Understanding Tavi Taxonomies

Journal Entry PoC — Purpose, Outcomes and Validation Evidence

XBRL Japan Advisor, Nobuyuki SAMBUICHI

Revised 14 November 2026 | Technical review report

Prepared as an update to the 2 November 2026 distribution handout and the 4 November XBRL Europe discussion material.

Basis: XBRL GL Next Requirements Specification; Taxonomy Framework Parts 1–4; the earlier Understanding Tavi Taxonomies handout; and the 14 November 2026 COR/PLT candidate design, XII Model Viewer observations, negative-test logs and EC2 deployment diagnostics. This report is not a new normative specification or approval of a production binding.

Executive summary

On 14 November 2026, I set out to demonstrate that Tavi can separate reusable COR business semantics from a purpose-specific PLT journal-entry palette, reconstruct Facts from a single Structured Tidy Data CSV, and report targeted data-quality errors. The demonstration moved beyond the 2 November handout's structural-only evidence: I observed 30 Facts and no errors when I tested the established PLT normal case in the XII Model Viewer. Targeted negative tests then confirmed datatype and dimensional validations. A deliberately malformed xs:QName value was not rejected and remains unresolved.

Outcome	Evidence / status
Normal PLT instance	XII Model Viewer: 30 Facts, 0 errors (author-observed run; PASS for this configuration).
COR/PLT imports	A wrong COR URL was identified and corrected; successful normal run subsequently reported.
Negative tests	E01–E05, E08 and E09B: expected failures observed; E09A: 30 Facts, 0 errors.
Open issue	E07 invalid xs:QName value was not reported as erroneous.
Account multiplicity	Separate repeatable COR account Fact Space was designed as a candidate; not validated in XII.
Publication / governance	No evidence of completed Canonical-to-Official publication for the 14 November candidate set.

Purpose and scope of the 14 November work

My investigation had four technical objectives:

(i) separate COR and PLT into importable JSON taxonomy modules;
(ii) retain different parent–child networks for reusable COR semantics and the flattened PLT view;
(iii) validate CSV-derived Facts and negative-test behaviour against the applicable Fact Spaces and datatypes; and
(iv) organise the test artefacts and their sourceMappings for repeatable review. The work was an isolated Journal Entry PoC and did not change the FSM → BSM → Candidate LHM → reviewed LHM → HMD process.

The underlying 18-column Structured Tidy Data sample contains seven data records: one AccountingEntries root row, two EntryHeader rows and four EntryDetail rows. Its expected 30 Facts

comprise 6 root attributes, 4 header attributes and 20 detail attributes. The PoC retains the three occurrence keys dAccountingEntries, dEntryHeader and dEntryDetail for the PLT view.

2. Architecture and revised taxonomy design

2.1 Source semantics and purpose-specific palette

COR, BTX and AAA are the reusable semantic "paint boxes". A reviewed HMD is the purpose-specific palette. The Tavi JSON modules used here are experimental realisations of selected COR meanings; an experimental binding is not an amendment to the reviewed HMD or an approved additional Taxonomy Framework Part.

Module	Primary responsibility	Example resources
COR taxonomy	Shared business Concepts and their semantic/concept hierarchy.	cor:p_entryDetail, cor:p_account, cor:accountNumber, cor:accountDescription; cor:journalEntriesHierarchy
PLT taxonomy	Purpose-specific occurrence Dimensions, Cubes, concept selection and independent presentation.	plt:dAccountingEntries, plt:dEntryHeader, plt:dEntryDetail; plt:h_entryDetail; plt:journalEntriesPresentation
Instance Module	Table Template, Fact Map and Fact Source; maps CSV fields to Concepts and dimensions.	plt:accountingEntriesJournalCSVInstance; external tavi-ins.csv

2.2 Distinct COR and PLT presentation hierarchies

The conceptual experiment deliberately preserves a COR account container while allowing a flatter PLT view. Their parent–child Networks are separate; presentation relationships do not redefine the identity of cor:accountNumber.

```
COR:  cor:p_entryDetail → cor:p_account → cor:accountNumber

PLT:  cor:p_entryDetail → cor:accountNumber
```

The working COR network places accountNumber under p_account. The PLT network places the same accountNumber directly under p_entryDetail. Both were visible as separate networks in the Model Viewer screenshot. This is evidence of different presentation paths, not by itself a rule for transferring values between them.

2.3 Domain Networks, Cubes and account occurrence

Concept Domain Networks specify the Concept members available to a Cube, while presentation Networks organise the displayed hierarchy. The 14 November candidate separates COR EntryDetail and repeatable Account Fact Spaces. COR uses the structural Concept cor:p_account (with no value column) and a proposed Account occurrence Dimension; PLT retains its independent, flattened entry-detail Fact Space without an Account occurrence Dimension.

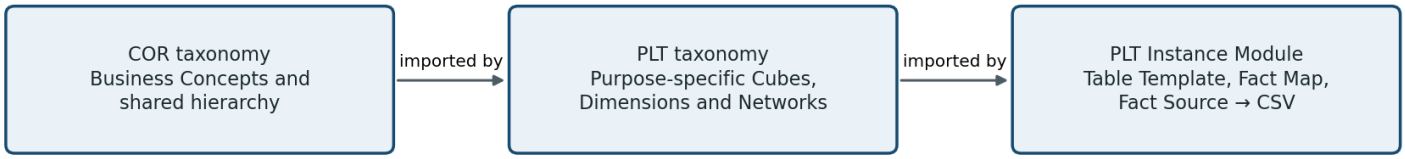
```
COR candidate:  EntryDetail (key) → Account (0..*; separate dAccount key)

PLT view:       EntryDetail (key) → accountCode/accountName on the Detail row
```

The separate COR candidate defines four Cubes, four typed occurrence Dimensions and four Concept Domain Networks, including cor:h_account and cor:dAccount. PLT retains three Cubes, three typed occurrence Dimensions and three Domain Networks, and imports COR Concepts without redefining them. The COR repeatable-Account path remains a candidate and has not been validated with a multi-account normal instance. The earlier 30-Fact PASS applies only to the tested PLT configuration.

The proposal also illustrates a conceptual issue: when multiple Account occurrences exist under one EntryDetail, flattening them to a single PLT accountNumber requires an explicit selection or specialisation rule. Such a rule is not established by a Cube reference alone.

2.3.1 Concrete taxonomy examples: COR and PLT

The previous handout illustrated a single review taxonomy with three Cubes and three Dimensions. The 14 November candidate instead separates two importable Taxonomy Modules: COR holds reusable accounting Concepts and the repeatable Account structure; PLT imports COR and defines the journal-entry palette. The following figures describe the uploaded candidate JSON files, not the older published integer-key and UUIDv7 review taxonomies.



The PLT taxonomy reuses COR Concepts without redefining the Concepts in PLT.

Figure 2A. COR business semantics are imported by PLT; the PLT Instance Module imports PLT and maps the CSV source.

COR (tavi-cor-tax.json). The current candidate contains 21 Concepts, 4 Domain Classes, 4 Dimensions, 4 Domain Networks, 4 Cubes and one parent–child Network. The fourth Dimension, cor:dAccount, distinguishes multiple Account occurrences under the same EntryDetail. The separate cor:h_account Cube uses cor:domain_account, whose two Concept members are cor:accountNumber and cor:accountDescription. The cor:h_entryDetail Cube instead uses cor:domain_entryDetail, which includes cor:p_account as a structural Concept but does not enumerate the two account attributes as flat EntryDetail members. COR's Account Fact Space is a proposal, not a demonstrated 30-Fact result.

PLT (tavi-plt-tax.json). The candidate imports cor:accountingEntriesConceptsPoC. It defines no new business Concepts, but has 3 Domain Classes, 3 Dimensions, 3 Domain Networks, 3 Cubes and its own parent–child presentation Network. The plt:h_entryDetail Cube uses plt:domain_entryDetail, whose members include cor:accountNumber and cor:accountDescription directly, together with EntryDetail description, amount and debit/credit indicator. The PLT CSV records the account fields on the EntryDetail row and has no dAccount column.

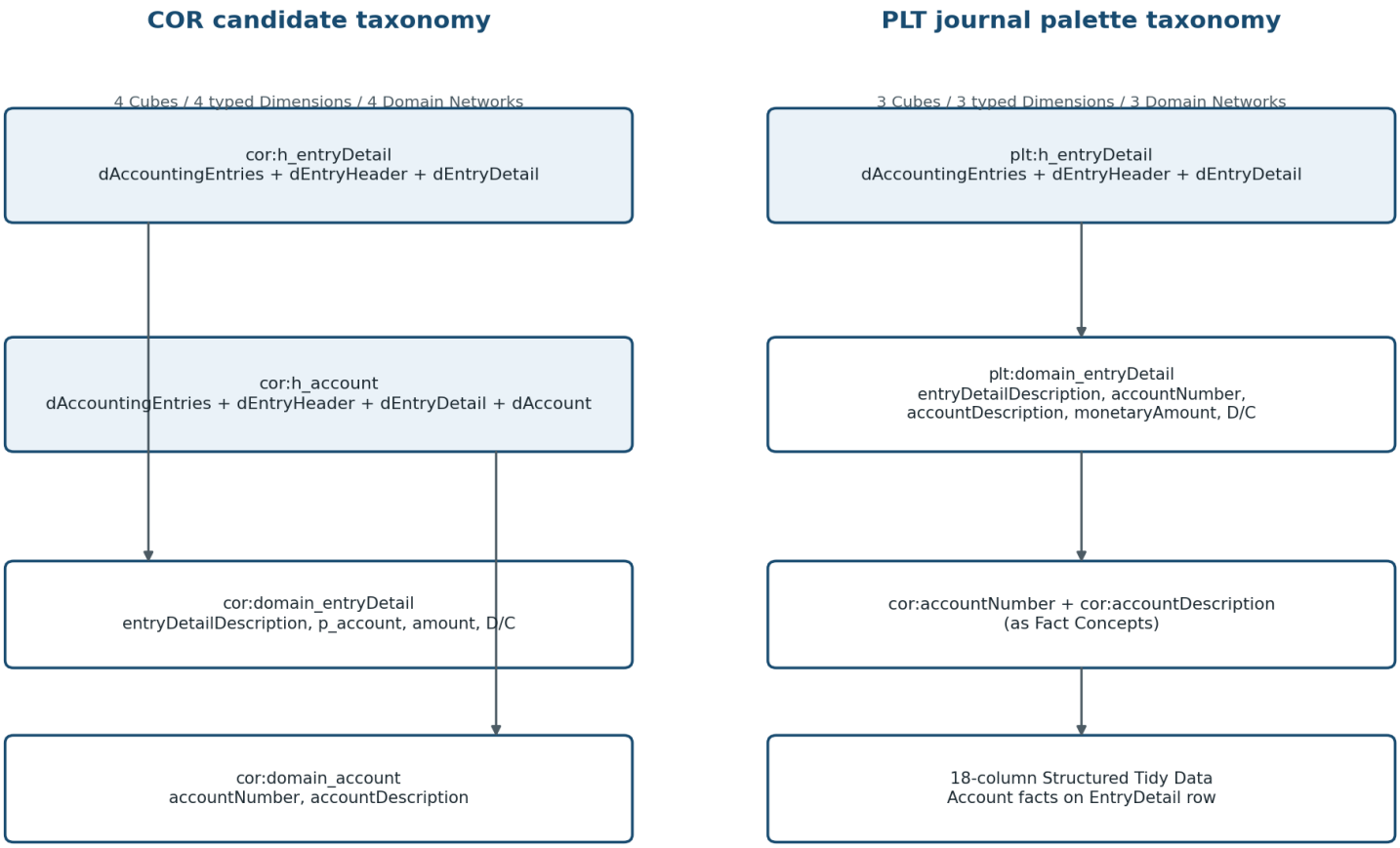


Figure 2B. Separate Cube/Domain Network structures in the COR candidate and the PLT journal palette. The remaining root and header Cubes follow the same three-level Journal pattern in PLT.

Their presentation hierarchies are also intentionally different. In COR the parent–child Network includes cor:p_entryDetail → cor:p_account → cor:accountNumber; in PLT the same cor:accountNumber is attached directly beneath cor:p_entryDetail. In the uploaded COR candidate, cor:accountDescription is attached directly beneath entryDetail in the presentation Network, whereas the separate cor:domain_account lists both account attributes. The figures keep these two Network roles distinct rather than silently revising the candidate.

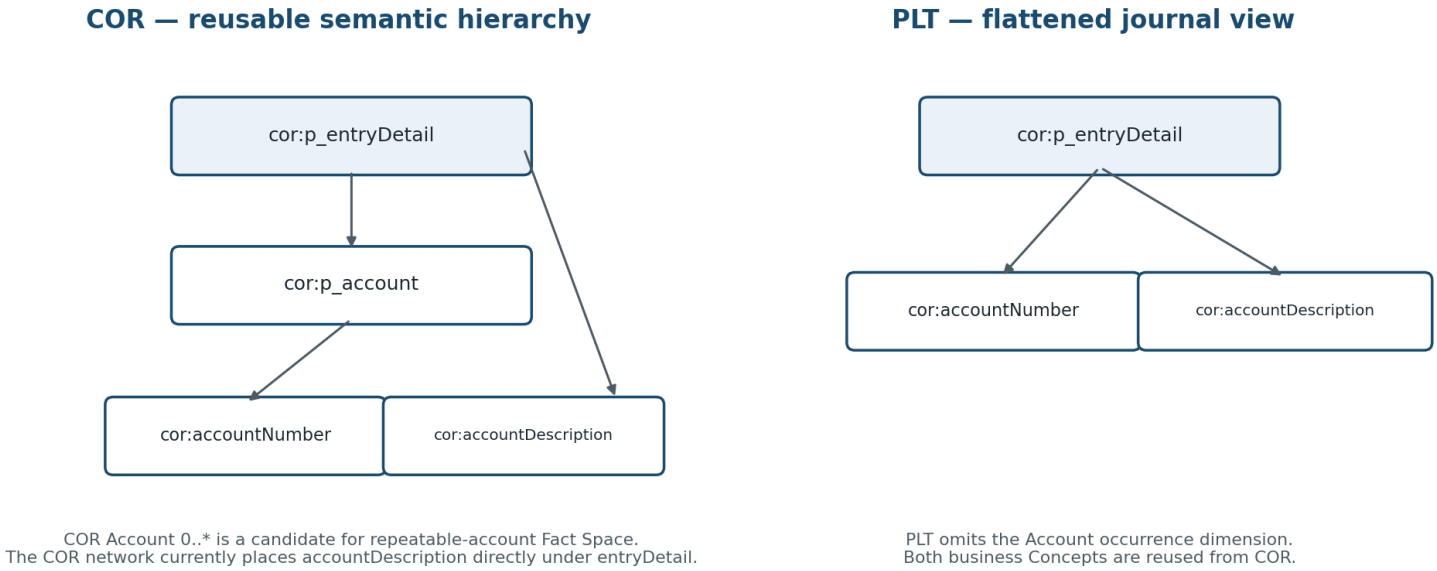


Figure 2C. Different COR and PLT presentation paths using the same COR business Concepts.

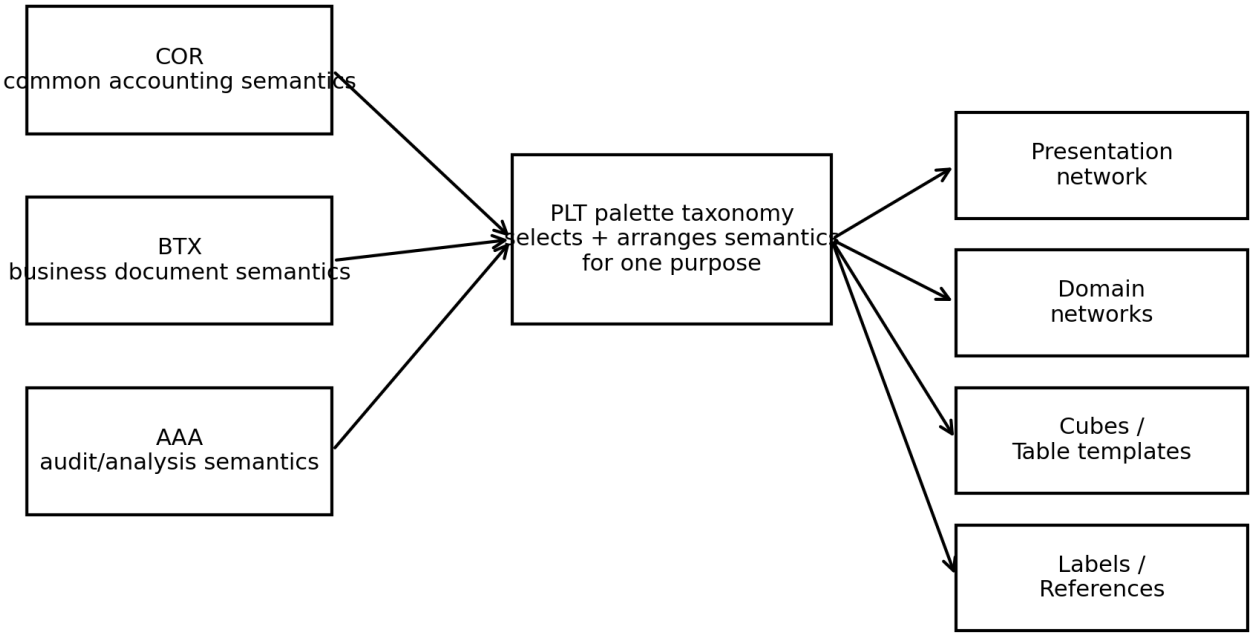
Cube dimensions do not by themselves enforce the existence of a matching parent row. The PLT negative test E08 established that missing required ancestor Dimensions can exclude Detail Facts from all Cube fact spaces; it did not establish referential integrity between distinct Class occurrences.

2.4 Why a palette taxonomy is necessary

A palette taxonomy is needed because a reusable semantic module and a purpose-specific reporting model solve different problems. The COR module should remain a stable pool of reusable accounting semantics, with Concepts and basic relationships that can be reused across many implementations. A palette taxonomy, by contrast, selects a subset of those semantics, arranges them for one reporting purpose, and adds the implementation-facing structures needed by that purpose, such as its Cubes, occurrence Dimensions, table templates, labels and presentation view. In this experiment, the PLT layer allows the same COR semantics to be presented in a journal-entry form suitable for the 18-column Structured Tidy Data sample without rewriting the underlying meaning of the business Concepts.

The practical necessity is strongest where the view differs from the source semantic model. Examples include flattening a repeated COR Account structure into a simpler PLT row view, choosing only the Dimensions needed for a given HMD, and supplying purpose-specific labels and references while keeping the common semantic source independent. Without a palette taxonomy, either the common COR model becomes cluttered with many purpose-dependent arrangements, or each implementation duplicates and forks the common business semantics.

Figure 1. Why a palette taxonomy is needed

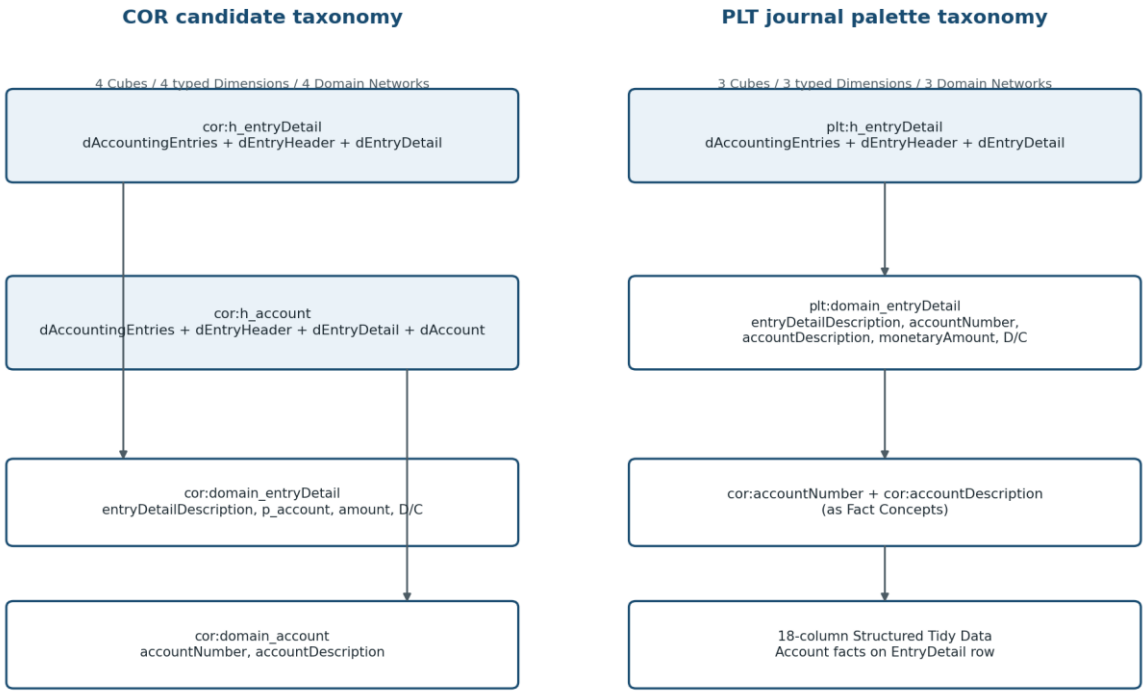


A palette taxonomy reuses stable common semantics but reshapes them for one reporting purpose without rewriting the underlying semantic modules.

2.5 Relationship among Cubes, cubeDimensions and Domain Networks

In Tavi, a Cube specifies a Fact Space through its cubeDimensions. A Concept Dimension uses a named domainNetwork to identify eligible Concepts, while typed occurrence Dimensions use their domainClasses to constrain values. The COR candidate separately defines EntryDetail and Account Fact Spaces so that Account occurrences can repeat; the PLT journal palette has a flatter entry-detail Fact Space and no Account occurrence Dimension. The two modules reuse business Concept identities but do not share the same Cube or Domain Network structure.

A presentation Network is independent of a Cube Concept Domain Network; different presentation paths do not change fact identity. In the tested PLT instance, each Detail Fact carries the required ancestor Dimensions. Missing the dEntryHeader Dimension in E08 caused the five affected Detail Facts to be rejected as outside any Cube fact space; it did not establish the existence of an actual parent Header row.



2.6 Difference from XBRL 2.1 Dimensions

The Tavi/OIM treatment of dimensions differs conceptually from XBRL 2.1 Dimensions, even though both identify the context in which a Fact is valid. In XBRL 2.1, the dimensional model centres on hypercubes, explicit or typed dimensions, domain-member relationships, and the all/notAll logic used by definition linkbases. In Tavi, the structure is expressed more directly in JSON terms: a Cube defines the fact space, cubeDimensions names the dimensions used in that space, and a domainClass constrains typed dimension members by data type. Domain Networks remain distinct from presentation Networks and act as concept-domain declarations rather than as the full 2.1 linkbase mechanism.

For the journal-entry PoC, the three occurrence keys behave like typed dimensions with integer values, but they are easier to inspect because the controlling data type is declared directly in the taxonomy JSON. This is why E01–E03 produced repeated invalidFactDimensionMember errors: the typed member values ABC, A and X did not conform to the integer domain class attached to the corresponding PLT dimensions.

Key contrast points:

- Tavi: Cube = fact space. XBRL 2.1: hypercube and dimensional relationships define the fact space.
- Tavi: cubeDimensions explicitly lists the identifying dimensions. XBRL 2.1: dimensions are linked through definition arcs.
- Tavi: typed member validation is driven by a domainClass data type. XBRL 2.1: typed dimensions depend on typed-domain definitions in the DTS.
- Tavi: presentation, domain and other networks are deliberately separate. XBRL 2.1: the same DTS often expresses multiple linkbase roles and arcroles.

2.7 Tavi Dimensions and Domain Networks: COR and PLT

Figure 3 illustrates the conceptual differences between Tavi and XBRL 2.1 Dimensions. In Tavi, a Cube defines a Fact Space through its cubeDimensions, while a Domain Network identifies the Concepts available within that Fact Space. A Domain Class specifies the permitted values of a typed Dimension, such as xs:integer. Unlike XBRL 2.1 Dimensions, which use hypercubes and dimensional relationships defined through XML and XLink linkbases, Tavi expresses these structures directly in JSON.

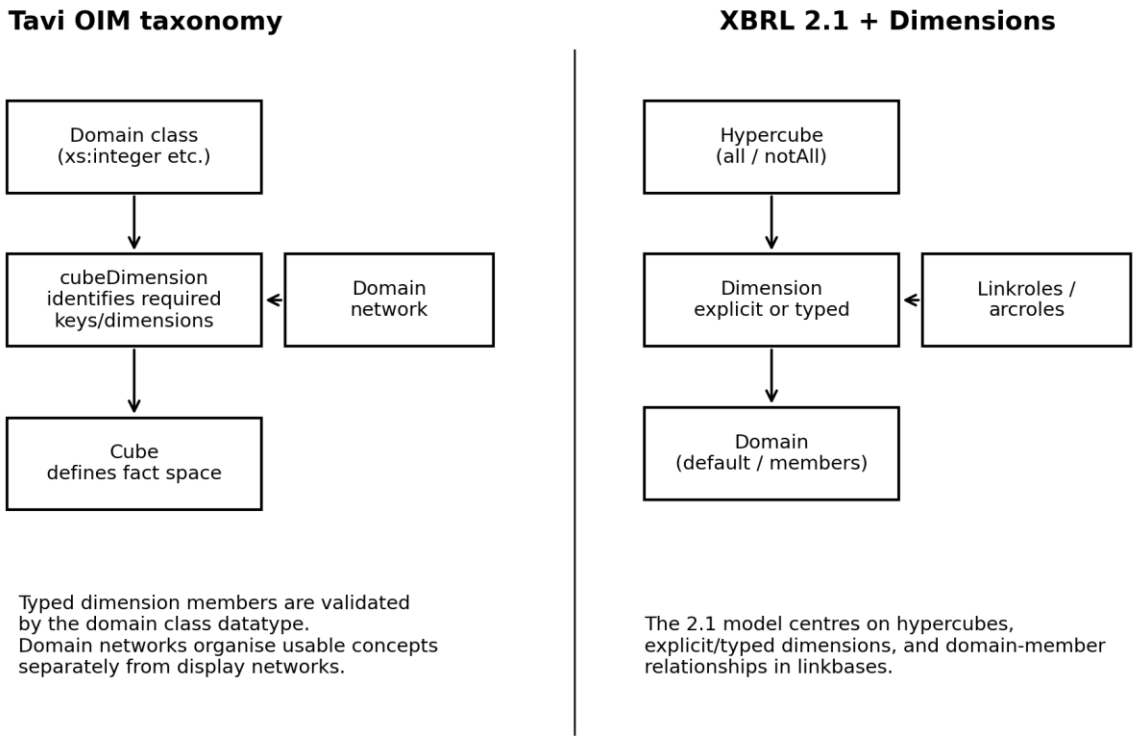


Figure 3. Conceptual difference between Tavi dimensions and XBRL 2.1 dimensions

Figure 3.a shows the four Domain Networks defined in the COR taxonomy: accountingEntries, entryHeader, entryDetail and account. The COR taxonomy preserves the reusable semantic structure in which p_account is a Class under entryDetail. Its attributes, accountNumber and accountDescription, are defined in the separate domain_account network. This separation supports multiple Account occurrences associated with the same EntryDetail, using an Account-specific Dimension and Cube.

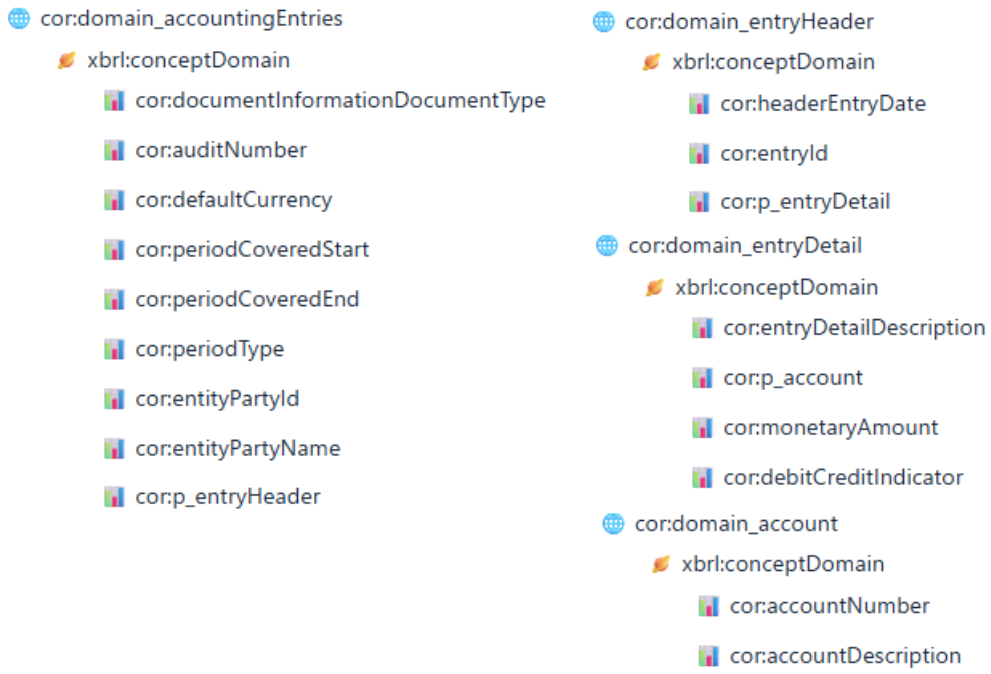


Figure 3.a. COR Domain Networks

Figure 3.b shows the three Domain Networks defined in the PLT taxonomy for the Journal Entry HMD View. The PLT taxonomy imports the COR Concepts but selects and arranges them for a specific reporting purpose. In particular, accountNumber and accountDescription are included directly in the

entryDetail Domain Network, without introducing an Account-specific Dimension. Their values can therefore be represented in the same Structured Tidy Data CSV row as the other EntryDetail attributes.

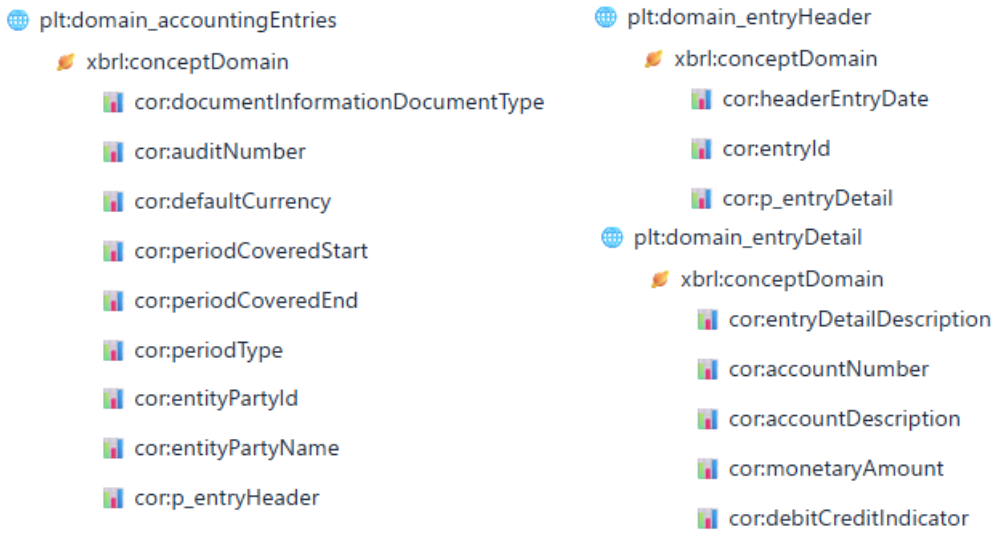


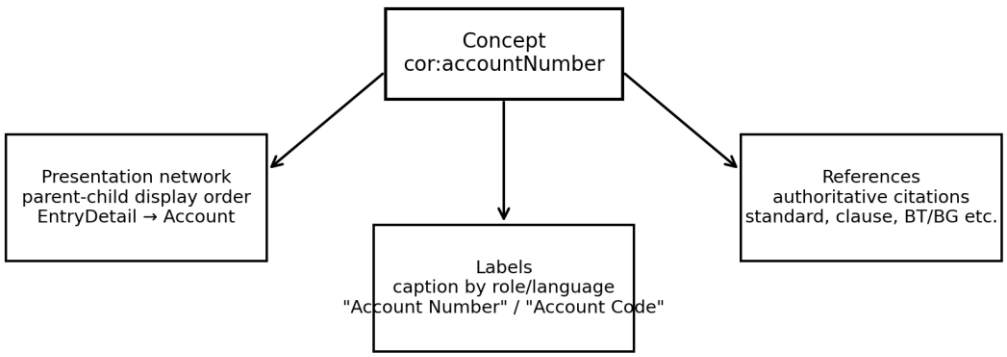
Figure 3.b. PLT Domain Networks

Together, Figures 3.a and 3.b demonstrate why a separate Palette Taxonomy is useful: COR preserves reusable semantics and repeatable structures, while PLT defines the Fact Spaces and dimensional arrangements required by a particular HMD View, without redefining the underlying business Concepts.

2.8 Presentation, label and reference definitions

These three structures should be treated as complementary, not interchangeable. A presentation Network defines how Concepts are displayed in a hierarchy for human understanding; it answers the question “where does this Concept appear in a visible tree?”. A label attaches one or more display captions to a Concept, usually by role and language; it answers “what text should a viewer show for this Concept?”. A reference attaches external or authoritative explanatory metadata, such as a clause, a specification citation, or a business-term correspondence; it answers “what source or interpretation supports this Concept?”.

Figure 4. Presentation, label and reference around a concept



Presentation answers “where is the concept shown in a hierarchy?”
Label answers “what caption is displayed?”
Reference answers “what external authority or meaning note supports it?”

In the COR/PLT split, presentation can vary by palette even when the underlying Concepts are reused unchanged. Labels may also differ when a purpose-specific palette needs different captions or language conventions. References are especially valuable when a palette taxonomy needs to state how a reused Concept relates to a business term, implementation guide, or source specification. The three together allow one semantic Concept to be reused across multiple views while keeping display, naming and authority clearly separated.

Figure 4 illustrates the three complementary ways in which Tavi provides information about a Concept. A Presentation Network defines the hierarchical arrangement of Concepts, while Labels provide human-readable descriptions in different languages or for different purposes. References associate Concepts with authoritative specifications, definitions or business terms. These definitions are independent of the Concept's identity and data type, allowing the same Concept to be reused across multiple taxonomies and presentation views.

Figure 5 compares the Presentation Networks defined in COR and PLT. The COR Network, cor:journalEntriesHierarchy, preserves the underlying semantic hierarchy in which cor:accountNumber is placed beneath the cor:p_account Class. In contrast, the PLT Network, plt:journalEntriesPresentation, places cor:accountNumber and cor:accountDescription directly beneath cor:p_entryDetail.

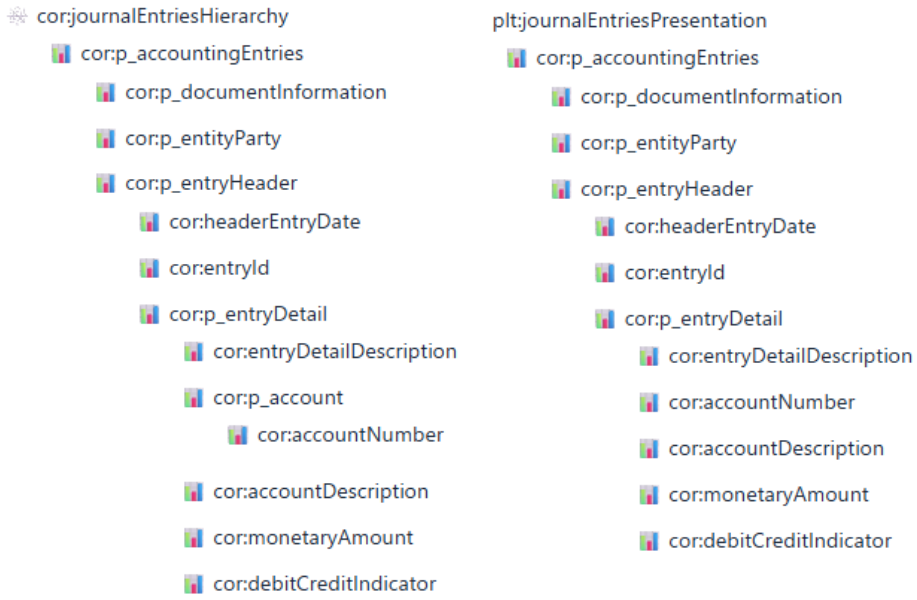


Figure 5. Networks

3. Instance construction and the successful normal test

The PLT Instance Module uses its own plt namespace for the instance model and the three occurrence Dimensions, while retaining cor business Concepts. The Table Template maps the existing CSV columns to those Concepts; monetaryAmount has iso4217:JPY, and the generated Facts carry the reporting period and entity.

Instance / source item	Observed design
Taxonomy chain	tavi-plt-ins.json → tavi-plt-tax.json → tavi-cor-tax.json
CSV	tavi-ins.csv, 18 columns × 7 data records
PLT Dimensions	plt:dAccountingEntries; plt:dEntryHeader; plt:dEntryDetail
Fact-level context	2026-04-01/2027-03-31; ent:ABC123; monetary unit iso4217:JPY
Normal-case result	XII Model Viewer reported no errors and 30 Facts on 14 November 2026

After correcting the COR taxonomy import URL, I confirmed that the XII Model Viewer displayed 30 Facts with no validation errors.

4. Negative-test design and results

Each negative CSV was derived from the same normal 18-column source, with a corresponding Instance Module. The test purpose is to distinguish a datatype violation, a required-dimension/fact-space violation and a physically present but unmapped CSV field. Expected-error detection is reported as PASS for that individual negative test, not as validation of the erroneous instance.

Case	Deliberate change	XII error / result	Assessment
E01	dAccountingEntries = ABC	invalidFactDimensionMember × 6	Expected error observed
E02	dEntryHeader = A	invalidFactDimensionMember × 2	Expected error observed
E03	dEntryDetail = X	invalidFactDimensionMember × 5	Expected error observed
E04	entryDate = 2026-13-40	factValueDataTypeMismatch × 1	Expected error observed
E05	amount = NOT_A_NUMBER	factValueDataTypeMismatch × 1	Expected error observed
E07	defaultCurrency = not:a:QName	No error reported	HOLD; anomaly unresolved
E08	Missing dEntryHeader on a Detail row	noFactSpaceForFact × 5	Expected error observed
E09A	Extra dUnknown CSV column, not mapped	0 errors; 30 Facts	Normal processing observed
E09B	dUnknown mapped as plt:dUnknown	invalidQNameReference × 30; noFactSpaceForFact × 30; FactSource undefinedDimension × 30	Expected error observed

E01, E02 and E03 show that one invalid typed-dimension value can generate multiple error reports because that value is attached to several Facts originating from one CSV record. E08 shows that the five Detail Facts fail the Cube requirements when the required ancestor dimension is absent. It does not prove that an existing Header record with a matching key must exist. E09A/B show that an extra physical CSV column is not the same as referencing an undefined taxonomy Dimension: the former can be ignored by the mapping, whereas the latter is rejected.

4.1 E07: unresolved xs:QName validation

For E07, I used not:a:QName for cor:defaultCurrency, whose Concept datatype was shown as xs:QName. This string has two colons and is not a lexical QName; I observed that the XII Model Viewer raised no error. The uploaded xbrlr:dataTypes file defines xbrlr:QNameSet as a collection over xs:QName, but does not itself specify the built-in QName lexical validator. A JSON structural Schema check is not equivalent to checking a CSV-derived Fact value against xs:QName. It remains unresolved whether this behaviour reflects the Tavi implementation's datatype handling, the applicable specification, or another runtime detail. The proposed alternate cases 123invalid and unknown:JPY were not reported as executed.

5. Deployment and test-case organisation

I deployed the test CSVs to the server and resolved the file access issue. The tests were conducted on an Amazon EC2 instance running Amazon Linux 2023.

I organised the test cases into separate directories, each containing a CSV and its corresponding Instance Module. The shell script passed its dry run; deployment and publication remain unverified.

6. Comparison with the 2 November distribution handout

Topic	2 November handout	14 November evidence / changed interpretation
Native Fact creation	New published Instance Module processing unverified	XII Model Viewer generated 30 Facts, zero errors in established normal PLT case
Taxonomy packaging	Standalone integer and UUIDv7 review variants	Additional experimental separation into COR Concepts and PLT Dimensions/Cubes
Concept presentation	Single Journal hierarchy described	COR and PLT independent hierarchies displayed and differentiated
Validation	Limited local checks; no confirmed native reconstruction	Seven negative conditions examined; six intended failures plus E09B observed; E09A processed normally; E07 unresolved
Publishing	2 November review publication evidenced	14 November experimental candidates not shown as Canonical/Official published

The November 2 report remains the historical evidence for the earlier published integer-key and UUIDv7 review sets. The November 14 viewer results do not retroactively establish full PWD conformance of those older releases, nor do they validate the proposed repeatable Account cube.

7. Evidence classification and outstanding issues

Classification	Statement
VERIFIED FACT — supplied logs	XII errors and counts for E01–E05, E08 and E09B; HTTP 301 incorrect COR URL; EC2 file modes; dry-run plan.
AUTHOR-OBSERVED TEST RESULT	Established PLT normal instance: 30 Facts/0 errors; E09A: 30 Facts/0 errors; E07: no error reported.
PROPOSAL / CANDIDATE	Repeatable COR Account: p_account (0..*), separate Account occurrence Dimension and Cube; PLT flattened EntryDetail row.
UNRESOLVED	E07 QName value validation; existence of parent Header occurrences; selection from multiple COR Accounts to a single PLT accountNumber.
NOT VERIFIED	Revalidation of latest COR repeatable-account taxonomy and complete regression suite; final server publication permissions; --apply completion; formal PWD conformance and GitLab variation suite.
PUBLICATION GOVERNANCE	No authority-based acceptance/CANONICAL/Official Git completion for 14 November artefacts is evidenced.

8. Conclusions and next review scope

The principal achievement was an operational PLT instance that imported separated COR and PLT taxonomy modules and reconstructed the expected 30 Facts from the existing 18-column Structured Tidy Data CSV without reported validation errors. The negative tests demonstrated useful enforcement of integer occurrence keys, date and monetary Fact types, required dimensions, and taxonomy-defined dimensions. E09A clarified that unmapped physical CSV columns need not become Facts or Dimensions.

The remaining technical questions are intentionally not labelled PASS. The malformed QName issue merits an implementation/specification investigation. The repeatable-account COR Fact Space requires

its own representative instance with multiple account rows and a separate validation result. A formal parent-existence constraint is not shown by the current Cube checks. Before any official promotion, the approved authority and standard candidate-to-publication workflow must be followed; this report itself is a technical review candidate.

9. Reference documents and artefacts

- Understanding Tavi Taxonomies, revised distribution handout, 2 November 2026 (uploaded Word and PDF reference).
- Understanding Tavi Taxonomies — Handout Corrected, 2 November 2026 (uploaded PDF reference).
- XBRL GL Next Requirements Specification and Taxonomy Framework Parts 1–4 (project governing semantic basis; detailed current authority not independently reconciled for this report).
- Tavi PWD 2026-09-01: https://www.xbrl.org/Specification/tavi/PWD-2026-09-01/tavi-PWD-2026-09-01.html
- XBRL GL Next public repository: https://github.com/pontsoleil/XBRL_GL_Next
- 14 November 2026 results from my XII Model Viewer tests, screenshots, negative-test error extracts, EC2 file listing and shell dry-run output.

Editorial note

ChatGPT prepared the test taxonomies, Instance Modules and CSV files. I reviewed their definitions and provided instructions for further extensions and refinements.

The initial draft of this report was also generated with the assistance of ChatGPT. I subsequently reviewed, edited and revised the text and figures, and I take responsibility for the final version.

■