determined from the file). Its hash module is tested against the three reference vectors of the AEAT hash specification. Because the instrument is ours, we excluded its own repository from the corpus and checked every ERROR independently (§5).

3. Method

Details are in `METHOD.md` ; the pipeline is reproducible with `run.sh` .

Collection. Nine code-search queries restricted to XML (`RegistroAlta` , `RegistroAnulacion` , `RegFactuSistemaFacturacion` , `RegistroEvento` , `sum1` , `IDEmisorFactura` , ...; 1,768 hits, 262 repositories) plus three repository searches (`verifactu` , `veri-factu` , `verifactu in:readme` ; 764 repositories), then a recursive walk of every repository's tree. Every XML file up to 1.5 MB was downloaded pinned to a commit SHA (29,491 files); 261 occurrences in 64 repositories (63 third-party) contained a Verifactu marker. No fork entered the corpus.

Classification before linting. Deterministic rules, in priority order: (c) AEAT responses and other non-records; (d) templates or schemas, including files whose key fields hold placeholder values; (e) files with the marker but no record; (b) deliberately invalid negative tests, detected from the path, from XML comments, or from test code that references the file inside a test case asserting an error; (a) everything else, i.e. records that purport to be valid. Occurrences in the instrument's own repository are removed before any of this. The heuristics err towards removing files from (a).

Deduplication. Identical files were linted once; a content takes the most conservative class among its third-party occurrences.

Linting. Each unique file was audited as its own chain, as the CLI does.

Statistics. Proportions at file and repository level with Wilson 95% intervals. The repository is the headline unit: files within a repository share a generator.

Verification. Every ERROR on a class-(a) file was reviewed one by one by the LLM-based study agent against the literal normative text, and every hash finding was recomputed with an independent implementation written from Orden HAC/1177/2024 art. 13 (§9).

4. Results

4.1 What the ecosystem publishes

	OCCURRENCES	UNIQUE FILES
Downloaded XML files	29,491	—
With a Verifactu marker	261	195
Instrument's own repository (removed)	7	4 only there
(c) AEAT responses	12	12
(d) templates, of which placeholder values	123 (91)	81
(e) marker but no record	73	62
(b) deliberately invalid tests	0	0
(a) records that purport to be valid	46	36
... readable and containing ≥ 1 record		35

Most repositories that talk about Verifactu publish no record. Of 1,008 repositories found, 63 third-party ones contain any XML with a Verifactu marker and 19 contain a record that purports to be valid.

The largest family is documentation with placeholders. 91 occurrences carry placeholder values in key fields. 62 use the values of the examples in the AEAT's *Descripción del servicio web*, section 9, which write `<sum1:Huella>HuellaRegistroAnterior</sum1:Huella>` , `<sum1:Huella>Huella</sum1:Huella>` , `<IDEmisorFactura>AAAA</IDEmisorFactura>` and `<NIF>NNNN</NIF>` ; the other 29 carry other dummies (`0123456789ABCDEF...` , `64 × A`). These examples are illustrative and were never meant to validate, but they sit in `tests/` , `fixtures/` and `examples/` directories next to real fixtures. An older variant of a signature example (`ejemploRegistro.xml`), whose previous hash is the literal word `huella` , survives in two repositories (six copies); a corrected variant in three others passes every rule.

No negative fixtures. None of the 254 occurrences outside the instrument's own repository is a deliberately invalid record kept to prove that a validator rejects it. The four occurrences our first heuristics placed in (b) were all positive *golden* files. Projects that test failure paths do it by mutating valid XML in memory.

4.2 Conformance of the records that purport to be valid (RQ1)

The 35 lintable class-(a) files contain 21 alta, 4 anulaci3n and 11 event records. 11 of the 35 files come from a single repository (R15: one-event fixtures).

OUTCOME	FILES	% [95% CI]
At least one ERROR	10	28.6 [16.3–45.1]
At least one AVISO	4	11.4 [4.5–26.0]
At least one INCOMPLETO	12	34.3 [20.8–50.8]
No finding at all	15	42.9 [28.0–59.1]

At repository level, **6 of 19 repositories (31.6%, 95% CI 15.4–54.0) publish at least one valid-looking record with an ERROR**, in every case in a file not found in any older repository of the corpus.

4.3 Which rules (RQ2)

RULE	WHAT IT CHECKS	FILES	REPOS WITH ERROR
RRSIF001	Hash present and recomputable from the record's fields	9	6
... of which: declared ≠ computed		7	5
... of which: no hash in the official field		2	1
RRSIF002	Hash format: 64 hex characters, uppercase	3	1
RRSIF011	System identified by <code>IdSistemaInformatico</code> + <code>NumeroInstalacion</code>	2	1
RRSIF030	<code>TipoFactura</code> present and in list L2	2	1
RRSIF040	Alta record has a breakdown (<code>Desglose</code>)	2	1
RRSIF043	Line tax = base × rate (±10 €)	1	1
RRSIF049	F1/F3/R1–R4 carry a recipient	1	1

AVISO: missing `TipoHuella` (2 files, 1 repo), `ImporteTotal` 5 € off, inside the AEAT margin (1), a stand-alone "end of NO VERI*FACTU operation" event (1). INCOMPLETO: no event summary in single-event files (10 files, 1 repo) and an undeclared chain start (2).

The causes we could reconstruct are mundane: a previous-record hash copied from the AEAT documentation into a record with different fields; a generator that writes lowercase hex and a timestamp without time-zone offset; a cancellation record that uses `alta` element names, so the hashed fields are empty; and one repository (R02) whose "Verifactu" XML follows an invented structure (nested `<Factura>`, `<Huella>` `<Hash>`, ISO dates), which accounts for the RRSIF011/030/040 rows and for the two "no hash" files.

Sensitivity. If the 50 lintable placeholder files were counted, rules that do not depend on the hash would add 7 files with ERRORS (totals, rate–tax consistency, recipients, recargo de equivalencia). Those were not reviewed and are reported only to show that the placeholder files are not otherwise clean.

5. Validity of the instrument

All 22 ERROR findings on class-(a) files were reviewed one by one by the LLM-based study agent (`VERIFICATION.md`). The seven hash mismatches were recomputed with an independent implementation that shares no code with the instrument: 7/7 confirmed. **No ERROR was found to be a false positive (0/22, upper 95% bound 14.9%).** The 22 findings are not independent (10 come from two files of one generator); over the 12 distinct repository–rule situations the upper bound is 24.3% (over the 20 distinct file–rule pairs, 16.1%). The independent implementation was written by the same team: it rules out coding bugs, not a shared misreading of the specification. One AVISO (1/4) was a severity misjudgement: a single-event fixture warned of "end without start" when the start may be in another file.

The review produced six issues for the instrument (`instrument-issues.md`): two coverage gaps (the time-zone offset of `FechaHoraHusoGenRegistro`, pending normative confirmation since the XSD types it as plain `xs:dateTime`; the `dd-mm-yyyy` format of `FechaExpedicionFactura`), two diagnostic problems, the severity issue above, and a usability one.

6. Discussion

The defect that matters is the one a schema cannot see. A wrong hash does not make a record schema-invalid, and the AEAT itself does not reject it: its validation rules say that if the hash does not follow the specification «se devolverá un aviso de error (no generará rechazo)» (*Validaciones y errores* v1.2.2, §3.1.3, item 23). The record enters, flagged, and only the submitter sees the flag. A developer who copies a fixture with an uncomputable hash gets a green build and a wrong reference.

Public XML examples almost never contain a chain, so they cannot serve as a reference for it; whether projects test chaining in code was not measured. 34 of 35 valid-looking files contain a single record, and 11 of them come from one repository. The chaining rules (RRSIF003, RRSIF004) had almost nothing to examine.

Copies of documentation are fixtures by accident. A test that "parses the official example" passes whether or not the project's hash logic is right.

The absence of negative fixtures is a finding. None of the 63 third-party repositories publishes a record that must fail.

7. Limitations

- **Coverage of GitHub search.** Code search indexes default branches, files under 384 KB and, in general, no forks; repository search depends on "verifactu" appearing in name, description or README. Other extensions, branches, releases and private repositories are not seen. The tree walk took at most 200 XML per repository and skipped ERP noise directories (`views/` , `security/` , `i18n/` ...).
 - **Small denominator.** 35 files, 19 repositories: wide intervals, which describe "repositories like these", not a margin of error over all Verifactu software.
 - **Classification.** Heuristic, biased towards excluding files from (a). The review of class (b) found and fixed 4 errors; classes (c), (d)-template and (e) were not reviewed. Errors there could move files in either direction; their effect on the rate is unknown.
 - **Instrument.** Partial coverage (no XSD validation, no NIF, date or offset checks): ERROR rates are lower bounds of non-conformance even when every ERROR is true.
 - **Review.** Findings and classification were reviewed by the LLM-based study agent and an adversarial AI reviewer (§9).
 - **Published examples are not production records.** Nothing here measures anyone's invoicing, only what is published as reference.
 - **Point in time.** One snapshot, 2026-10-02.
-

8. Conclusion

Most of the Verifactu XML that open-source projects publish is documentation copied with placeholders; a fraction is real, and about three in ten real examples break a rule, most often the hash. Public examples almost never contain more than one record, and none is meant to fail. A shared, openly licensed set of multi-record chains, valid and deliberately broken, would do more for the ecosystem than any single correction.

9. Automation and review

LLM-based agents carried out every phase of this study: writing the collection, classification, lint and statistics scripts and running them; designing and revising the classification rules; the one-by-one review of the 22 ERROR findings and of the classification of classes (a), (b) and (d)-placeholder; writing the independent hash implementation; and drafting this paper, the method and the divulgative version. A second LLM-based agent performed an adversarial review of the draft, whose corrections were applied. The operator of EasyByte decided the ethical framing (anonymisation, no contact with maintainers).

Competing interests

The authors' organisation develops verifactu-lint and offers commercial Verifactu services.

Data and ethics

Third-party files are not redistributed; `data/` contains only aggregates and anonymised identifiers. Repositories are published anonymised (R01–R19) and maintainers have not been contacted. No issue was opened.

References

- Real Decreto 1007/2023, de 5 de diciembre. BOE-A-2023-24840. <https://www.boe.es/buscar/act.php?id=BOE-A-2023-24840>
- Orden HAC/1177/2024, de 17 de octubre. BOE-A-2024-22138. <https://www.boe.es/buscar/act.php?id=BOE-A-2024-22138>
- AEAT. *Validaciones y errores*, v1.2.2. https://www.agenciatributaria.es/static_files/AEAT_Desarrolladores/EEDD/IVA/VERI-FACTU/Validaciones_Errores_Veri-Factu.pdf
- AEAT. *Detalle de las especificaciones técnicas para generación de la huella o hash de los registros de facturación*. https://www.agenciatributaria.es/static_files/AEAT_Desarrolladores/EEDD/IVA/VERI-FACTU/Veri-Factu_especificaciones_huella_hash_registros.pdf
- AEAT. *Descripción del servicio web* (examples in section 9). https://www.agenciatributaria.es/static_files/AEAT_Desarrolladores/EEDD/IVA/VERI-FACTU/Veri-Factu_Descripcion_SWeb.pdf
- AEAT. XML schema `SuministroInformacion.xsd`. https://www2.agenciatributaria.gob.es/static_files/common/internet/dep/aplicaciones/es/aeat/tike/cont/ws/SuministroInformacion.xsd
- verifactu-lint 0.4.0. <https://github.com/easybytehub/verifactu-lint>

Cite this study

CITATION

EasyxLab (2026). Conformance of open-source Verifactu implementations. Study S1. EasyByte Hub S. Coop. Mad. <https://github.com/easybytehub/easyxlab/tree/main/studies/s1-verifactu-conformance>

BIBTEX

```
@techreport{easyxlab_s1,
  title      = {Conformance of open-source Verifactu implementations},
  author     = {{EasyxLab}},
  institution = {EasyByte Hub S. Coop. Mad.},
  number     = {S1},
  year       = {2026},
  url        = {https://github.com/easybytehub/easyxlab/tree/main/studies/s1-verifactu-conformance}
}
```