QUBER RESEARCH — STRUCTURED FINANCIAL DATA

# XBRL: A Primer

## The machine-readable layer inside regulatory filings, how it is stored and accessed, and what docling can do with it

Every US public-company filing already holds a machine-readable tag on every reported number. This document explains that tagging standard from zero: the data model, the file formats, who maintains the dictionaries, where the data lives, and how to get it. It then examines docling's six-month-old XBRL backend against a real Apple 10-Q, measured with this project's own pinned version, and closes with possible directions. No prior XBRL knowledge is assumed.

Engineering Reference · Generated 2026-08-20
Source of truth: cited sources, section 13 · QUE-349

# Contents

## Audience and Scope

Written for the engineers on this project's extraction stack. It assumes fluency with PDFs, HTML, and docling's document model, and no knowledge of XBRL. Sources fall into four classes: the XBRL specifications at xbrl.org, SEC rules and staff pages fetched as primary documents, regulator and project sites, and this project's own measurements. Every web fetch was performed on 2026-08-20 by research agents whose findings cite exact URLs; the sources table in section 13 lists them. Claims about docling come from its source code and from conversions executed locally this session, using docling 2.113.0 (this repository's pinned version) and 2.121.0 (the release published 2026-08-20). Claims that rest on weaker evidence are marked in place: a note box, the word *unverified*, or the phrase *our synthesis*.

# 1 The Findings That Matter

Seven findings organize this document. Each names the section holding its support.

**Every number in a public filing is already machine-tagged.** Apple's most recent 10-Q holds 762 tagged numbers and 98 tagged text disclosures — XBRL calls each one a *fact*, the term section 2 defines — inside the same HTML document a human reads, each tag naming the accounting concept, the period, the unit, and the scale (sections 2, 3, 11). The tags are not optional. They have been mandatory for every US public company since 2021 and for EU-listed issuers since 2020 (section 5).

**Each SEC filing ships two machine readings, and the choice between them decides what a parser can recover.** The filing package holds the inline-tagged primary document and, beside it, an SEC-extracted pure-XML instance document with the same facts (section 6.2). Everything measured in section 11 follows from this split.

**Docling — the open-source document-conversion toolkit at the core of this project's parsing — shipped a real XBRL backend in February 2026, and it is already in this repository's installed version.** It wraps Arelle, the reference XBRL processor. It accepts only the pure-XML instance form and raises on inline XBRL — verified by execution, not inference (section 10).

**Measured: docling's XBRL backend cannot convert Apple's current 10-Q at all.** A typed dimension crashes it with an `AttributeError`, reproduced on 2.113.0 and on 2.121.0 released the day of writing. The crash sits in dead code: the block that computes each fact's segment qualifiers and then discards them (section 10.3). With a one-line guard the backend converts the filing in 2.6 seconds into 975 concepts, 3,040 value cells, and 3,999 relationship links, every value at full unscaled magnitude (section 11.1).

**Measured: a tagged filing routed through a generic text pipeline loses exactly what the tags carry — the priced cost of the wrong tool, not a discovery.** Through docling's HTML backend, 719 of Apple's 760 tagged numbers that display text (94.6%; two of the 762 display none) keep their digits, and every tag attribute is stripped: the concept name, the period, the scale that makes "109,417" mean $109.4 billion, the sign that makes a positive-printed number negative. All 41 misses are the em-dash a statement prints for nil; every one of the document's 118 printed nil dashes drops (section 11.2).

**The free SEC APIs silently drop every company-specific tag and every segmented fact.** Custom-extension concepts and any fact qualified by a dimension (business segment,

geography, share class) are absent from the aggregated JSON APIs. The filing package is the only complete source (section 7.3).

**XBRL covers regulated filings, not the documents this project mostly parses.** Quarterly earnings supplements and investor decks — the documents in this project's own test library — have no tags. Where XBRL exists it can serve as per-value ground truth beside the verification this stack already performs against page text; where it does not, extraction remains the only path (section 12).

# 2 What XBRL Is

XBRL — eXtensible Business Reporting Language — is a standard for tagging each number and disclosure in a business report so that software can read the report without parsing its layout. A filer attaches to every reported value a machine-readable record of what the value is, for whom, for when, and in what unit.

The standard began in 1998, when Charles Hoffman, a CPA at a Tacoma accounting firm, started exploring XML for financial reporting and the AICPA funded the work [13]. The base specification, XBRL 2.1, reached Recommendation status on 2003-12-31 and is still the foundation of everything below [1].

## 2.1 The fact

The unit of XBRL data is the **fact**: one reported value plus the metadata that gives it meaning. A fact has four essential parts:

- **Concept** — which line item this is, named by a qualified element from a published dictionary: `us-gaap:RevenueFromContractWithCustomerExcludingAssessedTax`.
- **Context** — the entity reporting (identified by its SEC CIK number) and the period, either an instant (a balance-sheet date) or a duration (a quarter, a year).
- **Unit** — the measure for numeric facts: `iso4217:USD`, `shares`, `USD-per-shares`, or the dimensionless unit `pure`.
- **Value** — the number or text itself, with a `decimals` attribute stating how far the value can be trusted: `decimals="-6"` means rounded to the nearest million. The spec forbids stating both `decimals` and its alternative `precision` on one fact [1].

The same fact at its three depths, taken verbatim from Apple's current 10-Q [47]:

**Exhibit 1 — One fact, three views** (total net sales, quarter ended 2026-06-27)

**1 · AS PRINTED**

| | Three months ended June 27, 2026 |
|---|---|
| **Total net sales** | $ 109,417 |

**2 · AS TAGGED IN THE PAGE**

```
<ix:nonFraction
 name="us-gaap:RevenueFrom
  ContractWithCustomer
  ExcludingAssessedTax"
 contextRef="c-18"
 unitRef="usd" scale="6"
 decimals="-6"
 format="ixt:num-dot-decimal"
 id="f-56">109,417
</ix:nonFraction>
```

**3 · AS DATA**

```
concept
 RevenueFromContractWith
 CustomerExcludingAssessedTax
entity    CIK 0000320193
period    2026-03-29
          to 2026-06-27
unit      iso4217:USD
value     109417000000
decimals -6 (to the million)
```

Column two is the fact as the filer wrote it: the printed "109,417" wrapped in a tag whose attributes name everything column three lists. The machine value is the displayed string times ten to the sixth — the `scale` attribute — rounded to the nearest million.

## 2.2 The taxonomy

The dictionary the concepts come from is called a **taxonomy**. A taxonomy is an XML Schema that declares the elements, plus a set of companion files called **linkbases** that relate them:

| Linkbase | What it records |
|---|---|
| **presentation** | The display ordering and nesting of concepts, as a parent-child tree per statement. |
| **calculation** | Summation relationships: which concepts add up to which totals, each contribution weighted +1 or −1. |
| **definition** | Dimensional structure: which axes apply to which line items (section 2.3). |
| **label** | Human-readable labels per concept, in multiple roles (standard, terse, documentation) and languages. |
| **reference** | Citations from each concept into the accounting literature, such as FASB Codification paragraphs. |

Two real entries from the calculation linkbase Apple filed with the 10-Q — gross profit defined as revenue minus cost of sales, the subtraction expressed as a −1.0 weight (attributes reordered and locator suffixes shortened for the page) [47]:

```
<link:calculationArc order="1" weight="1.0"
   from="us-gaap_GrossProfit"
   to="us-gaap_RevenueFromContractWithCustomer
       ExcludingAssessedTax"
   arcrole="https://xbrl.org/2023/arcrole/summation-item"/>
<link:calculationArc order="2" weight="-1.0"
   from="us-gaap_GrossProfit"
   to="us-gaap_CostOfGoodsAndServicesSold"
   arcrole="https://xbrl.org/2023/arcrole/summation-item"/>
```

A report never stands alone; it points at the taxonomies it uses, and those point at others. The full closure — every schema and linkbase reachable by following those references — is the **DTS**, the discoverable taxonomy set. The instance document that started the discovery is not itself part of the DTS [1]. The practical consequence: an instance separated from its schemas resolves no concepts, so a consumer must supply the extension files and fetch the standard taxonomies before a single fact can be read.

Filers extend the dictionary. When a company reports a line item the standard taxonomy lacks, SEC rule 405 requires it to create a company-specific element — an **extension** — "if and only if an appropriate tag does not exist in the standard list of tags" [14]. Extensions ship with the filing as a small taxonomy of their own (section 6.1). Extensions let every filer say exactly what it reported, and they defeat naive cross-company aggregation, because `aapl:EquitySecuritiesFVNIAccumulatedGrossUnrealizedGainBeforeTax` exists in no other company's filings (section 9).

## 2.3 Dimensions

Dimensions qualify a fact beyond entity and period. Apple's quarterly revenue appears in the filing once consolidated and once per product line: same concept, same period, same unit, different context. The qualifier is a segment block inside the context. The consolidated context (Exhibit 1's `c-18`) has none; the iPhone context adds one, verbatim from the filing [47]:

```
<segment>
  <xbrldi:explicitMember dimension="srt:ProductOrServiceAxis">
    aapl:IPhoneMember
  </xbrldi:explicitMember>
</segment>

<!-- revenue on c-18 (no segment):  109417000000
     revenue on c-57 (this segment):  54252000000 -->
```

`srt:ProductOrServiceAxis` is an **explicit** dimension [2, 3]: its value is a member declared in a taxonomy — here Apple's own extension member for iPhone. A **typed** dimension instead takes a raw value conforming to a schema type, for axes with no finite member list. This filing uses one to bucket remaining contracted revenue by the date recognition starts, and its member is a date, not a name [47]:

```
<xbrldi:typedMember
    dimension="us-gaap:RevenueRemainingPerformanceObligation
               ExpectedTimingOfSatisfactionStartDateAxis">
  <us-gaap:...StartDateAxis.domain>
    2026-06-28
  </us-gaap:...StartDateAxis.domain>
</xbrldi:typedMember>

<!-- the fact this context qualifies: 0.64, decimals="2" -->
```

The member of a typed dimension is a value with no concept behind it — no taxonomy anywhere declares "2026-06-28". Software written as if every dimension had a named member has nothing to look up there.

## 2.4 Calculation checking and its repair

The calculation linkbase lets a processor check that reported components sum to reported totals. The XBRL 2.1 mechanism has two named defects, in the words of its own successor specification: it "can cause erroneous consistency failures on rounded values, and missed consistency failures where duplicate facts are present" [8]. Calculations 1.1, a Recommendation since 2023-02-22, repairs both by comparing intervals instead of points: a value rounded to the nearest million is treated as the range it could have been rounded from, and a calculation is consistent when the computed and reported ranges overlap [8]. *The quoted passages in this subsection were transcribed through a fetch summarizer rather than from a downloaded copy; treat wording, not substance, as approximate.*

# 3 The Three Serializations

The same fact model has three wire formats. Which one a document uses determines which parser can read it.

## 3.1 The classic XML instance

An **instance document** is a pure-XML file: a root `<xbrl>` element holding context declarations, unit declarations, and one element per fact, with no human-readable presentation at all [1]. From 2009 to the inline mandate, SEC filers attached instance documents as exhibits beside their HTML filings. Today the SEC generates one from each inline filing (section 6.2), so the format remains everywhere even though filers no longer author it. The whole format in one excerpt — a context, a unit, and one fact wiring them together, verbatim from the instance in Apple's current filing package [47]:

```
<context id="c-18">
  <entity>
    <identifier scheme="http://www.sec.gov/CIK">
      0000320193
    </identifier>
  </entity>
  <period>
    <startDate>2026-03-29</startDate>
    <endDate>2026-06-27</endDate>
  </period>
</context>

<unit id="usd"><measure>iso4217:USD</measure></unit>

<us-gaap:RevenueFromContractWithCustomerExcludingAssessedTax
   contextRef="c-18" unitRef="usd" decimals="-6"
   id="f-56">109417000000
</us-gaap:...ExcludingAssessedTax>
```

## 3.2 Inline XBRL

Inline XBRL — iXBRL — embeds the tags inside the visible document instead of beside it. The filing is XHTML; each reported value's displayed text is wrapped in an element from the `ix:` namespace that holds the fact metadata as attributes. The specification is

Inline XBRL 1.1, a Recommendation since 2013-11-18 [6]. One real tag from Apple's FY2024 10-K, character for character [26]:

```
<ix:nonFraction unitRef="usd" contextRef="c-13" decimals="-6"
    name="us-gaap:RevenueFromContractWithCustomerExcludin-
gAssessedTax"
    format="ixt:num-dot-decimal" scale="6" id="f-60">
```

Three attributes deserve attention because losing them silently corrupts a number:

- **`scale`** — the displayed text must be multiplied by ten to this power. A filing that prints "in millions" once in a column header prints `scale="6"` on every affected fact.
- **`sign`** — when present it must be `"-"`: the fact's value is the negation of the displayed number. Filings print expenses as positive numbers in parentheses; the attribute records the true sign.
- **`format`** — names a transformation from the Transformation Rules Registry that converts displayed text to a canonical value: `ixt:num-dot-decimal` strips thousands separators, date transformations parse "June 27, 2026". The current registry is Transformation Registry 5, a Recommendation since 2022-02-16 [7].

Two structural elements matter for any text-oriented parser. `ix:hidden` holds facts that must exist in the data but have no displayed text. Apple's current 10-Q has 13 of them, and real filings wrap the hidden block in `style="display:none"`. `ix:continuation` lets one logical text fact span discontiguous stretches of the document. The full element set is fourteen tags [6].

## 3.3 xBRL-JSON and xBRL-CSV

The Open Information Model — OIM — is a syntax-independent definition of the fact model, adopted in 2021; each serialization is now derived from that model by mapping, rather than inherited from the standard's history [3]. Under it, xBRL-JSON and xBRL-CSV are full Recommendations, both dated 2021-10-13 [4, 5]. xBRL-JSON is a flat map of facts, each with a value and a dimensions object [4]:

```
{ "f923": { "value": "1234",
    "dimensions": { "concept": "tax:NumericConcept", "entity":
"cid:123456789",
```

```
      "period": "2015-01-01T00:00:00/2016-01-01T00:00:00",
 "unit": "iso4217:GBP" } } }
```

Arelle converts any XBRL document into this form (section 8). OIM also gives the standard its modern definition of duplicate facts and drops tuples — XBRL 2.1's legacy container for compound facts built from nested child elements; the 2026 US-GAAP taxonomy contains zero tuple declarations, measured directly against the schema file [11].

# 4 Who Maintains the Dictionaries

Four organizations own the moving parts a US filing depends on.

**XBRL International** maintains the specifications — XBRL 2.1, Dimensions, Inline XBRL, OIM, Calculations 1.1. It is a member-supported non-profit; the standards are freely licensed, and it reports more than 600 members and use in 65 countries [12].

**FASB** maintains the US-GAAP dictionary. Its annual release bundles three taxonomies: the GAAP Financial Reporting Taxonomy (GRT), the SEC Reporting Taxonomy (SRT, for elements GAAP filers need but GAAP does not define), and an employee-benefit-plan taxonomy [9, 10]. FASB took over maintenance from XBRL US in 2010 [13]. Deprecated elements survive exactly two annual releases, then are removed [9].

**The SEC** maintains the DEI taxonomy — document-and-entity information: filer name, CIK, form type, the cover-page facts — on its own namespace, `xbrl.sec.gov` [11]. A common error attributes DEI to FASB; the schema header states it was created by SEC staff.

**XBRL US** — the consortium's recognized US member body, a *jurisdiction* in its vocabulary — runs the Data Quality Committee, whose 196 published validation rules are the de facto quality bar for SEC filings (section 9) [45].

The sizes below were measured by downloading the 2026 schema files and counting element declarations directly — counts, not estimates [11]:

| Schema | Namespace | Element declarations |
|---|---|---|
| **us-gaap-2026** | `fasb.org/us-gaap/2026` | 17,182 total — 16,594 line items and abstracts, 330 hypercubes, 255 dimensions; 5,374 of the total are abstract |
| **srt-2026** | `fasb.org/srt/2026` | 620 |
| **dei-2026** | `xbrl.sec.gov/dei/2026` | 209 |

**What the count means.** The count is of element declarations in the core schema and includes abstracts (grouping headers that hold no value) and the axes, members, and hypercubes behind section 2.3's dimensions, not only reportable line items.

# 5 Where XBRL Is Required

XBRL is not one mandate but many, each regulator picking its own scope, taxonomy, and serialization. The US banking regulators moved first, the SEC made it universal for public companies, and the EU made the inline form the law for listed issuers.

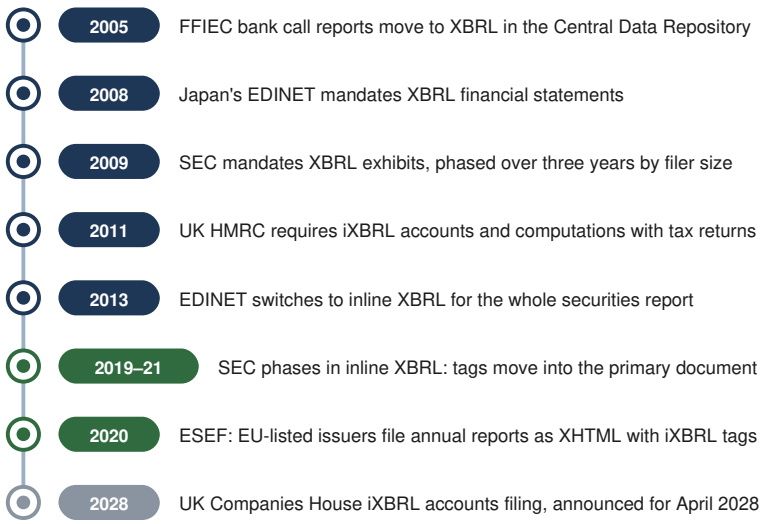


*Figure 1 — The major mandates in sequence. Green marks the inline-XBRL era, in which the tagged document and the human document are the same file. Grey is announced but not yet in force.*

**Two SEC rules set the US timeline.** The 2009 rule (Release 33-9002, effective 2009-04-13) required XBRL exhibits beside the filing, phased by filer size: the roughly 500 largest US-GAAP filers for fiscal periods ending on or after 2009-06-15, remaining large accelerated filers a year later, everyone else and IFRS foreign private issuers by 2011-06-15 [14]. The 2018 rule (Release 33-10514, effective 2018-09-17) replaced the exhibit with inline tagging on the same three-tier schedule: large accelerated filers from fiscal periods ending on or after 2019-06-15, accelerated filers from 2020-06-15, all others from 2021-06-15 [15]. Since mid-2021, every operating company's financial statements and cover page arrive tagged inside the primary document.

**ESEF** — the European Single Electronic Format, Commission Delegated Regulation (EU) 2019/815 — requires issuers on EU regulated markets to prepare annual financial reports entirely in XHTML, with IFRS consolidated statements marked up in iXBRL against a taxonomy extending the IFRS Accounting Taxonomy. It applies to financial years beginning on or after 2020-01-01, with a one-year deferral Member States could elect during COVID; block tagging of the notes applies from 2022 [16, 17, 18]. Primary statements are tagged value by value; notes are tagged as blocks — whole sections wrapped as single text facts.

**The rest, briefly.** UK companies have filed iXBRL accounts and tax computations with HMRC since April 2011 [19]; Companies House has announced mandatory iXBRL accounts from April 2028 [20]. Japan's EDINET has required XBRL since 2008 and the inline form since its 2013 modernization [21]. US bank call reports have been XBRL since the FFIEC's Central Data Repository went live in October 2005 [22, 23]. South Africa's CIPC mandated XBRL in 2018 [48]; India requires it of larger companies and Australia accepts it voluntarily, both per regulator summaries rather than fetched primary pages. The practical consequence for this project: regulated filings worldwide are tagged, and the further a document sits from a regulator — earnings supplements, investor decks — the more certainly it is not.

# 6 How a Filing Is Stored

## 6.1 The archive layout

Every SEC filing lives at a predictable URL: `sec.gov/Archives/edgar/data/<CIK>/<accession>/`, where the CIK is the filer's registration number and the accession number identifies the submission [25]. Each directory silently serves `index.json`, so a package can be enumerated programmatically. The listing below is Apple's FY2024 10-K, fetched and verified this session; the package holds 105 files, of which these are the XBRL-relevant ones [26]:

| File | Bytes | Role |
|---|---|---|
| `aapl-20240928.htm` | 1,503,780 | The primary document: the 10-K itself, inline-tagged |
| `aapl-20240928.xsd` | 58,200 | The extension taxonomy: Apple's custom concepts |
| `aapl-20240928_pre.xml` | 509,625 | Presentation linkbase |
| `aapl-20240928_cal.xml` | 150,261 | Calculation linkbase |
| `aapl-20240928_def.xml` | 232,678 | Definition linkbase |
| `aapl-20240928_lab.xml` | 779,708 | Label linkbase |
| `aapl-20240928_htm.xml` | 1,355,849 | SEC-generated pure-XML instance extracted from the inline document |
| `MetaLinks.json` | 1,047,227 | Renderer metadata: per-tag census, DTS listing, label and reference data |
| `FilingSummary.xml` | 44,798 | Manifest of the rendered report pages |
| `R1.htm` … `R53.htm` | — | Rendered report pages, one per statement or note detail |
| `Financial_Report.xlsx` | 107,340 | Generated Excel workbook of the rendered reports |
| `0000320193-24-000123-xbrl.zip` | 458,550 | The whole XBRL package as one download |

## 6.2 One filing, two machine readings

The package contains the same facts twice, in the two serializations of section 3. The filer authors the inline document; EDGAR extracts the pure-XML instance from it at acceptance time. Everything a downstream parser can do depends on which of the two it reads.

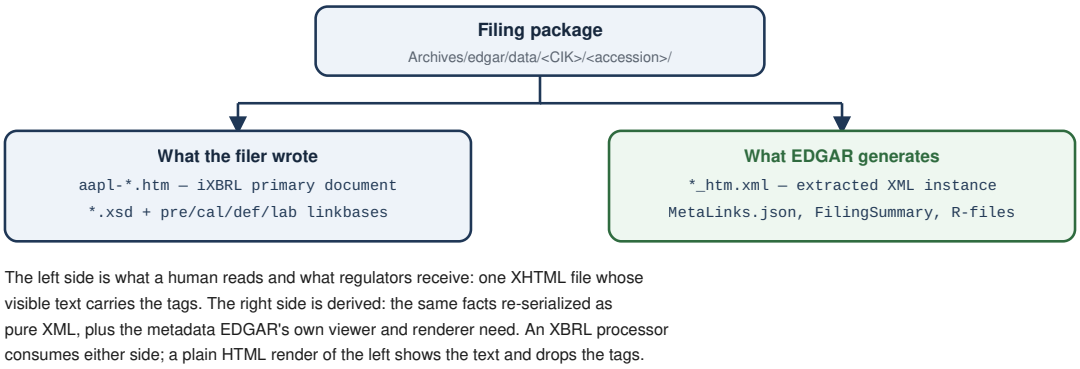


*Figure 2 — The two machine readings inside one filing package.*

## 6.3 The generated metadata

`MetaLinks.json` deserves attention because it answers questions that otherwise need a full XBRL processor. Its per-instance block includes a tag census with the standard/custom split precomputed: for Apple's FY2024 10-K, `keyStandard` 341, `keyCustom` 41 — a 10.7% extension rate for line items — and `memberStandard` 39 against `memberCustom` 31, showing that nearly half the dimension members are custom even when only a tenth of the line items are [26]. It also enumerates the full DTS and, per tag, the labels, documentation strings, and FASB Codification references.

`FilingSummary.xml` maps each rendered R-file to a statement role, with a `MenuCategory` field — Cover, Statements, Notes, Tables, Details — that is the reliable machine-readable discriminator between primary financial statements and everything else [26]. The SEC's inline viewer at `sec.gov/ix?doc=<path>` is a JavaScript shell over the same primary document; scraping it gains nothing over fetching the file [25].

# 7 How the Data Is Accessed

## 7.1 The consumption model

XBRL is consumed by an XBRL processor, not by a text pipeline. The working pattern, the same one the SEC itself runs at filing acceptance:

- **Obtain the filing package** — the instance or inline document together with its extension taxonomy files (section 6.1).
- **Load it into a processor** — Arelle in the open-source world (section 8). The processor assembles the DTS (section 2.2), fetching the referenced standard taxonomies, and resolves every fact against its concept, context, and unit.
- **Validate** — calculation consistency (section 2.4) and rule sets such as the DQC rules (section 9). The SEC's acceptance checks are this exact stack: Arelle plus the SEC's own plugins [31].
- **Query or export the fact table** — by concept, period, unit, and dimension; xBRL-JSON (section 3.3) is the natural export.

Page layout never enters this path. The processor reads contexts and concepts, not pages, and the rendered document is a view over the facts, not the source of them. Everything below — the SEC's APIs and bulk datasets — is this model run by the SEC on every filing, its results served precomputed.

## 7.2 The JSON APIs

The SEC serves aggregated XBRL data as free, keyless JSON on `data.sec.gov`. The endpoints, exactly as documented [24]:

```
https://data.sec.gov/submissions/CIK##########.json
https://data.sec.gov/api/xbrl/companyconcept/CIK##########/us-
gaap/AccountsPayableCurrent.json
https://data.sec.gov/api/xbrl/companyfacts/CIK##########.json
https://data.sec.gov/api/xbrl/frames/us-gaap/AccountsPayable-
Current/USD/CY2019Q1I.json
```

The CIK is zero-padded to ten digits. `companyconcept` returns one company's history for one concept; `companyfacts` returns every standard-taxonomy fact the company has ever filed; `frames` returns one concept across all filers for one calendar period — 3,390

entities for the 2019-Q1 accounts-payable frame verified this session. A single returned fact, verbatim [24]:

```
{"end":"2008-09-27","val":
5520000000,"accn":"0001193125-09-214859",
 "fy":2009,"fp":"FY","form":"10-K","filed":"2009-10-27"}
```

The `accn` field joins each fact back to its filing package. Three operational rules:

- At most 10 requests per second.
- A `User-Agent` header identifying the requester is mandatory. An empty one draws HTTP 403, verified by test.
- The `submissions` endpoint's `isInlineXBRL` flag says whether a filing has the package layout of section 6 [25].

The ticker-to-CIK mapping file `sec.gov/files/company_tickers.json` stores the CIK as an unpadded integer despite its `cik_str` name — pad it before building URLs [25].

## 7.3 What the APIs leave out

**WHAT THE APIS DROP**

The aggregated APIs drop every extension fact and every dimensionally qualified fact. Verified two ways this session: Apple's `companyfacts` response contains only the `dei` and `us-gaap` namespaces — no `aapl` extension concepts, and none of the filing's 19 executive-compensation-disclosure (`ecd`) facts — and requesting an extension concept through `companyconcept` returns HTTP 404 [24]. Segment revenue, geographic splits, share-class detail: anything qualified by an axis does not "apply to the entire filing entity" and is excluded. A FASB-hosted academic workshop states it plainly: "No access to custom extensions or dimensional data" [29].

`frames` adds a second caveat: facts are binned to the calendar period they best align with, so a member's period end can differ from the frame's nominal date by up to 30 days, and only the last-filed value per entity appears — restatement history is invisible [24]. The rule of thumb: the APIs answer standardized, consolidated, latest-value questions in one call; anything involving extensions, segments, footnotes, or as-first-reported values requires the filing package.

## 7.4 Bulk datasets and indexes

Four bulk paths exist, each fit for a different job [25, 27, 28]:

- **Nightly API archives** — the entire `companyfacts` corpus as one zip, rebuilt at about 3 a.m. ET. Same content, same exclusions, no rate limit.
- **Financial Statement Data Sets** — quarterly zips of every filing's numbers as flat tables (`num.txt`, 567.6 MB for 2025-Q4, verified by download). Since a December 2024 reprocessing they cover only the primary financial statements as rendered.
- **Financial Statement and Notes Data Sets** — the richer monthly sibling, adding dimensions (`dim.tsv`), calculation links, and text blocks. This is the only bulk source of dimensional data, closing part of the section 7.3 gap.
- **Index files** — per-quarter listings of every filing. `xbrl.idx`, 477 KB zipped per quarter, is the cheapest way to enumerate every XBRL filing in a period. The full-text search backend at `efts.sec.gov/LATEST/search-index?q=` is an undocumented but public Elasticsearch endpoint over filing text since 2001 — useful, and subject to change without notice.

# 8 The Tooling Landscape

One processor, Arelle, underlies the entire open-source ecosystem; most of what surrounds it is either a wrapper or abandoned. State of each project verified against PyPI and GitHub on 2026-08-20 [30–34].

| Project | State | What it is |
|---|---|---|
| **Arelle** | 2.44.4, released 2026-08-18; weekly cadence | The reference processor: DTS discovery, validation, Dimensions, the Formula rules language, inline XBRL, OIM export. GUI, CLI, Python API, plugin architecture. Stewarded by Workiva since 2022, Apache-2.0. The engine inside docling's backend. |
| **Arelle/EDGAR** | 26.1.3, released 2026-07-10 | The SEC's own validation and rendering plugin set, maintained by SEC staff. Successor to the archived EdgarRenderer. |
| **ixbrl-viewer** | 1.5.2, released 2026-08-13 | Arelle plugin producing a self-contained HTML viewer: click any tagged fact for its concept, period, unit, and dimensions. |
| **edgartools** | 5.51.0, released 2026-08-19; 2,596 stars | The most active EDGAR client: filings, financials, facts as dataframes. Consumes SEC-served data, so it inherits the section 7.3 exclusions where it uses the APIs. |
| **py-xbrl** | 3.0.3, released 2026-03-08 | Standalone instance and iXBRL parser. Maintained at a slower cadence. GPL-3.0 — copyleft matters if linked into a product. |
| **brel** | 0.8.2a1, alpha, 2025-02-26 | ETH Zurich academic library (PyPI name `brel-xbrl`). Its own README warns of bugs. Not production material. |
| **python-xbrl** | Last real commit 2023; last release 2016 | Abandoned. Its GitHub activity timestamps come from dependency bots. |

**Name trap.** The maintained Arelle package on PyPI is `arelle-release`. The bare name `arelle` is a third-party fork last uploaded in 2018. `pip install arelle` installs the dead one [30].

Two absences: no maintained general-purpose JavaScript XBRL processor exists, and the Rust entries are single-digit-star experiments. In this ecosystem star counts are not a maintenance signal; several 300-star Python projects have been dead for a decade [34].

# 9 Data Quality in Practice

Tagged data is not automatically clean data. Three facts calibrate how much to trust a filing's tags.

**Extensions are common, and every published rate uses a different denominator.** The SEC's own trend analysis counts custom line-item tags only and reports rates around 9–11% for US-GAAP filers; XBRL US reports "about 19% of concepts across reports"; the widest academic count, over all tags in 10-Ks, peaks at 25% in 2013 and settles near 21% [41, 43, 44]. These are measurements of different things and must never be quoted against each other. The per-filing census in `MetaLinks.json` (section 6.3) computes the SEC's variant directly — 10.7% for Apple's FY2024 10-K. Two SEC staff observations refine the picture. Small filers drive high line-item extension rates: 96% of high-rate filers are smaller companies, and the filing agent — the vendor that prepares the tagging — is a visible factor [41]. Custom *axis* creation runs the other way, rising with filer size; roughly half of all filers create at least one custom axis [42].

**The recurring error classes have names and automated detectors.** The XBRL US Data Quality Committee publishes 196 approved validation rules; EDGAR runs a subset as warnings at acceptance time — warnings, not rejections [45]. The classes map directly to rules: sign errors (a value tagged positive where the concept's declared debit-or-credit balance implies negative — DQC_0015 and kin), scale errors (a share count tagged in units instead of millions — DQC_0095, DQC_0192), and axis misuse (DQC_0001, the committee's first rule). These are exactly the failure modes of section 3.2's `scale` and `sign` attributes, now as filer mistakes rather than parser losses.

**Commercial vendors resell normalization, which is evidence the raw data needs it.** XBRL US states that most commercial aggregators dedicate significant resources to classifying extensions and consolidating concepts [44]. S&P's documentation concedes that even its "As Presented" product includes "some standardization to align data items across companies" — vendor "as reported" does not reliably mean "as the filer tagged it" [44, 29]. Extensions defeat naive cross-company aggregation; that is the narrow, defensible version of the standard criticism. Whether they harm human comparability is genuinely contested in the academic literature — studies exist on both sides, and the best-known US market-impact study found wider bid-ask spreads around 10-K filings in the mandate's first year, not narrower [49]. *That one-sentence summary of a mixed literature is our synthesis; the wider literature is not re-argued here.*

# 10 Docling's XBRL Capability

## 10.1 What shipped

Docling gained a first-class XBRL backend in v2.75.0, released 2026-02-24: `XBRLDocumentBackend`, registered for a dedicated `XML_XBRL` input format and wrapping Arelle as its processor [35, 36, 38]. v2.79.0 (2026-03-12) added fact metadata and linkbase relationships. The dependency is opt-in: `pip install "docling[xbrl]"` pulls `arelle-release`; the backend class itself ships in the base install — verified importable in this repository's pinned 2.113.0 — and raises a clear ImportError without Arelle [35]. Financial users drive the work: the feature went from requested to shipped in eight days, a FINOS evaluation effort cites the backend as in use, and the one published end-to-end application — a Red Hat graph-RAG pipeline over financial disclosures — runs exactly this backend over instance files [39, 40].

> **ACCEPTS INSTANCE DOCUMENTS ONLY**
>
> The backend accepts XBRL *instance documents* only. Its constructor contains `if model.modelDocument.type != Type.INSTANCE: raise ValueError("Document is not an XBRL instance")`, and Arelle types an inline filing as INLINEXBRL, not INSTANCE. Feeding it an SEC primary document raises `DocumentLoadError` — verified by execution [35]. No issue, pull request, or changelog entry in the docling organization mentions inline XBRL: it is unsupported and, as far as public evidence shows, unplanned. On EDGAR this is workable because every package includes the extracted `*_htm.xml` instance (section 6.2); it does mean docling reads the derived artifact, never the document as filed.

## 10.2 What the backend produces

The output is not pages of text; it is a key-value graph plus recovered narrative. For each fact, the backend emits one key cell holding the concept's local name and up to four value cells — `value:`, `period:`, `currency:`, `decimals:` — into the DoclingDocument's `key_value_items`, each cell a `GraphCell` object. Facts typed as text blocks — whole tagged note sections, the block tagging of section 5 — contain escaped HTML; the backend feeds each through docling's HTML backend and concatenates the results, which is where the texts and tables in the output come from. The document title is assembled from the DEI facts. Presentation and calculation relationships become graph links between cells [35]. Pagination is explicitly unsupported: there are no pages, because an instance document has none.

## 10.3 The gaps, verified in source

- **Dimensions are computed and then discarded.** The backend builds a list of each fact's dimensional qualifiers and never attaches it to anything — the identifier appears only at its own construction. Segment, geography, and share-class breakdowns collapse into indistinguishable duplicate facts [35]. Section 11.1 shows this same dead code crashing on typed dimensions.
- **The provenance hooks exist and go unused.** `GraphCell` has `prov` and `item_ref` fields — exactly the anchors needed to tie a fact to a document location — and the backend populates neither. Measured on the patched Apple run: zero of 4,015 cells [35].
- **The API is declared unstable.** The module docstring warns the key-value pair design "may soon change in a new release of the docling-core library", and the open issue tracking graph integration confirms a data-model refactor is expected [35, 37].

## 10.4 The HTML backend on inline XBRL

Since the XBRL backend rejects the document as filed, an iXBRL filing routed through docling lands in the HTML backend. Its parsing approach determines what survives. It parses with BeautifulSoup's `html.parser` builder, which is not namespace-aware: an `<ix:nonFraction>` element becomes a tag literally named `ix:nonfraction`, matches nothing in the backend's dispatch tables, and falls through to a default branch that recurses into the element's children [35]. The consequences, each verified by execution:

- Text content survives, in place, without fragmenting the sentence around it.
- Every attribute is destroyed: concept name, context, unit, `decimals`, `scale`, `sign`, `format`. The surviving string is the displayed text, not the fact's value — unsigned and unscaled, with no marker that modifiers were dropped.
- Content inside `style="display:none"` is removed permanently. Real filings wrap the `ix:hidden` fact block in exactly that, so hidden facts are not demoted to a recoverable layer — they are gone.

# 11 The Measurement

Subject: Apple's most recent 10-Q, filed 2026-07-31 for the quarter ended 2026-06-27, accession 0000320193-26-000020 — a 1.0 MB inline document with 762 visible numeric facts, 98 text facts, and 13 hidden facts, counted by parsing the `ix:` elements directly [47]. Both of the package's machine readings were converted with docling: the extracted instance through the XBRL backend — the intended path of section 7.1 — under both 2.113.0 and 2.121.0, and the inline primary document through the HTML backend under this repository's pinned 2.113.0, to price what a generic text pipeline loses. Scripts and raw outputs are committed with this document's evidence bundle [46].

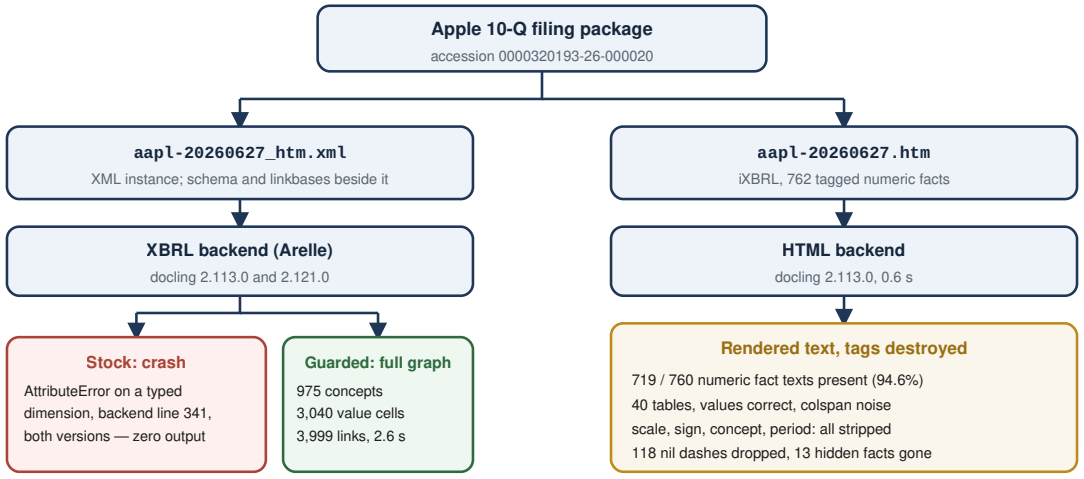


*Figure 3 — One filing, both paths, measured. Green is the intended path's guarded conversion, red the stock crash, gold the priced cost of treating the filing as a generic document.*

## 11.1 Path one: through the XBRL processor

The intended path: the extracted instance and its extension taxonomy into docling's XBRL backend, Arelle underneath. The conversion took three attempts.

- **Instance alone: silently empty.** The file converts "successfully" to a document containing one text item — the filename as a title. The instance references its companion extension schema; without it no concept resolves, and no error says so. The DTS closure rule of section 2.2, encountered in practice.
- **Instance plus its taxonomy files: crash.** With the six package files supplied via the backend's taxonomy option and remote fetch enabled for the FASB and SEC taxonomies, Arelle loads the model — and the backend raises `AttributeError: 'NoneType' object has no attribute 'localName'` at

`xbrl_backend.py:341`. The code assumes every dimension has a member name; section 2.3's typed dimension has none — its member is a date, and there is no concept to look up. Reproduced identically on 2.113.0 and on 2.121.0, released the day of this measurement, and the failing line sits in the dimensions block whose result is already discarded (section 10.3). Docling as shipped cannot convert the current 10-Q of the largest company on the exchange.

- **With a one-line guard: the full graph.** Substituting the typed member's string value where the member name is absent, the conversion completes in 2.6 seconds: 189 narrative texts and 28 tables recovered from text-block facts, a title composed from the DEI facts ("10-Q Apple Inc. 2026-06-27"), and a key-value graph of 975 concept cells, 3,040 value cells, and 3,999 presentation and calculation links.

## 11.2 Path two: the filing as a generic document

This path is the wrong tool used deliberately, to price its cost. A tagged filing that enters a generic document pipeline — docling's HTML backend here, but the point holds for any layout-and-text converter, this project's included — is read as a page, and a page reader keeps what a page shows. The numbers below are a baseline, not a defect report: they measure what such an ingest silently loses when handed a document whose meaning lives in markup the converter was never built to keep.

The conversion is fast and the visible text survives: 0.6 seconds for the 1.0 MB document, 598 text items, 40 tables, clean markdown. Of the 760 visible numeric facts with non-empty text, 719 — 94.6% — appear verbatim. Every one of the 41 misses is the same character: the em-dash a financial statement prints for nil. The document prints 118 of them as `&#8212;` entities — 41 inside tagged numeric facts, the rest in untagged cells — and zero survive to the output, so a nil cell and a truly empty cell become indistinguishable. The 13 hidden facts are absent, as section 10.4 predicts: their `display:none` wrapper is decomposed. The statement tables hold the right digits in the right rows, with structural noise — EDGAR filings are built on layout tables with heavy colspans, and the grids come through with duplicated header cells and spacer columns.

The predictable loss is the meaning: 726 of the 760 facts have a `scale` attribute and 54 have `sign="-"`, and none of it survives. The markdown holds "109,417" with nothing marking it as billions, and expense figures printed positive with nothing marking them negative. A reader model must re-derive both from column headers and context — the inference this project's pipeline performs on untagged PDFs, here performed on a document that carried the answer in attributes the converter dropped. The processor path of section 11.1 never pays this cost; it reads the attributes themselves.

## 11.3 What each path yields

The same revenue fact, side by side, states the tradeoff exactly:

| | Through the XBRL processor (guarded) | As a generic document (HTML backend) |
|---|---|---|
| **The fact as recovered** | `RevenueFromContractWithCustomerExcludingAssessedTax` = `109417000000`, period 2026-03-29 to 2026-06-28, USD, decimals −6 | `109,417` in a table cell |
| **Document form** | A fact graph plus recovered note text; no pages | Reading order, headings, tables, prose — pages of a report |
| **What is lost** | Dimensional qualifiers; any anchor back to the printed page | All fact metadata, nils, hidden facts |
| **Fails on** | Typed dimensions (stock); absent taxonomy (silently empty) | Nothing observed — degrades silently |

**One date nuance.** The backend prints the period end as Arelle's exclusive end datetime — 2026-06-28, one day past the quarter's printed close — while the instance itself records the inclusive end date 2026-06-27.

Neither path alone reads a filing the way this project's stack reads a PDF: one has the meaning without the layout, the other the layout without the meaning. The unused provenance hooks of section 10.3 are precisely the missing join.

# 12 Where This Could Go

This project has spent its effort making untagged documents yield trustworthy values — extraction, reconciliation, per-value grounding. For one class of documents, regulated filings, an authoritative record of every value already exists: maintained by the filer under legal obligation, free to fetch, and readable in seconds. The directions below are ways that record could bear on this stack. They are possibilities, not commitments, and none is scoped here.

**Ground truth for extraction, without hand labels.** The stack's reconciliation verifies extracted values against the page's own text; for a filing, the instance document offers a second, independent verification source — the value the filer asserted, at full precision, with period and unit attached. A join from extracted table cells to instance facts on value, period, and unit would score an entire extraction run against filer-asserted truth, on any filing, with no manual annotation. The same join, run the other direction, would measure what extraction misses.

**The scale and sign the pipeline currently infers.** This stack already infers each table's scale and currency from its caption — the units-capture step. Facts hold the answer as data: `scale="6"`, `sign="-"`, `unitRef="usd"`. On tagged documents those attributes could calibrate or audit the inference; disagreement between a caption-derived unit and a fact's declared unit is exactly the kind of two-readings-disagree flag the pipeline already raises for values.

**Authoritative structure for chunking.** The filing package publishes partitions this stack currently derives: `FilingSummary`'s statement-versus-notes categories, block-tagged note sections as explicit text facts, and `MetaLinks`' per-concept labels and Codification references. Ingesting a filing with those boundaries and labels attached — rather than re-deriving structure from layout — parallels how this stack already prefers a model-reported page structure over layout inference when one exists.

**A contribution upstream.** The typed-dimension crash is a one-line fix in a backend docling's maintainers flag as evolving; dimension capture and provenance population are the two gaps between the backend as shipped and the backend this project would want. Upstream work here is small, verifiable against the measurement in section 11, and aligned with docling's own open graph-integration issue.

**The boundary to respect.** None of this touches the documents that motivated this stack. Earnings supplements, investor decks, and anything else a company publishes outside a regulator's mandate have no tags, and no XBRL mechanism will ever verify them. The

tagged world can calibrate the extractors — a corpus where truth is known — and serve filings-based questions directly. The untagged world remains what the pipeline is for. The two uses share one requirement: reading the filing package, both serializations, well.

# 13 Sources

All sources fetched 2026-08-20 by this session's research agents or the author. Bracketed numbers in the body cite rows here. Rows 46–47 are this session's own artifacts.

| # | Source | URL |
|---|---|---|
| 1 | XBRL 2.1 Specification (Recommendation, errata-corrected) | xbrl.org/Specification/XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2006-12-18.htm |
| 2 | XBRL Dimensions 1.0 work-product index | specifications.xbrl.org/work-product-index-group-dimensions-dimensions.html |
| 3 | Open Information Model 1.0 (REC 2021-10-13) | xbrl.org/Specification/oim/REC-2021-10-13/oim-REC-2021-10-13.html |
| 4 | xBRL-JSON 1.0 (REC 2021-10-13) | xbrl.org/Specification/xbrl-json/REC-2021-10-13/xbrl-json-REC-2021-10-13.html |
| 5 | xBRL-CSV 1.0 (REC 2021-10-13) | xbrl.org/Specification/xbrl-csv/REC-2021-10-13/xbrl-csv-REC-2021-10-13.html |
| 6 | Inline XBRL 1.1, Part 1 (REC 2013-11-18) | xbrl.org/Specification/inlinexbrl-part1/REC-2013-11-18/inlinexbrl-part1-REC-2013-11-18.html |
| 7 | Transformation Rules Registry 5 (REC 2022-02-16) | xbrl.org/Specification/inlineXBRL-transformationRegistry/REC-2022-02-16/inlineXBRL-transformationRegistry-REC-2022-02-16.html |
| 8 | Calculations 1.1 (REC 2023-02-22) | xbrl.org/Specification/calculation-1.1/REC-2023-02-22/calculation-1.1-REC-2023-02-22.html |
| 9 | FASB 2026 GAAP Taxonomy release notes (PDF) | xbrl.fasb.org/resources/annualrelease/2026/GAAP_Financial_Reporting_Taxonomy_Release_Notes.pdf |
| 10 | FASB 2026 SEC Reporting Taxonomy release notes (PDF) | xbrl.fasb.org/resources/annualrelease/2026/SEC_Reporting_Taxonomy_Release_Notes.pdf |
| 11 | | |

| # | Source | URL |
|---|---|---|
| | Taxonomy schemas, counted locally: us-gaap-2026.xsd, srt-2026.xsd, dei-2026.xsd | xbrl.fasb.org/us-gaap/2026/elts/us-gaap-2026.xsd ; xbrl.fasb.org/srt/2026/elts/srt-2026.xsd ; xbrl.sec.gov/dei/2026/dei-2026.xsd |
| 12 | XBRL International: about, introduction | xbrl.org/the-consortium/about/ ; xbrl.org/the-standard/what/an-introduction-to-xbrl/ |
| 13 | XBRL US: about and history | xbrl.us/home/about/ |
| 14 | SEC Release 33-9002, Interactive Data to Improve Financial Reporting (PDF) | sec.gov/files/rules/final/2009/33-9002.pdf |
| 15 | SEC Release 33-10514, Inline XBRL Filing of Tagged Data (PDF) | sec.gov/files/rules/final/2018/33-10514.pdf |
| 16 | ESMA: electronic reporting (ESEF) | esma.europa.eu/issuer-disclosure/electronic-reporting |
| 17 | Commission Delegated Regulation (EU) 2019/815, OJ text | publications.europa.eu/resource/cellar/bc77a477-81d7-11e9-9f05-01aa75ed71a1.0006.03/DOC_1 |
| 18 | Regulation (EU) 2021/337 (ESEF deferral), OJ text | publications.europa.eu/resource/cellar/b077b858-77d4-11eb-9ac9-01aa75ed71a1.0006.03/DOC_1 |
| 19 | HMRC COTAX manual COM60040; XBRL guide for UK businesses | gov.uk/hmrc-internal-manuals/cotax-manual/com60040 |
| 20 | Companies House: accounts filing changes from April 2028 | gov.uk/government/news/companies-house-to-bring-in-changes-to-accounts-filing-from-april-2028 |
| 21 | Japan FSA: EDINET XBRL launch (2008); next-generation EDINET (2013) | fsa.go.jp/en/news/2008/20080317.html ; fsa.go.jp/search/20130821.html |
| 22 | OCC/FFIEC: Central Data Repository launch | occ.gov/news-issuances/news-releases/2005/nr-ia-2005-8.html |

| # | Source | URL |
|---|---|---|
| 23 | FFIEC CDR: taxonomy and bulk-data pages | cdr.ffiec.gov/public/DownloadTaxonomy.aspx |
| 24 | SEC: EDGAR application programming interfaces | sec.gov/search-filings/edgar-application-programming-interfaces |
| 25 | SEC: accessing EDGAR data | sec.gov/search-filings/edgar-search-assistance/accessing-edgar-data |
| 26 | Apple FY2024 10-K filing package, accession 0000320193-24-000123 | sec.gov/Archives/edgar/data/320193/000032019324000123/ |
| 27 | SEC DERA: Financial Statement Data Sets | sec.gov/data-research/sec-markets-data/financial-statement-data-sets |
| 28 | SEC DERA: Financial Statement and Notes Data Sets | sec.gov/data-research/sec-markets-data/financial-statement-notes-data-sets |
| 29 | FASB-hosted academic workshop on pulling XBRL data (PDF) | storage.fasb.org/XBRL__Academic_Research-A_Workshop_on_How_to_Pull_XBRL_Data.pdf |
| 30 | Arelle repository and stewardship announcement | github.com/Arelle/Arelle ; xbrl.org/the-future-of-arelle/ |
| 31 | Arelle/EDGAR plugin repository | github.com/Arelle/EDGAR |
| 32 | edgartools repository | github.com/dgunning/edgartools |
| 33 | py-xbrl repository | github.com/manusimidt/py-xbrl |
| 34 | Tooling survey: PyPI and GitHub metadata for all named projects | pypi.org ; api.github.com (per-project, see research record) |
| 35 | docling XBRL backend and HTML backend source | github.com/docling-project/docling/blob/main/docling/backend/xml/xbrl_backend.py ; github.com/docling-project/docling/blob/main/docling/backend/html_backend.py |
| 36 | | |

| # | Source | URL |
|---|---|---|
| | docling CHANGELOG (v2.75.0, v2.79.0, v2.121.0) | github.com/docling-project/docling/blob/main/CHANGELOG.md |
| 37 | docling issue #3035: XBRL graph integration (open) | github.com/docling-project/docling/issues/3035 |
| 38 | docling supported formats; XBRL conversion example | docling-project.github.io/docling/usage/supported_formats/ ; docling-project.github.io/docling/_generated/examples/xbrl_conversion/ |
| 39 | Red Hat: agentic graph-RAG over financial disclosures; companion repository | developers.redhat.com/articles/2026/07/22/how-we-built-agentic-graphrag-financial-disclosures ; github.com/caldeirav/agentic-graphrag-finance |
| 40 | FINOS ai-evals-framework issue #23 | github.com/finos-labs/ai-evals-framework/issues/23 |
| 41 | SEC staff observations: custom tag rates (2014) | sec.gov/data-research/structured-data/office-data-standards-Innovation-staff-observations-guidance-trends/staff-observations-custom-tag-rates |
| 42 | SEC staff observations: custom axis tags (2016) | sec.gov/about/divisions-offices/division-economic-risk-analysis/office-structured-disclosure-staff/staff-observations-custom-axis-tags |
| 43 | SEC: US-GAAP custom tags trend 2022–2024 (methodology note) | sec.gov/data-research/structured-data/us-gaap-xbrl-custom-tags-trend/us-gaap-xbrl-custom-tags-trend-2022-2024 |
| 44 | XBRL US: why normalize data (2025) | xbrl.us/why-normalize-data/ |
| 45 | XBRL US Data Quality Committee: rules and guidance | xbrl.us/home/priorities/data-quality/rules-guidance/ |
| 46 | This session's measurement scripts and outputs (evidence bundle committed with this document) | docs/evaluations/xbrl-docling-2026-08-20/ |
| 47 | Apple 10-Q filing package, accession | sec.gov/Archives/edgar/data/320193/000032019326000020/ |

| # | Source | URL |
|---|---|---|
| | 0000320193-26-000020, fetched and counted locally | |
| 48 | XBRL International news: CIPC XBRL mandate (2018-07-06) | xbrl.org/news/cipc-mandate-xbrl/ |
| 49 | Blankespoor, Miller and White (2014), "Initial evidence on the market impact of the XBRL mandate", Review of Accounting Studies 19(4) | doi.org/10.1007/s11142-013-9273-4 |