Apache Ossie 調査レポート

仕様と実装のあいだで何が起きているか

dobachi

2026-07-21

目次

1 はじめに

⚠ お読みいただく前に

本レポートは個人の調査メモであり、業務上の判断材料として用いることを想定していない。

- 個人による調査であり、査読を経ていない
- **AI と人間の協業によって作成している**
- そのため、事実誤認・出典の取り違え・論理の飛躍が残っている可能性がある。記述はできるかぎり一次情報で裏づけたが、誤りが残っていないという保証はない
- 記述の対象は 2026-07-19 時点のスナップショットであり、数日単位で状況が変わりうる領域を扱っている

導入判断や設計判断に用いる場合は、**必ず本レポートが挙げた一次情報にあたって検証されたい。**本レポートはそのための出発点として、参照した URL とアクセス日、調査範囲と限界（付録 A）、再現手順（付録 B）を明示している。

図 1.1: 両岸から多くの手で橋を架けている途中------中央はまだ接続していないが、作業は進んでいる。本レポートが描く Apache Ossie の現在地（概念図）

Apache Ossie（旧 Open Semantic Interchange, OSI）は、セマンティック層のベンダー中立標準であ

る。2026 年 6 月 22 日に Apache Incubator 入りし、OSI から Apache Ossie に改名した。

本レポートは、この標準が **2026 年 7 月時点で実際にどこまで使えるのか**を、一次情報の読解と動作確認の両面から確かめた記録である。

1.1 調査の動機

Snowflake が主導し 50 社超が参加、Apache 財団に移管------という見出しだけを追うと、セマンティック層の標準化が着々と進んでいるように見える。

一方で、実際に触ろうとすると最初につまずく。どのバージョンを見ればいいのか、公式サンプルはなぜ動かないのか、参加企業のうち何社が実装しているのか。公開情報を追うだけでは、この種の問いに答えが出ない。

そこで、仕様を一次読解したうえで**実際に動かし**、宣伝と実態の差を測ることにした。

1.2 何を調べたか

4 つの軸で調査し、そのうえで動作確認で裏を取った。

- **ASF 移管のガバナンス面** --- リリース・卒業見通し・ベンダー中立性の実態（チャプター 5）
- **仕様の技術的中身** --- 規範・データモデル・式言語・未決の設計論点（チャプター 4）
- **エコシステムの現状** ---「参加」と「実装」の落差、カタログ統合（チャプター 6）
- **動作確認** --- dbt を相手に実際に取り込ませる（チャプター 7）

動作確認は、リリース版 `0.1.1` と開発版 `0.2.0.dev0` の**両方**で行った。両者は揃っているツールチェーンが異なるため、別トラックとして設計している。

1.3 事実の扱い

- 出典は Tier 1（ASF 公式・仕様リポジトリ）／ Tier 2（プロジェクト広報）／ Tier 3（ベンダー発信・業界メディア）に区別し、事実関係は Tier 1 で裏を取った
- 個別の事実には出典の URL とアクセス日を付した。ただし**すべての記述を網羅できてはいない**。出典が明示されていない箇所については、各章の冒頭に示した参照元（仕様の固定コミット、動作確認の結果ファイル）を辿ってほしい
- 確認できなかったことは「未確認」と明記した。特に否定的な結論（「対応していない」）は不在の証明であり、本質的に弱い。その旨を都度断っている
- 動作確認は固定コミット（`07be0176`）に対して行った。上流が変われば結果も変わりうる
- 英文の引用には筆者による和訳を添えた

調査の手順、参照した文献の範囲、否定的結論の限界は 付録 A にまとめた。

1.4 本レポートの構成

チャプター 2 では、本題に入る前に**状況そのものを整理する**。名前・バージョン・ワーキンググループ一覧のいずれもが分裂しており、これを片付けないと以降の議論が読めないためである。

チャプター 3 で Ossie が解こうとしている問題とスコープを、チャプター 4 で仕様の中身を扱う。

チャプター 5 と チャプター 6 が調査の中心で、それぞれ運営体制と実装状況の実態を扱う。

チャプター 7 が動作確認、チャプター 8 が総括である。

付録に、調査の方法と範囲（付録 A）、動作確認の再現手順（付録 B）、情報源の一覧（付録 C）を収めた。本文の主張を検証したい場合は、そこから一次情報を辿れる。

1.5 再現について

動作確認の環境は同じリポジトリのルートにある。必要なのは docker だけ。手順は 付録 B を参照。

2 状況の整理

Apache Ossie を調べようとすると、技術の中身に入る前に**どれを見ればいいのか分からない**という壁に当たる。名前・バージョン・ワーキンググループのいずれもが分裂しているためだ。

本章はその地形を先に片付ける。以降の章はここを前提とする。

2.1 名前が 3 つある

呼称	何を指すか
Open Semantic Interchange (OSI)	2025 年 9 月の発足時の名称。旧サイトや初期の記事はこれ
Apache Ossie (incubating)	2026 年 6 月 22 日の Apache Incubator 入りに伴う現名称
OSI（別物）	Open Source Initiative、Open Systems Interconnection

改名の理由は公式に明言されている。**OSI という略称が他プロジェクトと紛らわしいため**で、仕様そのものに変更はない[1]。

ただし改名後の“Ossie”も完全に一意ではない。Virginia Tech 由来の SDR（Software Defined Radio）フレームワーク OSSIE と同名である。

検索するときは注意が要る。「OSI」で引くと Open Source Initiative の記事が大量に混ざり、2026 年 6 月以前の記事は旧名で書かれている。

2.2 バージョンが 2 つあり、どちらも中途半端

調べるうえで最も混乱しやすいのがここである。

0.1.1 は 2025-12-11 リリースの**唯一のリリース版**。出荷版 dbt が受理するのはこちらである。しかし**公式サンプルもコンバータも存在しない**。

0.2.0.dev0 は `main` ブランチの開発版。仕様本文に明記がある。

> Schema is mutable; **do not depend on this version in production**
>
> （スキーマは変わりうる。このバージョンに本番で依存しないこと。）

[1] Apache Ossie, “Apache Ossie (Incubating): The New Name for Open Semantic Interchange”, アクセス日 2026-07-19

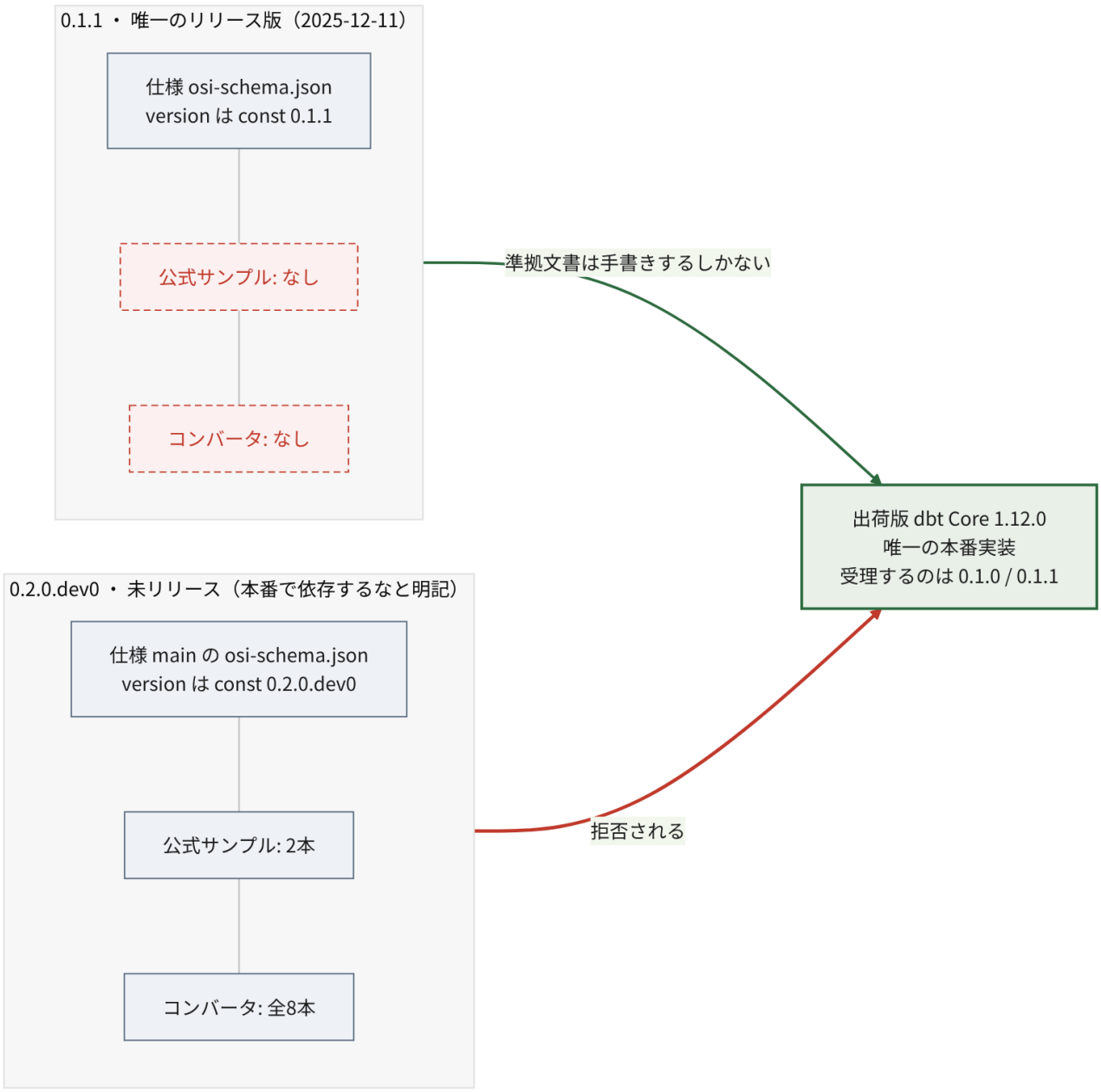

図 2.1: 0.1.1 と 0.2.0.dev0 という 2 つの世界

しかし**公式サンプル 2 本もリポジトリ内のコンバータ 8 本も、すべてこちらを対象**としている。

つまり「動く実装がある方」には手本がなく、「手本がある方」には動く実装がない。

この非対称が実害を生んでいることは チャプター 7 で示す。

2.2.1 ASF としての正式リリースは存在しない

さらに前提として、**Apache Release は 1 本も出ていない**。3 つの独立した一次情報が一致する[2]。

- Clutch ページが “No releases currently available”（利用可能なリリースなし）かつ **“No PGP Signing Keys”**（PGP 署名鍵なし）。署名鍵すら未登録であり、そもそも Apache Release を出せる状態にない
- `dist.apache.org/repos/dist/release/incubator/` に **ossie ディレクトリが存在しない**
- GitHub の Releases は空

存在するタグは **`osi-0.1.1-rc1` ただ 1 つ**。プレフィックスが `osi-` のままである点が重要で、**改名前・ASF 移管前に打たれたタグ**である。ASF のリリースプロセス（PPMC 投票 → IPMC 投票 → dist への配置）を通っておらず、`-rc1` の通りリリース候補止まりで正式リリースですらない。

したがって**「v0.1 系は Apache Release ではない」**。

2.2.2 「v1.0」表記は誤りではない — 後から番号が下げられた

ベンダーブログや技術メディアが仕様を「v1.0」と表記していることがある[3]。現在の仕様に 1.0 というバージョンは存在しないため、これを誇張表現と受け取ってしまうかもしれない。**実際にはそうではない。**

仕様リポジトリの履歴を追うと、**仕様は実際に `Version: 1.0` を名乗っていた**。

期間	`core-spec/spec.md` の宣言
2025-12-12 〜 2026-01-16 以降	**1.0**
2026-03-09 〜 2026-04-20	0.1.1
2026-05-19 以降	0.2.0.dev0

2026 年 1 月 27 日の仕様公開時点で、仕様は 1.0 だった。各社はそれを正確に引用していたにすぎない。

番号を下げたのは以下のコミットである[4]。

- **`2af09b2`（2026-03-09、Khushboo Bhatia）** --- “Version change from 1.0 to 0.1.1”（バージョンを 1.0 から 0.1.1 へ変更）

[2] Apache Incubator, “Clutch: ossie” / “dist.apache.org 配布領域” / “GitHub Releases API”, いずれもアクセス日 2026-07-19

[3] 例: Salesforce, “Ending Semantic Drift” / StartupHub.ai, “Open Semantic Interchange v1.0 Ends Metric Drift”, アクセス日 2026-07-19

[4] 旧リポジトリ open-semantic-interchange/OSI の履歴より。コミット `2af09b2`（2026-03-09）および `3f22030`（2026-05-19）。アクセス日 2026-07-19

- **`3f22030`（2026-05-19、同）** --- “Bump version to 0.2.0.dev0 on main”（main のバージョンを 0.2.0.dev0 へ引き上げ）。コミットメッセージに「release/0.1.1 を切って tag を打ったので、main は破壊的変更を受け入れる開発版とする。main の利用者が リリース済みの 0.1.1 と取り違えないようにするため」とある

> ❗ 読み解き
>
> **1.0 という完成を示唆する番号を、後から 0.1.1 へ引き下げた**という事実は、それ自体が状況を物語る。ASF 移管を前に、成熟度の表明を実態に合わせたと読める（ただし引き下げの理由はコミットメッセージに書かれておらず、**動機は未確認**）。
> したがって「v1.0」と書かれた記事を見たら、それは **2026 年 3 月以前の情報**である可能性が高い。誤りではないが、古い。
> 現在の正しい理解は「**リリース済みは 0.1.1 のみ、`main` は 0.2.0.dev0**」であり、これはプロジェクト自身のドキュメントの記述（“current version: 0.2.0.dev0, latest released: 0.1.1” ＝現行版は 0.2.0.dev0、最新のリリース版は 0.1.1）とも一致する。

2.3 ワーキンググループの一覧が 3 系統ある

公式情報の中に、粒度も名称も異なる 3 つの一覧が並存している。

出典	数	内容
`docs/working_groups.md`	5	Metric Language and Relationships / Composability / Catalog / Ontology / Sync API
公式サイト トップページ	5	Advanced Metrics & Expression Language / Composability / Catalog Integration / Ontology Representation / Model Converters & Developer Tools
`ROADMAP.md` の Current Efforts	3	Metric Semantics & Core Semantic Model / Catalog Integration & Semantic Services / Ontology & Semantic Interoperability
Incubator 入りアナウンス	3	“Three working groups (Metric Language, Catalog and Ontology)”

リポジトリ内のファイルはコミット `07be0176`（2026-07-17）時点、サイト側はいずれもアクセス日 2026-07-19 の内容である。

実態に最も近いのは 1 つ目である。

ワーキンググループ	リード	Slack チャンネル
Metric Language and Relationships	Will Pugh	`#metric-language-working-group`
Composability	Dianne Wood（AtScale）	`#wg-composability`
Catalog	Shubham Bhargav（Atlan）	`#wg-catalog`
Ontology	Kurt Stirewalt（RelationalAI）	`#osi-ontology`
Sync API	Francois Lopitaux（ThoughtSpot）	`#wg-sync-api`

各グループにリード担当者と Slack チャンネルが割り当てられており、**実際に会合が回っているのはこの単位である**と判断した。なお同ファイルのカレンダーリンクはいずれも “link TBD” のままで、会合日程は公開されていない。

`ROADMAP.md` にはこれとは別に「Future Efforts」7 項目があり、こちらは着手前の構想である。

- Dataset Abstraction & Logical Modeling
- Semantic Query Language & Reference Engine
- SQL Dialect, Expressions, and Execution Boundaries
- Dimensions, Hierarchies, and Time Semantics
- AI-Native Semantic Layer
- Governance, Identity, and Validation
- Industry / Domain-Specific Semantic Models

> **i** 起こりやすい取り違え
>
> Current Efforts の 3 件と Future Efforts の 7 件を一列に並べ、「10 個のワーキンググループがある」と紹介している記事がある。筆者自身の過去記事がそうだった[5]。
> ROADMAP の全リビジョンを追った限り、Current Efforts は 3 → 2 → 3 と推移しており、一度も 4 件以上になっていない。したがって**将来構想を現行の作業と取り違えたものと思われる。**
> ただし断定は避けたい。当該記事の執筆時点で ROADMAP がどう見えていたかを完全には再現できておらず、また節の見出しが “Current Efforts / Working Groups” と “Future Efforts” で 視覚的に紛らわしいという事情もある。

この不整合は 2026 年 7 月時点で解消されていない。実際に動いているワークストリームを知りたければ、リード担当者と Slack チャンネルが明記された `docs/working_groups.md` を見るのが確実である。

2.4 まとめ

本章で押さえた前提を再掲する。

- 現名称は **Apache Ossie**。2026 年 6 月以前の情報は **OSI** の名で書かれている
- リリース済みの仕様は **0.1.1 のみ**。`main` が示す `0.2.0.dev0` は 本番依存を禁ずる旨が明記された開発版
- **Apache Release は 1 本も出ていない**。「v1.0」は実在しない

[5] dobachi, “Open Semantic Interchange (OSI) 技術深掘り --- スコープと利用実例”（2026-07-10 初版）。本調査を受けて 2026-07-19 に訂正済み。

- ワーキンググループの一覧は公式内で 3 系統に分かれている。実態は `docs/working_groups.md` の 5 件

3 Ossie とは何か

3.1 解こうとしている問題

「売上」「解約率」「アクティブユーザー」といった指標の定義が、BI ツール・データウェアハウス・分析ツールのそれぞれに散らばって存在する。同じ名前でも定義が違えば、ダッシュボード間で数字が食い違う。

この「セマンティック層の断片化」自体は BI の時代から続く古い問題である。ダッシュボードを人間が見ている限り、数字の食い違いはいずれ気づかれる。「先週の資料と数字が違う」と誰かが言い出し、そこで定義の相違が発覚する。遅くとも意思決定の手前で止まることが多い。

AI エージェントが指標を参照して判断するようになると、この歯止めが外れる。

エージェントは受け取った定義をそのまま正しいものとして扱い、判断を下し、次の処理へ渡す。誤差は検知されないまま下流へ伝播し、その上にさらに判断が積み上がっていく。気づかれないまま業務の連鎖が進んでしまう点が、従来との違いである。

図 3.1: 一点の定義の食い違いが、人手のチェックを経ないまま下流全体へ広がっていく様子（概念図）

共通の定義基盤を持つ価値が質的に上がったというのが、Ossie の出発点である。

3.2 解法

Ossie は、セマンティックモデルを YAML / JSON で宣言的に記述する**ベンダー中立な交換フォーマット**を定める。掲げる原則は「Write Once, Query Anywhere」。一度書いた指標定義を、BI・AI・分析ツールが同じ意味で解釈できるようにする。

構造は素直である。

```
semantic_model[]
├─── datasets[]          論理データセット。source で物理テーブルを指す
│    └─── fields[]        行レベルの属性。式は方言ごとに併記できる
├─── relationships[]     データセット間の外部キー関係
└─── metrics[]           集約式。モデル全体のレベルで定義する
```

詳細は チャプター 4 で扱う。

3.2.1 AI 向けの文脈を仕様に組み込んでいる

特徴的なのは `ai_context` という要素が仕様レベルで用意されている点である。セマンティックモデル・データセット・フィールド・メトリクスのそれぞれに、AI 向けの指示・同義語・サンプル質問を付与できる。

```yaml
ai_context:
  instructions: "受注の売上金額と件数を扱う。status で絞り込める。"
  synonyms: ["売上", "受注"]
  examples: ["先月の売上は？"]
```

指標定義そのものだけでなく、**それをエージェントにどう説明するか**を交換対象に含めている。BI 時代のセマンティック層仕様との違いはここに出ている。

3.3 スコープの線引き — 何を解かないか

Ossie の位置づけを誤らないために、ここを押さえておきたい。

Ossie が与えるのは「フォーマットの相互運用」であって「意味の自動突き合わせ」ではない。

A 社と B 社が別々のセマンティックモデルを持つとする。Ossie は両社のモデルを同じ文法で表現させる。相手のメトリクス定義が機械可読で明示的になり、プロプライエタリなサイロから解放される。

しかし **A 社の churn と B 社の churn を自動で同一視することはしない。** 定義が食い違えば Ossie はその差分を正確に可視化するが、対応づけ（マッピングや統一定義の合意）は当事者のガバナンスに委ねられる。

プロジェクト自身もこれを認識している。`ROADMAP.md` の Ontology & Semantic Interoperability 節の Motivation にこうある[1]。

[1] Apache Ossie, “ROADMAP.md” L96、コミット `07be0176`（2026-07-17）時点。アクセス日 2026-07-19。和訳は筆者による。

> Ossie currently solves structural interoperability --- any tool can read and write semantic models in a common format --- but it does not yet solve conceptual interoperability, where different models may describe the same business concept using different names or structures.
>
> （Ossie は現時点で構造的相互運用を解いている。どのツールも共通フォーマットでセマンティックモデルを読み書きできる。しかし概念的相互運用はまだ解けていない。それは異なるモデルが同じ業務概念を別の名前や構造で記述しうる、という問題である。）

3.3.1 先行例として何が挙がっているか

同じ箇所は、意味をオントロジーで定義する先行例として **Palantir** と **Goldman Sachs Legend** を挙げ、「次元的なセマンティックモデルはその上に自然に載る」と述べている。

補足しておく。

Palantir Foundry の Ontology は、Customer や Order といった業務上の「もの」をオブジェクトとして定義し、それらの関係と操作を記述する層である。どのテーブルのどのカラムに実データがあるかとは切り離して、概念の側を先に定めるアプローチを取る。

Legend は Goldman Sachs が開発し 2020 年 10 月に FINOS へ寄贈したデータプラットフォームで、現在は FINOS のガバナンス下、Apache 2.0 で開発されている[2]。中核の Legend-Pure は UML と OCL に着想を得た関数型のモデリング言語で、業務モデルを先に定義し、そこから実装を導く作りになっている。

「その上に自然に載る」とはどういうことか。二層に分けて考えると分かりやすい。

層	何を定めるか	例
概念層（オントロジー）	業務上の「もの」とその関係	顧客とは何か、注文とは何か、両者はどう繋がるか
次元層（セマンティックモデル）	それをどう測るか	顧客ごとの売上合計、月次の注文件数

「顧客」という概念が先に定まっていれば、「顧客あたり売上」という指標はその概念を参照して定義できる。逆に概念層がなければ、指標は物理テーブルのカラムを直接指すしかなく、**同じ「顧客」を指しているのかどうかを機械的に判定できない**。

Ossie が現時点で持っているのは次元層だけである。Ontology ワーキンググループが目指しているのは、その下に概念層を敷き、異なるモデルが同じ業務概念を指していることを示せるようにすることだ。それが「概念的相互運用」の中身である。

> **「オントロジー」の語には幅がある**
>
> ROADMAP は Palantir と Legend をまとめて「オントロジーを使う」と括っているが、両者の実装は同じではない。Legend は UML / OCL 系のモデリング言語であり、RDF や OWL を用いる狭義のオントロジーとは系譜が異なる。
> Ossie の Ontology ワーキンググループが検討している言語も、n 項関係や role 名を扱う ORM 系

[2] FINOS, "Legend", アクセス日 2026-07-19。Legend-Pure は "an immutable functional language based on the Unified Modeling Language (UML) and inspired by the Object Constraint Language (OCL)"（UML を基礎とし OCL に着想を得た、イミュータブルな関数型言語）と説明される

の設計で、こちらもまた別系統である（セクション 4.7.5）。

概念的相互運用に向けた取り組みは Ontology ワーキンググループが担うが、これは現時点でロードマップ上の目標であり、完成した機能ではない。

> **過大評価を避けるための線引き**
>
> 「Ossie を採用すれば部門間・企業間で指標の意味が揃う」というのは誤りである。揃うのは**記述の文法**であって**意味**ではない。
> 意味を揃えるのは依然として人間の合意形成の仕事であり、Ossie はその合意を機械可読な形で書き留める場所を提供するにすぎない。
> ただしこれは小さな価値ではない。「何が食い違っているか」を正確に指し示せること自体が、合意形成の前提になるからである。

3.4 業界別の共通語彙という方向

この線引きを踏まえたうえで動き始めているのが、「業界単位で Ossie の上に共通ドメイン語彙を作る」というパターンである。

Financial Services Semantic Working Group が 2026-06-04 に初会合を開いている[3]。

> The Financial Services Semantic Working Group has held its first formal meeting, bringing together practitioners from across banking, insurance, asset management, and market infrastructure to solve the industry's semantic alignment challenge.
>
> （金融サービス・セマンティック・ワーキンググループが初の正式会合を開催し、銀行・保険・資産運用・市場インフラの各領域から実務者を集めて、業界の意味整合という課題に取り組む。）

> **この WG も一覧に載っていない**
>
> チャプター 2 で見たワーキンググループ一覧の 3 系統（`docs/working_groups.md` の 5 件、公式サイトの 5 件、ROADMAP の 3 件）の **いずれにも Financial Services Semantic Working Group は含まれていない**。
> WG の全体像を示す単一の情報源は、現時点で存在しないと考えたほうがよい。

異なる業界どうしをつなぐなら、双方が参照する共通リファレンス・オントロジー（金融の FIBO など）へのマッピングが鍵になる。ただし **FIBO との関係について、プロジェクトは公式に答えていない。** Issue と Discussion が立っているが、**いずれもコメントが付いていない**[4]。

なお「Net Asset Value のような業界共通の定義を揃える」という説明を第三者の解説記事で見かけることがあるが、**具体的にどの指標を対象とするかを示した一次情報は確認できなかった**。

[3] Open Semantic Interchange, "Updates" の 2026-06-04 のエントリ。アクセス日 2026-07-19。和訳は筆者による
[4] apache/ossie Issue #220（Documentation request: clarify Ossie's relationship with FIBO and financial-services ontology layering、コメント 0 件）および Discussion #219（同趣旨、コメント 0 件）。アクセス日 2026-07-19。関連する未決論点はセクション 4.8

4 仕様の詳細

本章は仕様リポジトリの一次読解にもとづく。

参照はすべて **固定コミット `07be0176`**（2026-07-17）に対するもので、比較対象はタグ `osi-0.1.1-rc1`。以下、`spec.md` の行番号はこのコミット時点のものである。アクセス日はいずれも 2026-07-19。

4.1 どれを規範として読むか

仕様リポジトリの `core-spec/` には複数のファイルがある。どれを軸に読むかを最初に決めておく必要がある。

ファイル	位置づけ
`core-spec/osi-schema.json`（330 行）	JSON Schema draft 2020-12。機械可読
`core-spec/spec.md`（594 行）	散文の仕様書。表形式の説明
`core-spec/spec.yaml`	YAML 形式の仕様スケッチ。第 3 の表現（後述）
`core-spec/expression_language.md`（35KB）	2026-07-15 追加。後述
`validation/validate.py`（290 行）	参照実装のバリデータ。上記スキーマを読む

README は 3 ファイルを並記するだけで、**どれが規範かを明示していない**。ただし参照バリデータ `validate.py` が構造検証に使うのは `osi-schema.json` であり、散文と JSON Schema が食い違う箇所では、機械的に「通る／通らない」を決めるのはスキーマのほうである。そこで本レポートは、**実装が実際に従う `osi-schema.json` を軸に読む**。これは公式な優先順位ではなく、検証の再現性を担保するための筆者の方針である。散文とスキーマが食い違う箇所は後述する。

同じ仕様が 3 つの形式で表現されている点には注意が要る。

- `osi-schema.json` --- 機械可読。バリデータが読む
- `spec.md` --- 散文と表。人が読む
- `spec.yaml` --- YAML のスケッチ。人が読む

`spec.yaml` は JSON Schema ではなく、**構造を YAML で書き下した見取り図**である。各キーに型と説明をコメントで添えた形式で、機械検証には使われない。

> ⚠ `spec.yaml` の書式は誤読を招きうる
>
> `spec.yaml` はトップレベルに `dialects:` と `vendor_name:` を並べている。これらは「# Enumerations」というコメントの下に置かれた**列挙値の定義**であって、文書に書くべきフィールドではない。
> しかし見た目は文書のひな型と区別がつかない。実際、`dialects` をルート直下に出力してしまう実装が存在する（セクション 7.3）。
> 一方 `osi-schema.json` のルートは `version` と `semantic_model` しか許しておらず、バリデータもこれ

> で検証する。書式の見取り図（spec.yaml）と機械検証の対象（osi-schema.json）は別物である。

参照バリデータは JSON Schema による構造検証に加え、名前の一意性、参照の整合、そして **sqlglot による SQL 式の構文検証**を行う。ただし MDX / TABLEAU / MAQL は sqlglot が解釈できないため検証をスキップする。

4.2 データモデル

4.2.1 全体像

構造は 5 つの要素からなる。まず関係を掴んでおくと、以降の表が読みやすい。

datasets が中心にある。 これは物理テーブルそのものではなく、「受注」「顧客」といった**業務上の実体**を表す論理的な単位である。source で物理テーブルを指すが、指すだけで、テーブルの構造をコピーはしない。

その下に fields が並ぶ。**行レベルの属性**、つまり 1 行を見れば値が決まるものだけを置く。「受注日」「ステータス」「金額」がこれにあたる。集約は書けない。

metrics は datasets の中ではなく、**モデル全体のレベル**に置かれる。「売上合計」「顧客あたり平均受注額」といった集約式で、複数のデータセットにまたがってよい。

relationships がデータセット同士を外部キーで結ぶ。

4.2.2 なぜ metrics だけ外側にあるのか

この配置が Ossie の設計上の特徴である。

Cube・MetricFlow・Malloy・Looker といった既存のセマンティック層は、いずれも**データセットの内側に measure（集約対象）を置く**。「orders テーブルの amount カラムを合計する」という形で、集約の定義がテーブルに紐づく。

Ossie はそうしない。フィールドはあくまで素材で、それをどう集約するかは上位の metrics が決める。

この違いは実装との摩擦を生む。 消費側が「measure への参照」を前提としている場合、OSI のメトリクスをどこに対応づければよいか決まらない。実際に情報が落ちることは セクション 7.4 で観測している。設計論争としても未決である（セクション 4.8）。

4.2.3 各要素の定義

4.2.3.1 Semantic Model

（spec.md L62-）

トップレベルは semantic_model（配列）。

フィールド	必須	備考
name	○	
datasets	○	

フィールド	必須	備考
description / ai_context / relationships / metrics / custom_extensions		

4.2.3.2 Datasets

(spec.md L96-)

フィールド	必須	備考
name	○	
source	○	物理テーブル参照 (database.schema.table)またはクエリ
primary_key		単一・複合いずれも配列で表現
unique_keys		配列の配列
fields		

source が物理層との唯一の接合点である。本レポートが参照する固定コミット(07be0176)時点では、仕様は形式を「database.schema.table 形式または query」としか定めておらず、**識別子のクォートの扱いを規定していない**。これが後の動作確認で問題を生む(セクション 7.3)。

なおスナップショット後の 2026-07-20、Snowflake のクォート付き識別子を扱う変更 (“quoted dot-separated identifier in snowflake”, #233)が main にマージされている[1]。以下の記述はあくまで固定コミット時点のものである。

4.2.3.3 Fields

(spec.md L203-)

フィールド	必須	備考
name	○	
expression	○	方言対応のオブジェクト
dimension		**is_time のみ**
label / description / ai_context / custom_extensions		

重要な点が 2 つある。

フィールドに measure(メジャー)の概念はない。スキーマに measure キーは存在せず、additionalProperties: false で追加も禁じられている。フィールドはあくまで「グルーピング・フィルタ・メトリクス式で使う行レベル属性」であり、集約は metrics 側の責務である。

フィールド式は集約を含んではならない。仕様に明記がある(spec.md L234)。

[1] apache/ossie PR #233 (2026-07-20 マージ)。本レポートの動作確認は 2026-07-17 の固定コミットに対して行っており、この変更は反映していない。アクセス日 2026-07-21

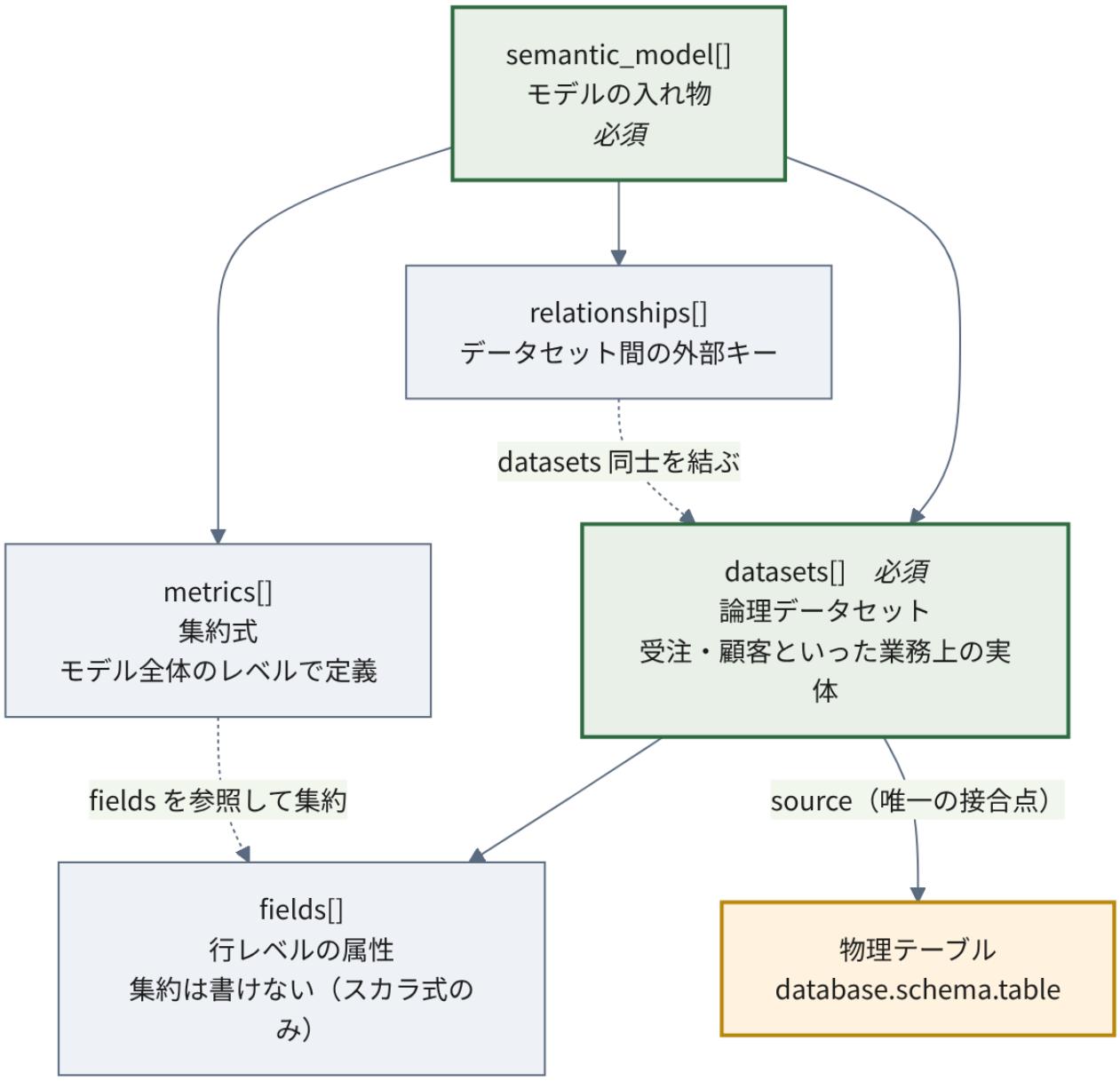

図 4.1: OSI 文書のデータモデル

Use scalar SQL expressions (no aggregations)

（スカラの SQL 式を使うこと。集約は用いない）

そして dimension が持つ情報は is_time の真偽のみである。**数値か文字列かといった型情報を仕様は運ばない。**消費側は物理テーブルのスキーマを見るか、独自に推論するしかない。この設計の帰結は セクション 7.4 で実測する。

この欠落は、プロジェクト側でも改善項目として認識されている。ROADMAP は is_time に代えてフィールドのデータ型そのものを持たせる Issue #84 “Support field datatype rather than is_time” を挙げている[2]。現行スキーマには未反映だが、既知の課題として作業領域に載っている。

4.2.3.4 Metrics

（spec.md L308-）

フィールド	必須	備考
name	○	
expression	○	方言対応のオブジェクト
description / ai_context / custom_extensions		

セマンティックモデルレベルで定義され、複数データセットにまたがれる。式はデータセット修飾付き（SUM(orders.amount)）で書かれており、フィールド式が裸のカラム名なのと対照的である。

4.2.3.5 Relationships

（spec.md L155-）

from（多側）→ to（一側）を from_columns / to_columns の配列で結ぶ。順序が対応し、要素数が一致する必要がある。複合キーに対応する。

4.3 式と方言

```
expression:
  dialects:
    - dialect: ANSI_SQL
      expression: "customer_id"
```

方言 enum は 7 種（spec.md L48-60）。

ANSI_SQL / SNOWFLAKE / MDX / TABLEAU / DATABRICKS / MAQL / BIGQUERY

同一フィールド・メトリクスに複数方言の式を併記できる。

[2] apache/ossie Issue #84: Support field datatype rather than is_time。ROADMAP の改善項目として参照。アクセス日 2026-07-21

```yaml
metrics:
  - name: total_revenue
    expression:
      dialects:
        - dialect: ANSI_SQL
          expression: "SUM(orders.amount)"
        - dialect: SNOWFLAKE
          expression: "SUM(orders.amount)"
        - dialect: MDX
          expression: "Sum([Orders].[Amount])"
```

ここが設計上の要点で、**方言間の変換は仕様の責務ではない**。

4.3.1 方言ごとの式はどこから来るのか

仕様は方言間の変換を定義していない。`expression` は方言別エントリの配列であり、複数方言の式を併記できる形にはなっているが、ある方言の式から別の方言の式を自動生成する仕組みは仕様に含まれない。変換を提供するかどうかは各実装に委ねられている。

したがって、Snowflake でも MDX でも動くモデルを配布したい場合、仕様が保証するのは「併記された式はそのまま運べる」ことまでで、併記されていない方言の式がどこから来るかは仕様の外にある。実装が変換器を持てばそこで補われるし、持たなければ書き手が併記することになる。

> ❗ Write Once の射程は構造に限られる
>
> 「Write Once, Query Anywhere」の Write Once が仕様として保証しているのは、おもに**構造**------データセット・フィールド・関係・AI 向けの文脈------を一度書けば運べる、という部分である。
> 式の可搬性については、仕様は併記の器を用意するだけで、**方言間の自動変換は提供しない**。ANSI_SQL しか書かれていないモデルを MDX 系のツールが読んだとき何が起きるかも、仕様は定めていない。その差を実装が変換で埋めるのか、書き手が併記で埋めるのかは、仕様の外の問題である。

この負担配分を変えようとしているのが、後述の expression language 提案である（セクション 4.7）。

4.4 拡張の仕組み

`custom_extensions` は `vendor_name` と任意の JSON 文字列 `data` の組で、ベンダー固有情報の退避先となる。

コア互換性を壊さずに拡張するための仕組みだが、裏を返せば**ここに入った情報は他ベンダーには解釈されない**。「無損失な round-trip」を謳う実装が、実際には ベンダー固有情報を `custom_extensions` に押し込んでいるだけ、という場合がある。round-trip として無損失であることと、interchange として無損失であることは別である。

4.5 仕様内の不整合

Metrics セクションの Expression Object 定義と、直後の Examples が食い違っている。

定義側:

```yaml
expression:
  dialects:
  - dialect: ANSI_SQL
    expression: "..."
```

例示側:

```yaml
- name: total_revenue
  expression:
    - dialect: ANSI_SQL
      expression: SUM(orders.amount)
```

例示には `dialects:` キーがなく、`expression` が直接配列になっている。JSON Schema の `Expression` は `dialects` を必須とし `additionalProperties: false` であるため、**例示のほうがスキーマ違反である**。

なお公式サンプル `examples/tpcds_semantic_model.yaml` は定義側に従っている。

4.6 0.1.1 と 0.2.0.dev0 の差分

散文の `spec.md` と JSON Schema の両方を比較した。

4.6.1 散文で分かる差分

変更	内容
`vendor_name` の緩和	閉じた enum →自由文字列。同じ一覧は非規範的な “well-known examples” として残り、`HONEYDEW` が追加
`BIGQUERY` 方言の追加	6 種 → 7 種
記述の修正	タイポ、ASF ライセンスヘッダ、“OSI” → “Apache Ossie”

4.6.1.1 変更の理由

いずれもコミットメッセージに理由が明記されている[3]。

`vendor_name` の緩和（`7e984cb`）:

[3]旧リポジトリ open-semantic-interchange/OSI の履歴より。コミット `7e984cb`（2026-05-21）、`c2233f0`（2026-05-28）、`39f6999`（2026-06-02）。アクセス日 2026-07-20。和訳は筆者による

> Allow any vendor or organization to define custom extensions without requiring changes to the core specification.
>
> （どのベンダーや組織であっても、コア仕様の変更を要さずにカスタム拡張を定義できるようにする。）

閉じた列挙のままでは、拡張したいベンダーが増えるたびに仕様本体を書き換える必要がある。その依存を切る変更である。ASF 移管を控え、特定企業の名前を規範に埋め込まない形にする意図とも読めるが、**そこまでは明言されていない**。

`BIGQUERY` の追加（`39f6999`）は外部コントリビュータによるもので、既存の 6 方言に GoogleSQL を並べる素直な追加である。

4.6.2 JSON Schema でしか分からない差分

散文の比較だけでは見落とす構造変更がある。

ルート直下の `dialects` / `vendors` が削除されている。

スキーマ	ルートで許されるプロパティ
0.1.1	`version`, **`dialects`**, **`vendors`**, `semantic_model`
0.2.0.dev0	`version`, `semantic_model`

いずれもルートは `additionalProperties: false`。したがって 0.1.1 で妥当だったルート直下の `dialects` を持つ文書は、**0.2.0.dev0 ではスキーマ違反になる**。

4.6.2.1 削除の理由と、リポジトリ内に残る不整合

このプロパティを削除したコミット `c2233f0`（2026-05-28）は、理由をこう書いている[4]。

> These top-level enumeration properties were **not referenced by any example or downstream consumer**. validate.py only reads dialects from each expression's per-dialect entries via the Expression definition.
>
> （これらのトップレベルの列挙プロパティは、**いかなる例からも下流の利用側からも参照されていなかった**。validate.py は Expression 定義を介して各式の方言別エントリからのみ dialects を読む。）

整理としては筋が通っている。ルートの列挙は宣言されているだけで、どこからも使われていない「死んだプロパティ」に見えた。

しかし同じリポジトリの dbt コンバータは、ルート直下に `dialects` を出力する。固定コミット `07be0176` 時点のコンバータ実装を実際に動かすと、`version` 文字列を `0.2.0.dev0` としながらルート `dialects` を含む文書を吐く（セクション 7.3）。つまり削除の根拠が言う「下流の利用側から参照されていない」プロパティを、リポジトリ同梱のコンバータ自身が出力していることになる。

[4] 同上、コミット `c2233f0`。和訳は筆者による

git 履歴をたどると、この出力はコンバータ導入時から続いており、version 文字列だけが後に 0.2.0.dev0 へ更新され、ルート dialects の出力は残されている[5]。削除の判断がこのコンバータを見落としていたのか、別トラックとして許容していたのかまでは、コミットメッセージからは読み取れない。

結果として、公式コンバータは **0.1.1 の構造に 0.2.0.dev0 のラベルを貼った文書**を出力する。同一コミットの公式バリデータを通らない成果物であり、これは動作確認で実測している（セクション 7.3）。

なお spec.yaml がトップレベルに dialects: を並べている書式も、この種の誤読を招きやすい（セクション 4.1）。

4.6.2.2 エンティティ定義は変わっていない

$defs を比較すると、SemanticModel / Dataset / Field / Metric / Relationship などエンティティ定義は**すべて同一**である。差分は Dialect（BIGQUERY 追加）と Vendor（enum →自由文字列）のみ。

> ❗ バージョン文字列は排他的な識別子である
>
> 各スキーマは version を **const** で固定している。enum でも pattern でもない。
>
> | スキーマ | properties.version |
> |---|---|
> | osi-0.1.1-rc1 | {"const": "0.1.1"} |
> | main | {"const": "0.2.0.dev0"} |
>
> エンティティ定義がほぼ同一であるにもかかわらず、各スキーマは自分のバージョン文字列だけを受け付ける。
> **構造的にほぼ同じ文書が、バージョン文字列だけで相互に排除される。**

この設計が実装にどう波及するかは チャプター 7 で実測する。

4.7 expression language — 提案が文書化された段階

core-spec/expression_language.md が **2026-07-15 付で main に存在する**（ステータス “Proposed Final”）。ASF 移管アナウンスの 3 週間後に入ったばかりである。

ただし「実装段階に入った」と読むのは早い。この文書が定義する新方言 Ossie_SQL_2026 は、**固定コミット時点の osi-schema.json の方言 enum にまだ入っておらず、既定方言化も実装されていない**。つまり現状は、口頭のアイデアではなく詳細な仕様案がリポジトリに文書として取り込まれた、という段階である。初期リリース（0.1.1、後述の RC タグ）にも当然反映されていない。

4.7.1 何を変えようとしているのか

現行の設計では、方言ごとの式を**モデル作成者が手で併記する**（セクション 4.3）。この提案はその負担配分を変える。

[5] 固定コミット 07be0176 の converters/dbt。ルート dialects を出力する挙動と、version="0.2.0.dev0" 化の経緯は同ディレクトリの git 履歴による。本レポートは実際の出力をセクション 7.3 で確認している。アクセス日 2026-07-21

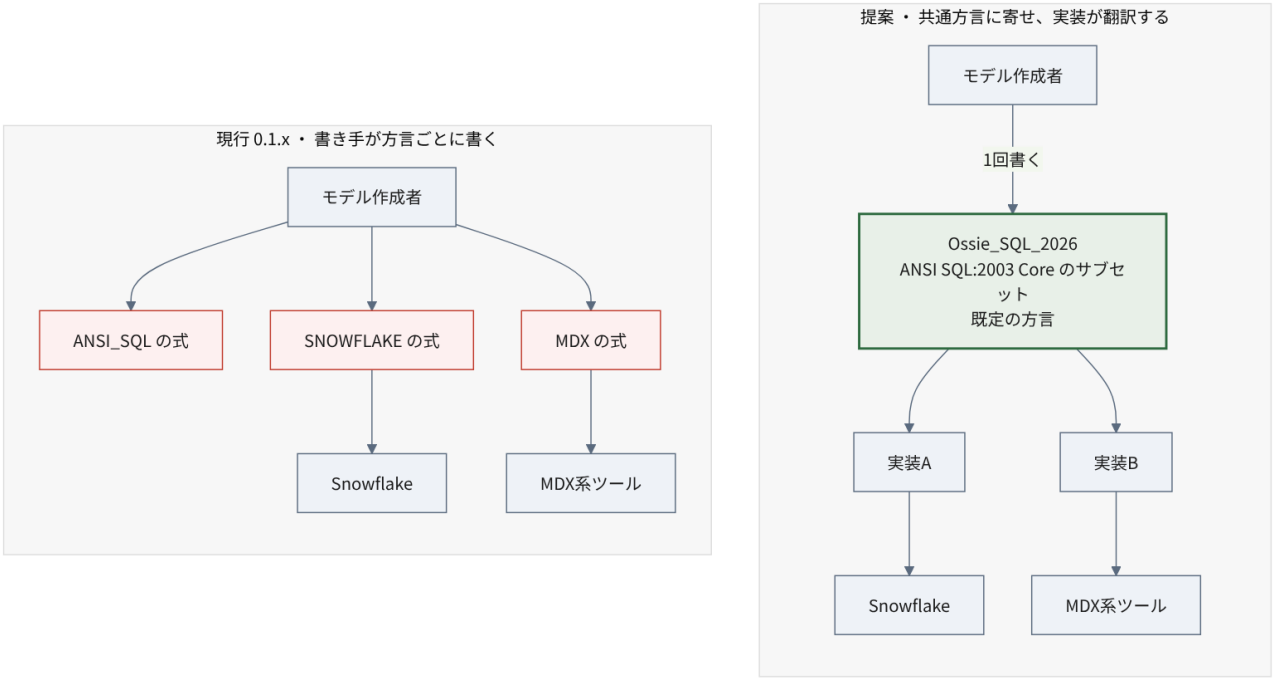

図 4.2: 方言をどう扱うか --- 現行と提案

やり方は、方言どうしを相互変換するのではなく、**Ossie_SQL_2026 という共通の方言を新たに定義し、そこに寄せる**というものである。仕様には次の 2 点が明記されている。

1) Create a new dialect in the Ossie spec: Ossie_SQL_2026, which refers to this language specification.
2) Make Ossie_SQL_2026 the default dialect if one is not chosen.

(1) Ossie 仕様に新しい方言 Ossie_SQL_2026 を作る。これは本言語仕様を指す。2) 方言が指定されなかった場合の既定を Ossie_SQL_2026 とする。)

N 個の方言があるとき、相互変換なら N×N 通りの対応が要る。共通方言を 1 つ置けば、各実装が「その方言を自分のエンジン向けに解釈する」だけで済む。**ハブを 1 つ設けて放射状にする**構図である。

4.7.2 負担が書き手から実装へ移る

この提案の要点はここにある。

	現行 0.1.x	Ossie_SQL_2026 提案
方言差を吸収する主体	併記した書き手／変換を持つ実装	共通方言を解釈する**実装**
書き手がすることは	併記したい方言の数だけ式を書く	共通方言で 1 回書く
可搬性の担保	仕様は保証しない	**採択・統合されれば**仕様が実装に要求する

提案文書は“the SQL expression language subset that Ossie-compliant implementations **MUST support**”（Ossie 準拠の実装が**サポートしなければならない** SQL 式言語のサブセット）と述べており、義務を実装側に置く設計になっている。ただしこの MUST は“Proposed Final”段階の提案文書内の記述であって、**現行の `osi-schema.json` に統合された要件ではない**。採択されて初めて「準拠実装への要求」として効く。

4.7.3 3 段階コンプライアンスは実装に課される

REQUIRED / RECOMMENDED / MAY は、**文書の書き方の区分ではなく、実装が満たすべき水準**を指す。

水準	意味
MUST Support（REQUIRED）	実装は必ず対応する。最小の可搬な式言語
SHOULD Support（RECOMMENDED）	対応が望ましい。全 DB にあるとは限らない分析関数
MAY Support（拡張）	対応は任意

基盤に **ANSI SQL:2003 Core**（ISO/IEC 9075-2:2003）を選んだ理由として、Snowflake・Databricks・PostgreSQL・BigQuery に広く実装されていること、意味論が定まっていること、window 関数や CTE といった分析機能を持つことが挙げられている。

ただし注意が要る。**CTE が選定の魅力として挙がることと、式で CTE を使えることは別である。**同じ提案文書の“Not Supported”は **CTE を明示的に対象外としている**。つまり「CTE を持つ SQL:2003 Core を土台に選んだ」が、「移植可能サブセットで CTE を書ける」わけではない。

ベンダー固有の機能は、従来どおり方言拡張から使える。共通部分の可搬性を仕様が担保し、その外は各ベンダーに委ねるという切り分けである。

4.7.4 適用範囲は論理層のみ

この提案が対象とするのは**論理層**（メトリクス・フィールド・フィルタ）であり、オントロジー層は対象外と明記されている。オントロジー層でも同じ式言語を使えるとよいが、それは別提案として扱う、とある。

なおオントロジー層の対応先として、仕様は OWL に加えて RelationalAI の (Py)Rel と Goldman Sachs の Legend を挙げている。チャプター 3 で触れた 2 つの系譜が、ここでも参照されている。

4.7.5 オントロジー層の言語は ORM 系である

オントロジー層そのものは別ファイル `ontology/ontology.md` で定義されている[6]。その設計は、BI 系のセマンティックモデルとも、RDF/OWL の狭義オントロジーとも異なる、**Object-Role Modeling（ORM）系**である。根拠は次のとおり。

[6] Apache Ossie, “core-spec/ontology/ontology.md”（`version: 0.2.0.dev0`）。コミット `07be0176`（2026-07-17）時点。アクセス日 2026-07-20

- **関係は n 項**である。`relationship` は “relate objects of **one or more** concepts” と定義され、`OneToOne` は “only for binary relationships” と但し書きされる。二項に限らない関係を前提にしている
- **role（役割名）**を第一級で扱う。各関係は `roles` を持ち、概念は「その関係で最初の role を演じる」形で関係を束ねる
- **verbalizes（自然言語化）が必須**。関係には `verbalizes: [ '{Person} is identified by {SocialSecurityNr}' ]` のように、リンクを自然言語で言い表すパターンを付ける

n 項関係・role・verbalizes を核に据える設計は、ORM / NIAM（fact-based modeling）の系譜そのものである。RelationalAI の (Py)Rel が同系統であることとも符合する。

つまり Ossie は、**論理層（次元的なセマンティックモデル）とオントロジー層（ORM 系）という、由来の異なる 2 つのモデリング伝統を 1 つの仕様の中で接ごうとしている**。チャプター 3 の「オントロジーの語には幅がある」で触れた論点の、仕様側の裏づけである。

4.7.6 集約関数の分解可能性分類

技術的に最も興味深いのはこの部分である。集約関数を 4 つに分類する。

分類	例	性質
Distributive	SUM, COUNT, MIN, MAX	部分集約の結果をそのまま再集約できる
Algebraic	AVG, STDDEV	中間状態を持てば再集約できる
Holistic	MEDIAN, COUNT DISTINCT	再集約できない
Sketch-based	---	近似構造を用いる

これは多段集約、すなわち **cross-grain 再集約**の土台になる。粒度をまたいで指標を積み上げられるかどうかは、セマンティック層が実用に耐えるかを分ける論点であり、それを仕様レベルで扱おうとしている点は評価できる。

ただし**再集約規則そのものは Google Doc にあり、Apache リポジトリにない**（セクション 5.3）。

4.7.7 移植不能な部分の隔離

日時フォーマット文字列は Oracle 系 `TO_CHAR` / `strftime` / Java LDML の 三つ巴で移植不能である。仕様はこれを **EXPERIMENTAL として明示的に隔離**したうえで、全主要エンジンで表現可能な「移植可能コア」だけを定義している。

解けない部分を解けないと書く姿勢は、仕様の書き方として誠実である。

4.8 未決の設計論点

GitHub Discussions には設計上の議論が公開されている。以下は結論が出ていない主な論点である。

これらは Discussion 上で散発的に交わされているだけではない。プロジェクトの `ROADMAP.md` は共通の式・クエリ言語（“Semantic Query Language & Reference Engine”）を明示的な取り組みとして掲げ

ており[7]、フィールドのデータ型は前掲 Issue #84、検証済みクエリは Discussion #82 として、それぞれ追跡対象になっている。「未決」は放置ではなく、作業項目として認識されたうえでの未決である点は補っておく。

4.8.1 先に用語を 2 つ

measure（メジャー）とは、**集約の対象となる数量と、その集約方法の組**を指す。Cube・MetricFlow・Looker といったセマンティック層では、これをデータセット（テーブル）の内側に定義する。

```
# dbt/MetricFlow のネイティブ記法の例
semantic_models:
  - name: orders
    measures:
      - name: amount_sum      # ← これが measure
        agg: sum              #   何を(expr)どう集約するか(agg)を持つ
        expr: amount
```

「orders テーブルの amount カラムを合計したもの」に `amount_sum` という名前を与え、メトリクスはその名前を参照する。**集約の定義がテーブルに紐づく**のが特徴である。

メトリクスツリーは、ひとつの指標を構成要素へ分解して木構造で表したものを指す。「売上 = 客数 × 客単価」「客数 = 新規 + 既存」のように辿れる形にして、どの要素が全体に効いているかを見るために使う。value driver tree などとも呼ばれ、**名称自体が定まっていない**。

4.8.2 集約方法をどう表現するか

3 案が並存し、未採択のままである。

案	提唱	内容
構造化 enum	AtScale	`aggregation_method` を列挙型で持つ
関数名前空間	Lloyd Tabb（Malloy）	`OSI_*` 関数として定義
SQL サブセット + エスケープハッチ	Snowflake	移植保証付きのサブセットを定め、外れる場合は逃がす

実装フェーズに進んだのは 3 番目である（セクション 4.7）。

4.8.3 メトリクスツリー

「メトリクスツリー」が派生メトリクスなのかどうかで、提起者と RelationalAI が真っ向から対立している（“I am definitely **not** talking about derived metrics” ＝私が言っているのは派生メトリクスでは断じてない）。用途も「相関検証」対「エージェントへの推論コンテキスト」で割れており、名称すら value driver tree / metric tree / derived metrics で定まっていない。

[7] apache/ossie の ROADMAP.md（固定コミット `07be0176`）。式・クエリ言語は “Future Efforts” に置かれ、専任ワーキンググループは未設置。アクセス日 2026-07-21

4.8.4 最上位 metrics か dataset 内 measures か

セクション 4.2 で見たとおり、Ossie は `metrics` をモデル全体のレベルに置く。既存のセマンティック層は、いずれもデータセットの内側に measure を置く。

どこが違うのか。

	dataset 内 measure 方式	Ossie の最上位 metrics 方式
集約の定義場所	テーブルに紐づく	モデル全体
「売上合計」を定義するには	orders の中に `sum(amount)` を置く	どのデータセットにも属さない式として書く
複数テーブルにまたがる指標	扱いにくい。measure は 1 テーブルに属するため	**書きやすい**。最初からモデル全体を見ている
消費側への写像	既存ツールと同型	**対応先が決まらない**

Ossie 方式の利点は、複数のデータセットにまたがる指標を素直に書けることである。「顧客あたり売上」のように、注文テーブルと顧客テーブルの両方を要する指標を、どちらかのテーブルの所属物にせずに定義できる。

一方の難点が、消費側との噛み合わなさである。dbt/MetricFlow は「メトリクス = measure への参照」を前提とするので、measure を持たない OSI のメトリクスを受け取ると**参照先が空になる**。実際に集約関数が脱落することは セクション 7.4 で実測している。

この噛み合わなさを背景に、議論スレッドでは複数の参加者から「分けることに何の価値があるのか」「名前空間の問題なのか、意味論的な違いがあるのか」という問いが投げられている。主要な実装が揃って measure 方式を採っている以上、そこから外れる Ossie 方式の側が理由を示すべきではないか------という含意の問いであり、いわば**立証責任を Ossie 側に置く論法**が参加者から提示された形である。これに対し**仕様側は「メトリクス言語ワーキンググループの重要な論点であり、提案がまとまり次第、公開文書を出す」という回答にとどまっている**[8]。

4.8.5 横断する軸 — どこまで仕様で構造化するか

個々の論点を貫いているのは、**何をどこまで仕様の構造として定めるか**という問いである。

対立する 2 つの立場がある。

構造化する側は、意味を明示的なフィールドとして仕様に持たせたい。検証できる、実装が解釈を誤らない、という利点がある。代わりに仕様が大きくなり、実装がすべてを解釈する義務を負う。

構造化しない側は、自由テキストや汎用の入れ物に留めたい。仕様が小さく保たれ、実装の負担が軽い。用途の変化にも追随しやすい。代わりに、解釈は消費側（近年はしばしば LLM）に委ねられ、同じ文書が実装によって違う意味に読まれうる。

要するに柔軟性と実装コストのトレードオフである。 仕様を厚くすれば相互運用の精度は上がるが、その水準を満たせる実装は減り、普及が遅れる。

この緊張は具体的な議論にも現れている。たとえば「検証済みクエリ（verified queries）を `ai_context` の中に埋めるのではなく、仕様の第一級要素にすべきだ」という提案には賛同が集まっている[9]。一方で「検

[8] apache/ossie Discussion #29: Top level “metrics” vs. dataset-level “measure”s, アクセス日 2026-07-20
[9] apache/ossie Discussion #82: Add `verified_queries` as a core element of the spec, アクセス日 2026-07-20

証やパーサを提供して品質を担保したいが、**方言を増やすほどそれが難しくなる**」という指摘もある[10]。後者は構造化を志向しつつ、その実装コストが方言の数に比例して膨らむことを述べている。

> **i** 立場と所属の相関は確認できなかった
>
> 「特定ベンダーの参加者が一貫して構造化に反対している」といった、**所属と設計上の立場の相関は、公開されている議論からは読み取れなかった。**
> 構造化を推す発言も慎重な発言も、複数の組織の参加者から出ている。実装コストへの言及も、特定ベンダーに偏ってはいない。

4.8.6 議論が付いていない論点

一方で、提起されたまま反応が少ない論点もある。

- grain（粒度）を first-class の概念として扱うべきか
- Ossie は交換フォーマットなのか、実行時の標準まで担うのか
- インポータは方言変換の義務を負うのか
- FIBO のような既存オントロジーとどう関係づけるか（チャプター 3）

いずれも仕様の性格を決める問いだが、2026 年 7 月時点でコメントが付いていないか、ごく少ない。

いずれも答えを出すのが難しい問いではある。ただし反応の少なさから需要の有無までは読み取れない。コメントが付いていないことは観測できるが、**公開 Discussion 上の反応量だけでは、関心が弱いのか、単に議論の順番が回ってきていないだけなのかは判定できない**。

需要の側から言えば、クエリ時にパラメータを渡せるメトリクスは、累積 window の議論と、金融領域の posted-at / settled-at の議論でそれぞれ独立に必要性が語られている。仕様にはまだ入っておらず、専用の議論スレッドも見当たらない。

[10] apache/ossie Discussion #35 の Kurt Stirewalt（RelationalAI）の発言。“I personally think we should lean into validation by providing parsers and type checkers that help ensure the quality of a model in the OSI. That gets harder to do the more we try to support different dialects.”（OSI のモデル品質を担保するパーサや型検査器を提供する方向に寄せるべきだと個人的には思う。ただし対応する方言が増えるほど、それは難しくなる。）アクセス日 2026-07-20。和訳は筆者による

5 ASF移管とガバナンス

各社のブログは「ベンダー中立になった」と書く。本章はそれが実態としてどこまで成立しているかを、ASF の一次情報で確かめる。

主な参照先は podling status page、Clutch、incubation proposal、dev@ アーカイブ、general@incubator アーカイブ。アクセス日はいずれも 2026-07-19。

結論を先に書く。**構造としては本物、実態はまだ偏在**。「ベンダー中立な運営体制の**入口に立った**」が正確である。

5.1 移管の経緯

項目	内容
Incubator 入り	2026 年 6 月（ASF 資料で日付が揺れる。後述）
Champion	JB Onofré（proposal 記載時は Dremio）
Mentors	JB Onofré / Holden Karau / Zili Chen / Russell Spitzer
初期コミッター（proposal の候補）	14 名（PPMC 5 + committer 9。実際に権限を持つ人数は別。後述）
リポジトリ	`apache/ossie`（作成は 2025-11-18）

> **入り日は ASF 資料内でも揺れている**
>
> Incubator 入りの日付は一次資料の中でも一致しない。Clutch のステータスは `Started: 2026-06-19` とする一方、同じ Clutch の News 欄は `2026-06-22 Project enters incubation.` とする[1]。本レポートでは おおむね **2026 年 6 月下旬**として扱い、日付そのものを論拠にはしない。

受入手続きは正規に踏まれている。

- `[PROPOSAL] Ossie, a data semantic specification and framework` --- 2026-06-11
- `[VOTE] Accept Ossie into the Apache Incubator` --- 2026-06-16
- `[RESULT][VOTE]` --- 2026-06-18

投票結果は **binding +1 が 6 票**（Tison, PJ Fanning, Russell Spitzer, Markus Weimer, JB Onofré, Larry McCay）、non-binding +1 が 2 票。

[1] Apache Incubator, “Clutch: ossie”（“Started: 2026-06-19” と “2026-06-22 Project enters incubation.” が併存）、アクセス日 2026-07-21

5.1.1 受入時のレビューで、確認できたライセンス指摘が 1 件

投票の議論中、IPMC メンバーの PJ Fanning がライセンスの件を指摘している[2]。

> Probably not a big issue but the OSI GitHub repo mentions CC-BY license. But ASF license docs have reservations about this license.
>
> （大した問題ではないと思うが、OSI の GitHub リポジトリは CC-BY ライセンスに 言及している。ASF のライセンス関連文書はこのライセンスに留保を置いている。）

指摘のトーンは穏当で、「おそらく大した問題ではないが」と前置きされている。続いて tison が ASF の Legal Resolved を示し、CC-BY コンテンツをソース形式で含めることへの制限を説明した。

対応は同日中に行われた。Russell Spitzer が **PR #157「Relicense Documentation to ASF 2.0」**を起票し、2026-06-16 19:19 UTC にマージされている[3]。PR 本文は、この変更が受入投票で提起されたライセンス上の懸念に対応するものだと述べる。ただし PR のマージが示すのは**表示上のライセンス変更**であって、権利処理や全著作者の同意といった IP clearance 全体の完了までは、これだけでは確認できない。投票スレッド上では、この後に追加の異議は確認していない。

指摘から解消まで約 5 時間半。受入プロセスに外部のレビューが挟まっていることの確認にはなる。ただし指摘自体は軽微なもので、修正を起票した Russell Spitzer は Snowflake 所属かつ Ossie の mentor でもある。

5.1.2 インフラ立ち上げの経過

podling status page に記録されている日付は以下のとおり[4]。

日付	出来事
2026-06-22	incubation 入り
2026-06-23	DNS 申請
2026-06-26	ML 開設 / git リポジトリ移管 / issue tracker 設定
2026-06-29	mentor アクセス付与
2026-07-08	dev@ で最初の実質投稿

incubation 入りから 4 日で主要インフラが揃っている。**他の podling と比較していないため「速い」とまでは言えない**が、少なくとも立ち上げが滞っている様子はない。

なお podling status page の Mailing Lists 欄は「dev@, private@, commits@, issues@ **polaris**.apache.org」となっており、ドメイン名が Apache Polaris のものになっている。テンプレートの流用時の修正漏れと思われる。

[2] PJ Fanning, “Re: [VOTE] Accept Ossie into the Apache Incubator”, general@incubator.apache.org, 2026-06-16 13:50:42. permalink（アクセス日 2026-07-19）

[3] open-semantic-interchange/OSI, “PR #157: Relicense Documentation to ASF 2.0”, merged 2026-06-16。アクセス日 2026-07-19

[4] Apache Incubator, “Apache Ossie podling status page” の Incubation Work 節。アクセス日 2026-07-19

5.2 リリースと卒業の現状

5.2.1 Apache Release はゼロ

チャプター 2 で述べたとおり、**Apache Release は 1 本も出ていない**。Clutch ページは “No releases” かつ **“No PGP Signing Keys”** を示す（収集日 2026-07-07 UTC 表示）。署名鍵はリリース準備の中で登録できるので「鍵がない＝直近で出せない」とまでは言えないが、少なくとも公開リリースへ向けた外形的な準備の痕跡は、現時点で確認できない。

dev@ のアーカイブにも、リリース計画に関する議論は見当たらなかった[5]。

5.2.2 卒業までの距離

新規 podling は、**最初の 3 か月は毎月、その後は四半期ごとに報告する**義務がある[6]。Ossie は 6 月下旬に受け入れられたため、この月次報告の対象になる。

実際の状況を追うと、**2026 年 7 月の Incubator report で Ossie は「新規 podling」として記載されるにとどまり、詳細報告は提出していない**[7]。受入が直近すぎたためで、“failed to report” 扱いでもない。最初の実質的なステータス報告は、通常なら 8 月の月次報告になる。

したがって「初回報告まで何もない」のではなく、**今後の月次報告が最初の外部評価の機会**になる。8 月・9 月の報告に、mentor の応答性・PPMC の自己評価・リリース計画が現れるかを見るのが有効である。

Clutch が挙げる未充足項目は次のとおり[8]。

- ドキュメント用 wiki の未整備
- podling website の未整備（`ossie.apache.org` は稼働しているが、Clutch の基準を満たしていないと判定されている。理由は未確認）
- **release distribution area の未設定**

卒業の判断は、固定した期間や報告回数で決まるものではなく、コミュニティの多様性、ASF 流の意思決定の定着、**Apache Release を作成・公開できる能力の実証**、IP clearance の完了、そして最終的にコミュニティ投票・IPMC の推薦・理事会の承認による[9]。

現時点の Ossie は、Apache Release がなく（チャプター 2）、release distribution area も未設定で、ML 上で追跡できる合意・投票の実績も乏しい。**卒業の時期は予測できない**が、少なくとも現状はリリース能力・意思決定の定着といった実績がまだ積み上がっていない段階である。参考までに、ASF は典型的な incubation 期間を約 1.5 年としている[10]。

---

[5] 2026-07-19 時点で dev@ossie.apache.org に存在する全 28 通の件名を確認し、`release` / `vote` / `rc` を含むものを探した結果による。不在の証明であり、本質的に弱い主張である点に留意されたい。私信や Slack での検討までは把握できない

[6] Apache Incubator, “Incubation Policy --- Podling Reporting”（“Podlings SHALL report monthly for their first three months, after that quarterly.”）、アクセス日 2026-07-21

[7] Apache Incubator, “July 2026 report”（“A new podling, Ossie … was accepted into the Incubator”。詳細報告のあった podling は Caldera, Casbin, Fluss, PonyMail, ResilientDB）、アクセス日 2026-07-21

[8] Apache Incubator, “Clutch: ossie”、収集日 2026-07-07 UTC 表示。アクセス日 2026-07-21

[9] Apache Incubator, “Guide to Successful Graduation” / “Release Management”。アクセス日 2026-07-21

[10] Apache Incubator, “トップページ”。アクセス日 2026-07-21

5.3 意思決定はどこで行われているか

5.3.1 dev@ の稼働実態

dev@ossie.apache.org のメッセージ総数は **28 通**（2026-07-19 時点）[11]。リスト開設は 2026-06-26、実質的な最初の投稿は 2026-07-08。**稼働してまだ約 2 週間**である。

立っているスレッドは、アナウンス、運営提案、参加表明、技術議論、そして GitHub Discussions からの転送に分かれる。投稿数上位は JB Onofré 6 通、Yong Zheng 6 通、Will Pugh 3 通。

複数社が絡む議論にはなっている。“Inconsistent projects/modules managements” スレッドには Emil Sadek（columnar.tech）、Khushboo Bhatia、Aniket Kulkarni（dremio.com）が返信している。

5.3.2 仕様変更の [VOTE] は一度も実行されていない

podling サイトは “a formal discussion-and-vote process for spec changes”（仕様変更に関する正式な議論と投票のプロセス）を掲げる。しかし **dev@ 上に仕様変更の [VOTE] スレッドは 1 本も存在しない。**

唯一 `[PROPOSAL]` が付いているのはコミュニティミーティングの開催時刻の件であり、仕様の変更に関するものではない。

実際の仕様議論は **GitHub Discussions で行われ、`[D]` プレフィックス付きで dev@ に転送されている**にとどまる。

ASF が求める「決定は ML 上で行う（If it didn’t happen on the list, it didn’t happen）」という規範から見ると、**議論の重心はまだ GitHub 側にある**。これは incubation 初期の podling によくある状態であり、mentor から是正が入る典型ポイントでもある。

5.3.3 GitHub Discussions からも決定が抜けている

中核的な意味論が Google Docs と Slack に置かれている。集約の再集約規則という、仕様の根幹に関わる内容が Google Doc にあり、Apache リポジトリに存在しない。

議論の所在が分散していることの一例として、オントロジー層の追加という近い主題が、別々の Discussion に立っている。2026-01-15 の #22「relational ontologies にもとづくモデル交換」と、2026-04-06 の #101「オントロジー層のサポート」である[12]。両者が同じ論点を重複して扱っているように見える一方、**後発の #101 が先行の #22 を「重複」と認識した形跡は、公開情報では確認できない**。情報分散が原因で重複したのか、単に並行して提案されたのかは断定できないが、関連する提案が複数の場所に散らばること自体は観測できる。

ワーキンググループも Slack チャンネル単位で運営されている（チャプター 2）。

[11] `lists.apache.org` の統計 API（`stats.lua?list=dev&domain=ossie.apache.org&d=lte=3M`）による。アクセス日 2026-07-19。個々のスレッドは同アーカイブから辿れる

[12] apache/ossie Discussion #22（2026-01-15）および Discussion #101（2026-04-06）。いずれもオントロジー層の追加を扱う。#101 のコメントに #22 への参照・重複の指摘は見当たらない。アクセス日 2026-07-21

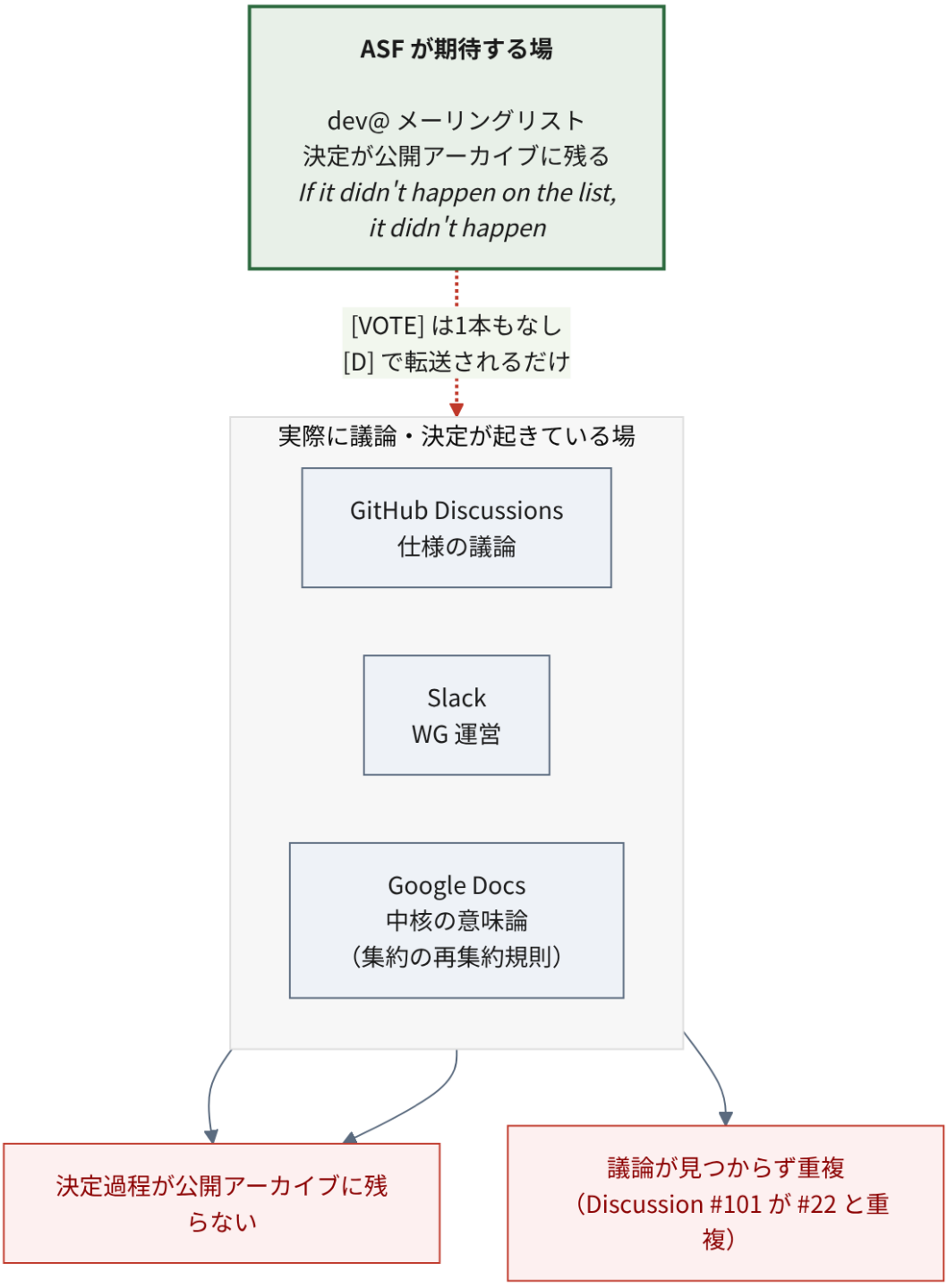

図 5.1: ASF が決定の場として期待する ML と、実際に議論・決定が起きている場のずれ

⚠ ASF に移管した意味が問われる部分

ML では決めず、GitHub Discussions で議論し、実質的な決定は Slack と Google Docs にある------という構図になっている。
ASF のガバナンスが提供する価値の中核は「決定過程が公開アーカイブに残ること」である。その点で、ML 上に追跡できる合意・投票の実績はまだ乏しく、ASF 流のプロセスが定着したとは言いにくい。ただし ML への転送や 公開 GitHub Discussion は使われており、完全に機能していないわけではない。
なお ML アーカイブ自体にも不備があり、プロジェクト側も Issue #193 で認識している。

5.4 ベンダー中立性の実態

5.4.1 PPMC の構成

incubation proposal では PPMC 5 名と committer 9 名に分かれており、**所属が明記されているのは PPMC 5 名のみ**である。

PPMC	所属（proposal 記載）
Khushboo Bhatia	Snowflake
Lior Ebel	Salesforce
Quigley Malcolm	dbt Labs
JB Onofré	Dremio
Russell Spitzer	Snowflake

この **proposal 時点の PPMC 候補 5 名のうち 2 名が Snowflake** である。ただしこれは proposal の候補構成であって、現時点の PPMC 全体ではない（後述のとおり、実際に権限を持つ人数は Clutch 上 4 名にとどまる）。

残り 9 名は proposal 上で所属が明示されていない。これ自体が透明性の観点で弱点になる。

5.4.2 コミットログから見える実態

`sfc-gh-` プレフィックスは Snowflake 社員アカウントの命名規則である。

コミット数の上位は次のとおり。

貢献者	所属	commit 数
Khushboo Bhatia	Snowflake	20（gmail 16 + snowflake.com 4）
Kurt Stirewalt	RelationalAI	15
Baruch Oxman	Honeydew	15
JB Onofré	Apache	9
Ralf Becher	個人	6
Emil Sadek	columnar.tech	8
Quigley Malcolm	dbt Labs	4

出典は GitHub API（`repos/apache/ossie/contributors` および `repos/apache/ossie/commits`）。所属はコミットのメールドメインによる推定を含む。アクセス日 2026-07-19。`sfc-gh-` プレフィックスが Snowflake 社員の命名規則である点は Snowflake のブログ著者ページで裏を取った。

Snowflake が単独最大ではある。しかし **RelationalAI と Honeydew の個人がそれぞれ Snowflake 主力とほぼ同等の commit 数**を出しており、「Snowflake のワンマンプロジェクト」とまでは言えない。

なお `karanasher`（19 commits、contributor 3 位）は committer リストに載っていない。所属も未確認である。

5.4.3 proposal 自身が偏重を認めている

Ossie の incubation proposal は、Known Risks の節でこう書いている[13]。

Homogeneous Developers（開発者の同質性）

> the contributor base is currently concentrated across three organizations. We view diversification as a critical incubation goal, not a stretch objective.
>
> （貢献者の基盤は現在、3 つの組織に集中している。我々は多様化を、努力目標ではなくインキュベーションの重要な達成目標と位置づける。）

対処として、good first issue のバックログ整備、オンボーディング・ワークショップ、コントリビュータ・スプリントの実施を挙げ、創設組織の外からの参加者を積極的に募るとしている。

Reliance on Salaried Developers（有給開発者への依存）

> While most of the contributors are affiliated with Snowflake, Dremio, dbt Labs, Salesforce, the commitments to open source ensure a genuine commitment to the Apache Way and open source principles.
>
> （貢献者の多くは Snowflake、Dremio、dbt Labs、Salesforce に所属しているが、オープンソースへのコミットメントが、Apache Way とオープンソースの原則への真摯な取り組みを担保する。）

エコシステムが成熟し他組織が実装を作るにつれ、創設企業の被雇用者を超えて貢献者が自然に多様化すると見込む、としている。

各社ブログの「ベンダー中立になった」という表現と比べると、proposal 本文のほうが現状を率直に記述している。

なお proposal は meritocracy についてこう述べる。

> contributors earn merit through sustained work of any kind and committership is earned by demonstrated contribution rather than employer
>
> （貢献者はあらゆる種類の継続的な作業を通じて信任を得る。コミッター権は、雇用主ではなく実証された貢献によって得られる。）

これは**方針の宣言であって現状の描写ではない**点に注意が要る。

同じ proposal は、仕様変更について **最低 7 日間の議論期間と +1 / 0 / -1 の投票**を要すると定めている。この手続きが実際に使われた記録は、2026 年 7 月時点で確認できていない（前節）。

[13] Apache Incubator, "Ossie Proposal", アクセス日 2026-07-19。和訳は筆者による

5.4.4 Clutch と status page の食い違い

Clutch は committers 4 名（全員 PPMC）と表示する一方、podling status page は 14 名を挙げる。

Clutch が committers を 4 名と数える一方、status page は 14 名を挙げる。14 名は proposal 上の「予定された初期コミッター」であり、実際に権限を持つのはその一部、という読みはできる。ただし Clutch の 4 名が正確に何を数えているか（LDAP アカウント作成済みか、実際の commit 権限保有者か）は一次資料から確定できない。**推定にとどまる**。

5.5 移管の前後で活動量は増えたか

移管後に活動量は増えている。ただし変動値のスナップショットであり、移管が原因で加速したという因果までは、この数字だけでは言えない。

以下は **2026-07-19 時点**の GitHub API の値である（日次で変わる）。

指標	値（2026-07-19 時点）
merged PR 累計	68
うち 2026 年 6 月下旬以降	**32**
6 月下旬以降の commit	52
リポジトリ star	1,342

出典は GitHub API（`search/issues?q=repo:apache/ossie+type:pr+is:merged` および `repos/apache/ossie`）。アクセス日 2026-07-19。数値は集計方法（merge 方式・共同著者・メール統合）で変わりうる。移管日（6/19 か 6/22 か）で前後区分も動く。

約 7 ヶ月（2025-11 〜 移管前）で merged PR 36 本に対し、移管後の約 4 週間で 32 本。活動量が増えたのは確かだが、次の理由で「移管が加速させた」とは断定できない。

- 4 週間という観測窓は短く、**アナウンス直後の一時的な効果と持続的な傾向を区別できない**
- ASF 移管に伴う機械的変更（LICENSE/NOTICE 追加、ヘッダ付与、CI 再構成、ライセンス変更作業）が PR 数を押し上げている可能性がある。**内訳は未確認**

定性的には、dev@ 開設からの 2 週間で committer リストにない人物からの参加表明が複数あり、**ASF ブランドが新規参加の敷居を下げた**兆候はある。

5.6 中立性をどう判定すべきか

「ベンダー中立になった」が妥当かは、宣言ではなく実績で判定すべきである。以下の 3 点が指標になる。

1. **仕様変更の VOTE で、所属に関係なく反対意見が検討され、理由付きの合意が公開の場に記録されるか**
2. **創設企業以外からの committer 昇格が起きるか**
3. **初回 Apache Release を IPMC 投票で通せるか**

いずれも 2026-07-19 時点では未達である。

6 エコシステムの現状

「参加 50 社超」という数字が独り歩きしている。本章では、それが実装状況としてどこまで裏付けられるかを、2026-07-19 時点で調べる。

> **本章の数値はスナップショットである**
>
> 参加社数・実装状況・各種の日付は、この分野では短期間で動く。以下は **2026-07-19 時点**の確認にもとづく。GitHub の状態を指す箇所はコミットや PR 番号を併記し、変わりうる値であることを明示する。

6.1 参加と実装の落差

参加組織は 50 社を超える[1]。一方、それを製品として使える状態にした実装は、**成熟度に大きな幅がある**。「対応を表明した」ことと「今すぐ使える」ことは別なので、段階を分けて数える。

段階	意味	該当（2026-07-19 時点で確認できたもの）
GA	一般提供	dbt Core 1.12（import + export、2026-07-16 GA。※変換に制約あり）[2]
Open Preview	出荷済みだがプレビュー	Snowflake（semantic view ↔ OSI YAML、2026 年 6 月導入）[3]
有償・限定提供	公開情報が乏しい	Honeydew（import 方向、要問い合わせ）[4]

[1] Apache Ossie, “Apache Ossie (Incubating): The New Name for Open Semantic Interchange”, アクセス日 2026-07-19。創設 17 社から 50 社超へ拡大した旨の記載による。「参加」は coalition の自己申告的な集計であり、実装企業数ではない。

[2] dbt docs: OSI semantic models、アクセス日 2026-07-19。1.12.0 は 2026-07-16 公開（PyPI）。export は未対応構文を落として警告する制約がある。

[3] Snowflake, “Preview features”（“Open Semantic Interchange (OSI) YAML format for semantic views”、Open Preview、2026 年 6 月）、アクセス日 2026-07-21。関数は SQL functions index 参照

[4] Honeydew は有償製品で、対応の詳細は要問い合わせ。公開ドキュメントからは import 方向・提供形態・対応バージョンを十分に裏付けられなかった。アクセス日 2026-07-19

[5] Salesforce, “Ending Semantic Drift” ほか。Tableau Semantics の重点として bi-directional metadata exchange を挙げる。Ossie リポジトリの converters/salesforce（jar `0.1.0-SNAPSHOT`）も参照。アクセス日 2026-07-21

[6] Databricks, “What’s new with Unity Catalog at Data + AI Summit 2026”, アクセス日 2026-07-20。原文は “Metrics is also open: it’s open source, available in Apache Spark and Unity Catalog OSS, and its Open Semantic Interchange (OSI) ready.”。和訳は筆者による

表明・重点領域	出荷は未確認	Salesforce/Tableau（OSI を co-lead、双方向交換を重点）[5]、Databricks（“OSI ready” と表明）[6]

今すぐ本番で使える実装は依然として少ない。 GA は dbt Core 1.12 の 1 件、Snowflake は Open Preview（後述）にとどまる。「参加 50 社」と「実際に動くもの」の落差は大きいが、その落差は成熟度の段階に分けて測る必要がある。

各段階の確認手段と参照文献は 付録 A に整理してある。

6.1.1 リポジトリ内のコンバータ 8 本は実装ではない

仕様リポジトリの `converters/` には 8 本のコンバータがある。

`dbt` / `gooddata` / `honeydew` / `omni` / `orionbelt` / `polaris` / `salesforce` / `snowflake`

これを「8 社が実装済み」と読むのは誤りである。

- いずれも**未リリース**。PyPI に公開されていないものもある
- **全 8 本が `0.2.0.dev0` を対象**としており、出荷版 dbt が受理しないバージョンである
- そもそも**公式コンバータの出力が公式バリデータを通らない**ものがある（セクション 7.3）

`orionbelt` は BlackRock のものと誤解されやすいが、**独立した開発者 Ralf Becher による OBML（OrionBelt Markup Language）**である[7]。`polaris` は Apache Polaris（Iceberg カタログ）を指す。

6.1.2 主要ベンダーの状況

主要ベンダーを OSI 対応の観点で個別に見ると、出荷段階に差がある。

- **Snowflake**: semantic view を OSI YAML として書き出す／OSI YAML から semantic view を作る機能を **Open Preview** で提供している（2026 年 6 月導入）。公式の preview features に「Open Semantic Interchange (OSI) YAML format for semantic views」として掲載され、関数 `SYSTEM$CREATE_SEMANTIC_VIEW_FROM_OSI_YAML` が挙がっている[8]。GA ではなく Open Preview である点に注意
- **Salesforce / Tableau**: OSI を co-lead し、「metrics・dimensions・hierarchies・relationships の双方向のメタデータ交換」を初期の重点に挙げている[9]。ただし記述は将来形・方針表明が中心で、**出荷済みの製品機能としては確認できなかった**（該当ヘルプページの本文は取得できず）。Ossie リポジトリには双方向変換器があるが、jar は `0.1.0-SNAPSHOT` で、これ自体は製品機能ではない
- **Databricks**: 製品ドキュメント（metric views と YAML リファレンス）に OSI import/export の手順は確認できなかった[10]。一方、Data + AI Summit 2026 の自社ブログでは metric views を

[7] 正典は ralfbecher/orionbelt-semantic-layer、PyPI パッケージ名は `orionbelt-semantic-layer`。アクセス日 2026-07-19
[8] Snowflake, “Preview features”（“Open Semantic Interchange (OSI) YAML format for semantic views”、Open Preview、2026 年 6 月）、アクセス日 2026-07-21。関数は SQL functions index 参照
[9] Salesforce, “Ending Semantic Drift” ほか。Tableau Semantics の重点として bi-directional metadata exchange を挙げる。Ossie リポジトリの converters/salesforce（jar `0.1.0-SNAPSHOT`）も参照。アクセス日 2026-07-21
[10] metric views / YAML reference、アクセス日 2026-07-19

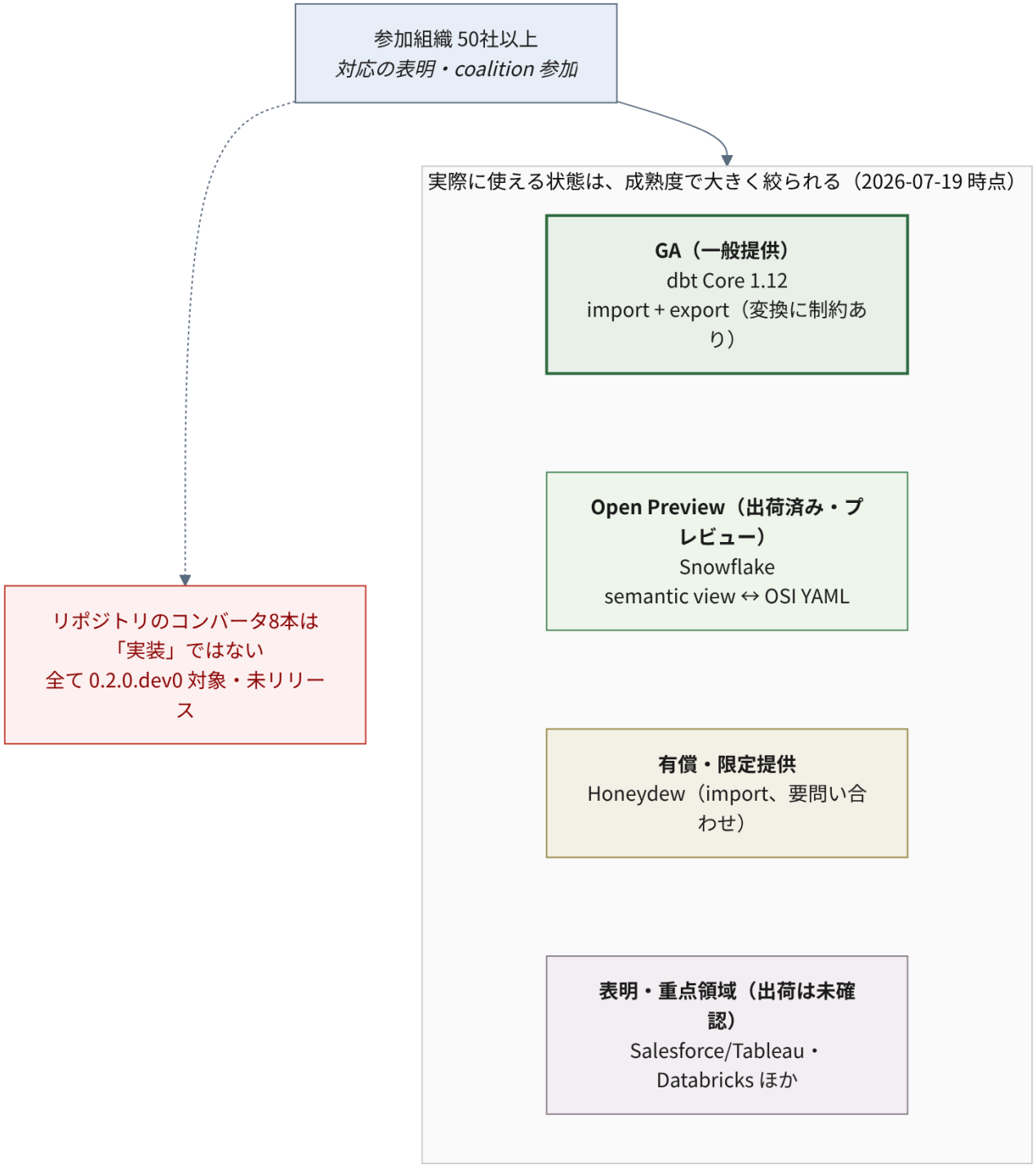

図 6.1: 参加と実装の落差（2026-07-19 時点）

“Open Semantic Interchange (OSI) ready” と述べている[11]。何をもって ready とするかは示されておらず、**互換機能の範囲と利用手順は未文書化**である
- **Dremio**: Dremio Cloud の changelog（2025-11-12 〜 2026-07-14 の範囲）に該当なし。製品側の記載は確認できず、自社ブログで Ossie に言及するにとどまる[12]
- **Cube**: ブログで「オープンアダプタを提供する」と表明してから約 10 か月経つが、製品ドキュメントにも Ossie リポジトリにも成果物が見当たらない[13]

> **「確認できなかった」は不在の証明であって存在しないことの証明ではない**
>
> 上記の否定的な判定（Dremio・Cube・Databricks の import/export 未文書化など）は、調べた範囲での不在であって、本質的に弱い。参照文献と調査範囲は付録 A に列挙した。公開情報でも、プレビュー機能などは掲載場所によって見つけにくいことがある。NDA 下のプライベートプレビューは公開情報に現れない。

6.2 カタログ製品 3 社の状況

Atlan・Alation・Select Star はいずれもローンチパートナーとして名を連ねているが、**いずれも動く実装を確認できなかった**。

この 3 社についてはドキュメント全体を機械的に走査したため、前節の 5 社より否定の確度が高い（走査対象と規模は付録 A）。検索語 `osi` / `ossie` / `open semantic interchange` に対するヒットは 3 社とも実質 0 件で、`apache/ossie` のコミット履歴にも 3 社のドメインからの貢献は存在しない。

6.2.1 出荷されている機能は Snowflake 固有形式に対応するもの

3 社が現に提供している機能を見ると、対応先が揃っている。

- **Atlan**: Context Engineering Studio（Private Preview）。独自形式を各ターゲットへ射影する設計で、Snowflake へのデプロイ先は **Snowflake Semantic View** である[14]
- **Alation**:「Data Products App が Snowflake semantic views と双方向連携」「Extract Semantic Views from Snowflake (Beta)」[15]
- **Select Star**: セマンティックモデル YAML は **「Snowflake Cortex Analyst specification」準拠**と明記[16]

3 社とも、接続先は Snowflake の semantic view または Cortex Analyst であり、ベンダー中立の交換形式を介してはいない。

これは各社の姿勢というより、**市場の順序として理解できる**。Snowflake は semantic view を実際に出荷しており、そこに接続すれば顧客の需要に今すぐ応えられる。一方 Ossie はチャプター 2 で見たと

[11] Databricks, “What’s new with Unity Catalog at Data + AI Summit 2026”, アクセス日 2026-07-20。原文は “Metrics is also open: it’s open source, available in Apache Spark and Unity Catalog OSS, and its Open Semantic Interchange (OSI) ready.”。和訳は筆者による
[12] Dremio Cloud changelog / Dremio ブログ、アクセス日 2026-07-19
[13] Cube: semantic layer sync、アクセス日 2026-07-19
[14] Atlan, “Deploy to Snowflake（Context Engineering Studio）”, アクセス日 2026-07-19
[15] Alation の 2026 年リリースノート（`docs.alation.com/en/latest/releases/releasenotes/`）より。走査の範囲は付録 A
[16] Select Star, “Semantic Models” のドキュメント全文より。走査の範囲は 付録 A

おりリリース版すら固まっておらず、対応先として選びにくい。すぐ使える実装が乏しい理由も同じところにある。

ただし「市場は Snowflake 固有形式への個別対応だけ」と一般化はできない。反対材料として、Databricks は自社ブログで Atlan・Collibra・Monte Carlo・Domo・Hex・Omni などとの連携（出荷済み・計画中を含む）を挙げている[17]。Snowflake を軸にした個別対応が目立つのは確かだが、Databricks 側のセマンティック層を軸にした連携群も並行して存在する。

そのうえで言えるのは、**Ossie が解こうとしている「ベンダー中立の交換形式を介した相互運用」は、現場ではまだ主流になっておらず、各ベンダー固有形式への個別対応が先行している**ということである。

6.2.2 発信の時制には幅がある

各社の発信を並べると、OSI をどの時点の話として語るかに違いがある。

Select Star は自社ブログで将来形を使う[18]。

> so they can travel across tools **as OSI matures**
>
> （OSI が成熟するにつれて、それらがツール間を行き来できるように）

Atlan の解説記事は現在形である[19]。

> This portability is backed by the Open Semantic Interchange (OSI) standard
>
> （この可搬性は Open Semantic Interchange (OSI) 標準に支えられている）

Databricks の Summit ブログの “OSI ready” も、準備状態を指すのか出荷済みの機能を指すのか、文面からは判別できない。

標準への賛同・準備・実装は、それぞれ別の段階である。発信の場（解説記事・イベント登壇・製品ドキュメント）によってどの段階を語っているかが変わるため、読み分けが要る。

> **i** この分野の情報を読むときの実務的な指針
>
> 導入判断のために「今それが使えるか」を知りたいときは、**製品ドキュメントとリリースノートを見るのが確実**である。プレスリリース・パートナー一覧・解説記事は、方向性の表明としては有用だが、出荷状況の証拠にはならない。
> 本章の否定的な判断は、製品ドキュメントの探索と `apache/ossie` のコミット履歴という 2 経路で確認している。ただしこれは不在の証明であり、本質的に弱い。調査範囲と限界は付録 A に明示した。

6.3 Superset の SIP-182 は OSI 対応ではない

Apache Superset の SIP-182「Semantic Layer Support」は 2026-06-22 に completed としてクローズし、約 26 本の PR が master にマージされている[20]。

ただし 2 点、注意が要る（確認手段はセクション A.2.3）。

[17] Databricks, “Redefining Semantics” の partner ecosystem 記載より。アクセス日 2026-07-21
[18] Select Star, “Snowflake AI-Ready Semantic Model”, アクセス日 2026-07-19。和訳は筆者による
[19] Atlan, “Context Catalog”, アクセス日 2026-07-19。製品ドキュメントではなく解説記事。和訳は筆者による
[20] apache/superset, “[SIP-182] Semantic Layer Support in Apache Superset”, アクセス日 2026-07-19

- **未リリース**。`superset/semantic_layers/` は 6.2 リリースブランチに存在せず、master 上のフラグは`"SEMANTIC_LAYERS": False`（`# @lifecycle: development`）
- **OSI 対応ではない**。`apache/superset` を検索しても `ossie` / `open semantic interchange` は 0 件で、`apache/ossie` 側にも Superset コンバータはない。SIP-182 はセマンティック層一般の基盤であって、OSI の実装ではない

Preset は「Preset と Superset の双方で OSI 形式の export / import をできるようにする作業を進めている」と表明しているが、未来形で日付もない。

6.4 意味をどこに置くか — Unity Catalog との対比

セマンティック層をどこに実装するかについて、2 つの異なる賭け方が並走している。一方は自社カタログの内側に作り込み、他方は中立仕様を格納する器として作る。前者は既に GA し、後者はまだ動いていない。

Databricks の Unity Catalog Business Semantics は **2026-04-02 に GA** している[21]。ただし GA は中核機能（metric views）についてで、materialization や UI authoring など 一部は Preview である。「GA」を全機能一律と読まないほうがよい。

- 中核は「metric views」
- metric view の実装を Apache Spark OSS へオープンソース化中（SPARK-54119）
- Unity Catalog OSS にも `METRIC_VIEW` テーブル型を追加する PR がマージ済み[22]。ただし**マージはコード上の進展であって、配布リリースへの収録とは別**。4/2 発表時点では OSS への対応を “coming soon” としており、現配布版への収録は未確認

比較にあたって前提を 2 つ置く。Ossie は**ツール間でモデルを受け渡す交換フォーマット**、Unity Catalog Business Semantics は**単一ベンダーのセマンティック層の実装**であり、層が異なる。また Databricks は Ossie の参加企業でもあり、実際に貢献している。競合として並べるのではなく、**設計上の選択が分かれた事例**として見る。

違いは、意味の定義をどこに置くかにある。

	Unity Catalog	Polaris + Ossie
置き場所	metric view を**テーブル型の一種**として既存カタログモデルに組み込む	Ossie 文書を**別エンティティ型**として持つ
状態	**GA**（2026-04-02、中核機能）	エンドポイントは 501、フラグは default false（固定コミット `07be0176` 時点。日次で変わりうる）
仕様の主体	自社仕様で先行出荷、OSI 対応も表明	最初から中立仕様を格納

読み取れるのは**速度と中立性のトレードオフ**である。

[21] Databricks, “Redefining Semantics: The Data Layer for the Future of BI and AI”, アクセス日 2026-07-19
[22] unitycatalog/unitycatalog, “PR #1493: Add METRIC_VIEW table type”, アクセス日 2026-07-19

自社仕様なら設計から出荷まで一社の判断で進められる。中立仕様を待つなら、仕様が固まるまで実装を出せない。Ossie がチャプター 2 で見たとおりリリース版すら確定していない状況では、後者が遅くなるのは構造的に避けがたい。

したがって「標準化陣営が周回遅れ」と読むのは、比較の仕方として適切でない。**中立性と速度のどちらを優先したかの違い**と見るほうが実態に近い。

そのうえで問うべきは、中立性のために払った速度のコストに見合うだけの中立性が実際に得られているかである。判断材料はチャプター 5 と チャプター 7 にある。

6.5 カタログ統合はまだ動かない

Ossie と Apache Polaris の連携は、**API の形が決まっただけ**の段階である。

6.5.1 実装済みのもの

- `apache/ossie` の Polaris コンバータ（PR #94、2026-05-12 マージ）。ただし中身は **Iceberg REST Catalog 経由で物理テーブル定義を吸い上げて Ossie YAML の雛形を作る**もので、**メトリクス定義は往復しない**。実 Polaris への結合テストと往復忠実性の検証は未チェックのままマージされている
- `apache/polaris` の semantic-models **OpenAPI 仕様とエンドポイントの器**（PR #4816、2026-07-01 マージ）

6.5.2 構想・未実装のもの

以下はいずれも 2026-07-19 時点（`apache/polaris` の当時の main）での確認であり、日次で変わりうる。

- Polaris の semantic-models エンドポイントは**全メソッドが `501 Not Implemented`**（`SemanticModelCatalogAdapt` の javadoc に明記）
- フィーチャーフラグ `ENABLE_SEMANTIC_MODELS` は default false
- CRUD 実装 PR #4961 は 2026-07-19 時点で**未マージ**。RBAC 統合もこの PR の中
- **リリース版 Polaris には入っていない**。1.5.0 も 1.6.0（2026-07-09）も `extensions/` は `auth` と `federation` のみ
- ROADMAP の “Standalone semantic service / registry” は設計提案すら見つからない

「Ossie モデルが Polaris から discover できる」はまだ動かない。

6.5.3 設計は理にかなっている

エンドポイントは未実装（後述）だが、設計の考え方には見るべきものがある。

Polaris は Ossie 文書の中身を読まない。`{version, semantic_model}` という JSON 文字列を、そのまま 1 個の値として保管するだけである。中に何のメトリクスが定義されているかも、どんなフィールドがあるかも解釈しない。

これは手抜きではなく、意図的な切り離しである。

もし Polaris が Ossie 文書の中身を理解する作りにすると、**Ossie の仕様が変わるたびに Polaris のコードを直してリリースし直す必要が出る。**現状の Ossie は チャプター 2 で見たとおりリリース版すら固まっていないので、これに追随するのは現実的でない。

中身を読まなければ、Ossie が 0.1.1 から 0.2.0 へ、さらにその先へ進んでも、Polaris 側は何も変えずに保管し続けられる。**仕様の進化速度とカタログのリリースサイクルを独立させる**ための判断である。

同じ発想は Iceberg にもある。Iceberg のカタログは、テーブルの詳細なメタデータを自分で持たず、`metadata.json` への**ポインタだけ**を管理する。実体の解釈はカタログの外に委ねる。Polaris の Ossie の扱いはその延長線上にある。

代償もある。中身を読まない以上、**カタログ側で「このメトリクスを含むモデルを探す」といった検索はできない**。保管と受け渡しに徹する設計になる。

なお両者の接点は Ossie の `datasets[].source`（`db.schema.table` 形式）ただ 1 つで、参照は Ossie → 物理テーブルの片方向である。Polaris が Ossie 文書を指すことはあっても、逆はない。

semantic-models はパスを `/polaris/v1/...` に分けて Iceberg REST Catalog の名前空間を汚さず、能力広告だけ Iceberg の `Endpoint` / catalog config 機構に乗せている。

一方で ROADMAP が言う “versioning” の実態は限定的である。Polaris の `EntityVersion` は**楽観的並行制御用の不透明文字列**であり、履歴取得 API はない。監査用途には答えられない。

6.5.4 人的なつながり

Ossie の Champion である JB Onofré は Polaris の中心人物でもあり、Mentor の Russell Spitzer は Iceberg の中心人物である。カタログ統合が優先課題に上がっている背景には、この人的配置がある。

6.5.5 他のカタログ製品

Apache Gravitino については、OSI / セマンティック層の実装を確認できなかった。リポジトリの Issue / PR を OSI 関連の語で検索した結果はゼロである。「Gravitino 上に OSI モデルのレジストリを作る」という趣旨のベンダー記述を検索結果で見かけたが、一次情報で裏が取れなかった。

Unity Catalog については セクション 6.4 で扱う。

7 動作確認

ここまでは文書の読解である。本章は実際に動かした結果を扱う。

本章の観測はすべて再現可能である。検証スクリプトと実行環境は ossie-playground のルートにあり、`make verify` で追試できる（付録 B）。実行結果の生データはリポジトリの `docs/results/track_a.json` / `track_b.json` に出力される。検証項目ごとの主張と反証条件は本章で順に示す。

固定の粒度には強弱がある。仕様リポジトリはコミット（`07be0176`）で固定し、消費側の `dbt-core` / `dbt-duckdb` は `==` で厳密に固定している。一方でベースイメージ（`python:3.12-slim`）はダイジェスト固定しておらず、一部の依存はバージョン下限（`>=`）指定で、完全なロックファイルは用意していない。したがって再ビルドの時期によっては下位依存の版が動きうる。この固定の限界と、正確な指定（`docker/Dockerfile`・`docker/requirements.txt`・`Makefile` の `OSSIE_COMMIT`）は 付録 B に記す。

7.1 検証の設計

7.1.1 なぜ 2 トラックに分けたか

チャプター 2 で述べたとおり、0.1.1 と 0.2.0.dev0 は**揃っているツールチェーンが異なる**。同じ手順を両方に適用することはできない。

	0.1.1	0.2.0.dev0
公式スキーマ	あり（タグ `osi-0.1.1-rc1`）	あり（`main`）
公式サンプル	**なし**	2 本
`converters/` 8 本	対象外	**全 8 本がこちら**
出荷版 dbt Core 1.12	**受理**	拒否

そこで 2 つの経路をそれぞれ別に検証した。

トラック A（0.1.1） --- 手書きの準拠文書 → 公式バリデータ → dbt 取り込み → `dbt run`。本検証の 2 トラックの中で、端から端まで通せた唯一の経路である（他のコンバータや `osi-to-msi` 方向、dbt Fusion 等は検証していない）。公式サンプルが存在しないため入力は自作せざるを得ない。

トラック B（0.2.0.dev0） --- 公式コンバータで dbt から生成 →公式バリデータ → dbt 取り込み。生成側の経路。

7.1.2 運用ルール

検証計画を先に固定し、各項目について**主張・手順・裏付けとなる観測・反証となる観測**を事前に書いた。

特に重要なルールが 1 つある。**投入する文書はすべて公式バリデータを通してから実装に食わせる。**

仕様に適合しない文書を実装に読ませると、実装がそれをどう扱おうと「実装の性質」としては解釈できない。仕様準拠を先に保証しておかないと、観測されたものが仕様由来か実装由来かを切り分けられなくなる。

7.1.3 環境

項目	値
仕様	`apache/ossie` @ `07be0176` / 比較タグ `osi-0.1.1-rc1`
消費側	dbt Core 1.12.0 / dbt-duckdb 1.10.1 / metricflow 0.211.0
実行環境	`python:3.12-slim` コンテナ

消費側の版は、`dbt-core==1.12.0` を起点に依存範囲から解決された結果であり、`metricflow` 0.211.0 もその解決で入った版である（明示的に固定して同梱したものではない）。

すべて Docker 上で動かし、ホストには何も入れていない。必要なのは docker だけである。再現手順は付録 B。

> **i** 本章の観測はスナップショット時点のもの
>
> 本章の結果は、いずれも固定コミット `07be0176`（2026-07-17）に対するものである。その後の 2026-07-20、`main` には複数のコミット（`0.2.0.dev0` のスナップショットファイル追加、Snowflake のクォート付き識別子対応 #233 ほか）が入っている。これらが後述の往復問題（セクション 7.3）を解消したかどうかは**未確認**であり、本章の観測には反映していない。

> **i** 検証中は外部ネットワークから切り離している
>
> 検証を走らせるコンテナは、Docker の設定（`network_mode: none`）で**外部と通信できない状態にしてある**。実行中にパッケージを取りに行ったり、仕様リポジトリを引き直したりできない。
> 理由は結果の再現性である。実行のたびに外部から何かを取得していると、**上流が変わった瞬間に結果が変わり、しかもそれに気づけない**。通信を断てば、結果を左右するのは固定したコミット（`07be0176`）とコンテナイメージだけになる。
> 外部を使うのは事前準備の一度だけで、仕様リポジトリを固定コミットで取得する `make spec` がそれにあたる。以降の `make verify` は取得済みのものだけを見る。
> 副次的な効果として、dbt が起動時に最新版を確認しようとして失敗する（`The latest version of dbt-core could not be determined!`）。これは切り離しが効いていることの確認にもなる。

7.2 結果の概要

検証は 6 項目。各項目は「こうなっているはずだ」という主張を先に立て、それが観測で裏付けられるか、あるいは覆されるかを見る形で設計した。

項目	何を確かめたか	結果
V1	構造が同じ文書でも、バージョン文字列の違いだけで相互に受け付けられなくなるか	裏付けられた。加えて別の要因も判明
V2	実装は、公式スキーマが禁じている書き方を受け入れてしまうか	裏付けられた。警告も出ない
V3	型情報を持たないディメンションを、実装はどう解釈するか	数値と文字列が区別されなかった
V4	メトリクスの集約式は、実装のモデルへどう写像されるか	集約関数が脱落。**実行しての確認はできなかった**
V5	仕様に書かれていない前提が、取り込みに必要になるか	3 件必要だった
V6	公式コンバータの出力を、出荷版の実装は受け取れるか	受け取れない。**想定していなかった要因が 2 つ**あった

全体として、**主張はおおむね裏付けられた**。ただし V4 は実行環境の制約で 最後まで確認できず、V6 は事前に想定していたのと別の理由でも失敗した。以下、順に見ていく。

7.3 本検証構成では dbt を挟んだ往復が成立しない

本章で最も紙幅を割く結果である。

仕様リポジトリ同梱の公式コンバータ（`ossie-dbt msi-to-osi`）で dbt プロジェクトから OSI 文書を生成し、同じ dbt に戻せるかを試した。以下は固定コミット `07be0176`・dbt-core 1.12.0・本検証の入力に対する観測であり、製品やアダプタ、現行 `main` を横断する一般命題ではない。

この構成では、少なくとも 3 つの要因が重なって往復が成立しなかった。以下の 3 つはそれぞれ別の層で起きており、いずれか 1 つを直しても残りで止まる。

7.3.1 破綻 1: 公式コンバータの出力が、同一コミットの公式スキーマで落ちる

```
[Schema] (root): Additional properties are not allowed ('dialects' was unexpected)
```

コンバータはルート直下に `dialects: ["ANSI_SQL"]` を出力する。これは **0.1.1 のルート構造**であり、0.2.0.dev0 では削除されている（セクション 4.6）。

にもかかわらずコンバータは `version: "0.2.0.dev0"` を出力する（converters/dbt/src/ossie_dbt/msi_to_osi.py L124 のハードコード。tests/test_msi_to_osi.py でもこの値がアサートされている）。

つまりコンバータの出力は、**ルート構造は 0.1.1 のまま、version 文字列だけ 0.2.0.dev0** という組み合わせになっている。この文書は、同一コミットの公式バリデータ（0.2.0.dev0 スキーマ）を通らない。

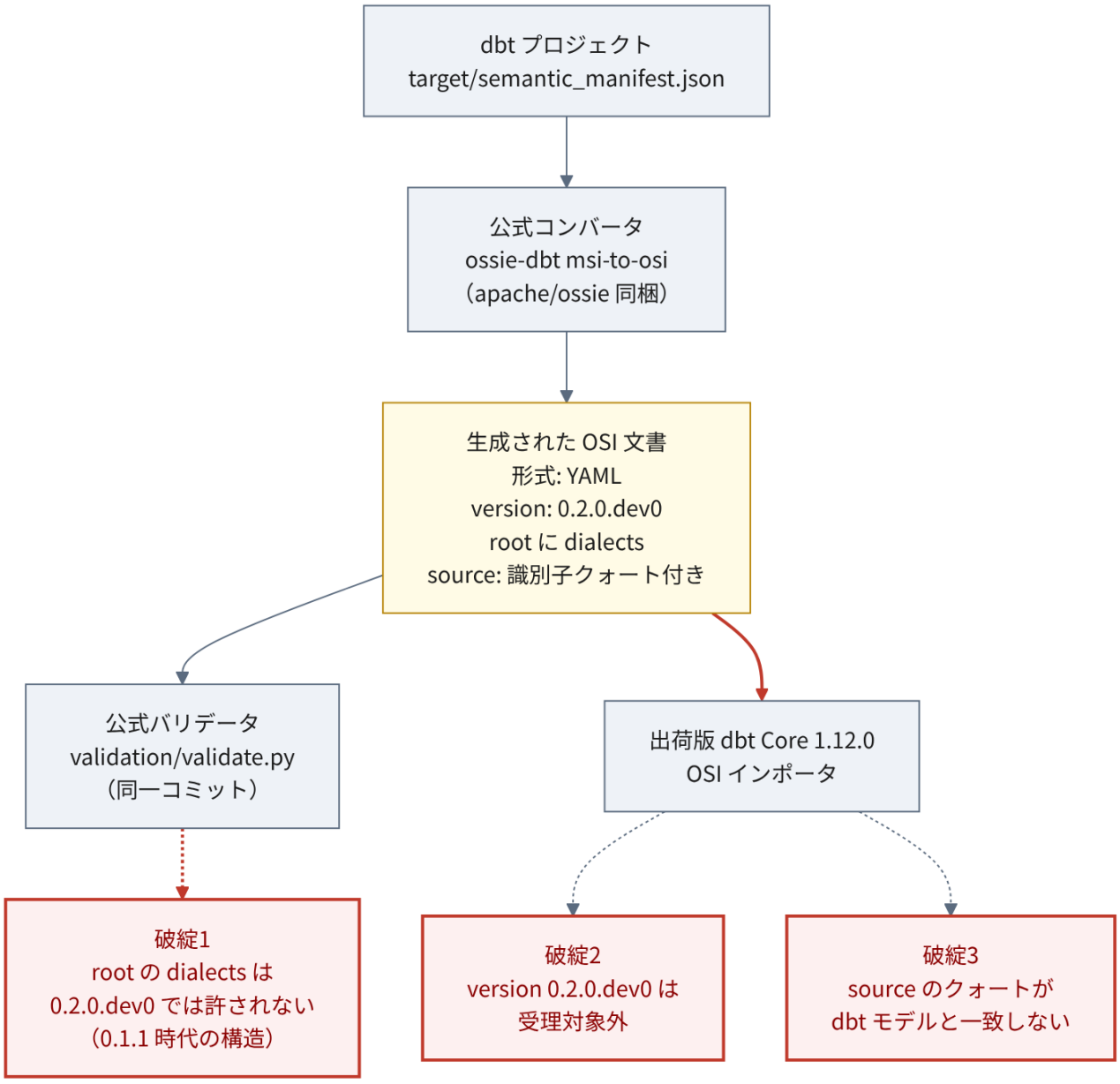
dbt プロジェクト
target/semantic_manifest.json
公式コンバータ
ossie-dbt msi-to-osi
（apache/ossie 同梱）
生成された OSI 文書
形式: YAML
version: 0.2.0.dev0
root に dialects
source: 識別子クォート付き
公式バリデータ
validation/validate.py
（同一コミット）
出荷版 dbt Core 1.12.0
OSI インポータ
破綻1
root の dialects は
0.2.0.dev0 では許されない
（0.1.1 時代の構造）
破綻2
version 0.2.0.dev0 は
受理対象外
破綻3
source のクォートが
dbt モデルと一致しない

図 7.1: 往復の破綻

7.3.2 破綻 2: バージョン文字列で拒否される

```
OSI file '...' uses unsupported version '0.2.0.dev0'.
Supported versions: ['0.1.0', '0.1.1']
```

dbt 側の受理集合は、インストールされた `dbt/constants.py` に `SUPPORTED_OSI_VERSIONS = frozenset({"0.1.0", "0.1.1"})` として定義されている（dbt-core 1.12.0。検証環境内で確認）。この制約は dbt のドキュメントにも “OSI documents must use version `0.1.0` or `0.1.1`”（OSI 文書はバージョン 0.1.0 または 0.1.1 を用いなければならない）と明記されている。

7.3.3 破綻 3: source の形式が一致しない

バージョンを `0.1.1` に書き換えると 0.1.1 スキーマは通過する。しかし dbt はなお拒否する。

```
contains dataset 'orders' ("ossie_track_a"."main"."orders") that does not match
any dbt model in this project.
```

コンバータは source を`'"db"."schema"."table"'` と**識別子をクォートして**出力する。dbt の突き合わせはクォートなしを期待する。手書きの `ossie_track_a.main.orders` は問題なく通った。

仕様は source を「`database.schema.table` 形式または query」としか定めておらず、**クォートの扱いを規定していない**（チャプター 4）。

エラーメッセージの `("ossie_track_a"."main"."orders")` という表記と、クォートなしの手書き文書が通ったことから、**クォートの有無が拒否の要因である可能性が高い**。ただし本検証はクォート以外の差分を厳密に統制したうえでの一点比較ではないため、これは原因の確定ではなく有力な観測として扱う。

7.3.4 帰結

この構成では、dbt → OSI → dbt の往復が成立しなかった。3 つの要因はそれぞれ別の層で起きており、どれか 1 つを直しても残りで止まる。ここで言えるのはこの検証構成についてであり、製品・アダプタ・現行 `main` を 横断する一般命題ではない。

> **つまずきの性質**
>
> 3 つの原因はいずれも、バージョン管理・出力形式・識別子の表記といった**取り決めの層にある**。メトリクスの意味をどう表現するかという設計上の難問とは別の種類の問題である。
> こうした食い違いは、仕様と実装が並行して動いている時期には起こりやすい。バージョンを進める、スキーマを整理する、コンバータを直すという作業がそれぞれ別のタイミングで行われれば、その間に齟齬が生じる。
> 裏を返せば、**取り決めを揃えれば解消しうる**種類でもある。

7.4 取り込み時に情報が落ちる

手書きの 0.1.1 準拠文書は、公式バリデータを通り、dbt に取り込まれ、`dbt run` まで到達した。**経路がまったく動かないわけではない。**

ただし取り込みの過程で情報が落ちる。

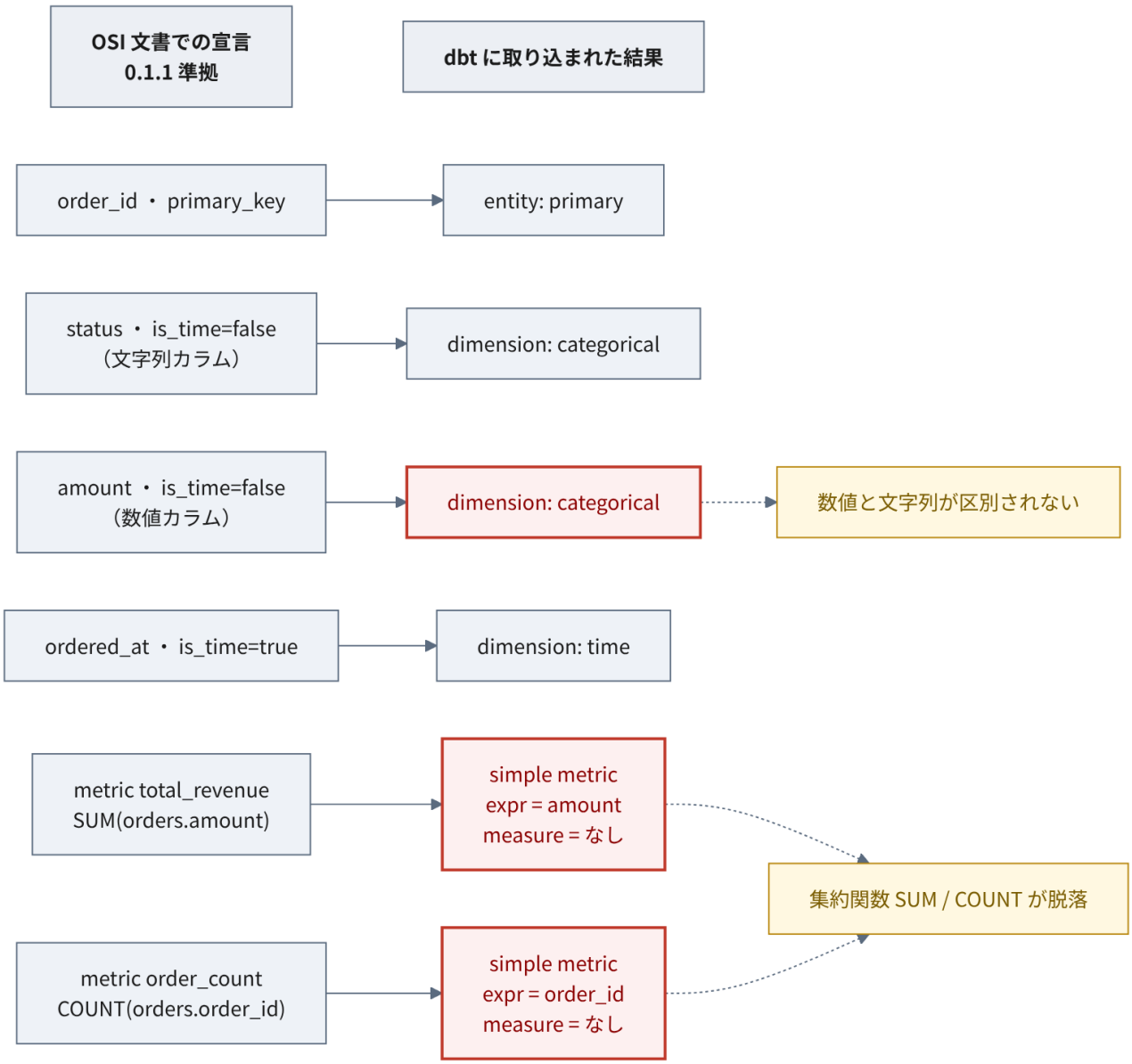

図 7.2: 取り込み時の情報欠落

7.4.1 型情報が運ばれない

OSI の `dimension` は `is_time`（時間軸かどうか）という真偽値しか持たない（チャプター 4）。数値なのか文字列なのかを示す欄がない。

取り込ませた結果はこうなった。

OSI 側の宣言	dbt での分類
`order_id`（primary_key）	entity (primary)
`status`（文字列カラム、is_time=false）	dimension: **categorical**
`amount`（数値カラム、is_time=false）	dimension: **categorical**
`ordered_at`（is_time=true）	dimension: time

数値の `amount` と文字列の `status` が、同じ categorical になっている。

> categorical とは何か、なぜ問題か
>
> セマンティック層では、列を大きく 2 つの役割に分ける。
>
> - **ディメンション**: 集計を「切る」ための軸。ステータス別、月別、地域別
> - **メジャー**: 集計「される」数量。売上金額、件数
>
> `categorical`（カテゴリカル）とは、**取りうる値の種類で分類するタイプのディメンション**を指す。「shipped / pending / cancelled」のようなラベルがこれにあたる。
> 金額である `amount` が categorical と解釈されると、**「金額の値ごとに行を切り分ける軸」として扱われる**ことになる。100.50 円のグループ、250.00 円のグループ、という具合である。本来は合計したい数量なので、意図とずれる。
> dbt は `source` を辿れば物理テーブルの列型を見に行けるが、**そこまではしていない**。OSI 文書に書かれた情報だけで判断している。

型情報を運ぶ欄が仕様にない以上、消費側が物理テーブルを見に行くか、名前から推測するしかない。**どちらを期待するかも仕様は定めていない。**

7.4.2 集約関数が脱落する

OSI 文書に 2 つのメトリクスを書いた。売上の合計と、受注の件数である。

```yaml
metrics:
  - name: total_revenue
    expression:
      dialects: [{ dialect: ANSI_SQL, expression: "SUM(orders.amount)" }]
  - name: order_count
    expression:
      dialects: [{ dialect: ANSI_SQL, expression: "COUNT(orders.order_id)" }]
```

取り込んだ結果がこれである。

OSI 側の宣言	dbt 側の結果
`SUM(orders.amount)`	type=simple, expr=`amount`, measure=None, input_measures=[]

OSI 側の宣言	dbt 側の結果
`COUNT(orders.order_id)`	type=simple, expr=`order_id`, measure=None, input_measures=[]

`SUM` と `COUNT` が消えている。残ったのは対象の列名（`amount` / `order_id`）だけで、「合計する」「数える」という集約の指示が失われた。2 つのメトリクスは対象列こそ違うものの、いずれも `type=simple` で集約関数を持たない、同じ骨格になっている。これは取り込み後の manifest 上での観測であり、実行時にどう出るかは後述のとおり未確認である。

> **なぜこうなるのか**
>
> セクション 4.2 で見たとおり、Ossie と dbt/MetricFlow では 集約の置き場所が違う。
> **dbt/MetricFlow の考え方**は、まずテーブルの中に「amount を合計したもの」という名前付きの部品（measure）を作り、メトリクスはその部品を指す、というものである。メトリクスの定義は「どの部品を使うか」だけを持ち、どう集約するかは部品の側が知っている。
> **Ossie の考え方**は、部品を作らず、メトリクスの式に直接 `SUM(...)` と書く。
> そのため OSI 文書を dbt に取り込むと、メトリクスが指すべき「部品」が存在しない。`measure=None`、`input_measures=[]` はその状態を表している。式から `SUM` を取り出して部品を組み立てる処理は行われず、中身の列名だけが `expr` に残った、と読める。

これは仕様の不備というより、**2 つの設計が噛み合っていない**ことの現れである。どちらの置き方が良いかは セクション 4.8 で議論が続いており、結論は出ていない。

7.4.3 実行検証はできなかった

計画では「取り込んだメトリクスが実際に計算できるか」まで確認するはずだった。**出荷版ツールチェーンでは実行できない。**

- `dbt-core` 1.12.0 単体にメトリクスをクエリするコマンドはない
- 依存として入った `metricflow` 0.211.0 はライブラリのみでコンソールスクリプトを持たない
- クエリ用の `dbt-metricflow`（最新 0.13.0）は **`dbt-core<1.12.0,>=1.10.4`** を要求し、1.12.0 を明示的に除外している（PyPI のメタデータ、アクセス日 2026-07-19）

OSI インポートが載ったバージョンでは、その結果をローカルでクエリできない。

したがって集約関数の脱落が実行時にどう出るかは未確認であり、マニフェスト上の観測にとどめる。

7.5 実装が公式スキーマより緩く受ける場合がある

`Field` は `additionalProperties: false` である。**`Field` に仕様外のキーを 1 つ足した**文書を用意すると、次のようになった。

検証者	結果
公式バリデータ（0.1.1）	拒否
dbt Core 1.12.0	**受理**。この追加キーについて警告は出なかった

寛容なパースは設計判断としてありうる。ただしこの観測は、**検証した「Field への追加キー」という一点**についてのものである。

dbt は、未対応の要素を落とす際に警告（`IO78`）を出す仕組みを持つ。今回の追加キーではその警告が出なかった、というのがここでの観測であり、「dbt はいかなる仕様違反も無警告で受ける」という一般化はできない。少なくともこの追加キーに関しては、利用者が仕様違反に気づく手がかりが得られなかった。

この項目の主張と反証条件も、他と同様に事前に定めてある。

7.6 仕様が定めていない前提

OSI 文書を dbt に取り込むには、仕様に記述のない要件が 3 つ必要だった。

1. **`.json` のみ走査される。** dbt は指定された `osi-paths`（既定では `OSI/`）配下を `rglob("*.json")` で探す（dbt-core 1.12.0 の `dbt/parser/osi.py`。検証環境内で確認）。`OSI/` に YAML を置いても無視され、parse は成功する（観測済み）。複数パスの指定はできるが、依存パッケージ内に置かれた OSI 文書は走査対象にならない。一方、本検証で用いた**仕様の公式サンプルは YAML であり、固定コミットのコンバータを CLI で実行した出力も YAML だった**（コンバータが YAML のみを出すのか、他形式が既定なのかまでは未確認）
2. **time spine モデルが要る。** メトリクスを含む文書を取り込むと、MetricFlow が granularity DAY 以下の time spine モデルを要求する。OSI 仕様に該当する概念はない
3. **source が dbt 管理下のモデルと一致する必要がある。**（`database`, `schema`, `alias`）で一致しなければならず、外部テーブルは参照できない

いずれも、**OSI に適合しているだけでは dbt での取り込みは保証されない**ことを示している。仕様の外にある消費側の事情を満たさなければ通らない。

7.7 この検証で確かめられなかったこと

- メトリクス実行時の挙動（ツールチェーン上不能）
- `0.1.0` を消費側が受理する一方、それに対応する公式スキーマ成果物をどこで検証できるのかが判然としない。dbt は `0.1.0` を受理集合に含めるが、仕様リポジトリの Version History には `0.1.0` の記載が見当たらない。「消費側が受理を表明する版を、どの公式成果物で検証できるか」という相互運用上の論点として残る
- 他の 7 つのコンバータの出力が同様の問題を持つか（`dbt` のみ検証）
- `osi-to-msi` 方向。dbt 同梱の `metricflow.converters.osi_to_msi` と リポジトリの `ossie_dbt.osi_to_msi` の関係も未確認

8 評価と展望

本章は前章までの観察を総合したものであり、新規の事実提示はほとんど行わない。各主張の根拠は、参照先として示した章と、そこから辿れる出典にある。

8.1 現在地

掲げられている「Write Once, Query Anywhere」は、まだ実現していない。

動作確認では、同じ dbt を挟んだ往復------公式エクスポータの出力を出荷版インポータが読む------が成立しなかった。しかも原因が 3 つ独立に立つため、どれか 1 つを直しても通らない（セクション 7.3）。

一方で、リリース版に準拠して手で書いた文書は、公式バリデータを通り、dbt に取り込まれ、実行まで到達している。**経路そのものは存在する。**

観測されたつまずきは性質が異なる 3 種類に分かれ、それぞれ解けやすさが違う。

8.1.1 (1) 運用規約の未整備

- バージョン文字列を `const` で固定した結果、構造的にほぼ同じ文書が相互に排除される
- 公式コンバータが 0.1.1 時代の構造に 0.2.0.dev0 のラベルを貼っている
- `source` の識別子クォートの扱いを仕様が定めていない

これらは**仕様の設計を変えずに解消できる**種類である。バージョンの進め方、コンバータの追随、識別子表記の規定といった取り決めを揃えれば片付く。

8.1.2 (2) 仕様そのものの設計上の制約

一方、次の 2 つは規約では解けない。

- **`dimension` が `is_time` しか持たず、型情報を運ばない。**数値と文字列が区別できない（セクション 7.4）
- **フィールドレベルの measure を持たない設計が、消費側のモデルと噛み合わない。**OSI は集約を `metrics` の式として持つが、MetricFlow は measure への参照を前提とする。結果として集約関数が写像で脱落する

これらは仕様を変更しなければ解決せず、しかも**どう変更すべきかが難しい**。

型情報を運ぶ欄を足すのは一見簡単に見えるが、どこまで型を持たせるかは設計判断である。物理層の型をそのまま写すのか、論理的な型を別に定めるのか、消費側に推論させるのか。セクション 4.8 で見た「構造化するか、消費側に委ねるか」という軸が、ここでも効いてくる。

metrics と measures の置き場所はさらに難しい。既存の主要な実装が揃って一方を採っているなかで、Ossie が別の設計を選んでいる。どちらにも理由があり、ワーキンググループでも結論が出ていない。

8.1.3 (3) エコシステムの薄さ

製品として動く実装が 2 件のみで、うち 1 件は有償・要問い合わせ。取り込んだメトリクスをローカルでクエリする手段さえない。

これは (1)(2) の結果でもあり、原因でもある。実装が増えなければ規約の不備も設計の噛み合わなさも表面化せず、修正されない。

8.1.4 見通し

(1) は取り決めの問題なので、時間と手当てで解ける見込みがある。

(2) は設計判断を要する。**技術的にも難しい問題**であり、時間をかければ自然に解けるものではない。そして難しい設計判断ほど、それを決める場が機能しているかどうかに左右される。

8.2 運営の側から見ると

チャプター 5 の観察と重ねると、もう一つの面が見えてくる。

運営の仕組みは、まだ立ち上がっている途中である。

- ASF 移管から 4 週間。Apache Release はまだなく、署名鍵も未登録
- 仕様変更の [VOTE] は実行された記録がない
- 仕様の議論は GitHub Discussions で行われ、中核的な意味論の一部は Google Docs と Slack にある
- ML のアーカイブにも不備がある（プロジェクト側も認識している）

これらは、**移管から日が浅いことを踏まえて読む必要がある**。ASF のガバナンスに沿った運営が定着するには通常それなりの期間を要し、incubation 初期の podling が同様の状態にあることは珍しくない。mentor から是正が入る典型的な段階でもある。

そのうえで、チャプター 2 で見た「ワーキンググループ一覧が公式内で 3 系統に食い違う」という現象や、バージョン管理の食い違いは、**何を正とするかが定まりきっていない**ことの表れと見える。

0.1.1 と 0.2.0.dev0 のどちらを基準にするのか、コンバータはどちらに追随するのか、公式サンプルはどちらで書かれるべきか。これらは技術的な難問ではなく、決めれば済むことである。決める場の整備が追いついていない、という順序で理解するのが妥当だろう。

8.3 導入判断

現時点で本番採用する理由は乏しい。

- 製品として動く実装は 2 件のみ（dbt Core 1.12、Honeydew）
- 仕様のリリース版は 0.1.1 だが、公式サンプルもコンバータもそちらを向いていない
- 取り込み時に型情報と集約関数が落ちる。しかもローカルでクエリ検証する手段がない
- カタログ統合（Polaris）は 501 のまま

一方で、**注視する価値はある**。直近数週間で実質的な変化が続いているためだ。

日付	出来事	参照
2026-06-22	ASF Incubator 入り	チャプター 5
以降 4 週間	merged PR 32 本（それ以前は月 5 本程度）	チャプター 5
2026-07-15	`expression_language.md` が `main` に追加	セクション 4.7
2026-07-16	dbt Core 1.12 で import + export が GA	チャプター 6

> **i** このレポートは古びやすい
>
> 上記の最後の 2 件は、本調査（2026-07-19）の 3〜4 日前の出来事である。これは「動きが速い」ことの証拠というより、**本レポートが短期間で陳腐化しうる**ことを意味する。
> 特に チャプター 7 の結果は、`dbt-metricflow` の対応バージョンが変われば、あるいは公式コンバータの出力が修正されれば、そのまま覆る。

現実的な向き合い方としては、以下が妥当だろう。

- **今すぐ乗るのではなく、自社のセマンティックモデルを Ossie で表現できる形に保っておく**。仕様の構造は素直で、dbt / Cube / LookML などから機械的に写像できる範囲にある
- 特定ベンダーのセマンティック層に固有機能で深く依存しない。依存する場合は `custom_extensions` に落ちる部分がどこかを把握しておく
- セクション 6.4 で見たとおり、中立性を取れば速度は落ちる。今必要な機能があるなら、単一ベンダーの実装を選ぶのは合理的な判断でありうる

8.4 何を見ていれば状況が分かるか

技術・運営の両面が絡むため、片方だけを見ても状況は掴めない。以下は、いずれも公開情報から判定できる指標である。

ガバナンス

1. **初回の Apache Release が出るか。** PGP 署名鍵の登録が前段階になる
2. **仕様変更が ML の [VOTE] を通るようになるか。** 所属に関係なく反対意見が検討され、理由付きの合意が記録されれば、中立な運営の実証になる
3. **Google Docs と Slack にある決定がリポジトリへ戻るか**
4. 8 月以降の月次 podling report の内容（mentor の応答性・リリース計画）
5. 創設企業以外からの committer 昇格

技術・エコシステム

6. **0.1.1 と 0.2.0.dev0 の分裂が解消されるか。** 具体的には、公式サンプルとコンバータがリリース版を向くか、あるいは 0.2.0 が正式リリースされるか
7. **公式コンバータの出力が公式バリデータを通るようになるか**（現状は通らない）
8. `dbt-metricflow` が dbt-core 1.12 系に対応し、取り込んだメトリクスをクエリできるようになるか
9. Polaris の `ENABLE_SEMANTIC_MODELS` が default true になり、リリース版に入るか
10. dbt と Honeydew 以外の 3 例目の実装が出るか

8.5 総括

Ossie が取り組んでいる問題------指標定義の断片化------は実在する。AI エージェントが指標を参照して判断を下すようになれば、定義の不一致は気づかれないまま下流へ伝播する（チャプター 3）。共通の記述形式を持つ価値は上がっている。

2026 年 7 月時点では、標準として依拠できる段階にはない。理由は一つではなく、少なくとも 3 つが重なっている。

技術的な難しさ。型情報をどこまで仕様に持たせるか、集約の定義をどこに置くか、方言差をどう吸収するかは、いずれも設計判断を要する問題である。既存のセマンティック層が別々の答えを出してきたことからも分かるとおり、自明な正解はない。概念的相互運用に至っては、プロジェクト自身が未解決と認めている。

運営の未成熟。何を正とするかを決める場が、まだ機能しきっていない。バージョン管理や公式コンバータの食い違いは、技術的に難しいからではなく、決めて揃える工程が追いついていないために生じている。

エコシステムの薄さ。実装が 2 件にとどまるため、仕様の不備も設計の噛み合わなさも表面化しにくい。実装が増えれば問題は顕在化し、修正の圧力もかかる。

3 つは互いに絡んでいる。技術的に難しい判断ほど合意形成の場を要し、場が整わなければ判断は先送りされ、実装は増えない。

ASF への移管は、このうち運営の側に効く手当てである。方向としては妥当で、受入プロセスに外部のレビューが挟まることも確認できた。ただし移管から 4 週間で成果を判定するのは早い。

判定の材料が揃うのは、8 月以降の月次 podling report と、初回 Apache Release の頃になるだろう。

8.6 本レポートの限界

- 2026-07-19 の一時点のスナップショットである。この分野の動きは速い
- 動作確認の消費側は dbt Core 1.12 のみ。Honeydew は有償・要問い合わせのため入手できず、他の実装は存在しない
- メトリクスの実行時挙動は確認できていない（チャプター 7）
- 否定的な結論は不在の証明であり本質的に弱い。調査範囲は付録 A に明示した
- 筆者は Ossie の開発に関与しておらず、ML や Slack の非公開のやりとりを見ていない

A 調査方法と範囲

本レポートには「対応が確認できなかった」という否定的な主張が複数ある。これは不在の証明であり、本質的に弱い。何をどこまで調べたのかを開示して、読者が確度を判断し追試できるようにする。

A.1 出典の階層

階層	内容	扱い
Tier 1	ASF 公式（incubator.apache.org, lists.apache.org, cwiki, dist.apache.org）、仕様リポジトリ	事実認定の根拠とする
Tier 2	プロジェクト広報（ossie.apache.org）	ASF インフラ上だが PPMC 自身の広報文。主張として扱う
Tier 3	ベンダーブログ、業界メディア	事実関係は Tier 1 で裏を取る。単独では根拠としない

A.2 ベンダーの実装状況をどう調べたか

「参加している」と「実装している」を区別することを最優先に置いた。参加表明・プレスリリース・パートナー一覧への掲載は、いずれも実装の証拠ではない。

判定は次の順序で行った。

1. **製品ドキュメント**に OSI / Ossie の記載があるか
2. 記載がある場合、それが **export / import のどちらか**、**GA か preview か**、**対応する仕様バージョン**は何か
3. **`apache/ossie` への貢献**があるか（コミット作者のメールドメイン、Issue/PR）

A.2.1 網羅的に走査した 3 社

ドキュメント全体を機械的に取得して検索したため、否定の確度が比較的高い。

製品	走査対象	規模	OSI 関連のヒット
Atlan	`docs.atlan.com` サイトマップ全 URL	2,815 URL	0 件（4 件は誤検出）

製品	走査対象	規模	OSI 関連のヒット
Alation	`docs.alation.com/en/latest/searchindex.js`（Sphinx 検索インデックス全体）	3.0 MB	0 件（2 件は “dossier” の部分一致）
Select Star	`docs.selectstar.com/llms-full.txt`（ドキュメント全文）	408 KB	0 件

検索語は `osi` / `ossie` / `open semantic interchange` / `interchange`。Alation は 2026 年のリリースノート 3 本（528 KB / 323 KB / 534 KB）も個別に確認した。Select Star は `llms.txt`、`changelog.md`、`features/semantic-models.md`、および 2.5 MB の過去 changelog も確認した。

参照した主な URL は本節の表と、第 5 章の各脚注に示したとおりである。

A.2.2 関連ページを個別に確認した 5 社

こちらは**網羅的走査を行っていない**。関連が見込まれるページとリリースノートを個別に参照したにとどまるため、否定の確度は上記 3 社より低い。

ベンダー	参照した文献
Snowflake	semantic views overview / feature releases 2026 / Tableau TDS 対応の release note / converters/snowflake
Databricks	metric views / metric views YAML reference
Dremio	Dremio Cloud changelog（2025-11-12 〜 2026-07-14 の範囲）/ self-serve semantic layer / Dremio ブログ
Cube	semantic layer sync
Salesforce / Tableau	Salesforce semantic layer guide / Tableau Semantics / converters/salesforce

いずれもアクセス日は 2026-07-19。

A.2.3 Superset SIP-182 の確認手段

Superset については「マージ済みだが未リリース」「OSI 対応ではない」という 2 つの判断をしている。根拠は以下のとおり。

マージ済みであること

- Issue #35003（SIP-182）が 2026-06-22 に `completed` としてクローズされている
- 関連 PR 約 26 本（`#37815` セマンティック層拡張、`#37816` モデル/DAO、`#37817` Explore 統合、`#38370` API、`#38376` UI、`#40475` ダッシュボードフィルタ、最終は`#42093`、2026-07-15）が `master` にマージ済み

未リリースであること

- superset/semantic_layers/ が **6.2 リリースブランチに存在しない**（GitHub API でパスを問い合わせて 404）
- アプリの最新タグは 6.1.0（2026-05-01）で、その config.py に SEMANTIC_LAYERS フラグが存在しない
- master の superset/config.py では"SEMANTIC_LAYERS": False に# @lifecycle: development の注記が付く
- CHANGELOG/6.1.0.md にセマンティック層関連の項目がない

OSI 対応ではないこと

- apache/superset に対する ossie / open semantic interchange の コード検索がいずれも 0 件
- OSI の語は散文中に 1 度、セマンティック層の一例として名前が挙がるのみ
- apache/ossie の converters/ に Superset 向けのものがなく、Superset に言及する Issue もない

Preset（Superset の商用提供元）の表明

- パートナーページに “We’re working to ensure Preset and Apache Superset™ can both export and import semantic definitions in the OSI format.”（Preset と Apache Superset の双方が OSI 形式でセマンティック定義を export / import できるよう取り組んでいる）とある。未来形で、日付の記載はない
- Preset のセマンティック層ドキュメントは自社のデータセットとメトリクスを説明しており、OSI への言及はない

いずれもアクセス日 2026-07-19。和訳は筆者による。

A.2.4 リポジトリ側の確認

apache/ossie について次を確認した。

- 全コミットの作者メールドメイン（API を 5 ページ分、計 500 件まで取得）
- Issue / PR の全文検索（repo:apache/ossie+<vendor>）
- converters/ 配下のディレクトリ構成と各コンバータが対象とする仕様バージョン
- GitHub Releases、タグ一覧

A.3 否定的結論の限界

以下は本レポートの主張の弱点として明示しておく。

- **NDA 下のプライベートプレビューは公開情報に現れない。**非公開で実装が進んでいる可能性は排除できない
- **ドキュメントに書かれていないだけで機能が存在する**ことはありうる。特に API 経由の未文書化エンドポイント
- 網羅走査した 3 社についても、検索インデックスやサイトマップがドキュメント全体を反映していない可能性がある
- 調査は 2026-07-19 の一時点のスナップショットである。実際、dbt Core 1.12 の GA（2026-07-16）と expression_language.md の追加（2026-07-15）は調査の数日前であり、**もう数日早く調べていれば違う結論になっていた**箇所がある

A.4 再調査時の注意

同種の調査を行う場合、以下の落とし穴に注意されたい。

- **未認証の GitHub API は偽陰性を生む。**`api.github.com/search/code` は 未認証で 401 を返し、org-repos エンドポイントはレート制限にかかる。どちらも「結果ゼロ」に見えるため、認証済みの `gh` コマンドを使うこと
- **SEO 記事と製品ドキュメントは食い違うことがある。**本調査では、ある製品の SEO 記事が現在形で OSI 対応を謳う一方、同社の製品ドキュメントには記載がないという事例に遭遇した（チャプター 6）
- **社名を冠したディレクトリ名は所属を意味しない。**`converters/orionbelt` は BlackRock ではなく独立開発者の OBML、`converters/polaris` はベンダー製品ではなく Apache Polaris である

A.5 動作確認の手順

次の順序で実施した。

1. **仕様を一次読解し、基礎文書として固定する**（チャプター 4）
2. **基礎から検証計画を導く。**各項目について、主張・手順・裏付けとなる観測・**反証となる観測**を事前に定める
3. **計画に沿って動作確認する。**計画外の観測は増やさない
4. 実装に投入する文書は、**必ず公式バリデータを通してから**使う

4 の理由を補足する。仕様に適合しない文書を実装に読ませた場合、実装がそれをどう扱おうと「実装の性質」としては解釈できない。仕様準拠を先に保証しておかないと、観測されたものが仕様由来か実装由来かを切り分けられなくなる。

2 で反証条件を先に定めるのは、結論ありきの検証を避けるためである。各項目の主張・反証条件・結果は第 6 章（チャプター 7）に示し、実行結果の生データはリポジトリの `docs/results/` に出力される。

環境は Docker で固定し、仕様リポジトリはコミットを固定（`07be0176`）、検証コンテナは外部ネットワークから切り離して実行した。再現手順は付録 B にある。

B 動作確認の再現手順

本レポートの チャプター 7 は、このリポジトリのルートにある環境で再現できる。

B.1 必要なもの

docker のみ。 Python も dbt もコンテナ内にある。

B.2 手順

```
git clone https://github.com/dobachi/ossie-playground.git
cd ossie-playground

make spec     # 仕様リポジトリを固定コミットで取得(ここだけネットワークを使う)
make verify   # トラックA・Bを実行
```

個別に実行する場合:

```
make verify-a   # トラックA (0.1.1) のみ
make verify-b   # トラックB (0.2.0.dev0) のみ。先に verify-a が必要
make shell      # コンテナ内でシェルを開く
```

検証結果は docs/results/*.json に出力される。

B.3 再現性の担保

固定の粒度は要素ごとに異なる。強い順に挙げる。

- **仕様リポジトリはコミットを固定している**（Makefile の OSSIE_COMMIT = 07be0176）。上流が変われば結果も変わりうるため、固定を外す場合は結果も取り直すこと
- **消費側の主要パッケージは厳密固定**。docker/requirements.txt で dbt-core==1.12.0 / dbt-duckdb==1.10.1 を== で指定している。本レポートの結論に直結する版はここで固定される
- **検証コンテナは外部ネットワークから切り離してある**（network_mode: none）。実行中に何かを取りに行かせないことで、結果を左右する要素を固定コミットとビルド済みイメージだけに限定する。外部を使うのは make spec の一度だけ

B.3.1 固定していない部分（既知の限界）

再現性は完全ではない。以下は固定していないため、再ビルドの時期によって動きうる。

- **ベースイメージをダイジェスト固定していない。** docker/Dockerfile は FROM python:3.12-slim とタグ指定のみで、@sha256:... を付けていない
- **一部の依存はバージョン下限指定（>=）**。jsonschema / sqlglot / PyYAML / hatchling / jinja2、および dbt-core 経由で入る metricflow などは、ロックファイルで固定していないため、解決される版がビルド時期に依存する

厳密な追試が必要な場合は、上記をダイジェスト固定・ロックファイル化したうえで再取得することを勧める。本レポートの中核的な観測（バージョン文字列による排除、ルート dialects の不整合、集約関数の脱落）は、== 固定した dbt-core 1.12.0 と固定コミットの仕様に依存しており、下位依存の版差では変わりにくい部分である。

B.4 構成

```
docker/           コンテナ定義と依存
compose.yaml      network_mode: none で実行
Makefile          入口
scripts/
  lib_ossie.py        共通処理（バリデータの展開など）
  verify_track_a.py   トラック A
  verify_track_b.py   トラック B
track-a/
  dbt-project/    検証用の最小 dbt プロジェクト（DuckDB）
  osi/            手書きの 0.1.1 準拠文書
track-b/
  generated/      公式コンバータの生成物（実行時に作られる）
spec/             仕様リポジトリ（make spec で取得。git 管理外）
report/           本レポート（Quarto）
```

B.5 実装上の注意

バリデータの挙動がバージョンで違う。 osi-0.1.1-rc1 タグの validate.py には --schema オプションがなく、__file__.parent.parent / "core-spec" を決め打ちで読む。scripts/lib_ossie.py はその構造を再現している。

コンバータは PyPI にない。 spec/python（apache-ossie）と spec/converters/dbt（ossie_dbt）は公開されていないため、リポジトリからソース導入している。オフラインで導入できるよう、ビルドバックエンド（hatchling）をイメージに含めてある。

B.6 本レポートのビルド

```
cd report
quarto preview   # ローカルで閲覧
quarto render    # _site/ へ出力
```

図版は `report/diagrams/*.mmd`（Mermaid ソース）から `mmdc` で PNG 化している。

C 情報源一覧

信頼性の階層と、否定的結論の扱いについては 付録 A を参照。

C.1 Tier 1 — ASF 公式・仕様リポジトリ

情報源	URL	用途
仕様リポジトリ	https://github.com/apache/ossie	仕様本体・コンバータ・バリデータ
固定コミット	`07be0176e48af67f0b46e0957a87a1545[illegible]`	本書が参照した時点
比較タグ	`osi-0.1.1-rc1`	リリース版 0.1.1 の内容
旧リポジトリ	https://github.com/open-semantic-interchange/OSI	移管前の履歴。バージョン変遷の追跡に使用
Incubator ステータス	https://incubator.apache.org/projects/ossie.html	podling 情報、mentors、入り日
Incubator Clutch	https://incubator.apache.org/clutch/ossie.html	リリース有無、卒業条件の充足状況
Incubation Proposal	https://cwiki.apache.org/confluence/display/INCUBATOR/OssieProposal	PPMC 構成、Known Risks
dev@ アーカイブ	https://lists.apache.org/list.html?dev@ossie.apache.org	podling 運営の実態
general@incubator アーカイブ	https://lists.apache.org/list.html?general@incubator.apache.org	受入投票のスレッド
dist 配布領域	https://dist.apache.org/repos/dist/release/incubator/	Apache Release の有無
June 2026 board report	https://cwiki.apache.org/confluence/display/INCUBATOR/June2026	初回レポート時期の確認

いずれもアクセス日 2026-07-19。

C.2 Tier 2 — プロジェクト広報

ASF インフラ上だが PPMC 自身の広報文であり、主張として扱う。

情報源	URL
公式サイト	https://ossie.apache.org/
Incubator 入りアナウンス	https://ossie.apache.org/updates/ossie-enters-apache-incubator/
旧公式サイト	https://open-semantic-interchange.org/

C.3 Tier 3 — ベンダー発信・業界メディア

事実関係は Tier 1 で裏を取る。単独では根拠としない。

情報源	URL
Snowflake（仕様確定告知）	https://www.snowflake.com/en/blog/open-semantic-interchanges-specs-finalized/
Snowflake（パートナー拡大）	https://www.snowflake.com/en/blog/osi-initiative-expands-partners/
Snowflake（Incubator 入り）	https://www.snowflake.com/en/blog/apache-ossie-open-semantic-interchange-incubator/
Dremio	https://www.dremio.com/blog/apache-ossie-incubating-the-new-name-for-open-semantic-interchange/
Databricks	https://www.databricks.com/blog/redefining-semantics-data-layer-future-bi-and-ai
Salesforce	https://www.salesforce.com/blog/ending-semantic-drift-unified-business-logic-foundation/

C.4 消費側実装のドキュメント

情報源	URL
dbt: OSI semantic models	https://docs.getdbt.com/docs/build/osi-semantic-models
dbt: osi-paths	https://docs.getdbt.com/reference/project-configs/osi-paths

ベンダー各社の製品ドキュメントについては 付録 A に一覧がある。

C.5 関連リポジトリ

情報源	URL
Apache Polaris	https://github.com/apache/polaris
Unity Catalog OSS	https://github.com/unitycatalog/unitycatalog
Apache Superset	https://github.com/apache/superset

C.6 本調査の成果物

情報源	URL
検証環境	https://github.com/dobachi/ossie-playground
既発の関連記事	https://dobachi.github.io/memo-blog/2026/07/10/Open-Semantic-Interchange-technical-deep-dive/