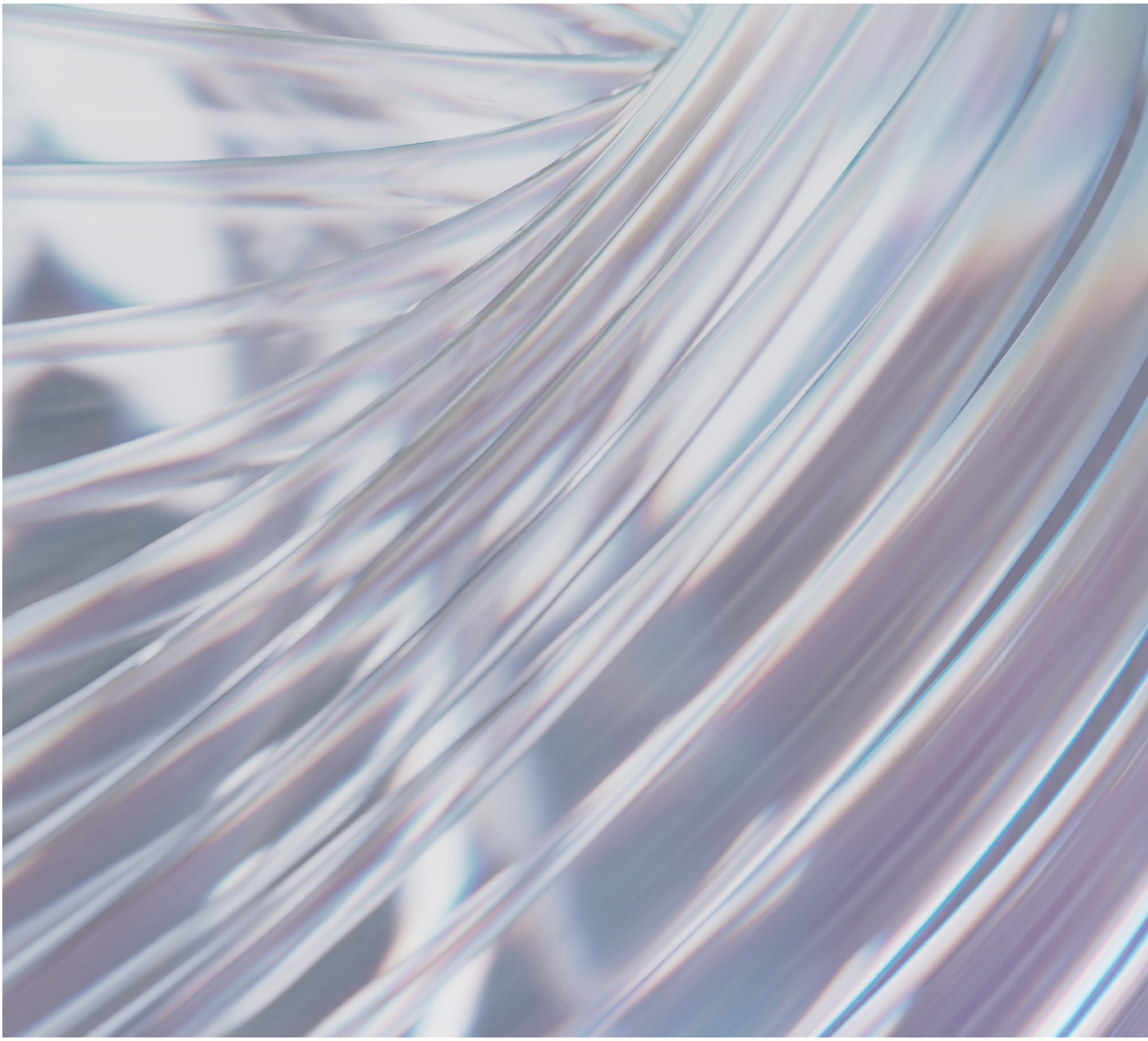
Why Open Dataspaces:

Design Philosophy and the Architectural Paradigm

Leveraging distributed data management
for inter-organization and global interoperability

April, 2026



Publisher Digital Architecture Design Center,
Information-technology Promotion Agency

Author & Editor Michitaka TSUDA
Chief Architect

Why Open Dataspaces:

Design Philosophy and

the Architectural Paradigm

Leveraging distributed data management

for inter-organization & global interoperability

KEY WORDS:

distributed data management, data mesh, dataspaces, semantic layer, ontology, usage control, Artificial Intelligence, Agentic AI, context engineering, Domain-Driven Design

Publisher

Digital Architecture Design Center,
Information-technology Promotion Agency, Japan

Editor & Author

Michitaka TSUDA, Open Data Spaces Chief Architect
Email: MichitakaTsuda@outlook.com

Cite As

Tsuda M. (2026). Why Open Dataspaces: Design Philosophy and the Architectural Paradigm. Information-technology Promotion Agency.

Purpose of This Document

This document presents the design philosophy of “Open Data Spaces (ODS)” — an open and scalable foundation for distributed data, built on organizational and national diversity by design — along with its core architectural paradigm.

Introduction:

The Age of Agentic AI and the Potential of Dark Data

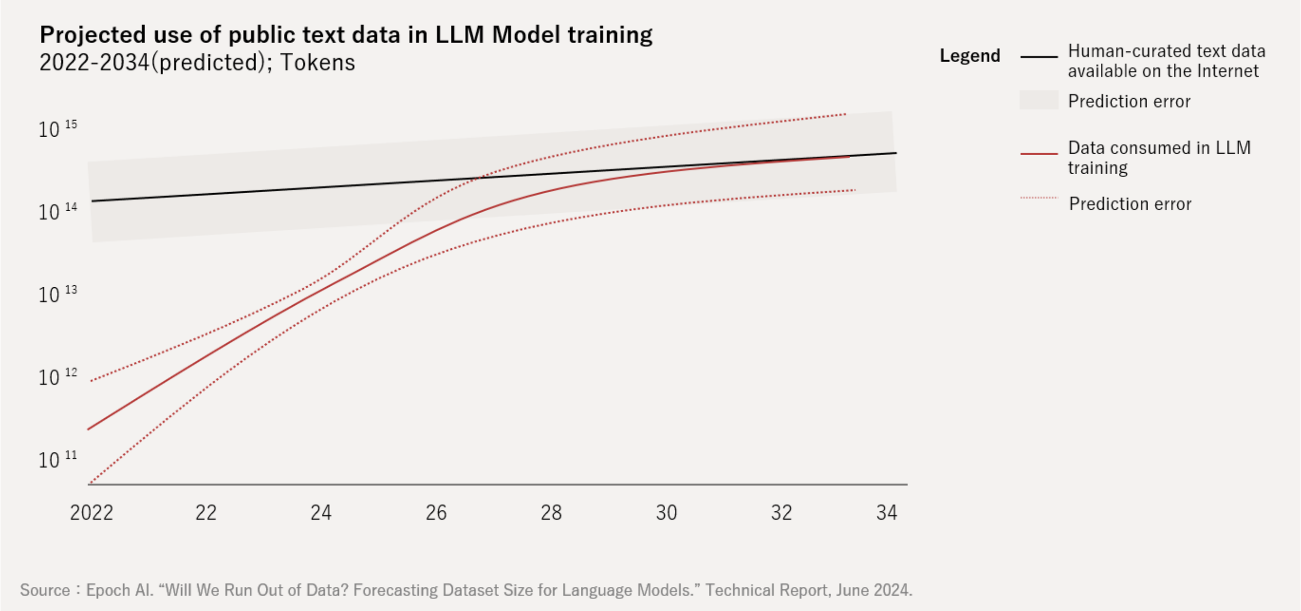
As AI continues to transform the fundamental structure of global industries, 2026 marks a pivotal turning point that many technology market participants may come to call the "**Dawn of Data Scarcity**". Any discussion of AI progress must account for three key variables driving model performance: (1) model size, (2) dataset size, and (3) the amount of compute — known empirically as the "**Scaling Laws**". From a practical standpoint, securing training data and computational resources has become the dual pillars of any AI strategy.

Market trends indicate fierce competition for computational resources encompassing semiconductor supply, data center construction, and power and communications procurement. The supply constraint on training data, by contrast, has not yet received sufficient attention from society at large.

The Finite Nature of Internet Data

Today's major Large Language Models (LLMs) rely on training datasets built from publicly available text on the internet — web pages, social media posts, digital books, and academic papers. The stock of data on the internet appears to grow continuously, but it has only a finite rate of growth. If the rate at which models consume data for training outpaces this growth, the quality and quantity of data available for training will become relatively scarce. This is what is called the "**data scarcity problem**". Epoch AI (2024) projects that if major benchmark LLMs continue training at their current pace, high-quality data could be exhausted between 2026 and 2032. (**Figure 1**)

This *exhaustion* does not mean the data itself disappears, but rather mean that the supply rate of new data capable of supporting training can no longer keep up.



Source：Epoch AI. "Will We Run Out of Data? Forecasting Dataset Size for Language Models." Technical Report, June 2024.

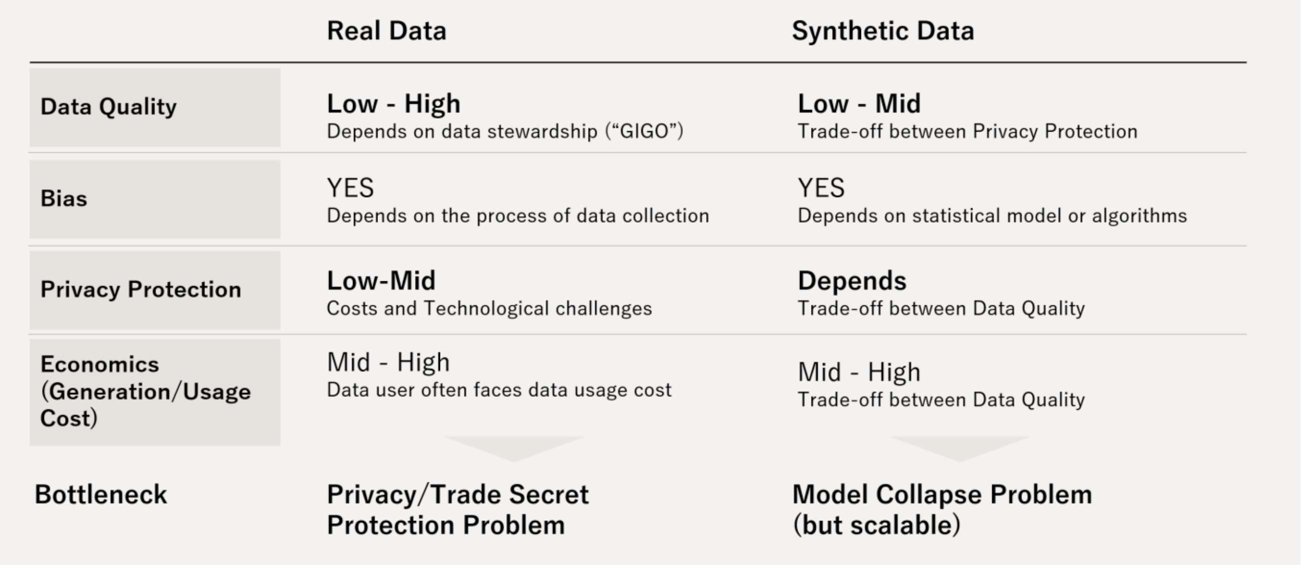
Figure 1 — Projected Use of Public Text Data in LLM Model Training

Real Data and Synthetic Data as Supply Sources

Two broad categories of data can supplement the scarcity:

- **Real Data**: Data obtained from real-world observations and transactions (operational data, design and engineering data, IoT sensor data, etc.)
- **Synthetic Data**: Data artificially generated by models or statistical methods

Synthetic data is easier to scale in total volume. However, **recursive use risks performance degradation through loss of distributional diversity** known as the "**model collapse problem**", making a supply of real data essential. Real data, on the other hand, often contains privacy-sensitive or proprietary information, which creates high barriers to acquisition and sharing. (**Figure 2**)



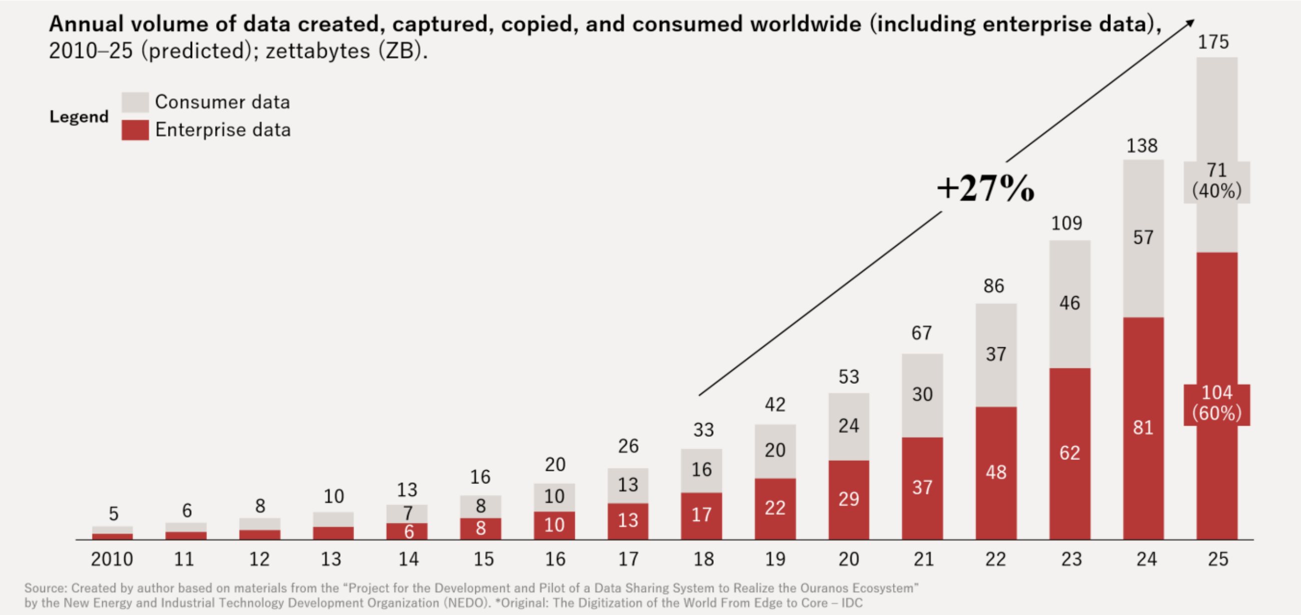
	Real Data	Synthetic Data
Data Quality	**Low - High** Depends on data stewardship (“GIGO”)	**Low - Mid** Trade-off between Privacy Protection
Bias	YES Depends on the process of data collection	YES Depends on statistical model or algorithms
Privacy Protection	**Low-Mid** Costs and Technological challenges	**Depends** Trade-off between Data Quality
Economics (Generation/Usage Cost)	Mid - High Data user often faces data usage cost	Mid - High Trade-off between Data Quality
Bottleneck	**Privacy/Trade Secret Protection Problem**	**Model Collapse Problem (but scalable)**

Figure 2 — Comparison of Real Data and Synthetic Data

Real data and synthetic data are therefore not simply in a trade-off relationship of superiority or inferiority. Rather, they complement each other depending on the use case. The optimal supply strategy is to retain the source real data while combining both types appropriately. Designers must also build in data strategies that minimize the impact of recursive contamination.

The Proportion of Dark Data within Real Data

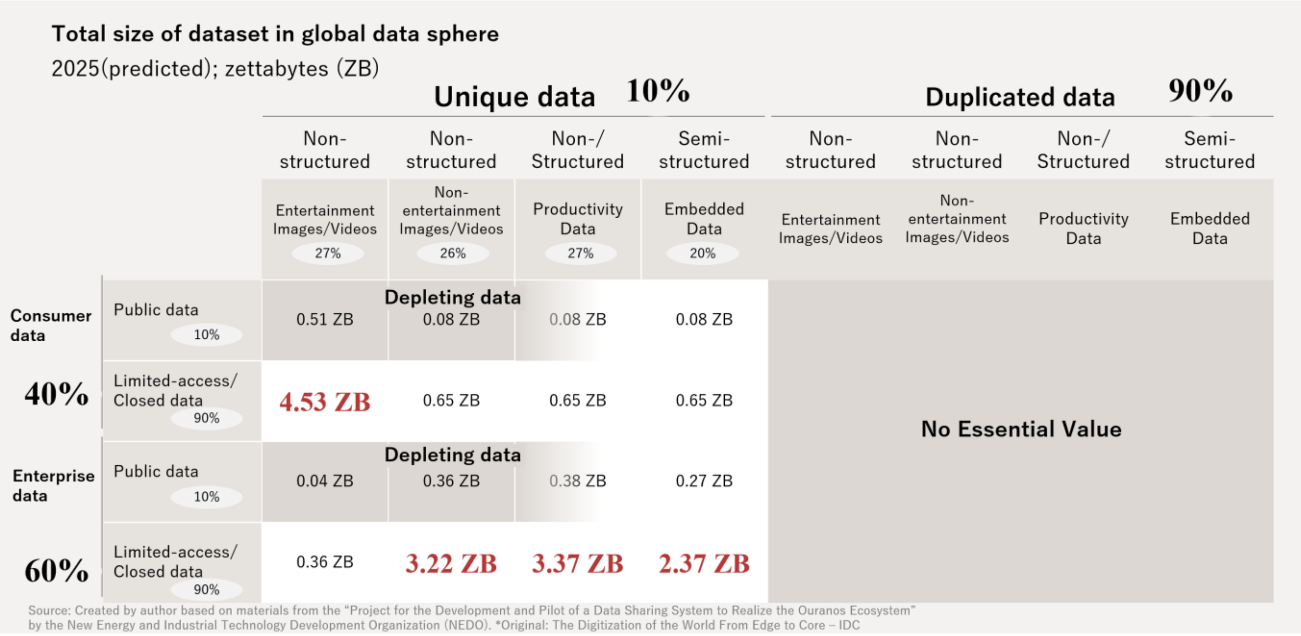
So how much real data actually exists? A survey by the New Energy and Industrial Technology Development Organization (NEDO, 2025) estimates that as of 2025, the total volume of real data created, captured, copied, and consumed worldwide amounts to approximately 175 ZB (zettabytes) per year. Of that total, roughly 71 ZB originates from consumers, while the remaining approximately 104 ZB is enterprise-derived. (**Figure 3**)



Source: Created by author based on materials from the "Project for the Development and Pilot of a Data Sharing System to Realize the Ouranos Ecosystem" by the New Energy and Industrial Technology Development Organization (NEDO). *Original: The Digitization of the World From Edge to Core – IDC

Figure 3 — Total Volume of Real Data and the Consumer vs. Enterprise Ratio

The same survey estimates the effective volume of real data at approximately 17.5 ZB, after removing duplicates. Of that, **roughly 16 ZB remains inside enterprises as "dark data"** — data not publicly available on the internet. (**Figure 4**) Dark data holds potential value as a resource for AI training and inference, yet in many cases, organizations cannot readily use it. Dark data frequently contains elements that introduce bias or contaminate data quality. Much of the data within enterprises also requires filtering and curation aligned with specific business purposes and contexts before it can be applied.



Total size of dataset in global data sphere
2025(predicted); zettabytes (ZB)

		Unique data 10%				Duplicated data 90%			
		Non-structured	Non-structured	Non-/Structured	Semi-structured	Non-structured	Non-structured	Non-/Structured	Semi-structured
		Entertainment Images/Videos 27%	Non-entertainment Images/Videos 26%	Productivity Data 27%	Embedded Data 20%	Entertainment Images/Videos	Non-entertainment Images/Videos	Productivity Data	Embedded Data
Consumer data 40%	Public data 10%	0.51 ZB	Depleting data 0.08 ZB	0.08 ZB	0.08 ZB	No Essential Value			
	Limited-access/ Closed data 90%	4.53 ZB	0.65 ZB	0.65 ZB	0.65 ZB				
Enterprise data 60%	Public data 10%	0.04 ZB	Depleting data 0.36 ZB	0.38 ZB	0.27 ZB				
	Limited-access/ Closed data 90%	0.36 ZB	3.22 ZB	3.37 ZB	2.37 ZB				

Source: Created by author based on materials from the "Project for the Development and Pilot of a Data Sharing System to Realize the Ouranos Ecosystem" by the New Energy and Industrial Technology Development Organization (NEDO). *Original: The Digitization of the World From Edge to Core – IDC

Figure 4 — Breakdown of the Total Volume of Real Data

GIGO, Domain Context, and Distributed Data

The greatest barrier to leveraging real data — whether as a training resource in pre-training or as an inference resource in post-training — is not volume but **quality**. A large accumulation of big data within an organization does not automatically make it useful for AI, BI (Business Intelligence), or DI (Decision Intelligence). Ingesting large quantities of data misaligned with the intended purpose can actually degrade the accuracy and generalizability of results, often called Garbage-In, Garbage-Out (GIGO). Beyond accuracy, running costs that generate no essential value also put pressure on finances.

Data quality depends on the **Domain**, the autonomous unit of responsibility within an organization. Each domain carries a **Context**, the totality of explicit and implicit circumstances specific to that domain, including its unique purposes, operations, and practices. Context is a major factor in determining data quality. Yet it exists distributed at the domain level, and therefore, uniform homogenization across domains is extremely difficult.
Against this backdrop, anyone thinking about architectural paradigms for data management in the Agentic AI era should keep the following key questions in mind:

- How can we assign a domain context to dark data?
- How can we make data enriched with domain context function as corporate capital that generates future cash flows?
- And how should we manage data and domain context that are inherently distributed across organizational and national boundaries?

Context is the essence of data in the Agentic AI era, and it is the proprietary asset of the domain. Domain owners assign context to data, provide it as a *"product"*, and control its use — and this drives enterprise value. A data strategy that treats data as corporate capital, combined with a software strategy that maximizes that data's value, will likely be the most fundamental survival strategy amidst "**Data is Eating the World**" — a world where software defines corporate value, and data determines the competitive advantage of that software.

Disclaimer

Open Data Spaces (ODS) is an open and scalable foundation for distributed data, built on organizational and national diversity by design. “Open Dataspaces” as a general term in this document refers to a new-generation distributed data management approach and its constituent concepts. Its design draws on the original dataspace papers in the U.S. (Franklin et al., 2005; Halevy et al., 2006) and data mesh (Dehghani, 2019; Dehghani 2022) as its core, and incorporates verification through collaborative R&D with private companies and industry groups at a commercial level. (**Table 1**)

Table 1 — Terminology of ODS and Open Dataspaces

Term	Meaning
Open Data Spaces（ODS)	The proper name of the technical concept for open and scalable foundation for distributed data, built on organizational and national diversity by design. The abbreviation ODS is used throughout this document.
Open Dataspaces (general technical term)	A general term used in this document to refer to the new-generation distributed data management approach and its constituent concepts, tracing their origins to the original dataspace papers and data mesh. Distinguished from the proper name ODS.

ODS is a different initiative from the dataspace architecture and related projects in Europe since around 2016. Open Dataspaces therefore follows different design principles from the technical specifications of International Data Spaces, Eclipse Data Spaces, and similar efforts. Currently, IPA serves as the technical design authority for ODS and as the secretariat for its global promotion activities. For the latest information on ODS activities as a technical concept, and for documents and source code related to Open Dataspaces as a distributed data management technology, please visit the official website.

Table of Contents

1. From Aggregation to Distribution: The Paradigm Shift in Data Management

Why Distributed Data Management?

To understand the necessity of this new paradigm, one must first grasp the history and challenges of non-distributed — that is, centralized — data management.

As the 21st century began, the volume of data that software handles grew explosively, ushering in the era of big data. Throughout this era, the focal concerns of data management evolved through volume, variety, and velocity. In 2001, data analyst Doug Laney famously proposed "**The 3Vs**" as a framework for explaining the fundamental characteristics of big data. From the early 2000s through the 2010s, the following technologies (among others) drove the data management field in response to the 3Vs:

- **Volume (around 2004–2006)**: HDFS in the Apache Hadoop ecosystem, and Amazon S3 as object storage
- **Variety (through around 2009)**: NoSQL databases like Document-oriented databases (MongoDB, etc.), Key-Value Stores (Redis, etc.), and Wide Column Stores (Apache Cassandra, etc.) that assumed flexible schemas; JSON as semi-structured serialization; and the expansion of data catalogs and ETL tools
- **Velocity (through around 2012)**: Apache Kafka for event-driven messaging with low latency; Apache Flink for stream processing; Apache Spark for large-scale real-time processing

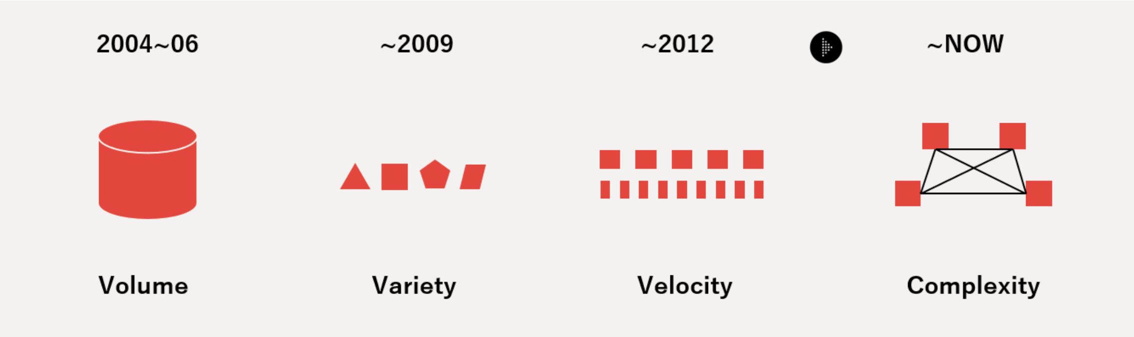


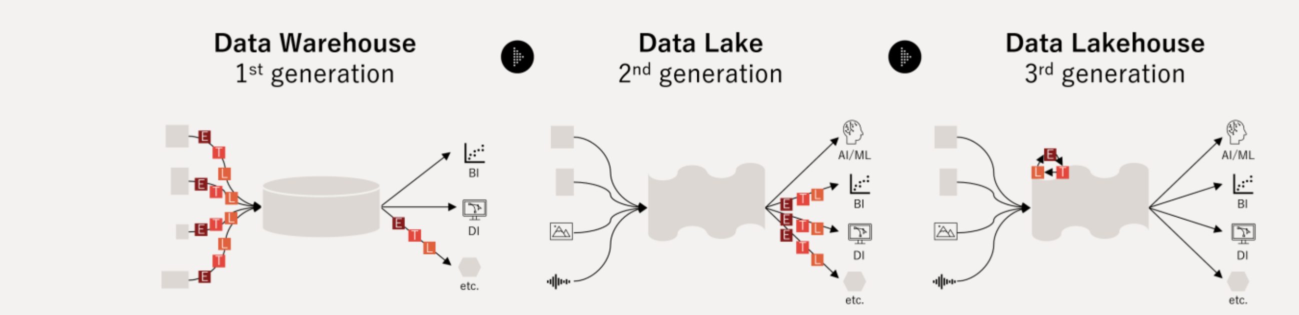
Figure 5 — Evolution of Focal Concerns in Data Management

Having navigated the 3Vs era, we now face the problem of **Complexity** in many contexts.

(**Figure 5**) This challenge is multi-dimensional. It spans not only technical aspects but also industrial, business, organizational, social, legal, and contractual dimensions.

On the technical side, to address first the 3Vs and then Complexity, the data management domain evolved from the classic Database Management System (DBMS) to the first-generation Data Warehouse (DWH), second-generation Data Lake (DL), and third-generation Data Lakehouse (DLH). (**Figure 6**)

With each generation, data use cases expanded beyond application-based BI and DI to embrace growing affinity with AI — reflecting recent training and inference needs. These technical advances also significantly improved Time to Value (TtV) for data users. From a business perspective, they contribute to expanding the overall software industry market size through data-driven decision-making, operations, and services.



Type of data	Structured Data	Structured, Semi-structured, Unstructured Data	Structured, Semi-structured, Unstructured Data
Usage	BI/DI	AI (ETL→BI/DI via DWH)	AI/BI/DI

Figure 6 — Generational Evolution of Data Management System Architectures

The Trade-offs of the Push and Ingest Paradigm, and Data Mesh

Despite these advances, conventional data management approaches remain in an aggregation-centric paradigm — one that stores and processes all data from inside and outside the organization in a single centralized location. This traditional "**Push and Ingest**" paradigm works very well for organizations with simple domain structures and limited use cases. But as domains diversify and data sources and consumers multiply, the growing volume of data increases technical complexity and raises utilization costs. The organization itself grows in complexity too, raising concerns that companies will hit the limits of scalability. In the age of Complexity, organizations need a solution for managing dark data — data that is scattered, siloed, and exists in heterogeneous structures and meanings — at a reasonable operational cost. In this environment, Zhamak Dehghani conceived the "**Data Mesh**" (Dehghani, 2019; Dehghani 2022) — a decentralized sociotechnical approach that offers a contrasting "**Serving and Pull**" paradigm to address the problems of Push and

Ingest. (**Figure 7**)

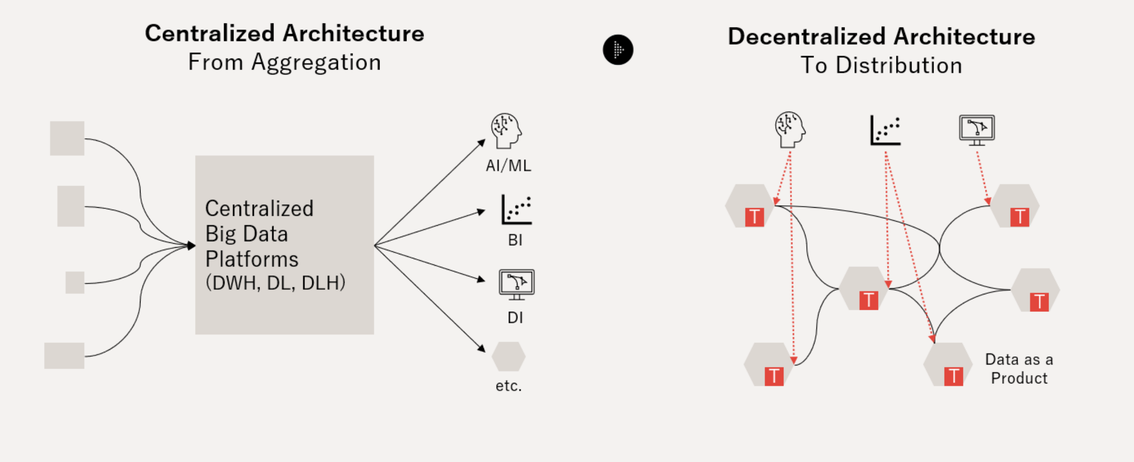
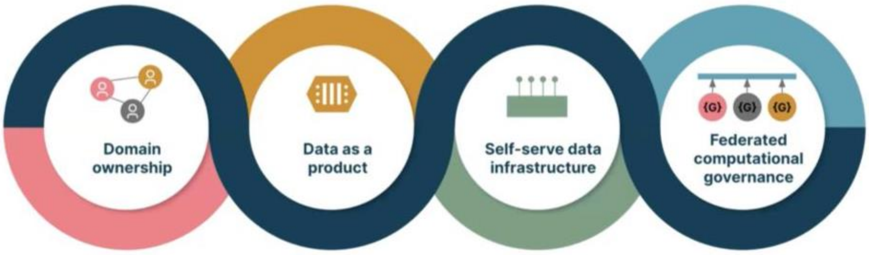


Figure 7 — Overview of the Distributed Data Management Architecture: Data Mesh

Data mesh achieves distributed data management through four principles: (1) Domain Ownership, (2) Data as a Product, (3) Self-Serve Data Platform, and (4) Federated Computational Governance. (**Figure 8**) While this document does not examine the data mesh paradigm or philosophy in detail, it was a critical turning point against conventional centralized architecture. Data mesh reconceptualizes **data as a reusable product across the department**.



Domain ownership
Data as a product
Self-serve data infrastructure
{G}
Federated computational governance

Figure 8 — The Four Principles of Data Mesh

The most notable achievement of this paradigm shift is that **data mesh returned the responsibility for data from the central data platform back to the business domains.** Enterprises have traditionally pursued organizational designs optimized for centralized data platforms, which led to the segmentation of Biz/Dev/Ops. This had the side effect of creating organizational silos. Data lost its context in the process of aggregation and transformation. Data consumers had to figure out meaning, constraints, quality, and usage conditions from code and documentation.

Data mesh asks the domain owners who generate data — PR teams, HR teams, product development teams, and others — to treat data not as a mere by-product of operations but as an explicitly designed and provided *product*. This conceptual shift made the fusion of Biz/Dev/Ops the optimal organizational design. Data utilization could then scale within the organization without relying on centralized coordination or pre-integration. Data mesh linked the technical Complexity of distributed data with the organizational Complexity of siloed Biz/Dev/Ops and presented the result as an architectural paradigm. This is precisely what justifies its claim to the title of “sociotechnical approach” and the reason the industry holds it in high regard. (**Figure 9**)

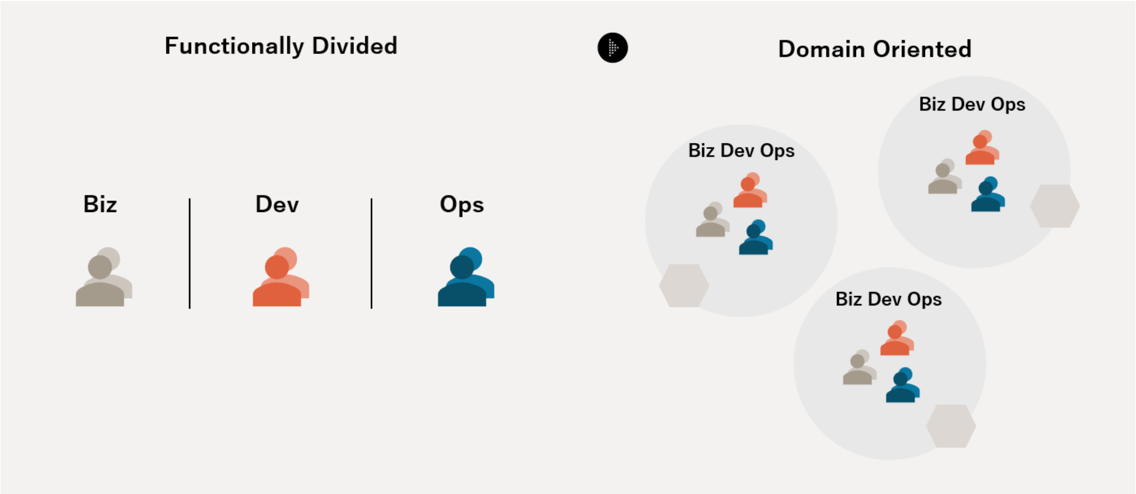


Figure 9 — Fusion of Biz/Dev/Ops Through a Domain-Driven Approach

However, data mesh is not a silver bullet for data management. Data mesh was proposed with an implicit assumption of collaboration across departments within a single organization. It has not fully addressed the **Governance Complexity** problem that emerges when crossing organizational boundaries.

The Governance Complexity problem has a cross-cutting nature that intertwines multiple dimensions: (1) the technical dimension, where data proliferates in heterogeneous sources, formats, identifiers, and meanings across and within organizations; (2) the organizational dimension, where ambiguity about data responsibility creates problems; and (3) the rights-based dimension, where access controls, usage conditions, and contractual practices become entangled when crossing corporate and national boundaries. This is the most typical challenge the distributed data management landscape faces today.

Is there a technical foundation that addresses Governance Complexity when crossing organizational boundaries? The answer lies in a concept called **dataspaces**, proposed in the U.S. 14 years before data mesh was born.

2. From Inter-Department to Inter-Organization: The Birth of "Dataspaces" in the U.S. Database Community

Why Dataspaces?

What exactly is the concept of dataspaces? As noted, the conceptual definition of **dataspaces** dates back to 2005. M. Franklin (UC Berkeley), A. Halevy (Google), and D. Maier (Portland State University) proposed "Dataspaces" to complement the database management system (DBMS) technologies of the time in handling Heterogeneous Collections of Data (HCoD) siloed within organizations. (Since activities calling themselves "dataspaces" have proliferated in many places, this document uses "**Classical Dataspaces**" to refer specifically to the concept first proposed in the United States, whenever distinction is needed.)

Franklin et al. (2005) and Halevy et al. (2006) focused their scope on two problem areas: (1) **architectural problems with relational DBMS and the associated** (2) **problems developers face**:

1. **Architectural problem**: It had become increasingly difficult to fit all data into traditional DBMSs, a single data model, or system.
2. **Problems developers face**: Developers had to address low-level data management challenges — search and query; enforcing rules; integrity constraints; naming conventions; tracking lineage; availability, recovery, access control; and managing evolution of data and metadata — across heterogeneous data collections on a case-by-case basis.

Classical Dataspaces introduced "a new abstraction of data management" to address these problems. The original papers defined dataspaces as follows (Franklin et al, 2005):

- A dataspace is defined as a set of participants and relationships.
- Participants are the individual data sources (e.g., relational databases, XML repositories, text databases, web services, software packages). Participants may be structured, semi-structured, or unstructured.
- A dataspace should be able to model any kind of relationship between two (or more) participants.
- Dataspaces can be nested within each other or overlap.
- A dataspace must include access rules between disparate dataspaces.
- The boundary of a dataspace can be fluid, but is clear in most cases.

These characteristics represent a clearly different approach from centralized Push and

Ingest. (**Figure 10**) Unlike data mesh, Classical Dataspaces does not prescribe how data sources themselves should be provided. Its most distinctive feature is establishing the concept of data management as "spaces" that exist in multi-faced, overlapping layers.

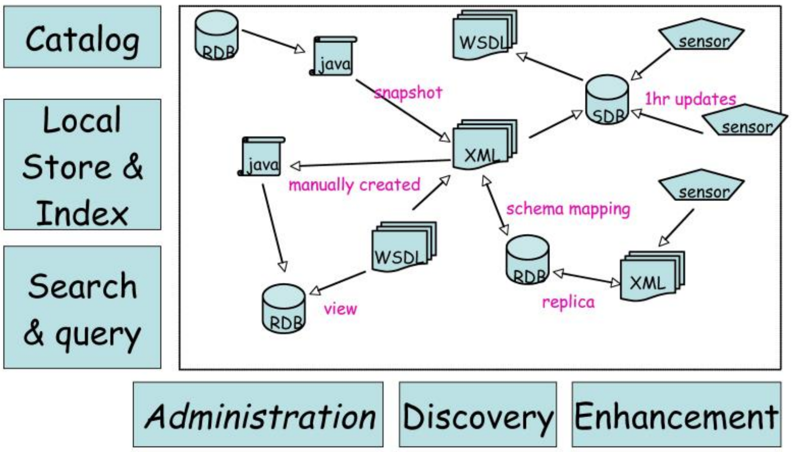


Figure 10 — An example dataspace and the components of a dataspace system. (Franklin et al. 2005: Figure 2.)

The term dataspace also appears in European initiatives. Starting around 2016, the Fraunhofer published a whitepaper (Otto et al., 2016) under the name "Industrial Data Space". A distinctly European architectural paradigm combining technology and institutional frameworks — renamed to "International Data Spaces" — has since been evolving.

What Did We Learn from Classical Dataspaces and Data Mesh?

Building on the original Classical Dataspaces paper (Franklin et al., 2005), Open Dataspaces was designed as a Distributed Data Management Architecture that escapes centralized management of HCoD and enables data management across organizations and national boundaries. Open Dataspaces reduces the burdensome data management costs for application developers dealing with HCoD, while ensuring the autonomy of both data providers and data consumers, and enabling the return of economic value derived from data. We can view this as the technical realization of the Data Free Flow with Trust (DFFT) principle — proposed at the World Economic Forum Annual Meeting (Davos) in Geneva, Switzerland in 2019, and incorporated into the Leaders' Declaration at the G20 Osaka Summit in June 2019 with the endorsement of world leaders (Digital Agency, n.d.).

Open Dataspaces inherits the architectural paradigm and four basic principles of data mesh. (**Figure 11**) It accepts the paradigm shift from the traditional Push and Ingest model to a Serving and Pull model, and therefore adopts a data management approach based on **Domain-Driven Design** (**DDD**).

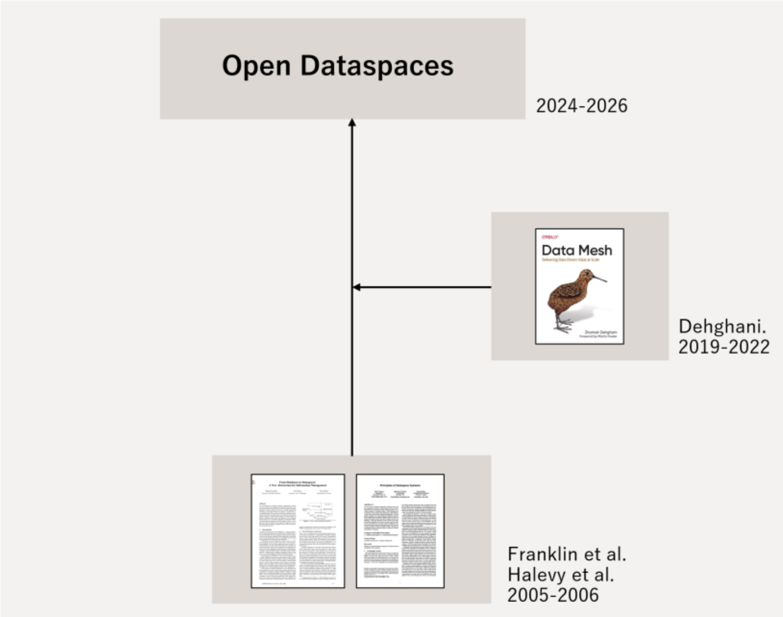


Figure 11 — The Theoretical Origins of Open Dataspaces

The most significant difference from data mesh is that Open Dataspaces is **a paradigm designed for inter-organizational data management, going beyond the inter-departmental** (**or, intra-organizational**) scope. (**Figure 12**)

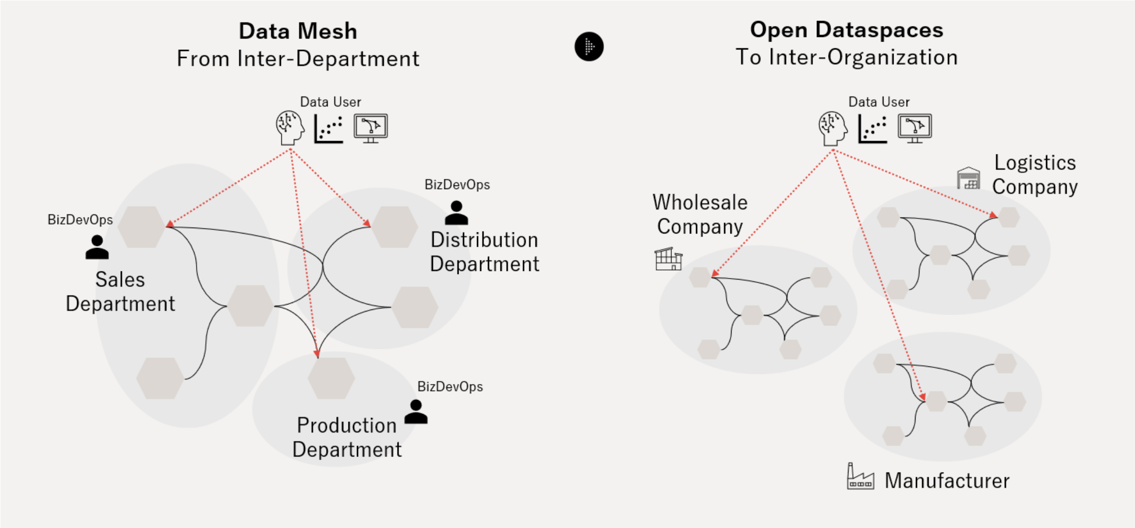


Figure 12 — Evolution from Data Mesh to Open Dataspaces

Why does a paradigm shift occur when the unit moves from departments to a cross-organization?

As noted earlier, data mesh has already encountered several operational challenges as its commercial adoption has progressed. These include schema and semantic inconsistency and fragmentation from each domain providing its own Data Product, security concerns in access control, and other Governance Complexity issues. We can broadly categorize them into three problems: **(1) Where to get**, **(2) What to mean**, and **(3) Who and How to use**:

1. **Where to get:**
   - **Where is the data in the first place?** (e.g., Where can I obtain production performance data from the manufacturing division of a company with whom I have an OEM contract?)
   - **Does *this* data refer to the same thing as *that* data?** (e.g., Is the Manufacturer's "6AX-10K Industrial Robot" the same as the wholesaler's "Robot 10kg Standard Model"?)
2. **What to mean:**
   - **What does this data mean?** (e.g., Is Alice from the government PR team affiliated with the PR department?)
   - **Is the meaning of *this* data consistent with *that* data?** (e.g., Does "altitude" in this field mean elevation above sea level, or height above ground level?)
3. **Who and How to use:**
   - **Who is trying to access the data?** (e.g., Which company operates this Agentic AI (Client App)? Does that company exist, and is it trustworthy?)
   - **Who can access this data?** (e.g., Can the HR department of a company commissioned by a partner airline view this data?)
   - **How must this data be used?** (e.g., What is the price per unit time for this streaming data? Can it be used for secondary purposes beyond BI?)

This is the Governance Complexity problem that arises with scaling. Data mesh is not unaware of these issues, but when domains cross organizational boundaries, and those organizations cross national boundaries, the governance costs that inter-domain coordination could once handle will explode exponentially. To deal with this growing pain of scaling, we must build solutions into the architectural paradigm from the outset. We must also account for the disruptive presence of (Agentic) AI, which had not yet reached this level of influence in 2005 when the original papers were written, nor in 2019 when data mesh was proposed. To briefly summarize the design philosophy to this point:

- Classical Dataspaces complement, rather than conflicts with, DBMS and data management platforms such as DWH, DL, and DLH.
- Open Dataspaces inherits the principles of Classical Dataspaces and the paradigm of data mesh. It is a complementary architectural paradigm and technical specification for realistically driving data sharing across organizations and national boundaries.

3. The Notability, Design Principles, and Technical Principles of Open Dataspaces

Why Open Dataspaces?

Put another way, Open Dataspaces proposes an architecture that addresses the Governance Complexity problem in handling HCoD across departments, organizations, and national borders. It also provides *order and benign discipline* as an open standard to achieve interoperability.

The notability of the Open Dataspaces paradigm is that it realizes transparent Single Source of Truth (SSOT) and a value-return mechanism for domain owners (as data providers) through a distributed architecture with three major pillars (**Figure 13**), premised on cross-organizational operation and the presence of Agentic AI:

1. Where to get: **Data Addressability and Discoverability** (**DAD**)
2. What to mean: **Ontology and Semantic Interoperability** (**OSI**)
3. Who and how to use: **Identity and Usage Control** (**IUC**)

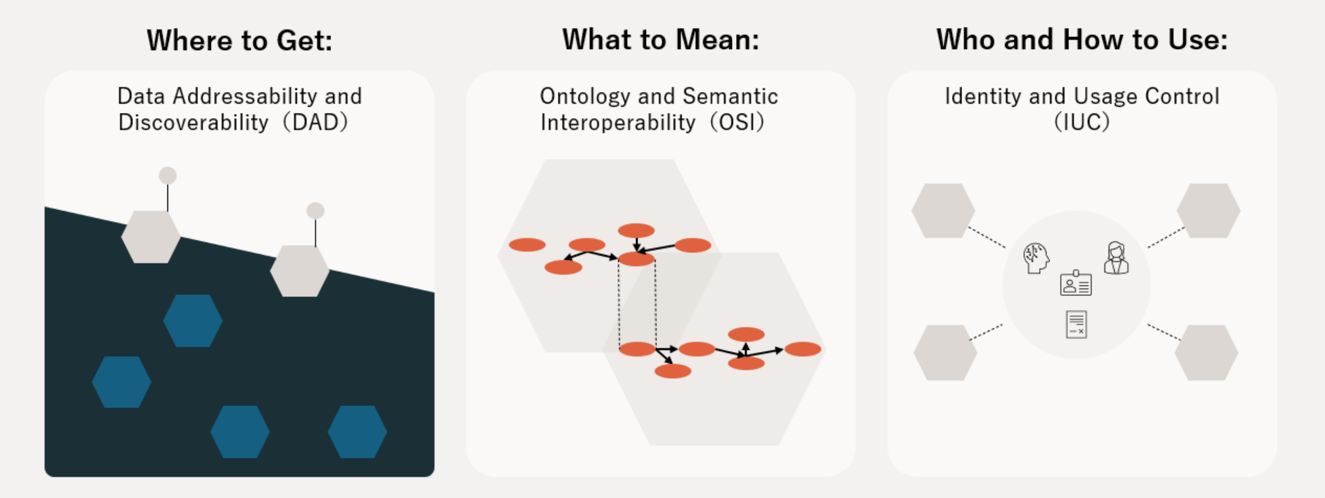

Where to Get:
Data Addressability and Discoverability（DAD）
What to Mean:
Ontology and Semantic Interoperability（OSI）
Who and How to Use:
Identity and Usage Control（IUC）

Figure 13 — The Three Pillars of Open Dataspaces' Distributed Architecture

DAD, OSI, and IUC each combine technical disciplines for ensuring Open Dataspaces' interoperability with rights-based disciplines for securing revenue opportunities for data providers. Importantly, in the Open Dataspaces architecture, protocols deliver the means of implementing discipline. The architecture assumes **"incremental adoption" in line with the characteristics and maturity of the market.** Open Dataspaces does not impose the full suite of protocol compliance costs on the market. Instead, it emphasizes a "Minimal Yet Viable" design that allows opt-in based on market demand driven by Make Money, Save Money principles. To achieve this, protocols are composed in a loosely coupled manner, with backward compatibility built in at the design stage. This philosophy of *benign discipline* draws on Classical Dataspaces' principle of "incremental payoff for incremental investment, and not exist only as monolithic solutions" (Franklin et al., 2005). For specific technical details on backward compatibility, see the Reference Architecture Model (ODS-RAM) and protocols (ODS Protocols).

Design Principles of Open Dataspaces

To realize the three pillars of the distributed architecture, Open Dataspaces adopts three core design principles: (1) **Vendor-agnostic**, (2) **Institution-agnostic**, and (3) **Product-Like, Service-Oriented Design**:

1. **Vendor-agnostic**: Open Dataspaces assumes multi-cloud and cloudless operation and adopts a vendor-agnostic design that does not depend on any specific company's services or products.
2. **Institution-agnostic**: Open Dataspaces explicitly separates the institutional and regulatory requirements of specific jurisdictions from its technical specifications. It is designed for localization under various institutions and regulations and provides an architectural paradigm and technical specifications that are adaptable globally.
3. **Product-Like, Service-Oriented Design**: The problems to be solved and the functional requirements always reside in the market. Essential needs do not lie in legal systems, regulations, or the products of specific vendors. Designers must work backwards from the innovations the market latently demands and pursue Product Market Fit (PMF) through agile validation. Rigid technical specifications become obsolete, and the market rejects them. Open Dataspaces places high importance on the question of whether something contributes to Make Money, Save Money.

Note that the "open" in ODS and in Open Dataspaces does not mean making data publicly available on the internet. This "open" signifies freedom from vendor lock-in and institutional lock-in, and the achievement of global-level interoperability — a point that is often misunderstood. More precisely, it refers to the mechanism that lets **data providers choose and control who they share data with, based on their own will, while simultaneously obtaining appropriate compensation**.

The above constitute the design principles of Open Dataspaces. We now define the minimum constituent unit of the architectural paradigm.

4. The Minimum Unit of the Architectural Paradigm: Architectural Quanta/Quantum

Architectural Quanta (AQ) is the minimum constituent unit of architecture in distributed data management. Data Product is the AQ of data mesh. To address Data Product's interoperability problem, Open Dataspaces newly introduces **the concept of "Ontology Product", which has a symmetric relationship with Data Product.** Open Dataspaces then **designates both Data Product and Ontology Product together as the constituent unit AQ**. This is the greatest differentiator from the data mesh paradigm. Open Dataspaces calls this AQ paradigm the "**Double-Product Quanta Model (DPQM)**". (**Figure 14**)

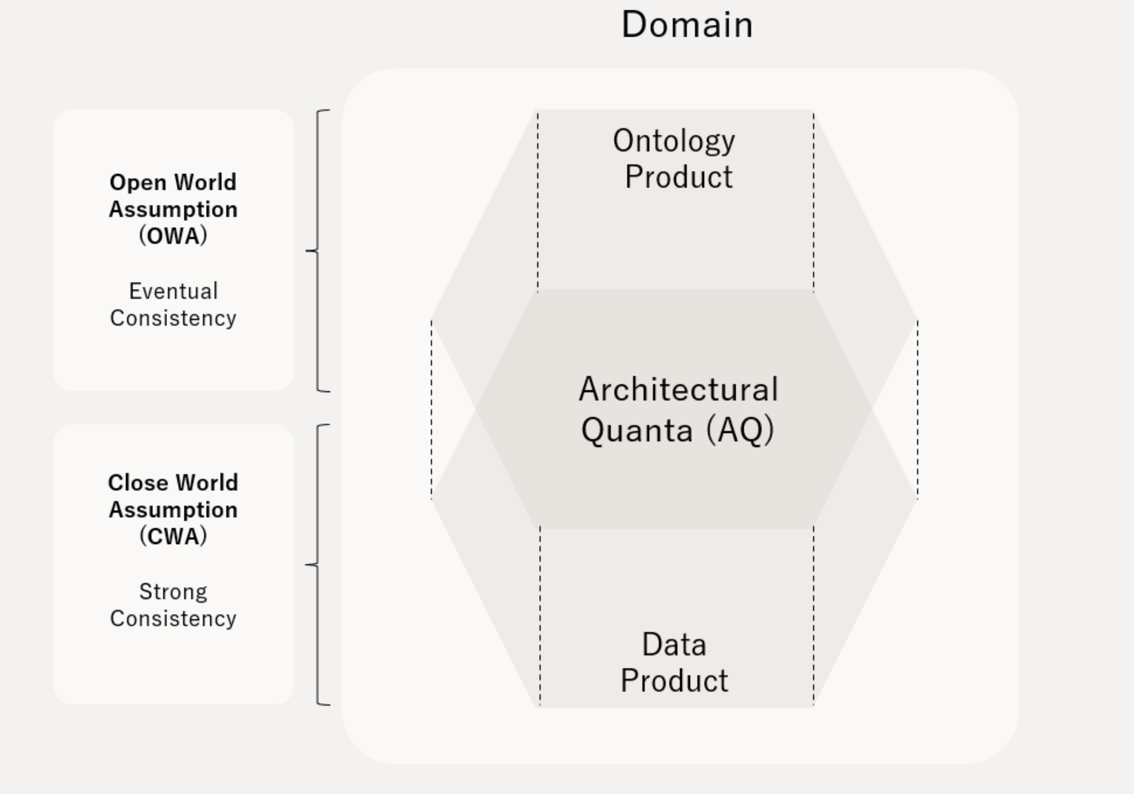

Domain
Open World Assumption (OWA)
Eventual Consistency
Close World Assumption (CWA)
Strong Consistency
Ontology Product
Architectural Quanta (AQ)
Data Product

Figure 14 — Double-Product Quanta Model (DPQM)

Assumptions of Open Dataspaces and the Strengths of DPQM

Open Dataspaces adopts the Serving and Pull paradigm and is oriented toward DDD for cross-organization and cross-border distributed data management. It places the following three points as *assumptions*:

1. Data providers provide data self-declaratively.
2. Data providers offer incomplete or evolving descriptions, but such incompleteness should not be grounds for exclusion.
3. Data consumers require explicit assurances regarding reliability, completeness, and consistency.

DPQM is the paradigm for resolving the trade-off between the expectations of data providers and consumers in distributed data management. The true strength of DPQM **lies not in enforcing homogenized completeness but in achieving selective robustness while maintaining the openness of the entire system**.

Open Dataspaces operates under the **Open World Assumption** (**OWA**) — the assumption that “what cannot be known as true is not false.” OWA is the most compatible approach for actively embracing the diversity of Data Products that domain owners self-declaratively provide. Under OWA, however, completeness cannot be guaranteed for undeclared information. The conventional centralized data management approach of the Push and Ingest paradigm rests instead on the **Closed World Assumption** (**CWA**) — the assumption that “what cannot be known as true is false”. This rigidness has historically satisfied data consumers’ expectations for reliability, completeness, and consistency.

Open Dataspaces therefore adopts a two-layer structure: OWA governs Ontology Product, while CWA is selectively introduced for Data Product. This design bridges the trade-off between the expectations of data providers and consumers. (**Figure 14**)

1. Consider two examples in the aviation industry:
   Static data searches — such as route information — can tolerate *Eventual Consistency*, which requires consistency to hold eventually but not necessarily at the moment of query. This is highly compatible with OWA-style management.
2. Flight operations management systems, on the other hand, must do more than search data. To prevent collisions, they must guarantee the consistency of dynamically changing flight plans and require synchronization that ensures the latest status. This is *Strong Consistency*, a domain requiring CWA-style management.

This represents a fundamental shift in the design philosophy of data management. Open Dataspaces emphasizes **liberation from the constraints of set theory**. The relational model that RDBMS pioneer Edgar Codd defined presupposes CWA — the idea that “all real-world data can be treated as a pre-defined set”. But the real data world is infinitely expansive; no one can pre-define it as a finite set. In cross-organizational and cross-border distributed data management, it is fundamentally impossible to pre-define who holds what kind of data. Open Dataspaces resolves this through a technical principle named “**Schema Flexible**” — instances (data) come first, and designers attach meaning afterward. This design draws its support from consistency with OWA. Readers will develop a deeper appreciation of DPQM’s robustness as they continue through this document.

Basic Terminology and Relationships

Let us also organize some basic terminology. (**An**) **Open Dataspace** is an Architectural Quantum (AQM) composed of two or more AQ. For example, the product domain AQ of a wholesale enterprise and the warehouse domain AQ of a logistics company together form a Wholesale Distribution Dataspace (AQM). Likewise, the product domain AQ of a wholesale enterprise and the production domain AQ of a manufacturing company form a Manufacturing Logistics Dataspace (AQM).

An Open Dataspace is dynamically and plurally composed. If a manufacturing company wishes to form an Open Dataspace for supply-demand optimization with a wholesale enterprise, it would encompass the AQs of those companies. The totality of such plurally and multi-layered set relationships is referred to as **Open Dataspaces** or “**The Open Dataspace**”. (**Figure 15**) (For convenience, this document uses “Open Dataspaces” hereafter.)

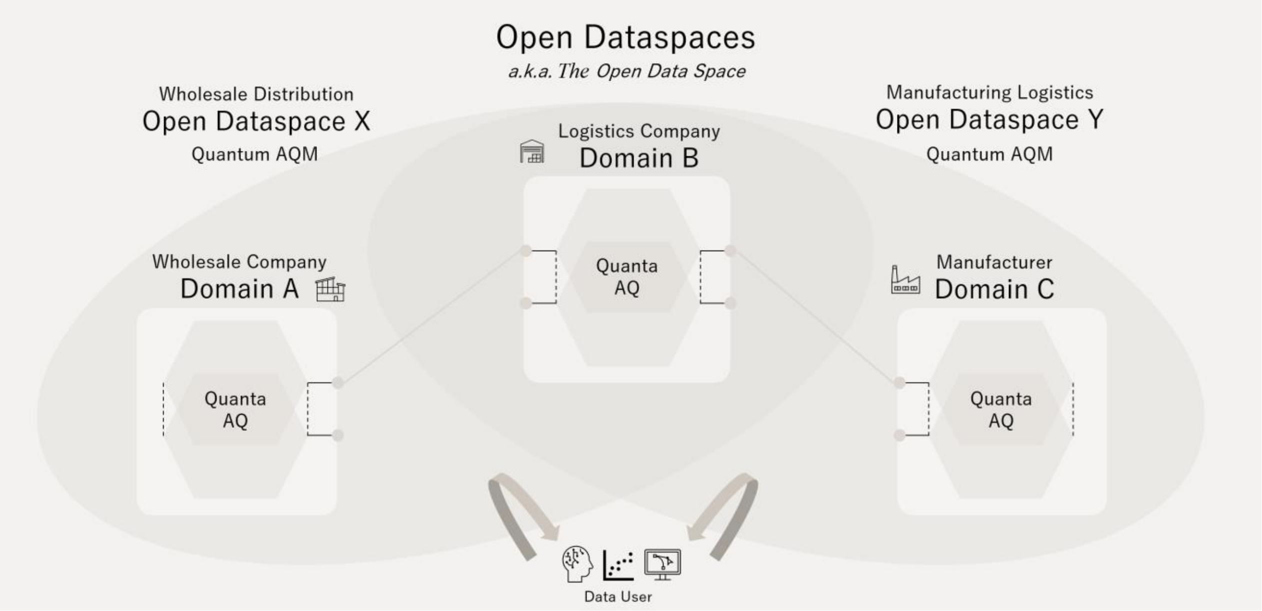


Figure 15 — The Constituent Unit (AQM) of Open Dataspaces

Between Ontology Product and Data Product in the AQ sit the API gateway, multimodal data source abstraction, and transaction management (structured, semi/unstructured, synchronous, and asynchronous data communication, etc.). Authentication and authorization functions as part of IUC are also built here in a loosely coupled manner. To achieve this, Open Dataspaces decomposes AQ into four logical layers for functional organization. (**Figure 16**) This document covers only the correspondence here; for details, refer to ODS-RAM. The reason for this layer structure will become clear as one reads through the document.

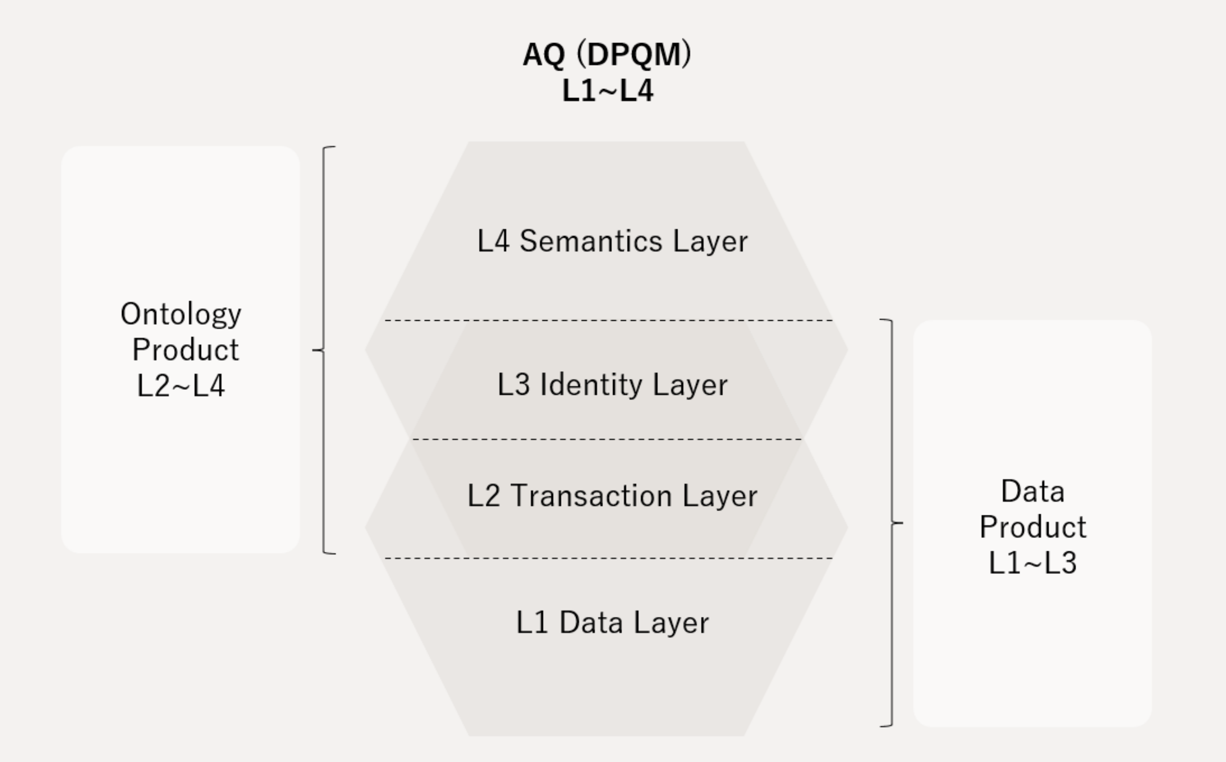


Figure 16 — Correspondence between DPQM and Functional Layers

Let us now examine OSI, DAD, and IUC at higher resolution.

5. Pillar 1: Ontology and Semantic Interoperability (OSI)

Ontology as a Product —

Separating the Information Model from the Data Model

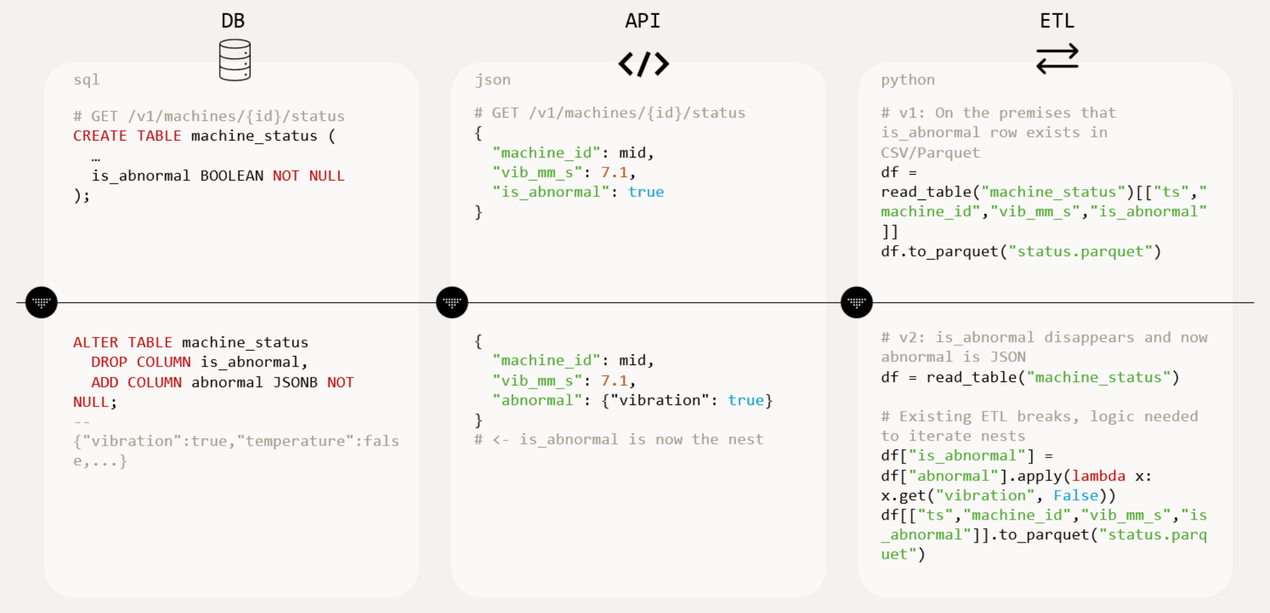
In distributed data management that crosses organization and national boundaries, the most primitive problem Data Products face — and the one that shackles scalability — is **semantics.** To address this problem, Open Dataspaces adopts DPQM, which explicitly separates the **Information Model** from the **Data Model.** This separation corresponds to the separation of Data Product and Ontology Product, respectively. The two are often conflated in the data management world, but they hold different roles and responsibilities in Open Dataspaces:

- **Data Model**: A structure and set of relationships that express the elements of observed facts — operational data, sensor values, transaction records, logs — and the elements generally equivalent to DBMS schemas and tables. It prioritizes efficient storage, retrieval, and updates, while minimizing meaning and interpretation. Facts once recorded are, in principle, not changed retroactively.
- **Information Model**: Expresses the meaning of how to interpret, associate, constrain, and operate on facts. (Cf. Lee, 1999, etc.) Semantic structures such as classifications, identifications, preconditions, and prohibitions are the responsibility of the information model (and its abstracted concept, Ontology). These can be updated as operations change.

Conventional information systems mix these two roles within the data model itself. As a result, changes in meaning are treated as changes in schema, which cascades into historical data, applications, and analytical logic. (**Figure 17**) Then, why is it effective to separate the data model from the information model? Consider a factory example:

- In manufacturing, data structures for equipment, parts, and processes serve for long periods. Meanwhile, the *meanings* assigned to them continuously change as quality standards are revised, safety regulations evolve, and operational improvements are made.
- Suppose a manufacturing plant initially tracked equipment only in terms of “operating/stopped” status. Following a quality incident, evaluation from the perspectives of “abnormal vibration,” “overload,” and “presence of manual intervention” becomes required. Conventional design would demand

changes to table definitions and API specifications, migration of historical data, and application modifications.



```sql
# GET /v1/machines/{id}/status
CREATE TABLE machine_status (
  …
  is_abnormal BOOLEAN NOT NULL
);
```

```sql
ALTER TABLE machine_status
  DROP COLUMN is_abnormal,
  ADD COLUMN abnormal JSONB NOT
NULL;
--
{"vibration":true,"temperature":fals
e,...}
```

```json
# GET /v1/machines/{id}/status
{
  "machine_id": mid,
  "vib_mm_s": 7.1,
  "is_abnormal": true
}
```

```json
{
  "machine_id": mid,
  "vib_mm_s": 7.1,
  "abnormal": {"vibration": true}
}
# <- is_abnormal is now the nest
```

```python
# v1: On the premises that
is_abnormal row exists in
CSV/Parquet
df =
read_table("machine_status")[["ts","
machine_id","vib_mm_s","is_abnormal"
]]
df.to_parquet("status.parquet")
```

```python
# v2: is_abnormal disappears and now
abnormal is JSON
df = read_table("machine_status")

# Existing ETL breaks, logic needed
to iterate nests
df["is_abnormal"] =
df["abnormal"].apply(lambda x:
x.get("vibration", False))
df[["ts","machine_id","vib_mm_s","is
_abnormal"]].to_parquet("status.parq
uet")
```

Figure 17 — A Typical Case Where “Change in Meaning” Causes Structural Disruption

The data model records observed facts, so reality’s changes strongly constrain it. The information model, which handles how those facts are interpreted, associated, and constrained, may be reinterpreted retroactively. If both are forced into the same schema, the needs of frequently changing data destroy meaning, and enormous database redesign costs follow. Indeed, **many system modifications in organizations are caused not by the data itself changing, but by the meaning of the data changing.**

In a design where the information model is separated from the data model, engineers can respond to changes not by modifying the equipment data itself, but by adding new concepts such as “abnormal state” or “safety risk” on the ontology. Updates to the information model conditionally classify existing equipment instances into new classes or attributes, while the original data is preserved. New evaluation axes can then be applied consistently across all equipment without destroying historical operational logs or maintenance records. Applications and analytical processes reference the information model to interpret meaning, so display items and determination logic extend without breaking existing interfaces.

In other words, Open Dataspaces treats the information model as an independent knowledge unit, not to preserve meaning, but to keep it in a state that can be reinterpreted. Separating the information model allows the inevitable evolution of meaning in real operations to be treated as reinterpretation rather than structural destruction. This is the strength of DPQM shown in **Figure 14**.

To reflect this design philosophy, Open Dataspaces uses Resource Description Framework (RDF) and RDF* to retain observed facts, vocabulary, and provenance as knowledge units. RDF Schema (RDFS) provides vocabulary and structure. Web Ontology Language (OWL) defines constraints such as validity and exclusivity. (**Table 2**) RDF is a model defined as a graph structure consisting of Subject–Predicate–Object. It is a knowledge unit designed to be distributable and extensible while retaining relationships even at stages where meaning has not been fully determined.

Table 2 — Differences in Scope between Data Product and Ontology Product

AQ	Data Product	Ontology Product		
Level	Data Model	Information Model（narrow definition）	Semantics	Ontology
Representation	Any Multi-Modal Raw Data	RDF (and RDF*)	RDF Schema	OWL
Purpose	Reflecting Reality	Knowledge-Unit	Vocabulary and Structure	Validation and Reasoning

Semantic and Ontological Interoperability — From Guess to Knowledge

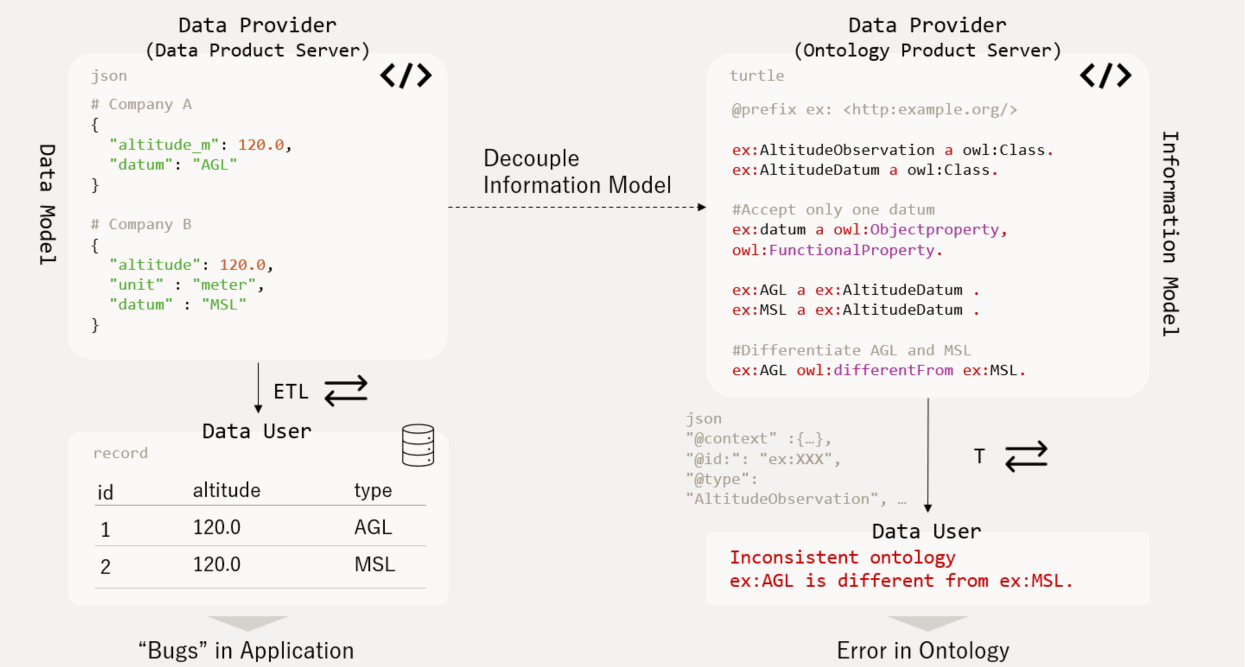
Let us dig one level deeper into the problem of meaning. In Open Dataspaces that handle HCoD across organization and national boundaries, the **semantic gap** arising from self-descriptions that a specific domain within a specific organization defines is a very significant interoperability challenge.

For example, “Altitude” carries two different meanings in the aviation industry: elevation above Mean Sea Level (MSL) and height Above Ground Level (AGL). Suppose an aviation operations dataset defines “altitude” as MSL, where cross-organizational integration proceeds on. A data consumer then erroneously assumes both usages refer to the same concept, unaware that another organization uses AGL for the same term. On the surface, integration appears to progress, but in reality, incorrect identifications accumulate, forming a debt that cannot be verified after the fact. This debt can then lead to incidents in flight operations management.

DPQM demonstrates its strengths here as well. By explicitly separating Ontology Product from Data Product, the design gives the ontology responsibility for explicitly constraining and guaranteeing semantic consistency — that is, detecting contradictions. At the implementation level, RDFS provides foundational vocabulary and relationships while OWL enables semantic reasoning, constraints, and error returns for semantic contradictions. **The origin of meaning lies in the domain. Domain owners, as data providers, define the Ontology; then data consumers interpret it.** This role assignment forms the cornerstone that ensures consistency with OWA.

Consider a more concrete example. In ordinary integration without separating the information model, differences in altitude reference standards are part of the data structure. Semantic misalignment then depends on application logic, which is implementation quality, and when consistency guarantees are insufficient, it surfaces as bugs. (Database constraints cannot express that AGL and MSL are mutually exclusive.)

When the information model is separated as an ontology, however, semantic constraints such as "altitude must belong to exactly one reference standard" and "AGL and MSL are different concepts" can be logically declared. Incorrect integration then returns as a logical inconsistency error, not a bug. (**Figure 18**) This is not a difference in data design. It is a difference in design philosophy — moving the location of meaning from code to a logical system.



```
json
# Company A
{
  "altitude_m": 120.0,
  "datum": "AGL"
}

# Company B
{
  "altitude": 120.0,
  "unit" : "meter",
  "datum" : "MSL"
}
```

record

id	altitude	type
1	120.0	AGL
2	120.0	MSL

```
turtle
@prefix ex: <http:example.org/>

ex:AltitudeObservation a owl:Class.
ex:AltitudeDatum a owl:Class.

#Accept only one datum
ex:datum a owl:Objectproperty,
owl:FunctionalProperty.

ex:AGL a ex:AltitudeDatum .
ex:MSL a ex:AltitudeDatum .

#Differentiate AGL and MSL
ex:AGL owl:differentFrom ex:MSL.
```

```
json
"@context" :{…},
"@id:": "ex:XXX",
"@type":
"AltitudeObservation", …
```

```
Inconsistent ontology
ex:AGL is different from ex:MSL.
```

Figure 18 — Conceptual Overview of Resolving Semantic Gap Through Separation of Information Model and Introduction of Ontology

Even when ontology resolves the semantic gap, data providers define each ontology self-declaratively and serve it as an Ontology Product. The gap problem between ontologies, therefore, continues to exist, referring to this as the **ontological gap**. The ontological gap is a significant problem for Open Dataspaces, which presupposes crossing organization and national boundaries. Two points in particular can become bottlenecks for scalability: ensuring consistency between domains in vocabulary and ontology, and the excessive burden that unit conversion and identification place on data consumers. In areas where the same parties act as both data consumers and data providers, those parties are expected to share ontological gap costs through duplication-avoidance decisions, and standardization at a higher conceptual level is expected to progress to some extent.

In areas with a high diversity of data providers, however, such cooperative behavior is extremely difficult. To perform the crosswalk needed to overcome the ontological gap in a dynamic and scalable way, Open Dataspaces incorporates LLM-based improvement of the ontological gap. Historically, the mapping work to bridge the ontological gap relied on labor-intensive, manual means — an approach with scalability challenges. By supplementing this work with LLMs, it becomes possible to bridge the ontological gap while maintaining scalability.

Why does this design work? The LLM proposes hypotheses about possible correspondences (or mappings) between different ontologies. Those hypotheses are then verified and constrained by the structure of each ontology itself. Through this interaction, even at stages where meaning has not been fully determined, organizations can recursively narrow the ontological gap during operations without stopping data sharing. Open Dataspaces calls this “**Dynamic Ontology**”. (**Figure 19**) Dynamic Ontology is also useful for advanced queries, such as hypothetical queries.

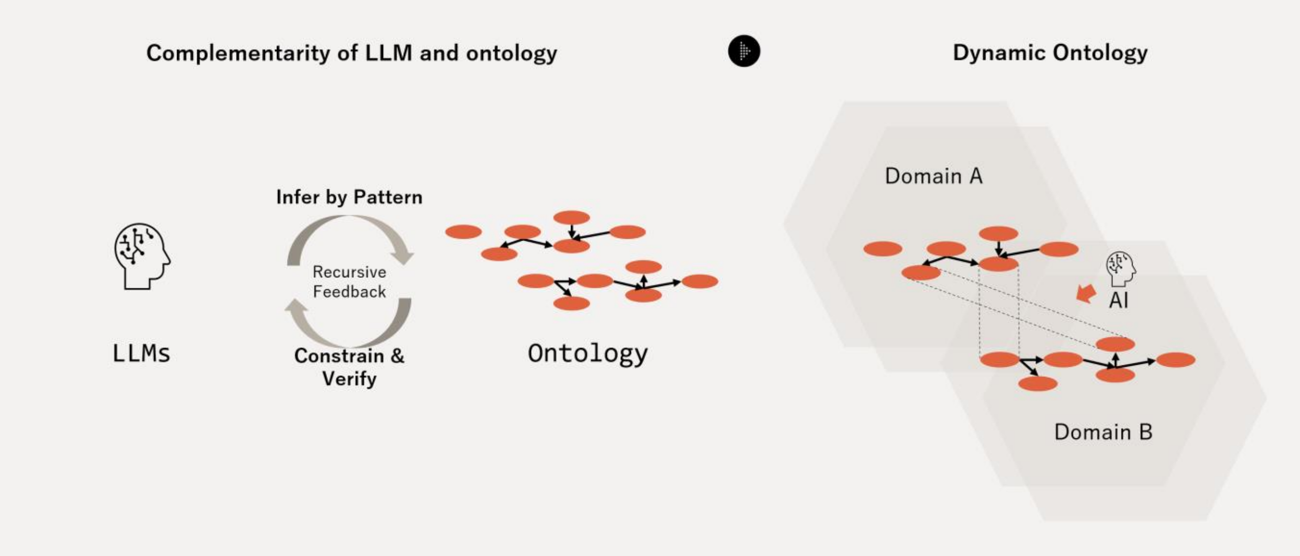


Figure 19 — Complementarity of LLM and ontology, and Dynamic Ontology

As described, the context to give to AI is expressed as a graph (“**Graph is Context**”). Open Dataspaces promotes a paradigm shift **from guess to knowledge** as an Agentic AI-native architecture. Note that this thinking also carries through to IUC in Chapter 7. Ontology and semantics in Open Dataspaces are not pre-completed designs. They are objects that operations continuously update. Ontology Product directly reflects the design philosophy’s awareness that carelessly accumulated data has low value and hence GIGO. Market forces will weed out quality problems in the ontologies that domain owners self-declaratively provide as data providers. This challenge is common to the effort to make real data AI-Ready. For scalability, domain teams should incorporate a human-in-the-loop — for making tacit knowledge explicit — into the dynamic ontology cycle, then gradually remove these “training wheels” over time. That said, semantics and ontology need not be viewed as enormous undertakings. As the saying goes, “A Little Semantics Goes a Long Way”.

OWL is a logical reasoner that can detect semantic contradictions; however, it cannot treat undeclared information as inconsistent since OWL itself operates under OWA. Deficiency detection is a validator problem. It requires the application side to declare completeness requirements, or the introduction of CWA-style validation, such as SHACL (Shapes Constraint Language). But enforcing CWA is incompatible with Open Dataspaces’ philosophy of pursuing loose integration through self-declaration and best effort. To resolve this trade-off, Open Dataspaces introduces a two-stage query concept in the next chapter.

6. Pillar 2: Data Addressability & Discoverability (DAD)

Data Addressability —

Guaranteeing the Existence and Identification of Data

Next, let us look from the perspective of potential data consumers vis-à-vis data providers. In distributed data management, the first barrier data consumers face is **addressability** — how to reach the data. Addressability has two components: *existence* and *identification*.

First, consider the problem of existence. As noted earlier, many organizations live with dark data that exists but cannot be used. From the perspective of distributed data management, however, this framing is not quite accurate. Data lacking its addressability is equivalent to non-existence. In an environment where data is distributed across organizations, consumers cannot know “what kind of data with what meaning and context exists in which domain” a priori. Even if the domain owner clearly manages the data on their end, it remains invisible and inaccessible to consumers. (**Figure 20**)

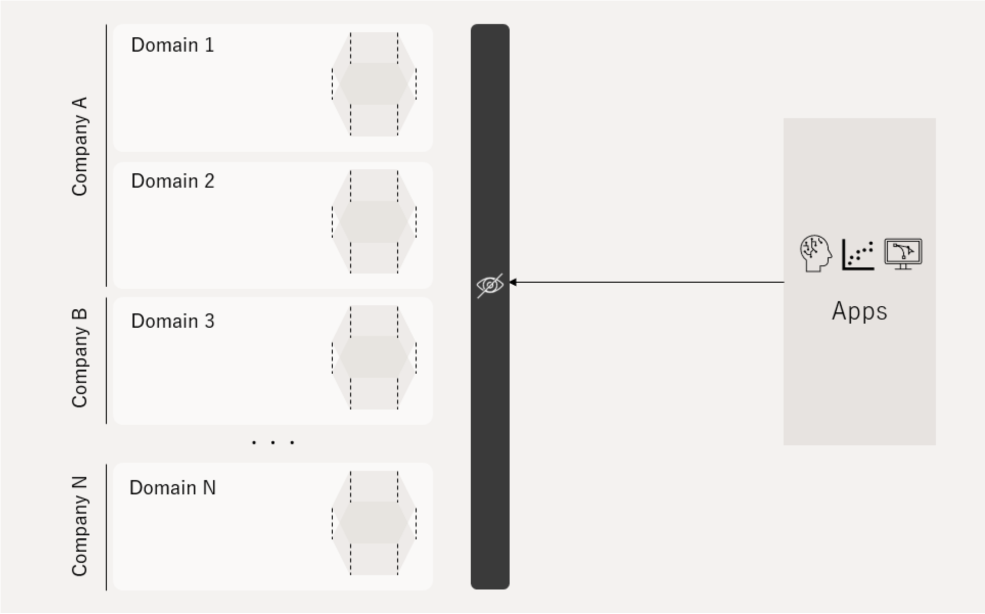


Figure 20 — The Addressability Problem

This structure resembles the challenges the Web has faced. Information on the Web is managed by actors distributed around the world and is unreachable without knowing the URL.

The Web overcame this problem not by managing everything centrally, but through a design premised on discoverability. The key was not fully enumerating destinations but providing a state from which discovery can begin. Open Dataspaces positions addressability as one of its pillars for precisely this reason — to make data that is invisible to others exist. Open Dataspaces, which adopts DPQM, exposes two independent interfaces to the outside: an *Ontology Endpoint* and a *Data Endpoint*. (**Figure 21**) These Web-based endpoints are the sole existence proof for an AQ to appear in Open Dataspaces, and the entry point for beginning discovery.

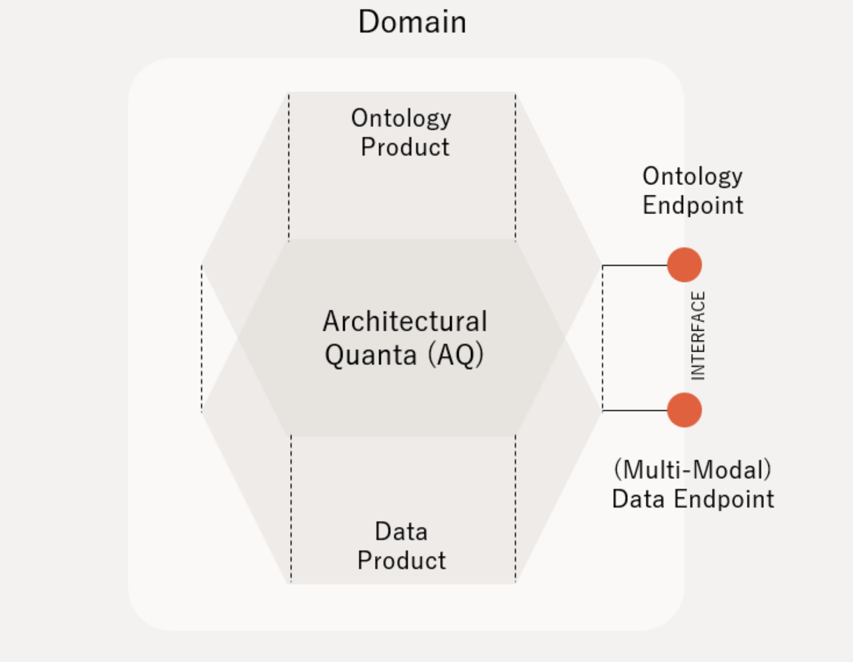


Figure 21 — Ontology Endpoint and Data Endpoint

With endpoints in place, data consumers can theoretically become aware of data's existence. However, if endpoints use different identifiers per domain, they cannot recognize the same entity across domains and organizations. This is the problem of identity.

Consider an industrial robot manufacturer. When each domain designs its own identifiers, the following problems arise within the organization:

- Is the “Prototype Rev.B” (e.g., design BOM) that the product development domain named the same thing as the “6AX-10K Line3” (e.g., manufacturing BOM) that the production domain defined?

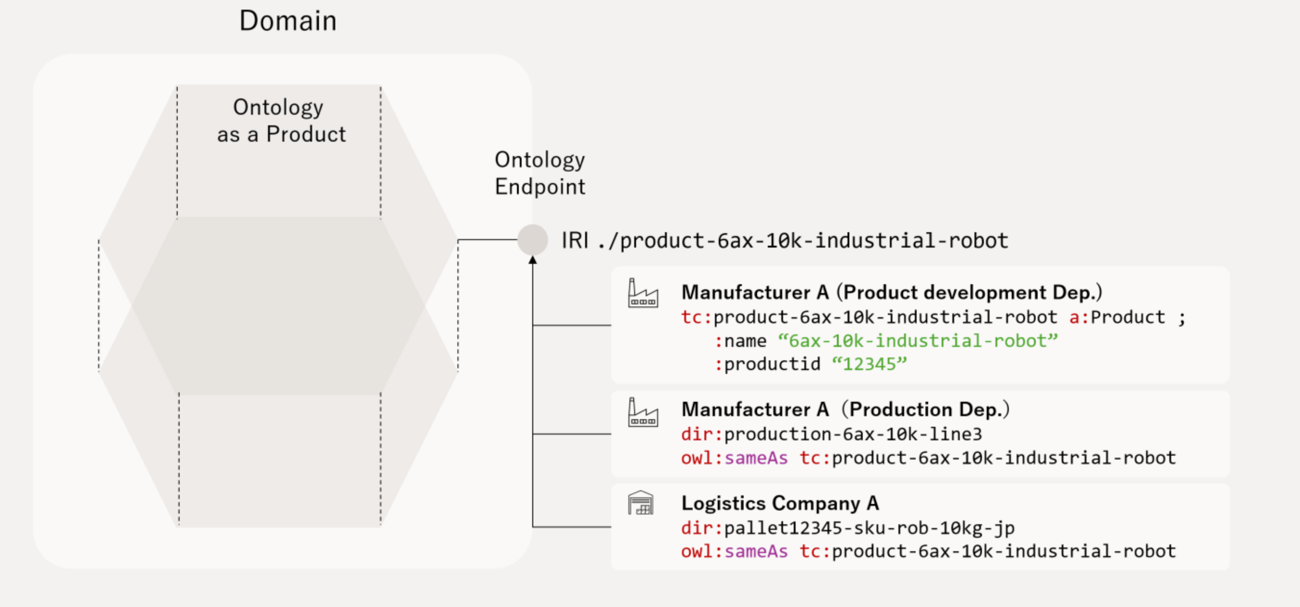
Product development and production departments handle products from their respective domain perspectives and contexts; therefore, it is natural to have different names and internal identifiers. But handling them in an integrated manner creates Data Product mapping costs in practice. At the AQM unit that crosses enterprises, when the same finished robot reaches a customer through B2B distribution, additional problems arise:

- The Manufacturer’s “6AX-10K Industrial Robot,” the wholesaler’s “Robot 10kg Standard Model,” and the logistics company’s “Pallet#12345/SKU-ROB-10KG-JP” all refer to the same unit. But names and internal identifiers differ depending on the contracts tied to the product, creating high costs when referencing in the contexts of returns, recalls, and maintenance contracts.
- Opening a new overseas factory requires multilingual support for identifiers.
- The same model number receives different identifiers at each installation site, making it impossible for the production department to track operational history, defect recurrence tendencies, and causal relationships in parts replacement for the same equipment.

Solving this through the traditional Push and Ingest paradigm requires — every time a department or company links to the data model — canonical ID redesign, entity pipeline reconstruction, mapping table creation, multilingual support, and so on. That is, an endless data integration. But in Open Dataspaces, where AQs are dynamically generated and connected, pre-defining identifiers that are correct for all cases is unrealistic.

Importantly, the identification problem Open Dataspaces addresses does not rest on the assumption that a single objective identity exists in the world and that the system guarantees it. The goal is to provide a means by which entities managed with different internal identifiers and contexts per domain can agree after the fact that they are “talking about the same entity” based on business necessity.

Given this background, DPQM of Open Dataspaces explicitly separates the Ontology Endpoint from the Data Endpoint and adopts IRI (International Resource Identifier) for the Ontology Endpoint. (**Figure 22**) In brief, IRI extends the URL concept to resources. An Ontology Endpoint self-defined and published via IRI functions as a globally unique identifier. This is wisdom from Semantic Web research, grounded in the concept of name resolution on which the Internet relies.


Domain
Ontology as a Product
Ontology Endpoint
IRI ./product-6ax-10k-industrial-robot
Manufacturer A (Product development Dep.)
Manufacturer A (Production Dep.)
Logistics Company A

```
tc:product-6ax-10k-industrial-robot a:Product ;
   :name “6ax-10k-industrial-robot”
   :productid “12345”
```

```
dir:production-6ax-10k-line3
owl:sameAs tc:product-6ax-10k-industrial-robot
```

```
dir:pallet12345-sku-rob-10kg-jp
owl:sameAs tc:product-6ax-10k-industrial-robot
```

Figure 22 — Global Existence Identification via IRI

IRI does not negate the different internal identifiers each domain uses. Rather, it provides a minimum point of agreement about existence on the Open Dataspaces, while preserving those identifiers. As shown in **Figure 22**, the production department of Manufacturer A and Logistics Company A can link their internal identifiers through post-hoc ontology expressions. This is the strength of the extensible IRI system. IRI provides the foundation for linking domain-specific identifiers in a mutually referenceable form.

Discoverability — A Loose Search Mechanism Providing Data Relationships

So far, we have focused on the addressability problem regarding data location. Data consumers can discover the Ontology Endpoint as an entry point and can externally judge the uniqueness of the entity that the pointer indicates. But at this stage, data consumers still do not know how to perform a standard query to search and cluster the relationships in the information model constrained by ontology — that is, to obtain a list pointing to actual data.

This problem necessarily arises when adopting a distributed architecture. Within a single organization, data is centrally aggregated, schemas and naming conventions as data models are controlled, and a comprehensive catalog independent of them can be maintained. In an environment premised on autonomous management per domain, however, no one can pre-construct a single correct catalog or comprehensive list.

To address this problem, Open Dataspaces introduces the concept of **discovery**. Two stages of query processes are required before reaching the actual data source. (**Figure 23**)

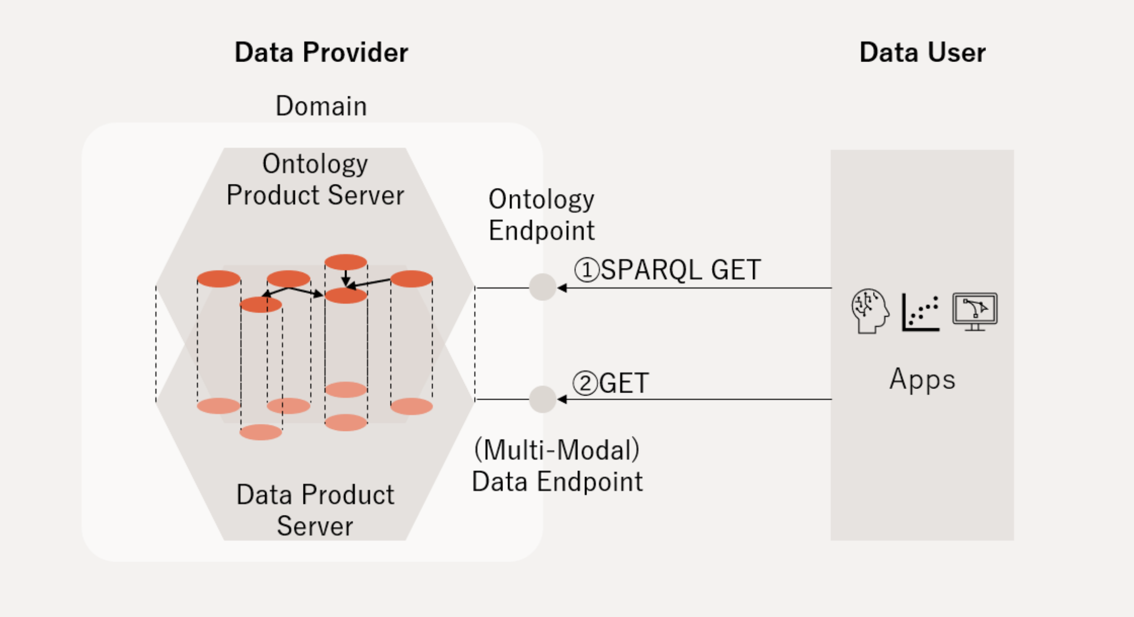


Figure 23 — Minimum Relationships Constituting Discoverability

1. **Ontology query** (**Stage 1**): An everything query presents a best effort result ("data catalog") for any keyword.
2. **Data query** (**Stage 2**): Access the endpoint of the data source linked to the graph presented as a best effort result.

Discovery in Open Dataspaces does not return complete answers. Data consumers who need data with guaranteed completeness must go directly to the data source — that is, Data Product. **A "Data Catalog" in Open Dataspaces is therefore not a static repository that someone permanently hosts (e.g., a data portal using Comprehensive Knowledge Archive Network (CKAN), a clearing house repository). It is rather a dynamic viewer that each query generates as a best effort result.** (**Figure 24**)

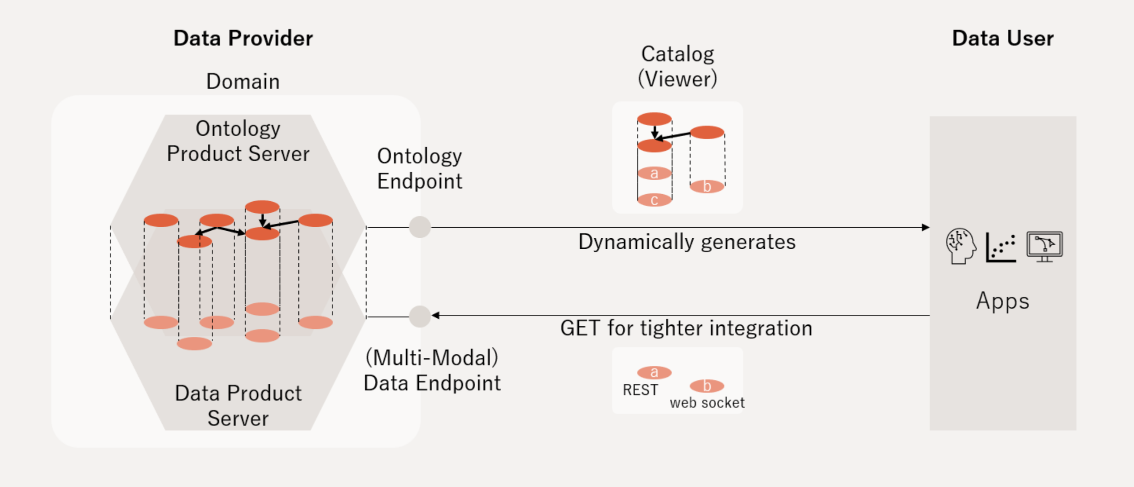


Figure 24 — Data Catalog as a Dynamic Viewer

All-to-all queries across HCoD at each instance are inefficient. For computation and running cost optimization, it is necessary to introduce mechanisms such as local stores and indexes — caches of discovered relationships — and cross-cutting crawling. Open Dataspaces therefore introduces a mechanism for Ontology indexing and crawling modeled on web search engines, called “**Distributed Catalog**”. (**Figure 25**)

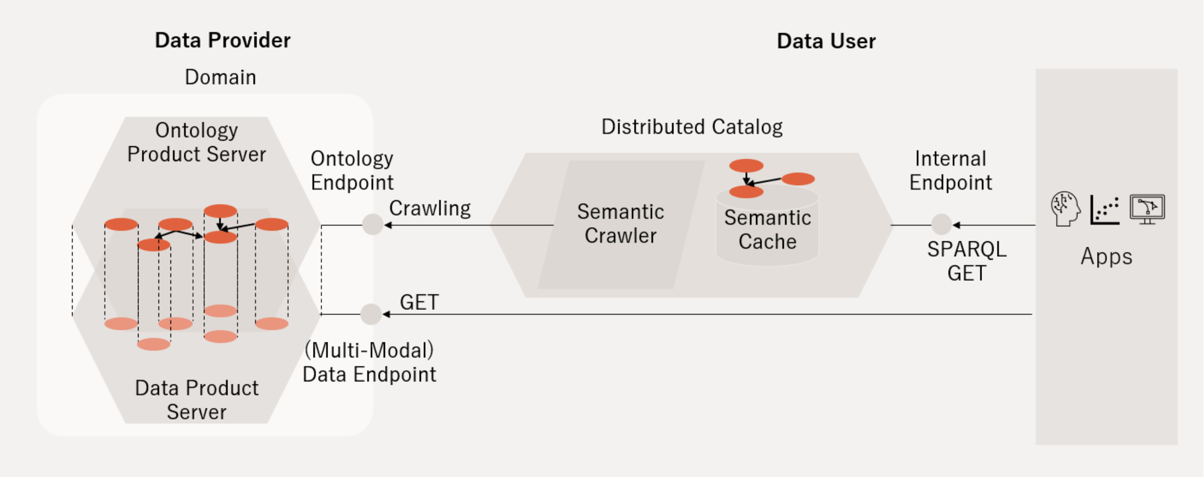


Figure 25 — Overview of Distributed Catalog

Services that provide the function of discovering such distributed catalogs and the first Ontology Endpoint — the search entry point for HCoD — are collectively called “**Discovery Service**”. Through these intermediary functions, data consumers will be able to explore Open Dataspaces more efficiently and reach more data sources. (**Figure 26**) Note that ontologies themselves often qualify as trade secrets. Accordingly, the IUC concept applies to Ontology Product in the same way it does to Data Product; see the next chapter for details.

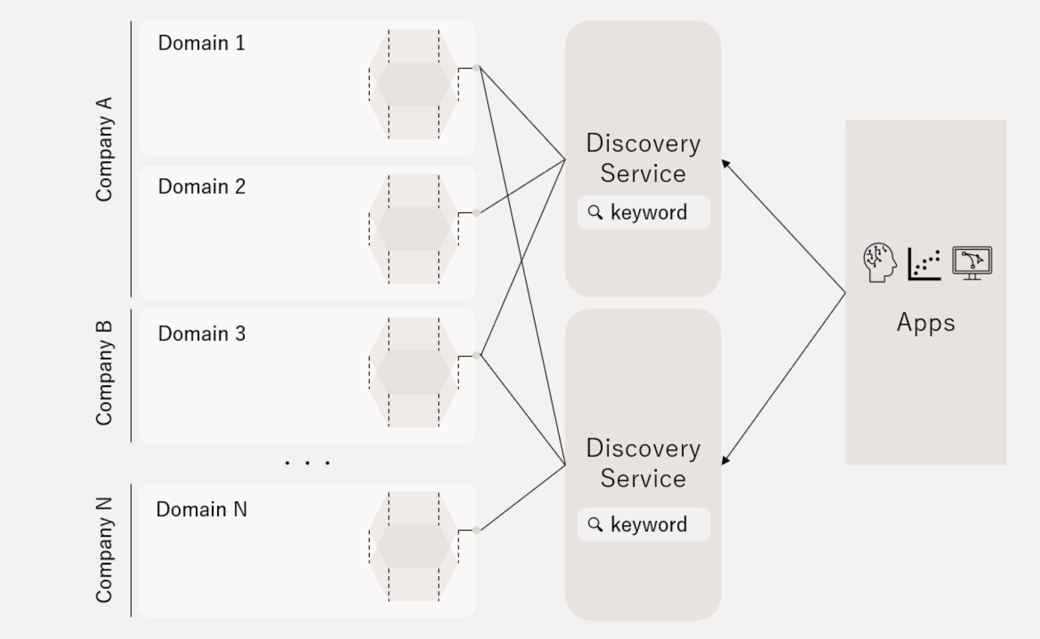


Figure 26 — Overview of Discovery Service

Supplementary Note: On Data Trust and Data Trustworthiness

Open Dataspaces operates under OWA, but it does not exclude the option of introducing CWA-style validation within limited boundaries — beginning with self-declaration at the unit of Data Product on the data model side. The two-stage query described in this chapter is the means Open Dataspaces uses to achieve this.

To introduce partial CWA-style validation, it is necessary to prepare a technically verifiable third-party interface for Data Trust — that is, assurance against uncertainty about the completeness of Data Product — and for lineage. ODS Protocols provides this as an optional choice through the “**Data Trust Assessment Protocol**”. Similarly, for quality, ODS Protocols optionally provides the “**Data Trustworthiness and Quality Assessment Protocol**” — an interface through which third parties can verify the indicators that domain owners present as Service Level Objectives (SLOs). This can include dynamic quality management, such as quality assessments like Great Expectations, and reduction of data downtime. Prescribing these in unified standards independent of domains is unrealistic. By leaving verifiability as a protocol, third parties can take on functions such as data auditing.

7. Pillar 3: Identity and Usage Control (IUC)

Identity — Trust by Design

Finally, let us look at IUC, the third pillar. In information systems within a single organization, *trust* operates as an implicit assumption between data consumers and data providers. In distributed data management, however, this premise breaks down and tends to become a more serious problem:

- **Failure of the assumption that organizational boundaries are fixed.** (e.g., Equipment manufacturers change on a project basis. Maintenance contractors change annually. A joint venture formation may have the same individual holding a position in a different legal entity.)
- **Failure of the assumption that the identity of a principal is self-evident.** (e.g., The same individual Alice acts as an employee of a pharmaceutical company, a contractor for a trading company, and a primary responsible person of an overseas local subsidiary.)
- **Failure of the assumption that an authenticated principal can be trusted.** (e.g., Login is possible with a legitimate identity, but an NDA has expired, a contract has ended, or the jurisdiction has changed outside the target business scope.)

Ignoring these structural problems causes operational failures: restricted logs become viewable; access temporarily breaks even when a contract is valid; teams cannot explain why a person or Agentic AI was able to view certain data. Designs that equate authentication with trust produce not merely incidents, but breakdowns of operations and auditing.

Open Dataspaces **treats trust not as an assumption but as a design target** (**Trust by Design**), taking such operational confusion arising from organizational fluidity as its premise. For Data management across organizations and national borders — not just within specific domains — trust must be built through design. This is why Open Dataspaces positions trust as a cross-cutting perspective, explicitly separated from the Identity Layer (L3), one of the layers into which AQ is logically divided. Only the correspondence is covered here; for details, see ODS-RAM. (**Figure 27**)

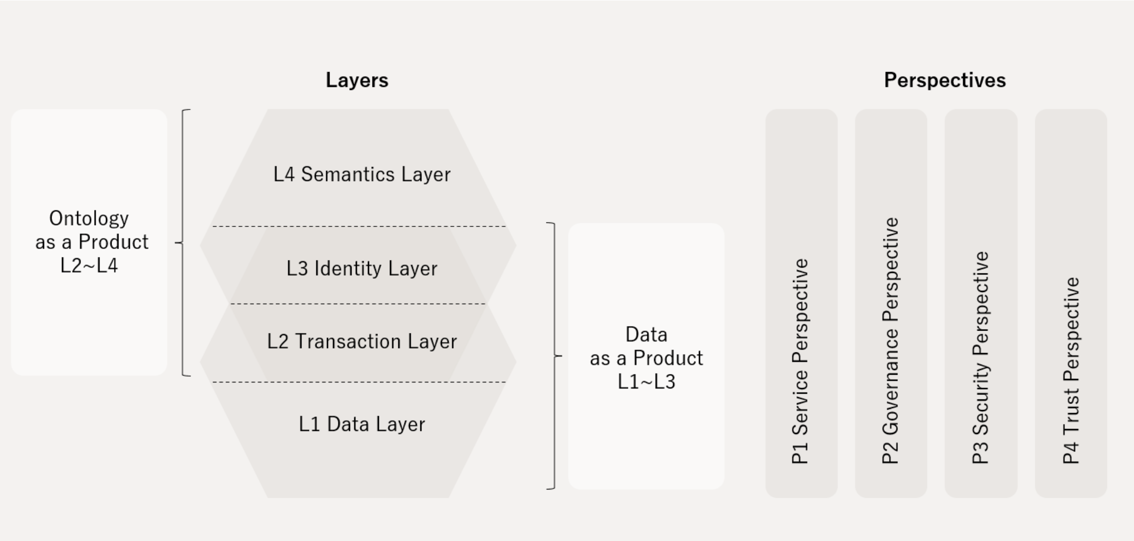


Figure 27 — Explicit Separation of Trust Perspective from the Identity Layer Logically Constituting AQ

From the perspective of Trust by Design, Open Dataspaces decomposes identity into three elements and assigns different responsibilities to each:

- **Identity Proofing:** Guarantees that the principal exists in the real world. To which individual or organization in the real world does this principal correspond? (e.g., Which legal entity employs or operates them? What position do they hold, linked to which contract, and until when is it valid?)
- **Authentication:** Confirms the principal's claim. Is this principal the owner of the claimed identity? (e.g., account, certificate, API Key, etc.)
- **Authorization:** Permits actions based on the relationship between principal and resource. May this principal now perform this operation on this resource? (e.g., operational logs for Equipment A are viewable, the control API for Equipment B is not.)

From the perspective of Trust by Design, the most critical principle **is not to equate an authenticated principal with a trustworthy principal**.

Open Dataspaces explicitly handles the basis for trustworthiness that corroborates that the principal exists. That basis must, for example, explain the correspondence with the real-world individual or organization, support expiration and renewal over time, and enable mutual explanation across companies and national borders. Importantly, Open Dataspaces **does not enforce a single basis for trustworthiness.** This reflects the design principle of institution-agnostic — corporate registration and personal identification systems differ by country and jurisdiction, and contractual practices and responsibility demarcation concepts differ by industry.

In Open Dataspaces, parties agree on the required trust level according to the risks and use cases of distributed data management. They then combine means such as verification, signature, and time assurance to build a use-case-appropriate trust structure. This allows Open Dataspaces to achieve not complete unification, but mutual connection of trust based on order and benign discipline. This is also an important concept for managing Non-Human Identity in the Agentic AI era.

As described later, Open Dataspaces treats the relationship between principal and resource as a graph for authorization. In this model, identity is not merely a collection of attributes. Rather, it is a node designed as the starting point of relational inference. From the perspective of Trust by Design, three requirements follow:

- A principal must be referenceable across organizations and systems.
- Attributes and relationships attached to a principal can change over time.
- Authorization decisions must always reflect the latest state of relationships.

If the actual existence of identity is not structurally guaranteed, these relational inferences lose legitimacy, and therefore, authorization results cannot be explained nor audited.

Open Dataspaces does not centrally integrate authentication and authorization. At each organizational unit that bundles domains, it manages the minimum identity of the principal and its correspondence with trust and authorization management units. At the domain-unit AQ, it manages relationships and authorization rules aligned with the operational context.

This separation of responsibilities produces three outcomes that constitute a trust structure guaranteed by design:

- Each organization can maintain its own operational policies.
- Even between organizations, responsibility boundaries can be maintained at the individual level.
- Trust expiration and renewal are immediately reflected in authorization decisions.

Identity design in Open Dataspaces is not merely about identifying someone. It is about maintaining trust — as a structure — that can be explained across organizations, time axes, and responsibility boundaries.

Access Control — Graph-to-Graph Control

After resolving authentication, the next view needed is permission of access to data as a resource — **access control**. Open Dataspaces operates under a zero-trust architecture. Rather than relying on lower network-level access management, it strengthens per-access-unit permission authorization from authenticated access principals (users, companies) to access targets (resources such as data and services).

For this, Open Dataspaces adopts a **PEP/PDP model** (Policy Enforcement Endpoint (PEP) / Policy Decision Endpoint (PDP)). **(Figure 28)**

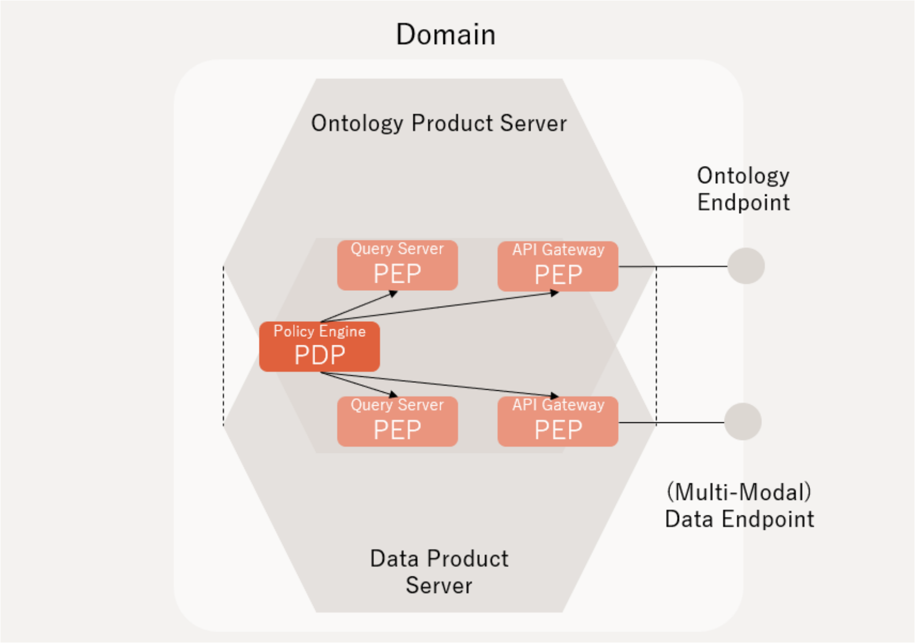


Figure 28 — Conceptual Overview of PEP/PDP Model in Zero-Trust Architecture

PEP sits at the point where actual requests pass through, such as the API gateways and query servers of both Data Product and Ontology Product. PDP sits as a policy engine within each domain and handles access decision-making. The essential benefit of this separation extends beyond simply combining coarse-grained and fine-grained controls:

- Centralizing policy decisions in the domain makes auditing and accountability easier.
- Policy changes propagate immediately to all PEPs, ensuring operational agility.
- Multi-layered enforcement at both the API level and data query level becomes possible.

Note that the AuthZEN specification, which seeks to standardize, has been progressing in recent years.

For the authorization decision model, Open Dataspaces adopts "**ReBAC (Relationship-Based Access Control)**", which enables sophisticated access management with ease. In ReBAC, the PDP expresses relationships between principal (identity) and resource — affiliation, contract, delegation, attributes, and so on — as a graph, then evaluates reachability and conditions to determine permissions. The key point is that the PDP does not perform probabilistic guesswork, but evaluation and derivation of relationships through access policy graphs. Note that Open Dataspaces does not deny adoption of other policy languages such as RBAC (Role-Based Access Control) or ABAC (Attribute-Based Access Control).

“**Graph-to-Graph Control**” (**Figure 29**) combines ReBAC with ontology graphs, and its true value is as follows:

- Given an identifier (IRI) called Alice, the PDP can **infer** (not guess) that she is human, belongs to a manufacturing company, that company has an NDA, and therefore she has access rights to certain resources.
- Given a pointer called Resource A, the PDP can infer the ontology linked to it and grant access to an appropriate range of graphs.

In this architecture, Alice can be replaced not just by a human being, but also by a crawler or an Agentic AI.

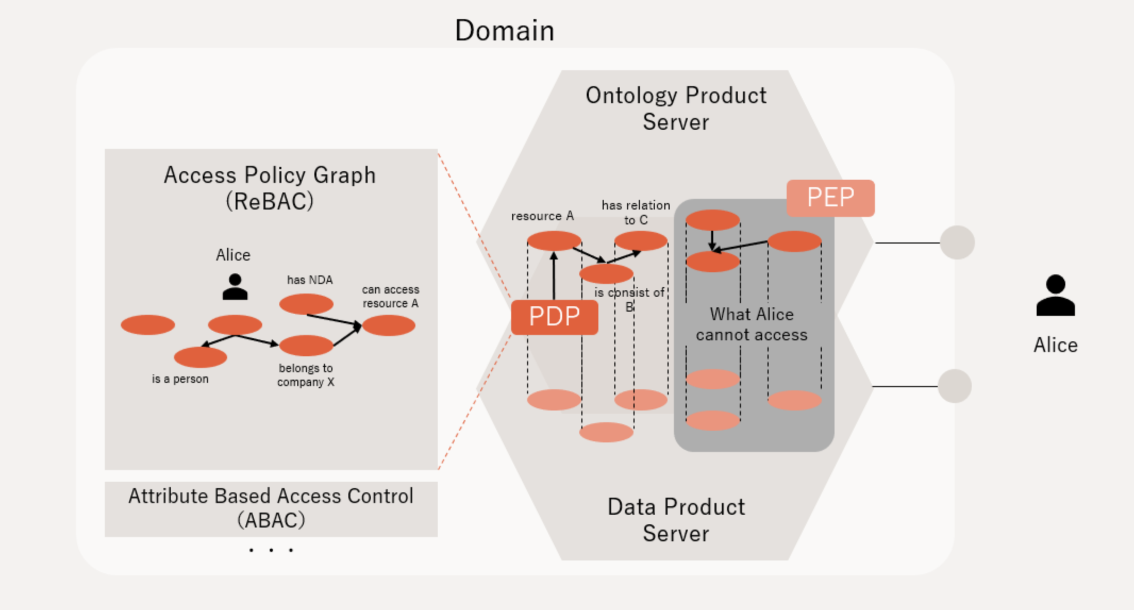


Figure 29 — Relationship between Access Policy Graph and Ontology

In access control, ontology does not perform authorization decisions. Its role is to define semantic boundaries — what constitutes the same resource or the same context. For example, if a certain Resource A is composed of multiple data elements and derived data, the Ontology expresses the compositional relationships and identifications. The PDP determines the scope of the resource graph to evaluate based on that ontology, then applies ReBAC to that scope. The division of roles is clear: Ontology provides semantic boundaries; access policy graphs determine permissions and prohibitions. This separation prevents the extension or reinterpretation of meaning from unintentionally expanding the access scope, while enabling flexible control.

Usage Control — Flexible Rights Relationship Options Independent of Institutional and Technical Dependencies

When we view data as a resource and product that generates future cash flows, introducing the concept of **usage control** becomes important as a market incentive for providing value returns to data providers.

Usage control extends access control to focus on the rights and obligations aspects of handling data. Steinbuss et al. (2021) define usage control as "the specification and enforcement of restrictions regulating what must (not) happen to data," governing the Provisions and Obligations of data processing. (**Figure 30**)

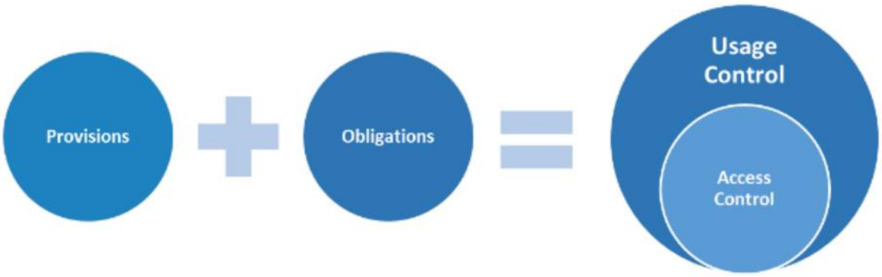


Figure 30 — Usage Control consists of provisions and obligations (Steinbuss et al. 2021: Figure 4.)

Usage control is a retrofitted concept that did not originally exist in Classical Dataspaces. Why does Open Dataspaces introduce it? To incorporate, from the design stage, corrections for two asymmetries between data providers and consumers:

1. **Asymmetry in rights-obligations**: If a data provider has no means of controlling the purpose and conditions of data use for a party to whom access to a data source endpoint has been granted, then the provider is blindly supplying data as a resource. Data providers must have appropriate incentives within market principles and must be able to control the use of their own data. This is an important factor that affects the scalability of Open Dataspaces itself.
2. **Asymmetry in pricing power**: If market principles enter distributed data management that approves self-declarative data provision while rights-obligations asymmetry exists, pricing power will effectively rest with data consumers. As stated in the premises of Chapter 4, data

consumers demand explicit assurances regarding reliability, completeness, and consistency. This means only data providers who can meet data consumers' expectations will be selected.

This mechanism brings positive effects, such as improvement of ontology quality on the data provider side. But since data consumers' optimal strategy is to use data at the lowest price and under the broadest purposes and conditions, the market structure will likely normalize cheap and disadvantageous data provision. This will inhibit the entry of providers with high-value data and can lead to market failure.

Planning for the concept of usage control at the architecture level is therefore essential. "**Data Contract**" plays a very important role in complementing this asymmetric rights-obligations relationship. There is, however, one important caveat: binding the means of usage control — which is closely tied to rights and obligations — to a single, inflexible technical protocol creates a high risk of becoming a major bottleneck for market introduction.

The technical architectures and means for achieving the objective of usage control are diverse, for example:

- Blockchain-based confidential computation methods in the BI/DI domain (e.g., calculating total carbon footprint across suppliers)
- Differential privacy training in data clean rooms in the AI domain
- The cutting-edge research advancing in Machine Unlearning

Therefore, it is the market that selects the means for usage control based on economic rationality. Under this awareness, Open Dataspaces does not restrict technical means related to usage control. It provides only interfaces, allowing various methods that do not presuppose jurisdictional or institutional homogeneity. (**Figure 31**)

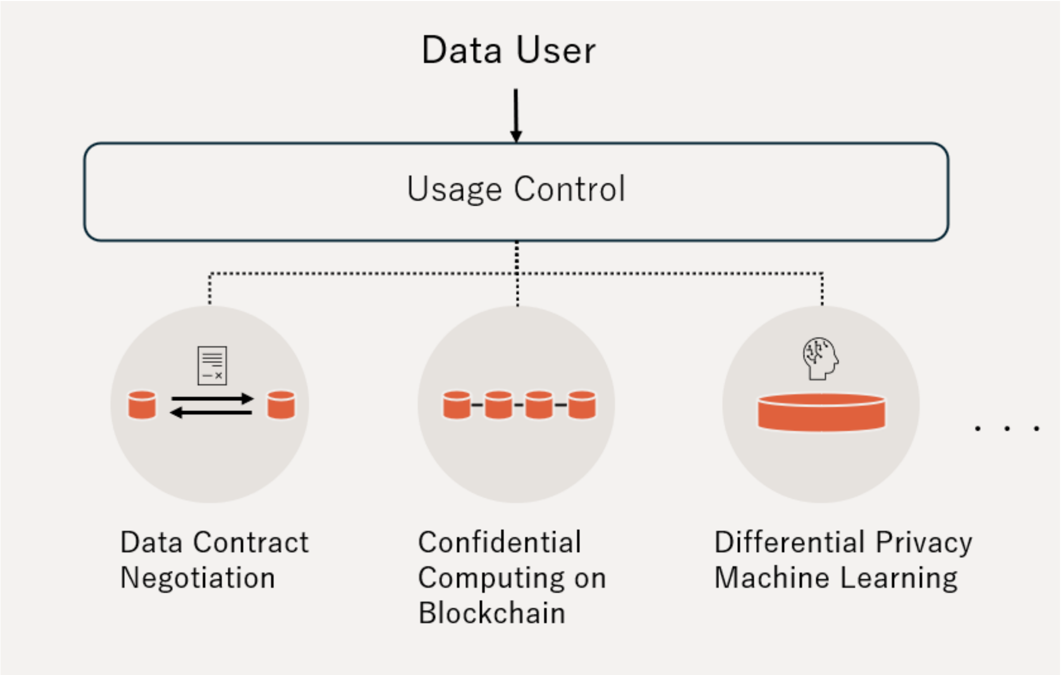

Data User
Usage Control
Data Contract Negotiation
Confidential Computing on Blockchain
Differential Privacy Machine Learning
. . .

Figure 31 — Diversity of Usage Control

In that sense, approaches such as contract negotiation—which link data with rights for exchange and negotiation—should be treated as just one of several possible methods for usage control. ODS also conducts international activities for interconnection verification that assume contract negotiation as one pattern, but this is merely one option among many technical means. It is important to note that forcing technical means derived from institutional requirements rather than market requirements may impose adaptation costs on companies that exceed the benefits of complying with that standard.

Supplementary Note: On Electronic Contracting

To support the various legally and technically existing usage controls in Open Dataspaces, a technical interface for electronic contracting (e-Contracting) is optionally available as the "**Heuristic Contracting Protocol**". The Heuristic Contracting Protocol covers the scope from electronic contract formation through to reflecting the results in the PDP. (**Figure 32**) This protocol does not include its own contract negotiation procedures. Rather, it targets connections with third-party applications that provide electronic contract services where a large market already exists.

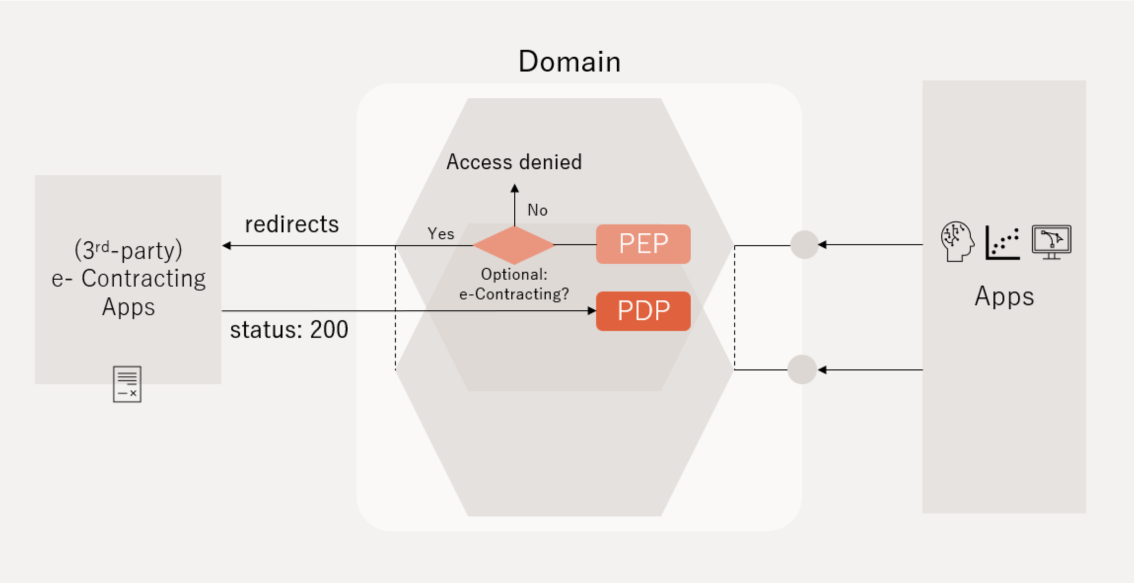


Figure 32 — Conceptual Overview of Heuristic Contracting Protocol (Optional)

Currently, data contracts in distributed data management are often operated through pre-contracts based on pre-established contract templates. Based on prior discussions in UN/CEFACT e-Negotiation (**Figure 33**) and the reality of global corporate legal practice and commercial customs, **it is unrealistic at this time to require automated completion of contract negotiation for data sharing *as mandatory*. Contracts always contain a margin that cannot be reduced to machine-readable form.**

Allowing asynchronous human-in-the-loop callbacks for final judgment by corporate legal departments is also important in real business, not just machine-only completion. This is a measure that takes into account the relationship between field operations handling day-to-day contract processing and the legal department providing supportive judgment for complex interpretations. Toward a future where human-in-the-loop approaches near-zero even across jurisdictions, we look forward to technical progress in electronic automated contracting and legal/audit practices, and to the maturation of society.

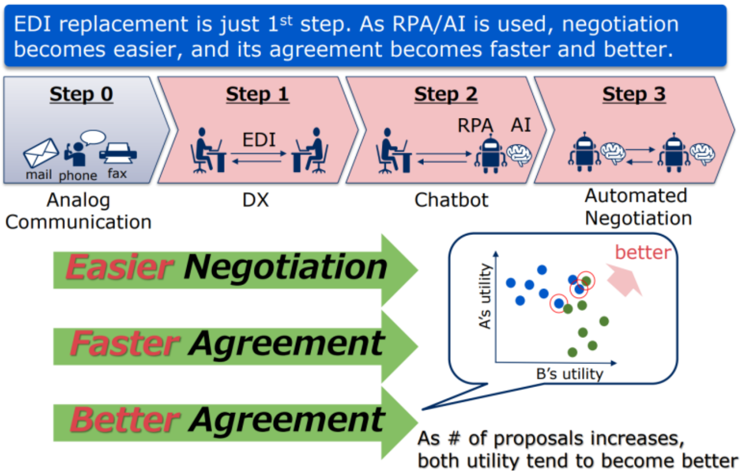

EDI replacement is just 1st step. As RPA/AI is used, negotiation becomes easier, and its agreement becomes faster and better.
Step 0
mail
phone
fax
Analog Communication
Step 1
EDI
DX
Step 2
RPA
AI
Chatbot
Step 3
Automated Negotiation
Easier Negotiation
Faster Agreement
Better Agreement
better
A's utility
B's utility
As # of proposals increases, both utility tend to become better

Figure 33 — Gradual Evolution of e-Negotiation Assumed in UN/CEFACT (Nakadai et al. 2020: p.7)

Supplementary Note: On Clearing, Billing/Payment

Likewise, an optional technical interface called "**Clearing and Payment Protocol**" is available for settlement and billing upon use of Data Products by data consumers, following the same concept as for contracting. (**Figure 34**)

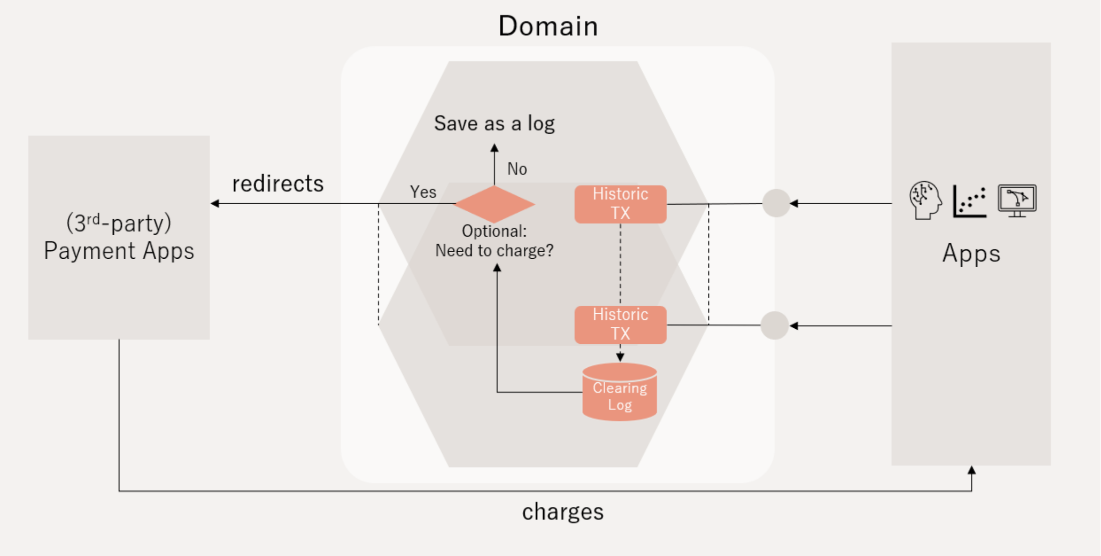


Figure 34 — Conceptual Overview of Clearing and Payment Protocol (Optional)

Combining these technical interfaces for usage control — heuristic contracting, settlement, and billing/payment — enables advanced services such as data marketplaces capable of handling distributed data.

To realize these optional technical interfaces and functions, Open Dataspaces positions cross-cutting transaction logging and monitoring as "**Common Functionality**", leaving room for advanced monitoring functions such as event detection, alerts, and notifications. This document does not cover everything; please refer to the technical specifications in ODS Protocols for details.

8. Service Model

As a final topic, let us describe the two implementation patterns as an assumption at the architecture design phase — developed through dialogue with the market — for how to implement the DPQM and distributed data management based on DDD.

The most orthodox method of implementing the Open Dataspaces architecture is (1) **the approach in which domain owners themselves build a Self-Serve Data Platform and provide Data Product/Ontology Product based on DPQM (referred to as the "Distributed Service Model")**. This approach, however, is only feasible for large enterprises with significant digital resources that can build and operate their own environments. SMBs, for example, often lack the financial capacity to provide the functions necessary for compliance as a domain owner and face implementation challenges.

Therefore, Open Dataspaces also assumed (2) **an approach in which a managed service provider delivers the basic software stack that constitutes DPQM on behalf of the domain owner, while the domain owner retains responsibility for providing Data/Ontology Product (referred to as the "Federated Service Model")**. In Open Dataspaces, the managed service business providing this core technology is referred to as a "**Dataspace Service Provider (DSSP)**", following the terminology used in Classical Dataspaces.

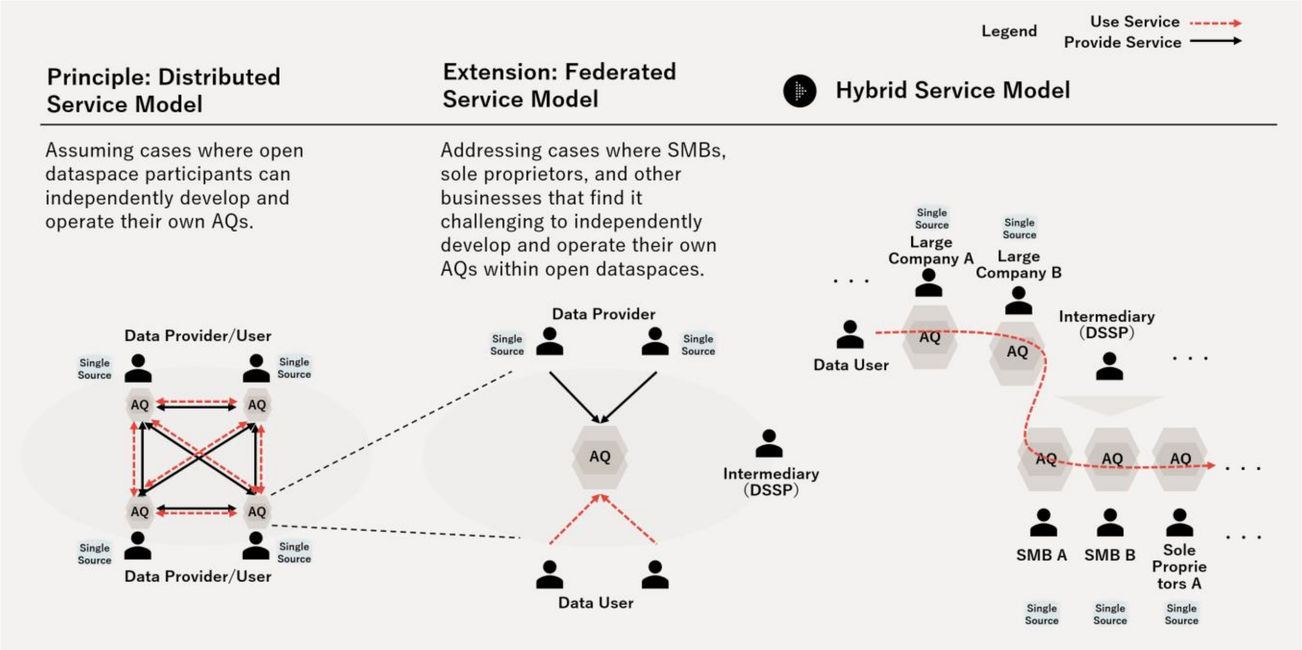
Reference: Definition of DSSP in Classical Dataspaces (Franklin et al. 2005)

Note: In classical data spaces, DSSP is referred to as a "Dataspace Support Platform."

- Must deal with data and applications in a wide variety of formats accessible through many systems with different interfaces. A DSSP is required to support all the data in the dataspace rather than leaving some out, as with DBMS.
- Although a DSSP offers an integrated means of searching, querying, updating, and administrating the dataspace often the same data may also be accessible and modifiable through an interface native to the system hosting the data. Thus, unlike a DBMS, a DSSP is not full control of its data.
- Queries to a DSSP may offer varying levels of service, and in some cases may return best effort or approximate answer. For example, when individual data sources are unavailable, a DSSP may be capable of producing the best results it can, using the data accessible to it at the time of query.
- Must offer the tools to create tighter integration of data in the space as necessary.

Businesses providing DSSP are sometimes called "**intermediaries**". Intermediaries expand the reach of Open Dataspaces by establishing service relationships (that is, federating) with domain owners who perform usage control of data, through the provision of DSSP.

Open Dataspaces calls the mixture of domain owners onboarding on their own and onboarding through DSSP intermediation a "**Hybrid Service Model (HSM)**". (**Figure 35**) In the use case for realizing traceability management of vehicle storage batteries, for example, Tier 2 and beyond suppliers to OEMs and Tier 1 consist of a high proportion of SMEs. Cases have emerged where a neutral service operating entity — jointly established by the automotive and battery industry — has begun providing DSSP for battery CFP data as an intermediary.



Principle: Distributed Service Model

Assuming cases where open dataspace participants can independently develop and operate their own AQs.

Extension: Federated Service Model

Addressing cases where SMBs, sole proprietors, and other businesses that find it challenging to independently develop and operate their own AQs within open dataspaces.

Hybrid Service Model

Figure 35 — Overview of HSM

For the role of intermediaries in HSM and the concept of ensuring usage control for domain owners, please refer to related documents, including ODS-RAM.

Conclusion

The above constitutes the design philosophy, architectural paradigm, and characteristics of Open Dataspaces. For more specific explanations, readers are referred to the design artifacts: the Reference Architecture Model (ODS-RAM), protocols (ODS Protocols), and the source code of the reference implementation OSS (ODS Middleware) for the functional means to deploy them.

Open Dataspaces has completed market-level verification and enterprise commercial-level validation. Its core components are close to a major release. How to develop this in a scalable and sustainable manner is, however, extremely important for implementing distributed data management in society.

We wish to advance this innovation called Open Dataspaces with the spirit of the Cathedral and the Bazaar, trusting in the power of the open-source community worldwide. The technical specifications and components of Open Dataspaces are still a living document (and living source) that has just been born, and they contain much room for improvement. This artifact deserves management through a global, democratic process, divorced from the logic of specific powerful vendors or regulatory authorities.

IPA is considering the form of such an open scheme and needs the cooperation of companies and developers worldwide. If this document has resonated with you, we strongly welcome your participation.

Michitaka TSUDA
Open Data Spaces Chief Architect

Contact:
Information-technology Promotion Agency (IPA)
Digital Architecture Design Center
dadc-info@ipa.go.jp

Bibliography

- **Dehghani Z.** (2019). How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh.
- **Dehghani Z.** (2022). Data Mesh: Delivering Data-Driven Value at Scale. O'Reilly Media.
- **Digital Agency.** (n.d.). Overview of DFFT. https://www.digital.go.jp/en/policies/dfft/dfft-overview
- **Epoch AI.** (2024). Will We Run Out of Data? Forecasting Dataset Size for Language Models. Technical Report, June 2024.
- **Franklin M, Halevy A, Maier D.** (2005). From databases to dataspaces: a new abstraction for information management.
- **Halevy A, Franklin M, Maier D.** (2006). Principles of Dataspace Systems.
- **Nakadai S, Sugamata H.** (2020). 35th UN/CEFACT Forum Webinar Supply Chain Management and Procurement Domain meeting: eNegotiation Project.
- **New Energy and Industrial Technology Development Organization (NEDO).** (2025). Market Size Study of Data Spaces and Impact Modeling & Scenario Analysis Report. https://www.nedo.go.jp/content/800039315.pdf
- **Ministry of Economy Trade and Industry, Information-technology Promotion Agency.** (2025). Whitepaper: Ouranos Ecosystem Dataspaces Reference Architecture Model.
- **Otto B. et al.** (2016). WHITEPAPER: Industrial Data Space. Fraunhofer.
- **Steinbuss S. et al.** (2021). Usage Control in the International Data Spaces. International Data Spaces Association.
- **Y. Tina Lee. (1999).** Information modeling from design to implementation. National Institute of Standards and Technology.

Acknowledgement

Review Contributors:
Kimitaka TAMURA Senior Expert, Architecture Strategy Department, IPA
Nobuaki AOKI Researcher, Engineering Group, IPA
Motonobu SAITO Researcher, Engineering Group, IPA
Yoshikazu HANATANI Researcher, Engineering Group, IPA

Akiyoshi MATONO; Akihisa USHIROKAWA; Daiki MATSUZAKI; Hirotsugu SEIKE; Kazuo NAKASHIMA; Kengo SEKI; Kyota SEKI; Masaaki FUKUDA; Masazumi HIRONO; Nari WOO; Remon KATAYAMA; Ryuta NAKAMURA; Satoshi TAJIMA; Shinji NAKADAI; Shinsuke KOBAYASHI; Takayuki KANEKO; Takehito BANNAI; Takashi KAI; Takuro MORI; Teruyoshi FUJIWARA; Toshihiro IMAI; Wataru FUKATSU; Yoshio TANAKA; Yuichiro KOJIMA; Yuki SAWADA（Alphabetical）

Cover/Document Design
Michitaka TSUDA