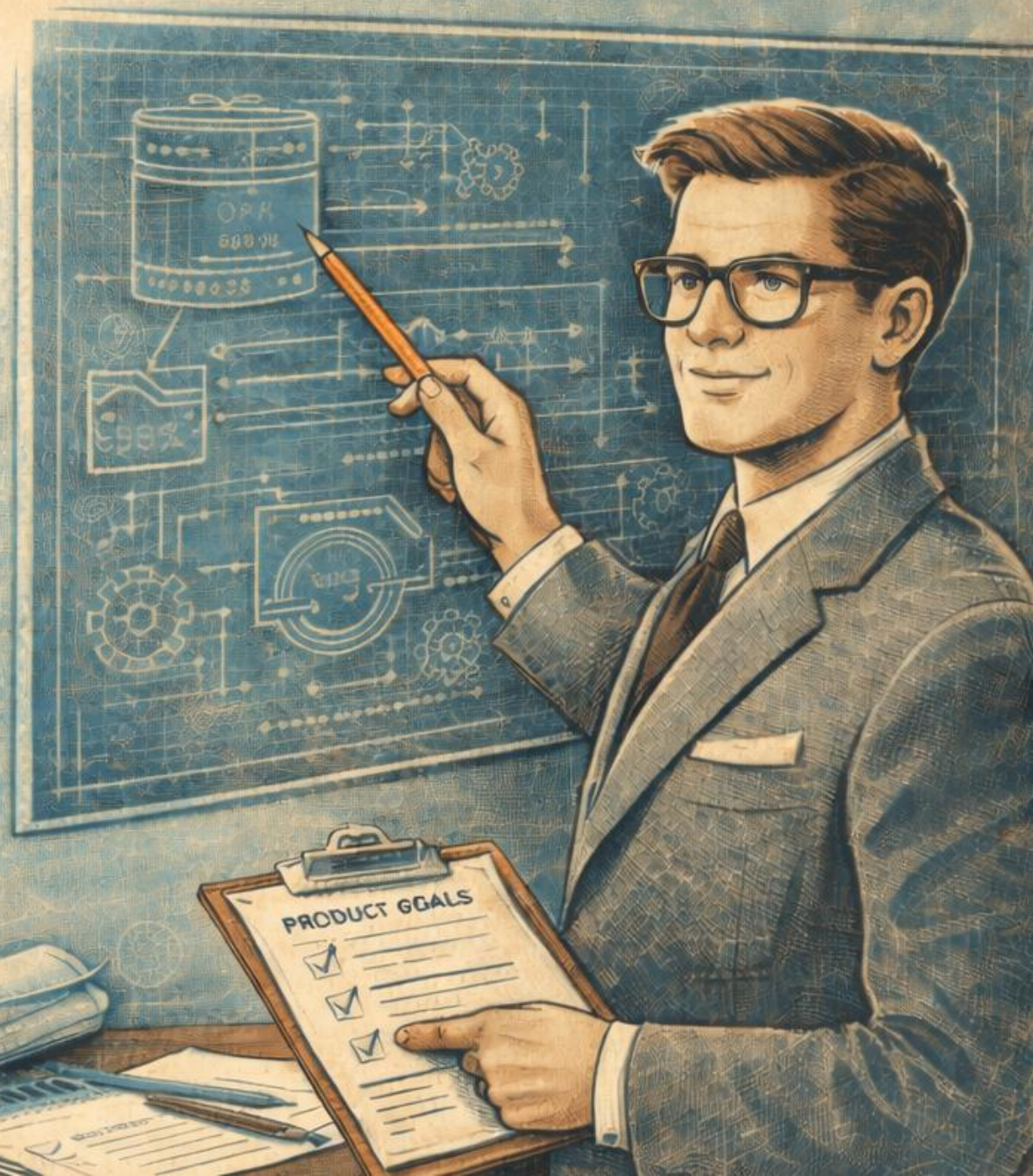
The 5 Lessons of the Open Data Product Specification

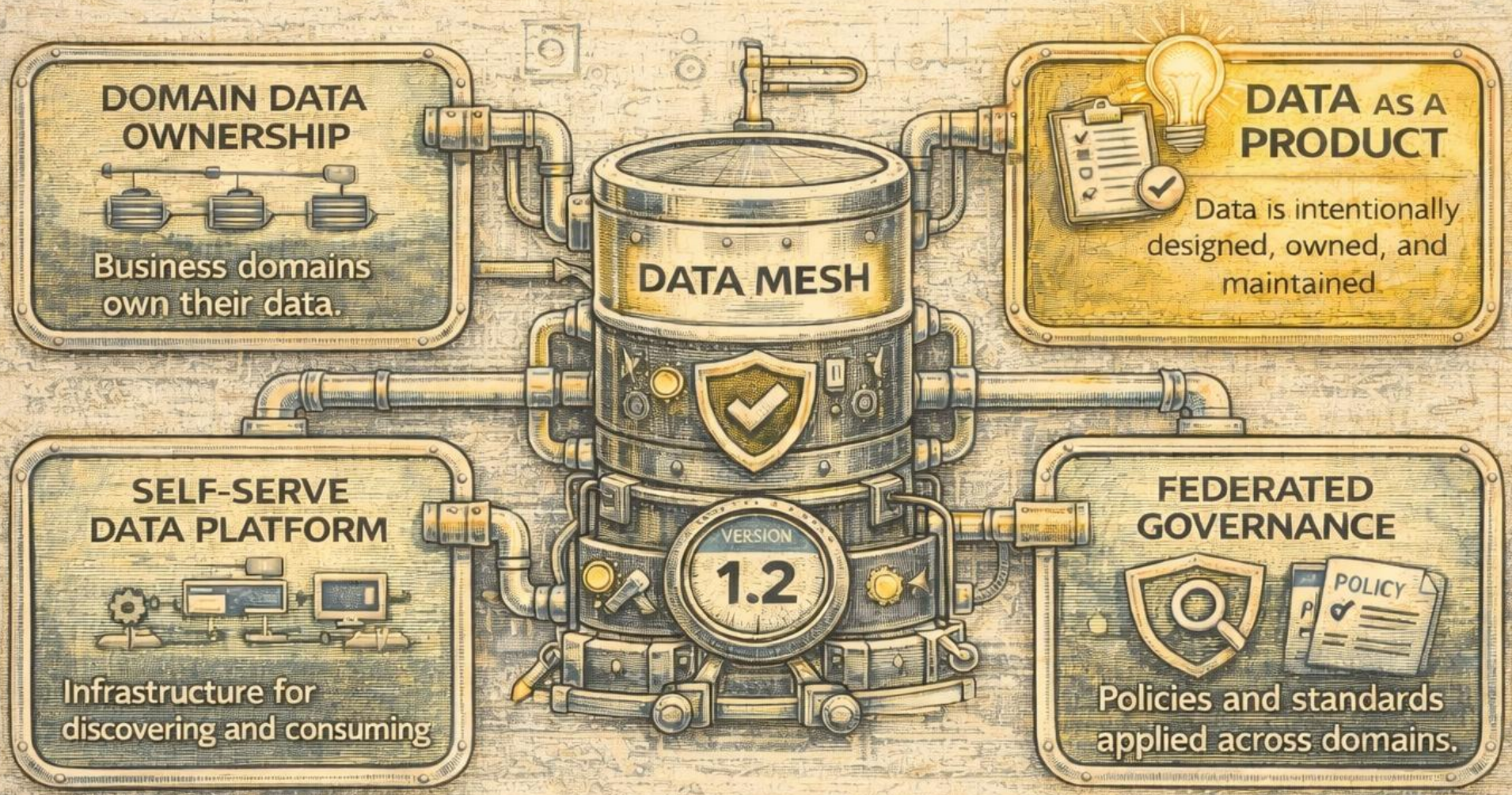
Datawarehousing &
Business Intelligence Summit

Ron Tolido
TechstraOrdinary
March 24, Utrecht




OFFICIAL
CHECKLIST
SHAMPOO
QUALITY
TESTED
WARRANTY

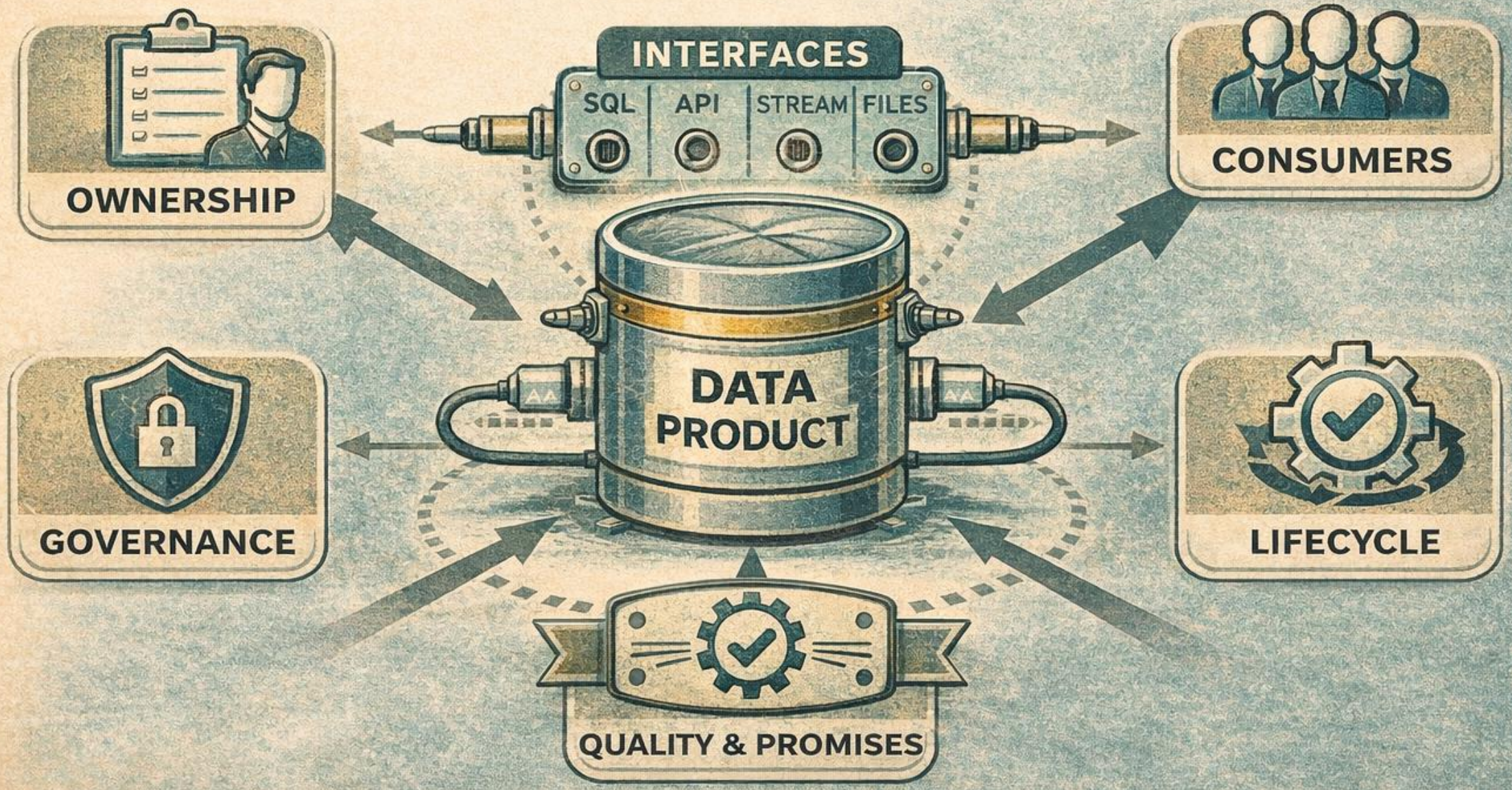
THE FOUR PILLARS OF DATA MESH



Data Mesh gave us the philosophy


SHARE DATA VIA
D!t%& CLOUD!
OUT OF DATE
V2
DATA
PRODUCT
UNKNOWN
OWNER
BEST-EFFORT
FRESHNESS

A **Data Product** is a reusable, governed data asset that is intentionally designed, owned, and delivered to consumers with clear interfaces, quality guarantees, and lifecycle management.



HOW TO BUILD A PROPER DATA PRODUCT

Lessons from the Open Data Product Specification

1 Purpose Before Plumbing

DATA PRODUCT SPECIFICATION

WHAT: _____

WHO: _____

WHY: _____

A product must clearly state **what it is, who it serves,** and **why it exists.**

2 No Owner, No Product

Every product has a named **owner** responsible for value and success.

DATA PRODUCT

3 Interfaces Are Promises

Stable interfaces make consumption **predictable** and **scalable.**

4 Trust Is Engineered

Reliability, freshness and **accuracy** must be **explicit guarantees.**

5 Governance by Design

Compliance, access and versioning are built into the product itself.

TechstraOrdinary

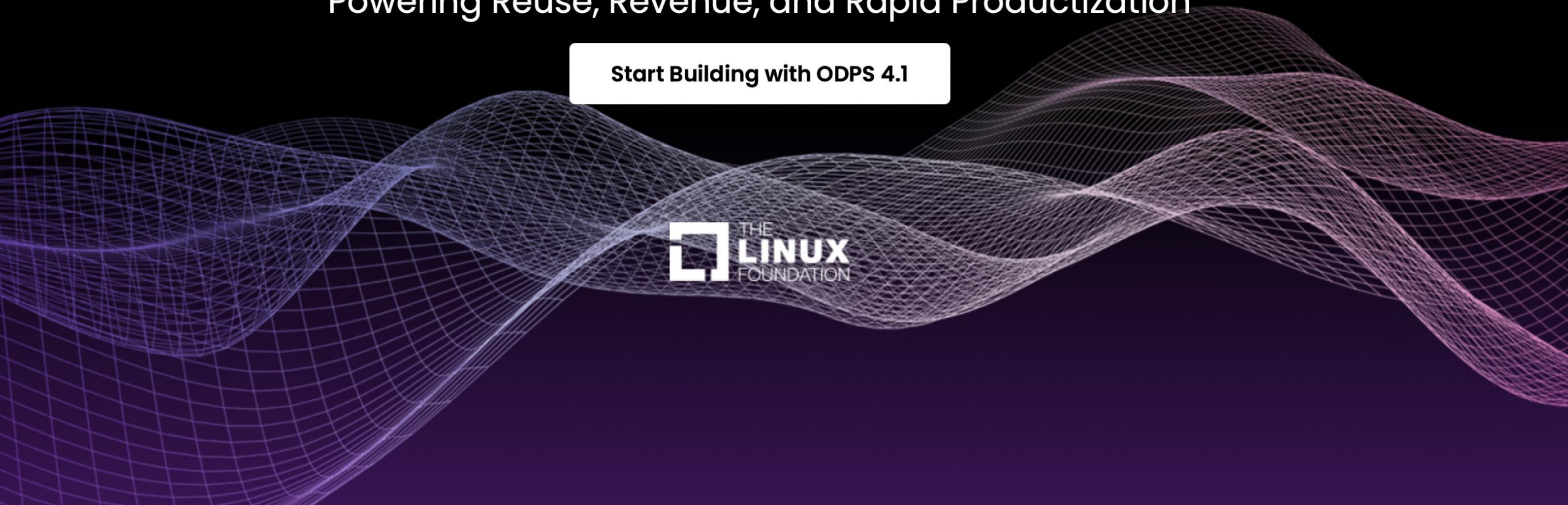
ODPS

What is ODPS | Versions | HowTo | Use cases | Testimonials | Development

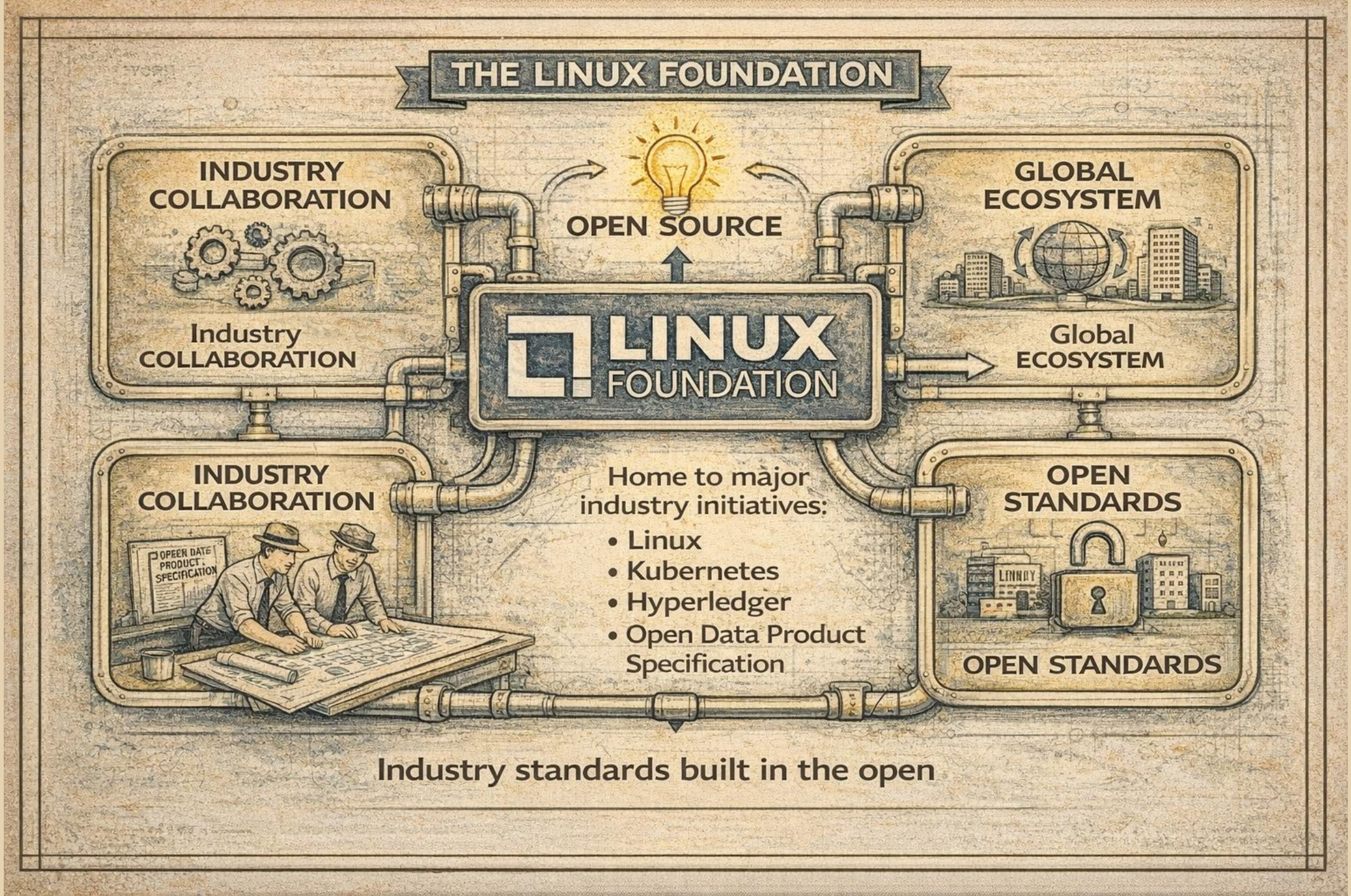
Open Data Product Specification

Powering Reuse, Revenue, and Rapid Productization

Start Building with ODPS 4.1



THE LINUX FOUNDATION

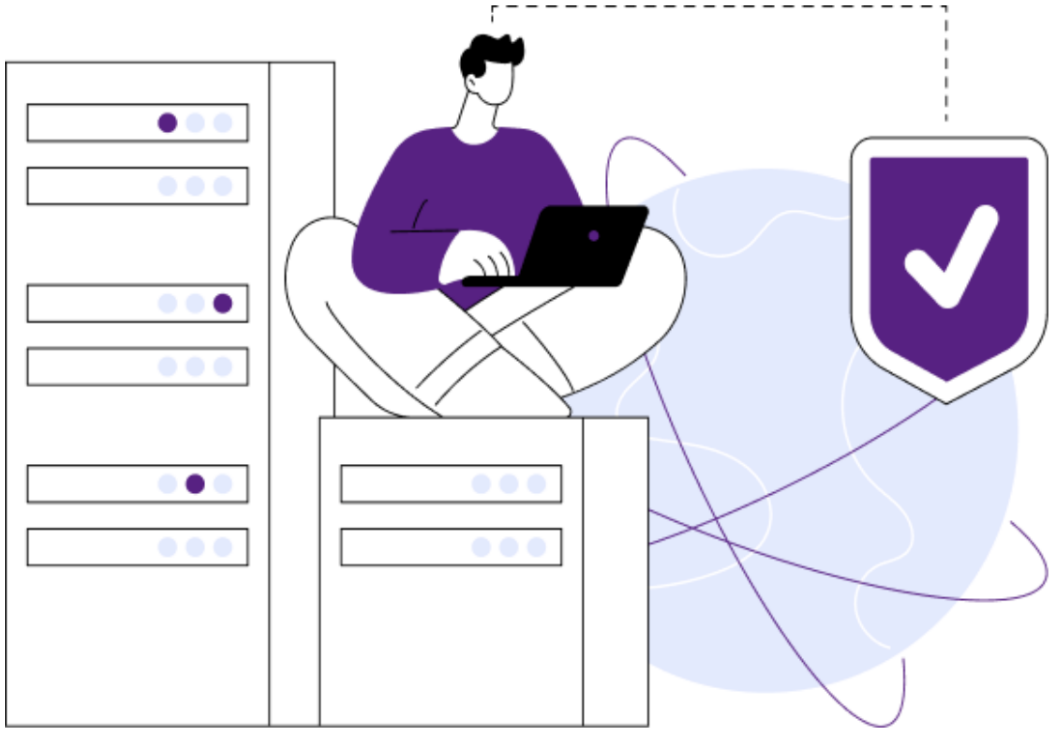

THE LINUX FOUNDATION
INDUSTRY COLLABORATION
Industry COLLABORATION
OPEN SOURCE
LINUX FOUNDATION
GLOBAL ECOSYSTEM
Global ECOSYSTEM
INDUSTRY COLLABORATION
OPEN STANDARDS
Home to major industry initiatives:
• Linux
• Kubernetes
• Hyperledger
• Open Data Product Specification
OPEN STANDARDS
Industry standards built in the open

ODPS is designed for both open and commercial data products, enabling scalable governance, monetization, and AI-native integration — all through a modular, machine-readable YAML specification. Whether you're offering public sector open data or building monetized data APIs, ODPS provides a standardized structure to define and manage your data products. With support for twelve flexible pricing plan templates, organizations can easily adapt to freemium, subscription, enterprise licensing, or usage-based models.

ODPS 4.1 introduces Product Strategy linking data product KPIs to shared business objectives. Together with in-spec referencing for SLAs, Data Quality, Access, and Pricing Plans, it enables measurable, outcome-driven governance and consistent, reusable metadata across large-scale data ecosystems.

The specification embraces a computational, Everything-as-Code philosophy. Data quality rules, SLAs, pricing enforcement, and contract logic can all be automated and integrated into modern workflows. ODPS supports leading platforms such as Stripe, Checkout.com, PayPal, SodaCL, Monte Carlo, DQOps, and Prometheus — making it compatible with both technical operations and business processes.

Hosted by the Linux Foundation since May 14th, 2024, ODPS is a vendor-neutral, open-source initiative built to reshape how organizations govern, share, and monetize data. Its core mission is to provide a machine-readable, modular, and extensible specification that makes data products interoperable, governable, and AI-ready by design.



```yaml
schema: 'https://opendataproducts.org/v4.1/schema/odps.yaml'
version: 4.1
product:
  productStrategy:
    objectives:
      - en: Reduce emergency response time
    strategicAlignment:
      - en: Smart City Vision 2030
    contributesToKPI:
      id: bizkpi-city-response-time
      name: City Emergency Response Time
      description: Average minutes from incident to first responder arrival
      unit: minutes
      target: 5
      direction: at_most
      timeframe: "by Q4"
    relatedKPIs:
      - id: bizkpi-traffic-congestion
        name: Traffic Congestion Index
        unit: percentage
        target: -10
        direction: decrease
    productKPIs:
      - id: kpi-detection-coverage
        name: Event Detection Coverage
        description: % of reported incidents captured in real time
        unit: percentage
        target: 95
        direction: at_least
        frequency: hourly
        calculation: detected_events / reported_events

      - id: kpi-time-to-insight
        name: Average Time to Insight
        description: Median time from event occurrence to product update
        unit: seconds
        target: 60
        direction: at_most
        calculation: p50(update_ts - event_ts)
contract:
  id: 02323M123
  type: ODCS
  contractVersion: 2.2.2
  contractURL: 'https://datamesh-manager.com/urbanltd/dataproducts/9bd530'
details:
  en:
    name: UrbanPulse Events
    productID: urbanpulse-events-001
    valueProposition: >-
      Enable smarter city experiences by providing structured, real-time
      public event data ready for integration into travel apps, tourism
      platforms, and smart city services.
    description: >-
      UrbanPulse Events is a SmartCity data product that aggregates and
      structures public event information — concerts, exhibitions, festivals,
      sports events — making it accessible through APIs and dashboards for
      internal and future external use.
    productSeries: SmartCity Living Data Products
    visibility: internal
    status: production
    productVersion: 0.1.0
    versionNotes: >-
      Initial internal release with basic event metadata structure and shadow
      pricing model implemented.
    issues: >-
      Current limitations include manual ingestion of some event sources and
      partial metadata for smaller events. These will be addressed in the next
      update with automated feeds and metadata enrichment.
    categories:
      - city-events
      - tourism
      - smartcity
    standards:
      - ODPS 4.0
    tags:
      - smartcity
      - events
      - tourism
      - public-data
    brandSlogan: Turning City Buzz into Business Value
    type: dataset
```

HOW THE LESSONS MAP TO ODPS

1	PURPOSE BEFORE PLUMBING	DETAILS + STRATEGY
2	NO OWNER, NO PRODUCT	OWNER / DOMAIN
3	INTERFACES ARE PROMISES	CONTRACT + ACCESS
4	TRUST IS ENGINEERED	SERVICE LEVELS
5	GOVERNANCE BY DESIGN	GOVERNANCE POLICIES

Each part of the standard maps to a lesson

LESSON 1: PURPOSE BEFORE PLUMBING

Clearly state what it is, who it serves, and why it exists...

1. **WHAT:** A brief description of the data product's purpose.
2. **WHO:** Identify the target users and beneficiaries.
3. **WHY:** Explain the value and outcomes it delivers.

A product must be clearly defined before you start building pipelines.

			The pattern to implement multilanguage support for data products was adopted from de facto UI translation practices. The attributes inside this element are commonly rendered in the UI for the consumer and providing a simple way to implement that was the driving reasoning. See for example JSON - Multi Language
name	string	max length 256 chars	**REQUIRED** The name of the product.
productID	string	max length 256 chars	**REQUIRED** Product identifier.
visibility	string	one of: private, invitation, organisation, dataspace, public	**REQUIRED** The publicity level eg who can see this product. Private - just the creator. Invitation - visible only to parties explicitly invited. Organisation - visible to all in your organisation. Dataspace - visible to all existent members of the data space. Public - visible to all publicly.
status	string	one of: announcement, draft, development, testing, acceptance, production, sunset, retired	**REQUIRED** The status of the product. Lifecycle model discussed in details in here (link).
type	string	one of: raw data, derived data, dataset, reports, analytic view, 3D visualisation, algorithm, decision support, automated decision-making, data-enhanced product, data-driven service, data-enabled performance, bi-directional.	**REQUIRED** The type of the product. Options are derived from examples and lists found from academic literature.

The `productStrategy` object captures the business intent behind a data product and links it to a **single higher-level business KPI** it is primarily accountable for. It defines the objectives, the primary KPI at the business level, and the **product-level KPIs** that measure its direct contribution.

Optionally, it can also include **related KPIs** — secondary or cross-unit measures that:

1. **Capture additional value** — show extra benefits beyond the main KPI, useful for demonstrating broader impact.
2. **Track side effects** — monitor unintended positive or negative impacts on other KPIs.
3. **Highlight cross-department value** — reveal benefits for other business units, strengthening prioritization and funding cases.

Beyond KPIs, `productStrategy` can include:

- **objectives** — natural-language statements describing the business outcomes the product supports.
- **strategicAlignment** — references to corporate initiatives, programs, or policy documents that the product contributes to (e.g., Smart City Vision 2030).

By embedding both primary and related KPI connections directly into the product specification, `productStrategy` ensures that data products remain technically sound, strategically aligned, and transparent in how they deliver value.

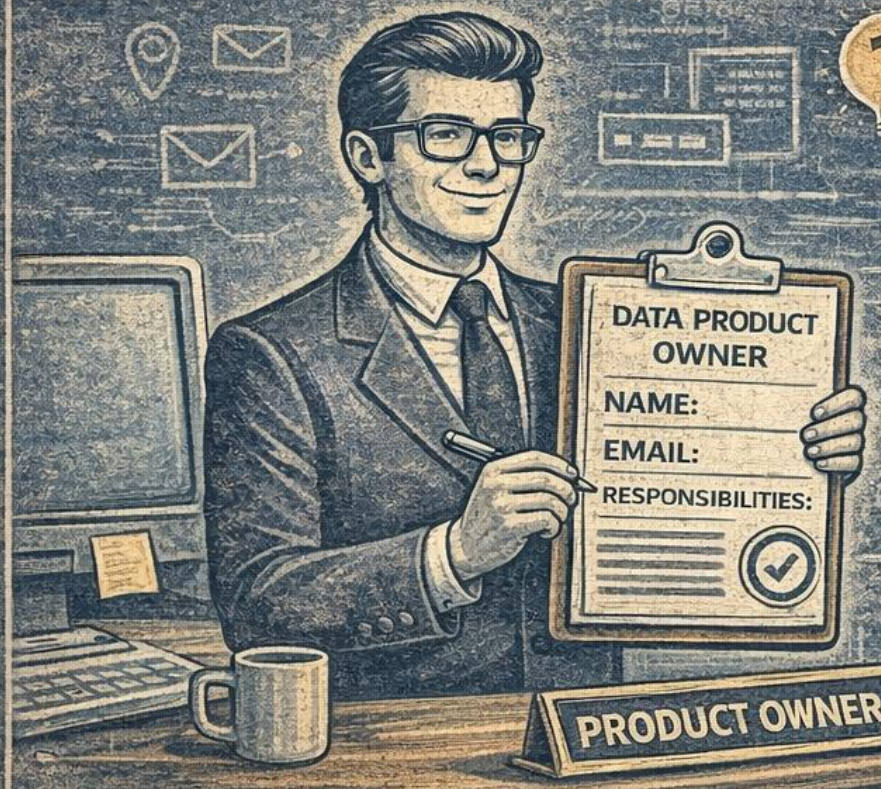
Attributes and options

Element	Type	Options	Description
productStrategy	object	–	Top-level block that connects the data product to business goals and KPIs.
objectives	array of objects	language-tagged strings	Business objectives the product supports, written in natural language.
contributesToKPI	object	required	**Single higher-level business KPI** (from SMART objectives) that this product is accountable for.
id	string	optional	Identifier of the business KPI (use shared IDs for roll-ups).
name	string	required	KPI name.

Example of catalog object usage:

```yaml
schema: https://opendataproducts.org/v4.1/schema/odps.yaml
## JSON schema: https://opendataproducts.org/v4.1/schema/odps.json
version: 4.1
product:
  productStrategy:
    status: Planned
    startDate: 2026-01-12
    endDate: 2026-06-30
    objectives:
      - en: Reduce emergency response time
    strategicAlignment:
      - en: Smart City Vision 2030
    contributesToKPI:
      id: bizkpi-city-response-time
      name: City Emergency Response Time
```

LESSON 2: NO OWNER, NO PRODUCT



1 Every data product has a named owner responsible for value and success.

1. **WHAT:** A brief description of the data product purpose.
2. **WHO:** Identify the target users and beneficiaries.
3. **WHY:** Explain the value and outcomes it delivers.

No clear ownership? No data product to deliver.

description	string	optional	Human-readable description.
unit	string	e.g., `percentage`, `minutes`, `s`	Unit of measurement.
target	number/string	–	Target value for the KPI.
direction	enum	`increase`, `decrease`, `at_least`, `at_most`, `equals`	Desired direction of movement.
timeframe	string	optional	When the target should be met.
frequency	string	Measurement cadence (e.g., hourly, daily, monthly).	
owner	string	Responsible role/team (optional).	
calculation	string	Human-readable formula (optional).	
productKPIs	array	optional	KPIs measured **at product level** that influence `contributesToKPI`. Useful for contribution analysis and governance checks.
relatedKPIs	array	optional	Secondary/cross-unit KPIs to monitor side-effects and additional value (informational; not for prioritization).
strategicAlignment	array	language-tagged strings	Strategic initiatives, policies, or visions the product aligns with.

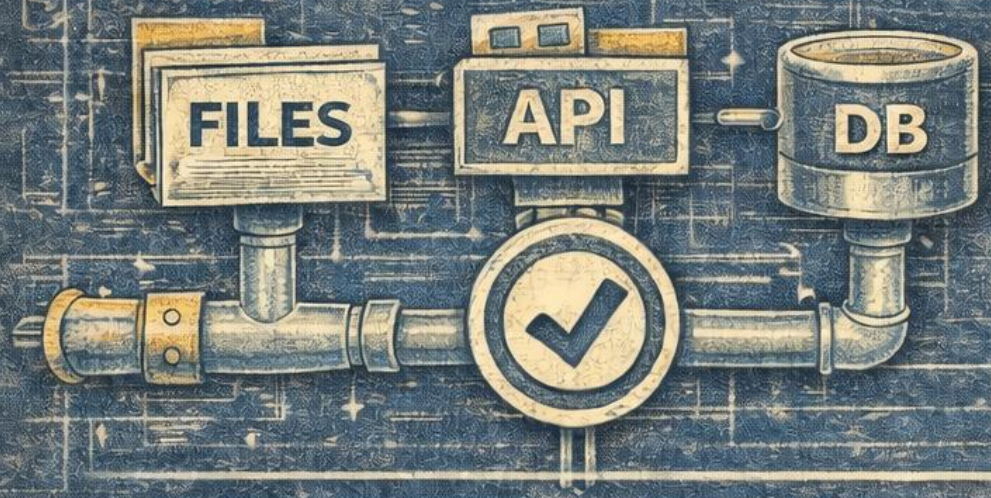
Governance-by-Design checks (Minimum Lovable Gates)

- **Required:** `contributesToKPI.name` present.
- **Recommended:** ≥1 `productKPIs` with `unit`, `target`, and `calculation`.
- **Optional:** `relatedKPIs` for secondary/side effects.
- **Traceability:** use shared KPI `id`s to enable cross-product roll-ups against the same business KPI.

```
    unit: percentage
    target: 95
    direction: at_least
    frequency: hourly
    calculation: detected_events / reported_events

  - id: kpi-time-to-insight
    name: Average Time to Insight
    description: Median time from event occurrence to product update
    unit: seconds
    target: 60
    direction: at_most
    calculation: p50(update_ts - event_ts)
```

LESSON 3: INTERFACES ARE PROMISES

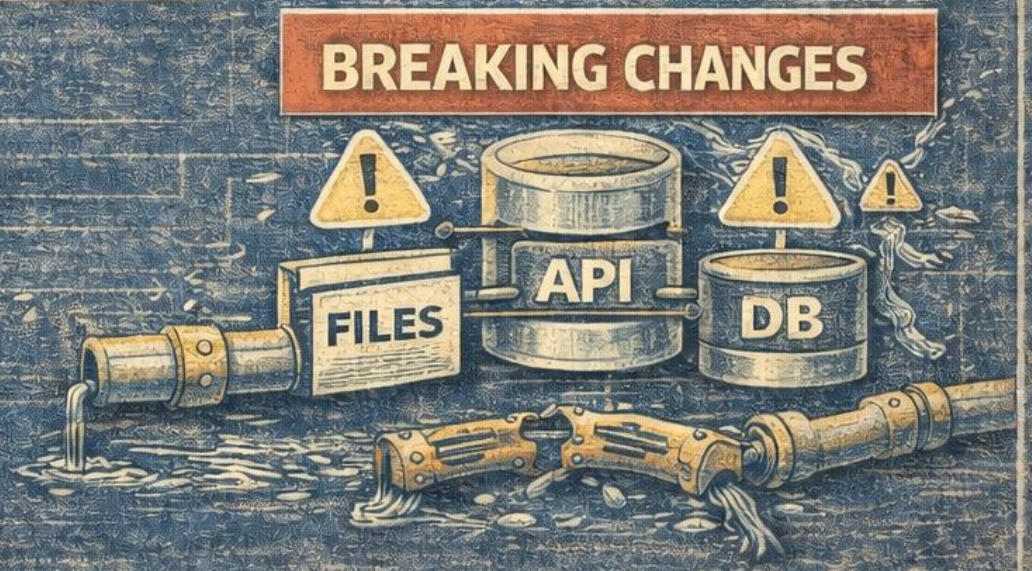


SQL | API | Stream | Files
estabiished expectations.

Stable inferfaces make consumption predictable and scalable.

- Clear data contracts.
- Backward-compatible changes.
- Documentation and support.

Interfaces set expectations,
like à promise you must keep or face



The `contract` object in the ODPS schema defines the structure for linking a data product to a formal data contract. It includes an `id` as a unique identifier, a `type` field that specifies the contract standard (either ODCS or DCS), a `contractVersion` indicating the version of the standard, and a `contractURL` that points to an external contract document in a contract management service. Optionally, an inline YAML spec element can be used instead of a URL to embed the contract directly in the product file. These specifications allow data producers to clearly declare governance, usage rights, and technical expectations, supporting interoperability across platforms using standards such as the Open Data Contract Standard (ODCS) or the Data Contract Specification (DCS). More details in the Knowledge Base.

Contract can be defined in 3 ways:

- Use ODPS given attributes to define basic information of the data contract and add URL to details found in contract management system
- Same as above but add contract details as YAML on `spec` element.
- Use `$ref` to define contract object content in a separate file.

Optional attributes and elements

Element name	Type	Options	Description
contract	element	-	Binds together data contract details. You can use both URL and inline (YAML) for data contract content.
id	string	-	UUID of the data contract
type	string	one of: ODCS, DCS	Defines the standard used in data contract. Currently supported options: ODCS and DCS.
contractVersion	string	-	Version of the standard used to define the Data Contract. Type attribute defines the standard/specification. NOTE! This is not the possible iterated version of the data contract itself.
contractURL	URL	Valid URL. See more from RFC 3986.	URL pointing to data contract in data contract management service or alike. Optionally you can use *spec* to add data contract details as YAML inline element.

Example of contract object usage:

```yaml
schema: https://opendataproducts.org/v4.0/schema/odps.yaml
version: 4.1
product:
  contract:
    # Optionally se URI of external file, contains all contract details
    $ref: 'https://example.org/contracts/default.yaml'
    # Or then define with attributes
    id: 02323M123
    type:  ODCS
    contractVersion:  2.2.2
    contractURL: https://demo.datamesh-manager.com/demo834016807886/dataproducts/9
    # Or use inline style and add contract under spec element
    spec:
        ....
```

The `dataAccess` object defines how users—or machines—can technically access the data product. It allows publishers to describe **multiple, named access methods** tailored to different consumer needs: from simple file downloads and APIs to AI agent integration via protocols like **MCP**.

Each entry under `dataAccess` (such as `default`, `API`, or `Agent`) represents a distinct access interface with its own metadata, authentication requirements, and documentation references. This structure makes it possible to:

- Offer **flexible access modes** for various user personas (analysts, developers, AI agents, etc.)
- Support **multilingual UI presentation** through localized `name` and `description` fields
- Clearly declare **security expectations** using `authenticationMethod`
- Link to both **machine-readable specs** (`specsURL`) and **human-readable guides** (`documentationURL`)
- Promote **reusability** by referencing these interfaces throughout the ODPS YAML using `$ref`

Including an AI agent-specific access interface (`outputPorttype: AI`) supports MCP-based agent interactions, aligning your product with **AI-native data delivery patterns**.

Referencing Examples

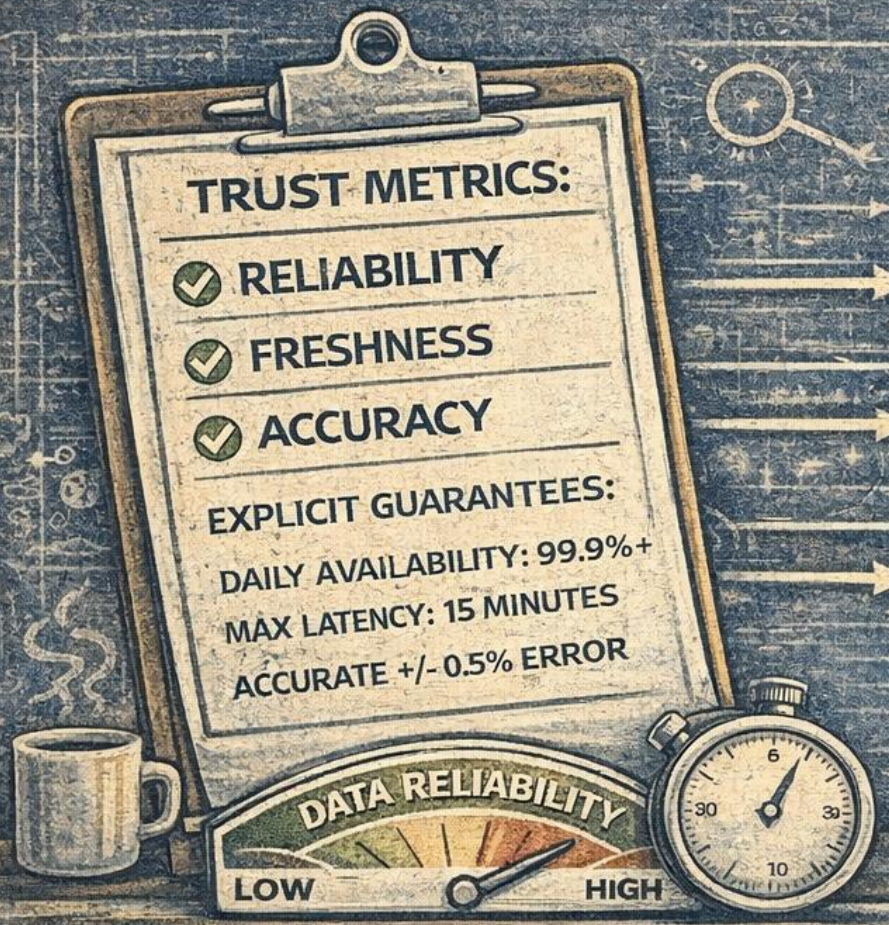
For example in your `access` section in Pricing, you can reuse any defined method from `dataAccess` like this: *$ref: '#/Product/dataAccess/default'*

Optional attributes and elements

Element name	Type	Options	Description
dataAccess	object	-	Root-level object containing named access configurations. Each key (e.g., `default`, `API`, `Agent`) defines an access method that can be reused across the ODPS YAML.
$ref	filepath or valid URL	-	Define the Data Access in external file for reuse purposes, example `$ref: 'https://example.org/dataAccess/all-packages.yaml'` See example. This makes it easy to keep related profiles (e.g. default, API, agent) together, apply versioning and validation once, and publish all variants from a single repo or source.

outputPorttype	string	file, API, SQL, AI, gRPC, sFTP, etc.	Describes the technical method for delivering data (e.g., `file` for file downloads, `API` for web services).
format	string	TOON, JSON, XML, CSV, Excel, zip, plain text, GraphQL, MCP	Specifies the data format made available through this access channel.
authenticationMethod	string	OAuth, Token, API key, HTTP Basic, none	Security model required to access the data.
specification	string	OAS, RAML, Slate, MCP	Defines the type of API or protocol specification used to describe access (e.g., OpenAPI, RAML, or a custom protocol like MCP).
specsURL	URL	Valid URL	Points to the machine-readable technical documentation (e.g., OpenAPI YAML).
accessURL	URL	Valid URL	The direct access point to retrieve the data – can be for example an API endpoint or a file link.
documentationURL	URL	Valid URL	A human-readable documentation or guide for access setup, authentication steps, or onboarding.
hashType	string	SHA-1, SHA-2, SHA-256, MD5, etc.	(Optional) Defines hash algorithm used when providing file integrity verification.
checksum	string	any string	(Optional) File hash/checksum value, useful for verifying data integrity after download.

LESSON 4: TRUST IS ENGINEERED



Reliability, freshness, and accuracy must be explicit guarantees.

RELIABILITY Ensures uptime and daily availability.

FRESHNESS Guarantees updates hourly at most.

ACCURACY Offers correct, precise data on key metrics.

Trust in data must be built, measured, and **maintained.**

ODPS
OPEN DATA PRODUCT SPECIFICATION

Search

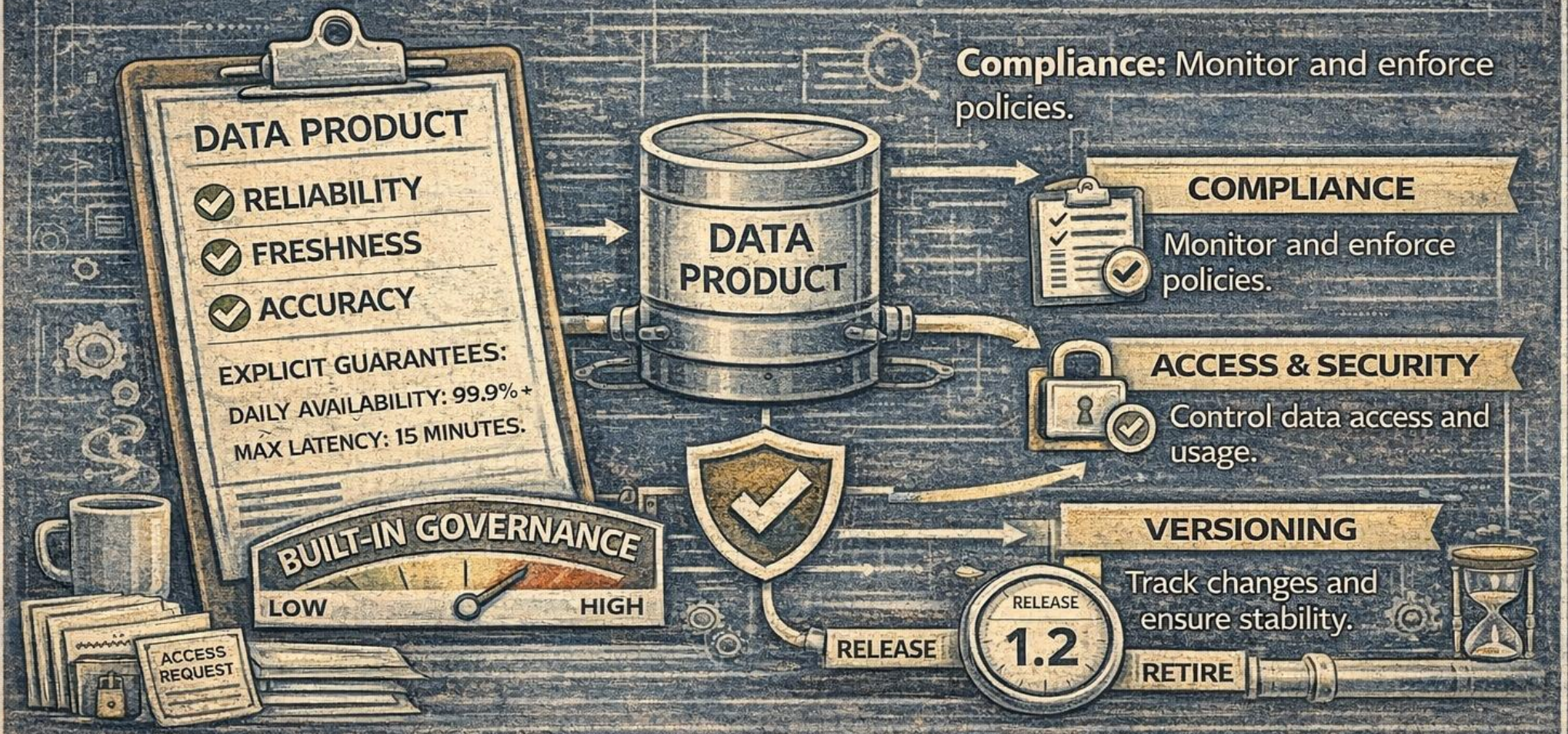
SLA can be defined with 11 standardized dimensions with decoupled Everything as Code monitoring

SLA Dimension	Description
latency	minimal amount of time before getting any response.
uptime	Uptime is a measure of system reliability, expressed as the percentage of time a machine, typically a computer, has been working and available. See more https://uptime.is/.
responseTime	amount of time to process external request.
errorRate	Maximum tolerated errors in data, percentage.
endOfSupport	The date at which your product will not have support anymore.
endOfLife	The date at which your product will not be available anymore. No support, no access.
updateFrequency	how often data is updates.
timeToDetect	How fast can you detect a problem?
timeToNotify	Once you see a problem, how much time do you need to notify your users?
timeToRepair	How long do you need to fix the issue once it is detected?
emailResponseTime	How long do you need to respond to email support requests?

Example of SLA component usage:

```yaml
SLA:
  declarative:
    - dimension: uptime
      displaytitle:
        - en: Uptime
      objective: 99
      unit: percent
    - dimension: responseTime
      objective: 200
      unit: milliseconds
```

LESSON 5: GOVERNANCE BY DESIGN



COMPLIANCE: Monitor and enforce

ACCESS & SECURITY: Control data access and usage.

VERSIONING: Track changes and ensure stability.

Governance, access, and **versioning** must be built into the product itself.

MINIMUM LOVABLE GOVERNANCE FOR DATA PRODUCTS

Aspect	Data Product Management (MLG)
Primary Goal	Deliver value and outcomes
Focus	Data as a **product** with lifecycle, owner, and customers
Approach	Lightweight, embedded governance within each product
Process	Iterative, automated, tied to delivery
Success Measure	Business KPIs, adoption, trust, and impact

Core Principles of MLG

1. **Value First** — Every data product must tie to at least one business KPI. Governance begins with outcomes, not just controls.
2. **Lean but Strong** — Only apply rules that matter. Automate wherever possible. Don't create work that nobody uses.
3. **Governance by Design** — Governance is built into the product itself: metadata, contracts, SLAs, and quality checks travel with the product.
4. **People-Centric** — Governance should be simple, clear, and supportive. Product teams should see it as an enabler, not a blocker.

THE ODPS TOOLBOX

Using ODPS in Practice

1 DEFINE DATA PRODUCTS CONSISTENTLY

Use ODPS as a standard template for describing data products.

• identity • owner • domain • use cases • strategy

→ **Every data product starts with the same structure**

2 STANDARDIZE DATA CONTRACTS

Describe schemas and interfaces in a machine-readable way.

• tables • APIs • streams • fields

→ **Clear contracts between producers and consumers.**

3 ENABLE DATA PRODUCT DISCOVERY

Use ODPS to feed:

• data catalogs • data marketplaces • internal discovery tools

→ **Makes data products discoverable and reusable.**

4 AUTOMATE GOVERNANCE

Embed policies directly in the specification.

• access control • privacy classification • compliance rules

→ **Governance becomes built into the product.**

5 SUPPORT PLATFORM AUTOMATION

Platforms can use ODPS to automatically

• deploy interfaces • validate contracts • monitor service levels.

→ **From documentation** to automation.

HOW TO BUILD A PROPER DATA PRODUCT

Lessons from the Open Data Product Specification

1 Purpose Before Plumbing

DATA PRODUCT SPECIFICATION

WHAT: _____

WHO: _____

WHY: _____

A product must clearly state **what it is, who it serves,** and **why it exists.**

2 No Owner, No Product

Every product has a named **owner** responsible for value and success.

DATA PRODUCT

3 Interfaces Are Promises

Stable interfaces make consumption **predictable** and **scalable.**

4 Trust Is Engineered

Reliability, freshness and **accuracy** must be **explicit guarantees.**

5 Governance by Design

Compliance, access and versioning are built into the product itself.

TechstraOrdinary

The 5 Lessons of the Open Data Product Specification

Datawarehousing &
Business Intelligence Summit

Ron Tolido
TechstraOrdinary
March 24, Utrecht