The Machine-Interpretable Standards Ecosystem

Standards, Profiles, Building Blocks, and Registers

An Architectural Framework for a Composable, Discoverable, and AI-Ready Standards Infrastructure

OGC Document: 26-021

cite as: Simonis, I., Atkinson, R., Villar, A., Noardo, F., & Toscano, M. (2026). The machine-interpretable standards ecosystem: Standards, Profiles, Building Blocks, and Registers. An architectural framework for a composable, discoverable, and AI-ready standards infrastructure (OGC Document No. 26-021). Open Geospatial Consortium. https://doi.org/10.62973/26-021

Table of Contents

Executive Summary

The Thesis

AI does not eliminate the need for standards — it raises the economic value of getting them right. When an AI agent consumes data whose meaning is implicit (undeclared units, assumed coordinate systems, locally-understood feature types), it is forced to infer, approximate, or hallucinate. In operational environments, that failure mode is unacceptable. The paper argues that the canonical form of a standard should no longer be a PDF or HTML; it should be a machine-interpretable package from which human documentation is derived — not the other way around.

The Cost Being Addressed

IBM estimates poor data quality costs the US economy $3.1T annually; Gartner pegs the hit to an average enterprise at $12.9M per year. A large share of that cost is not wrong data, but ambiguous data — values without declared units, formats, or reference systems. The Mars Climate Orbiter ($125M loss, pound-force vs. newton-seconds) is the famous case; the unglamorous daily equivalents are date formats, ellipsoidal vs. orthometric height, and undeclared coordinate reference systems that can introduce positional errors of up to 21 km. Fifteen years and millions of euros into INSPIRE, Europe still has not closed this gap with document-centric standards.

The Proposed Architecture

Four interlocking concepts, designed to be adopted incrementally:

- **Machine-readable standards** — every specification has a stable web address, formal structural definitions, executable validation rules, a declared list of dependencies, and automatically tested examples. The human-readable version is generated from the machine-readable source, not authored separately.
- **Profiles** — machine-readable specializations of a base standard, for example, for a given jurisdiction or domain (a Stuttgart municipal profile extending a German national profile extending a European profile extending a global base standard). Conformance is inherited down the chain and testable at every layer.
- **Building blocks** — reusable specification components (how to represent a geometry, a time interval, a provenance record, a contact detail, or a versioned object) that can be combined, refined, or substituted. Define once, reuse everywhere — the same logic that underpins modern software package ecosystems, applied to standards.
- **Registers** — federated catalogs of every asset (standards, profiles, building blocks, implementations, conversion tools, vocabularies, validators), organized in the same hierarchical way the internet's domain name system organizes web addresses. A root authority defines the structural patterns; domain authorities run their own conforming nodes. Cacheable, replicable, and operable in disconnected environments.

The Real Leverage: Machine-Interpretable Data Offerings

The paper is framed as a proposal about standards, but the commercial payoff lands one layer higher — on the data itself. Machine-interpretable standards are a means; machine-interpretable data offerings are the end. When a dataset is published against a registered profile, every value it carries arrives with its meaning declared in a form an AI agent can consume safely: units of measure point to a shared reference, coordinate systems are explicit, feature types resolve to catalog entries, date and time conventions are declared rather than assumed, and provenance is a traversable chain rather than a PDF footnote.

This is a capability any data publisher — public or commercial — can adopt independently of whether upstream standards bodies move in lockstep. A provider that packages its offerings this way gains three things immediately: its data becomes directly actionable by AI agents without a custom integration layer; its consumers can verify conformance automatically rather than relying on a data sheet; and its provenance becomes inspectable, which is increasingly a procurement requirement in regulated sectors. The same architecture that makes a standard AI-ready makes every dataset published against it AI-ready by inheritance.

The safety dimension matters as much as the capability dimension. An AI agent consuming ambiguous data must guess at what the values mean, and guesswork at the semantic layer is where hallucinations and silent errors originate. An agent consuming a dataset whose every element carries explicit declarations can validate its own interpretation against the declared rules before acting, converting a probabilistic failure mode into a deterministic one. For any organization whose AI systems are expected to operate on third-party data in high-stakes contexts (defense, finance, healthcare, regulatory reporting, critical infrastructure), this shift from inferred meaning to declared meaning is the difference between an agent that can be deployed and one that cannot.

Why This Matters Commercially

AI reliability becomes an infrastructure question, not a model question

If an enterprise's AI agents are expected to transform data, compute indicators, or act on third-party inputs, their failure rate is bounded below by the ambiguity of their sources. Machine-interpretable standards convert guesswork into validation — the declared rules act as guardrails that let an agent verify its own outputs before submitting them. Investment in this semantic layer now compounds across every downstream AI use case.

Integration cost collapses at the margin

The building-permit example in the paper is the clearest commercial signal: a software vendor serving Germany's ~11,000 municipalities currently builds bespoke integrations for each. Under this architecture, the vendor builds once against the national profile and absorbs each municipality by consuming a small, registered adjustment rather than a new integration project. The every-pair-connects-to-every-pair cost structure becomes linear.

Regulatory reporting gets an auditable meaning

The EU Deforestation Regulation example shows how the architecture makes compliance claims inspectable: a company reports not just a conclusion but the baseline, indicator

definition, processing chain, and provenance that produced it. Regulators and auditors can interrogate the logic instead of accepting a PDF. The same pattern applies to the EU's corporate sustainability reporting regime, California's climate risk and greenhouse gas disclosure laws, the international sustainability reporting standards issued by the IFRS Foundation, and any other regime where the cost of the audit is currently driven by reconciling inconsistent definitions.

Vendor lock-in weakens by design

Because every data contract is a composition of registered, individually-addressable components, switching providers or integrating a new one becomes a mapping exercise against a public register, not a reverse-engineering project. This is a structural shift in buyer power for any organization that consumes geospatial, regulatory, or scientific data as an input.

What's Actually New Here

The individual technologies this architecture relies on — formats for describing data structures, rule languages for validation, semantic vocabularies for meaning, protocols for publishing web APIs — are all mature and in production use. The novelty is the packaging: treating them as a coherent, federated infrastructure with built-in support for inherited specialization, reusable components, and a shared discovery layer. The OpenAPI specification, widely used to describe web APIs, demonstrated the commercial value of this inversion in a narrower domain — turning specifications that were aspirational prose into assets that are actionable and testable. This paper generalizes that pattern to the full standards stack, with the geospatial community as the first-mover domain.

The Risks Worth Tracking

- **Adoption velocity**. INSPIRE took fifteen years to partially succeed under binding legal requirements. The OGC approach is voluntary and member-driven; the paper itself is careful to frame its policy suggestions as proposals for membership discussion rather than mandates.
- **Tooling maturity**. Generating human documentation from canonical machine-readable forms is a hard engineering problem. The paper acknowledges that the toolchain requires significant investment and should be co-designed with practicing standards authors.
- **Governance drift**. A federated catalog works in theory when the root authority and the domain authorities agree on structural patterns. The internet's domain name system is the template — and it also has fifty years of accumulated operational scar tissue. The unglamorous governance work of keeping registers consistent across organizations will determine whether this scales.
- **AI consumption as a first-class test**. The paper argues that AI-readiness must be continuously tested, not retrofitted. An earlier OGC research initiative showed servers being overwhelmed by AI-generated requests when the available metadata was too thin for agents to query efficiently — a foretaste of what insufficient semantic infrastructure produces once AI agents are deployed at scale.

The Authors' Ask

The paper is explicitly a proposal for discussion, not a specification. The authors want feedback on which parts are ready for near-term adoption, which need additional pilots, and which policy directions the OGC membership should formalize. Eight suggested policy initiatives are offered — stable web addresses for standards components, collaborative and version-controlled publication environments, human documentation generated from machine-readable sources, automatically tested examples, formal validation rules, machine-readable dependency declarations, clearly documented profiling patterns, and universal registration of components — all designed to be individually valuable and incrementally adoptable rather than requiring an all-at-once commitment.

Bottom Line for an Executive Reader

If your organization produces, consumes, or regulates data that crosses organizational, jurisdictional, or domain boundaries — and if AI agents are expected to operate on that data — this architecture describes the substrate those agents will need to be reliable. The paper discusses standards, but the practical leverage is that the same architecture makes data offerings AI-consumable in a safe, validated, provenance-bearing form. A data publisher does not need to wait for a global standards shift to capture this value; publishing against a registered profile is a unilateral move. The question is not whether this direction is right (the economics of AI reliability make it close to inevitable), but how early to invest, which profiles and registers to influence or establish, and where the competitive advantages of early adoption accrue. The geospatial community is first; financial reporting, clinical data, supply chain, and energy markets are structurally similar.

1. Introduction and Motivation

Standards are the foundation of interoperability. They reduce friction in data exchange and the design costs of systems for publishing, processing, and using that data. To maximize effectiveness, we need standards that allow systems, organizations, and communities not only to exchange data but to do so with shared meaning, so that what one side produces, the other can interpret correctly and use with confidence.

In the age of AI, a question is increasingly heard: Do we still need standards at all? If AI can interpret almost anything, why invest in the careful and often painstaking work of standardization? The answer is that AI does not remove the need for standards; it raises the value of doing them well. AI systems perform best when the semantics of data, services, and rules are explicit rather than implicit, and when the artifacts they consume are constrained, discoverable, and interpretable rather than left to inference.

This is why semantic interoperability is becoming more, not less, important. Severe digital failures rarely occur because data cannot be exchanged. They occur because data can be exchanged without a shared understanding of what it means. In geospatial systems, this occurs when units are undeclared, coordinate reference systems are assumed rather than specified, feature types are understood only locally, or code lists and temporal conventions are not aligned across communities. Systems may be able to move data, but they cannot reliably interpret it. In addition to interpretation failures, the inability to discover, connect, and reuse available data represents a significant lost opportunity. These opportunity costs—though difficult to quantify—can be substantial, as poorly described data prevents systems and people from realizing its full value.

AI can magnify or mitigate this problem, depending on how much context we can provide it and how big a task we set it. A human reader can sometimes reconstruct missing context from experience, surrounding documentation, or institutional knowledge. An AI agent can process standards, data, and service descriptions at a far greater scale, but where semantics are missing, it must infer, approximate, guess, or worst, hallucinate. That is precisely what we should not want in high-value operational environments. Well-performing, well-behaving AI depends on explicit, formal, and machine-interpretable semantics.

This has an important consequence for how standards should be designed. In an AI-native environment, the objective is not merely machine-readable specifications, but machine-interpretable standards that expose meaning, constraints, dependencies, and applicability in directly usable form. From that foundation follow machine-interpretable data and service offerings. Reliable AI depends on this chain: if standards are not machine-interpretable, then the data and services derived from them remain partly opaque, forcing AI systems to infer intent where they should instead be able to interpret and validate it explicitly. To validate that this objective is being met, standards should be systematically tested for both human and AI readiness — for example, by assessing whether community-specific profiles can be developed and applied successfully within targeted sub-domains.

That is the role of building blocks. Standards exist in many forms, such as legacy, current, and emerging, each optimized for different environments and communities. A key role of

building blocks is to provide a common operational view through which these diverse capabilities can be understood and related to one another. Building blocks package specification components in forms that machines can discover, interpret, validate, and reuse, including schemas, semantic mappings, constraints, examples, transformations, and dependencies. By expressing standards and profiles as compositions of such building blocks, the ecosystem shifts from document-centric publication toward machine-interpretable semantics. AI systems no longer need to infer intent primarily from prose; they can operate on explicit components, formal constraints, declared dependencies, validated examples, and reusable transformation assets.

The question, then, is not whether we still need standards, but how we should do standards today so that they deliver semantic interoperability for both human and machine consumers. This document argues that standards should be authored and managed as machine-interpretable, interconnected assets, composed from reusable building blocks and supported by registers that make them discoverable, traceable, and trustworthy. In this model, human-readable documents remain essential, but they are derived views of a deeper semantic architecture rather than the sole authoritative form.

These requirements imply more than a better publication format. Standards must be uniquely identifiable and resolvable so that humans and machines can refer to the same artifact unambiguously. They must also be adaptable to the needs of particular communities, jurisdictions, and applications, because interoperability at scale depends not only on shared base standards, but also on transparent ways to express local constraints, vocabulary bindings, and extensions without severing conformance to that shared base. Their content must be decomposable into reusable specification components that carry explicit meaning, constraints, examples, transformations, and dependencies. And all of these assets must be discoverable, traceable, and governable across organizational boundaries.

This document elaborates an architectural approach that responds to these needs through four interconnected principles:

1. Every standard must be referenceable in a machine-readable way.
2. Every standard can be profiled for specific community needs.
3. Profiles and standards are composed of reusable building blocks.
4. All assets — standards, profiles, building blocks, implementations, transformation engines, vocabularies, and validators — are stored in interconnected registers that enable discovery, reuse, and trust.

Quality assurance and testing are concerns that cut across all four principles rather than constituting a separate one. They are nonetheless essential. Federation, in particular depends on shared testing and validation procedures to establish trust across organizational boundaries.

Together, these principles create a composable, federated standard ecosystem in which standards are not merely published, but made operational. By making standards machine-interpretable, the ecosystem enables data and service offerings to become

machine-interpretable as well. That, in turn, provides the semantic foundation and operational guardrails required for reliable, scalable, and trustworthy AI.

2. The Cost of Ambiguity

The principles outlined above are not abstract ideals. They respond to a concrete and measurable problem: the cost of ambiguous, undeclared, or inconsistent semantics in data exchange. IBM estimates that poor data quality costs the US economy $3.1 trillion per year, while Gartner puts the average impact on a single organization at $12.9 million annually. A significant share of these costs stems not from data being wrong, but from data whose meaning is unclear: values without declared units, formats, or reference systems that force every downstream consumer to guess.

The most dramatic examples are well known. NASA's Mars Climate Orbiter was destroyed in 1999 because one system expressed thruster force in pound-force seconds while another expected newton-seconds, and nothing in the data exchange declared which unit was intended. The result was a $125 million loss from a single missing semantic declaration. During Hurricane Katrina in 2005, emergency responders could not share geospatial data because agencies communicated locations using incompatible reference systems and formats; it took weeks to establish a functioning shared GIS capability while lives hung in the balance.

But not every interoperability failure makes headlines. The everyday equivalents are pervasive and, if unnoticed, carry serious consequences of their own. Consider something as simple as a date: the value "03/04/25" means 3 April 2025 in most of Europe (DD/MM/YY), March 4, 2025 in the United States (MM/DD/YY), and April 25, 2003 in Japan (YY/MM/DD) (see W3C Date Formats and Date Formats by Country). When timestamps are attached to sensor readings, environmental observations, or geospatial features and exchanged across borders, a misinterpreted date silently corrupts the data. No error message, no warning. Or consider height: a GPS receiver reports ellipsoidal height — the distance above a mathematical reference surface — while engineers and citizens expect orthometric height, the elevation above mean sea level. The difference between these two values can reach tens of meters depending on the location. A dataset that simply states "elevation: 120 m" without specifying the vertical reference system forces every consumer to guess, potentially with serious consequences for flood modeling, construction, or aviation safety.

In the horizontal plane, coordinate reference system mismatches are arguably the most common everyday failure in geospatial work: datasets arrive with coordinates, but no declaration of which coordinate system they belong to, or with the wrong one declared. Research has shown that failing to account for the difference between spheroidal and spherical datums alone can introduce positional errors of up to 21 kilometers. Features simply appear in the wrong place on the map — a school plotted in a field, a pipeline crossing the wrong property boundaries. The European INSPIRE Directive, one of the most ambitious attempts to harmonize geospatial data across borders, has demonstrated just how difficult this problem is at scale: the complexity and diversity of data formats among EU member states continues to hinder seamless integration, despite more than fifteen years of effort and legally binding requirements.

Every one of these failures shares a common cause: the semantics of the data — its units, its reference system, its format conventions — were not declared in a machine-readable form that both human implementers and automated systems could unambiguously interpret. Standards exist to provide exactly these declarations. The remainder of this document describes an architecture for making them work effectively for both audiences.

In geospatial systems, semantic failure often appears in the gap between formal exchange and practical usability. Datasets may be published through interoperable services and encoded in accepted formats, yet still fail in downstream use because feature types, code lists, lineage, temporal validity, or coordinate reference assumptions are not aligned. The problem is not that machines cannot access the data; it is that they cannot reliably interpret it. This is exactly the gap that machine-readable standards, profiles, vocabulary bindings, and registers are intended to close.

3. Core Concepts

3.1 Machine-Readable Standards

The foundational requirement is that every standard — and every component within a standard — must be addressable, describable, and constrainable in a machine-readable form. This does not mean that human-readable documents disappear. Rather, it means that the canonical form of a specification is machine-readable, and human-readable documents are derived from it. This inversion is critical: it ensures that the machine-readable form remains authoritative and complete, rather than a secondary artifact that may drift out of sync with the document. This approach doubles down on emerging best practices already gaining traction across the industry. OpenAPI, for example, embodies this principle, with specifications designed to be actionable and testable from the outset.

A machine-readable standard includes several layers of information. At the most basic level, it carries its identity — a stable URI that resolves to a landing page providing both human-readable documentation and machine-readable metadata. This metadata describes the standard, its domain, the version it represents, who maintains it, and its normative status. In practice, once a standard is established, these resources do not need to be actively accessed during routine use. They are primarily needed to validate implementation libraries and tooling. Day-to-day users operate within their own domains, referencing these canonical resources only when defining or validating the additional components of standards they need for their specific contexts.

Beyond identity, a machine-readable standard expresses its structural model. For data-encoding standards, this means formal schemas, currently expressed in languages such as JSON Schema, XML Schema, or SHACL shapes. For API standards, this means interface contracts such as OpenAPI or AsyncAPI descriptions. For conceptual models, this means ontologies expressed in OWL or SKOS vocabularies. Architecturally, these are reference implementation technologies: the ecosystem is designed so that these specific technologies can evolve or be extended as the landscape matures.

Critically, the machine-readable form also captures constraints that go beyond what schemas alone can express. These are rules that define valid usage, such as co-occurrence constraints, conditional requirements, value-dependent behaviors, and

cross-element validation logic. These constraints are expressed in formal rule languages (currently SHACL, Schematron, JSON Path-based rules, or similar) and are automatically executable, meaning that any instance claiming conformance to the standard can be validated programmatically.

Finally, a machine-readable standard declares its dependencies. If a standard builds on, extends, or constrains another standard, that relationship is expressed in a machine-readable dependency graph. This enables automated reasoning about compatibility, impact analysis when standards change, and intelligent discovery of related specifications.

3.2 Profiles

A profile is a constrained and potentially extended version of one or more base standards, tailored to the needs of a specific community, jurisdiction, application domain, or use case. Profiling is the mechanism through which generic standards become practically useful in specific contexts.

Profiles serve multiple purposes. They narrow optionality: where a base standard permits multiple encoding choices, a profile may mandate one. They add domain-specific semantics: a profile of a generic feature model may define the specific feature types, attributes, and value domains relevant to a national mapping programme. They impose stricter constraints: a profile may require elements that the base standard leaves optional. And they may add extensions: additional elements, code lists, or behaviors that the base standard does not address but the community requires.

The key architectural principle is that a profile is not a divergent, independent copy of the base standard. A profile maintains a formal, machine-readable relationship to the standard(s) it derives from, and remains explicitly bound to them. An instance conforming to a profile must also conform to the base standard. This inheritance chain is explicit and testable: the base standard's validation rules apply, along with the additional rules introduced by the profile.

Profiles can be layered. A national mapping agency may profile a regional (e.g., European) profile, which itself profiles a global OGC standard. Each layer adds specificity without breaking conformance to the layers below. This creates a hierarchy of interoperability: data conforming to the national profile is guaranteed to be processable by any system that understands the regional or global standard, even if some community-specific semantics are lost.

Profiles are themselves first-class, registered assets in the ecosystem. They have URIs, landing pages, machine-readable descriptions, validation rules, and dependency declarations — exactly like the standards they profile. This means that a consumer discovering a dataset described as conforming to a particular profile can programmatically trace the full inheritance chain, retrieve all applicable validation rules, and understand exactly what guarantees that conformance provides.

3.3 Building Blocks

Building blocks are the reusable, scale-independent components from which standards and profiles are composed. A building block encapsulates a coherent piece of specification — a data model fragment, a set of constraints, an API operation pattern, a code list, a conceptual element — packaged for independent reuse.

A building block can be as small as a single concept — for example, a geometry fragment or a contact-details pattern — or it can be an assembly of smaller building blocks grouped into a higher-level pattern. An assembly of building blocks is itself a building block, and can in turn be extended, refined, or combined with others. The result is that standards are built up in layers, from narrow foundational patterns to rich domain-specific compositions, and the same foundational patterns can be reused across many domains. For an application community, the set of building blocks relevant to their work becomes the practical interface between their domain and the wider standards ecosystem.

The building block concept addresses a pervasive problem in standards development: redundancy and inconsistency. Multiple standards often define overlapping concepts, such as temporal properties, spatial geometry, contact information, and provenance metadata, with subtle differences that create unnecessary friction. Building blocks allow these common elements to be defined once, maintained in one place, and reused wherever needed with guaranteed consistency.

A building block is more than a schema fragment. It is a self-contained package that includes, though is not strictly limited to, schemas and data models. A building block may include schemas in one or more formats, associated constraints and validation rules, examples that are automatically tested, documentation for human consumers, and machine-readable metadata including dependencies on other building blocks. This self-containment means that a building block can be independently validated, tested, and versioned.

Composition is the central operation. A standard or profile is composed by assembling building blocks and adding any additional specifications needed to bind them together. This composition is explicit and machine-readable: the bill of materials for any standard or profile can be inspected to see exactly which building blocks it uses, which versions, and how they are constrained or extended in that context.

Building blocks also serve as the primary vehicle for extensibility. When a community needs to extend a standard, they create new building blocks rather than modifying existing ones. These new building blocks can then be registered and made available to others facing similar needs, creating a growing library of composable specification components. Communities can either build their own registries or contribute directly to the OGC community, and these registers can optionally be federated to create scalable ecosystems. A scalable and secure approach to registers is key to easy adoption, as the next section elaborates.

3.4 Registers

Registers are the connective tissue of the ecosystem. They are authoritative, curated catalogs that make all assets — standards, profiles, building blocks, implementations, validators, transformation engines, code lists, and any other standards-related resources — discoverable, addressable, and interconnected.

A register is not merely a list. It is a governed resource with clear ownership, curation policies, and machine-readable content. Each entry in a register has a stable URI, rich metadata, and declared relationships to other entries within and across registers. Registers expose their content through standardized APIs, enabling both human browsing and programmatic access.

The register architecture is federated and hierarchical, following a pattern analogous to the Domain Name System. OGC maintains root-level registers that define the canonical namespaces and structural patterns. Domain authorities, national bodies, and community organizations operate their own registers that conform to the root patterns but are independently managed. This federation ensures that no single organization needs to catalog everything, while the shared structural conventions enable cross-register discovery and traversal.

A good register architecture supports efficient caching and offline modes for both testing and secure air-gapped operational contexts that need access to shared content. Security is a first-class requirement of the register infrastructure, not merely an operational afterthought. Submission, update, replication, and governance actions should therefore be protected by strong authentication and authorization. Digitally signed register entries or signed register snapshots allow users and systems to verify provenance and integrity even when content is retrieved from caches or offline replicas, which is particularly important in disconnected or air-gapped environments where controlled replication must preserve traceability to the authoritative source.

Several distinct types of registers operate within the ecosystem. The *Standards Building Blocks Register* catalogs all reusable specification components. *Profile Registers* catalog community-specific specializations, organized by domain or jurisdiction. *Implementation Registers* catalog software that implements standards or building blocks, with links to conformance test results. *Transformation Registers* catalog available mappings between standards, profiles, or data models. *Validation Registers* catalog available conformance tests and validators. *Feature Type Catalogs* describe the types of real-world features that data can represent, enabling semantic discovery of datasets.

The register infrastructure also provides the foundation for provenance and trust. Because every asset has a registered identity, a governed lifecycle, and declared relationships, consumers can trace exactly where a specification came from, who maintains it, how it relates to other specifications, and whether implementations have been tested against it. This Integrity, Provenance, and Trust (IPT) capability is essential for building confidence in a federated, distributed ecosystem.

4. The Architecture in Detail

4.1 Anatomy of a Machine-Readable Standard

Every standard in the ecosystem is represented as a structured package of machine-readable artifacts, organized according to a common pattern. The following table describes the components:

Component	Description

Canonical URI	A persistent, resolvable identifier for the standard. Resolves to a landing page with content negotiation, serving HTML for browsers and RDF/JSON-LD for machines.
Metadata Descriptor	A machine-readable description including title, abstract, version, status, maintainer, domain classification, license, and temporal validity. Expressed in a standard metadata vocabulary (e.g., DCAT, Dublin Core extended with OGC-specific terms).
Structural Schemas	Formal data models in one or more schema languages: JSON Schema for JSON encodings, XML Schema for XML encodings, SHACL or OWL for semantic models. Multiple schema representations may coexist, linked by equivalence declarations.
Constraint Rules	Machine-executable validation rules that go beyond schema validation: co-occurrence constraints, conditional requirements, cross-element rules, value-dependent logic. Expressed in SHACL, Schematron, or equivalent rule languages.
Dependency Graph	Machine-readable declarations of all dependencies on other standards and building blocks, including version constraints and the nature of each dependency (imports, extends, constraints, profiles).
Tested Examples	Reference instances that conform to the standard, automatically validated against the schemas and constraint rules as part of a continuous integration pipeline.
Human Documentation	Derived from the canonical machine-readable forms. This is the traditional specification document, but generated rather than authored independently, ensuring consistency with the machine-readable artifacts.

4.2 The Profiling Mechanism

Profiling is the systematic process of specializing a base standard for a particular context. This may involve selecting optional elements, defining content rules such as controlled vocabularies, or using specific extensions. The architecture supports profiling through a formal, machine-readable framework that ensures profiles remain traceable to their bases and interoperable across community boundaries.

4.2.1 Profile Structure

A profile is structurally identical to a standard in terms of its packaging: it has a URI, metadata, schemas, constraints, examples, and documentation. What distinguishes a profile is its dependency declaration, which explicitly states which standard(s) or profile(s) it profiles, and the nature of its specializations.

Well-designed profiles should also be easily profiled further for specific sub-domains of application. Profiles thus behave as Building Blocks and, therefore, can and should be developed as such.

Specializations fall into defined categories. A restriction narrows the set of options by mandating specific choices from those permitted by the base standard. Constraint tightens requirements by making optional elements mandatory, restricting value ranges, or imposing stricter cardinality. An extension adds new elements, attributes, or behaviors not present in the base standard. Vocabulary binding associates specific code lists or controlled vocabularies with generic fields. Each specialization category has specific rules about how it affects conformance inheritance.

4.2.2 Profile Layering

Profiles can be layered in a hierarchy. Consider a concrete example from the geospatial domain:

- The OGC Features Core standard defines the base model for geographic features, including geometry, temporal properties, and minimal metadata.
- An INSPIRE profile in Europe requires specific metadata elements, coordinate reference systems, and code lists mandated by the INSPIRE Directive.
- A German national profile further constrains the INSPIRE profile by requiring German-specific feature-type classifications, attribute naming conventions, and quality indicators.
- A municipal profile further constrains the national profile for urban planning use, mandating building-specific attributes and local code lists.

At each layer, the profile inherits all constraints from below and adds its own. The machine-readable dependency chain enables a validator to automatically assemble the complete set of applicable rules by traversing the profile hierarchy. A dataset validated against the municipal profile is guaranteed to conform to all standards and profiles in the chain.

4.2.3 Cross-Profile Interoperability

Because profiles share common bases, data from different communities can be exchanged at the level of their shared ancestor. A German municipal dataset and a French regional dataset may use incompatible local profiles, but both conform to the INSPIRE profile and are therefore processable by any INSPIRE-compliant system. The register infrastructure makes this interoperability visible: a consumer can query the register to find the common ancestor of any two profiles and understand the guaranteed level of interoperability. Importantly, if two organizations see a need to standardize and share information outside the scope of an ancestor model, they can define and register a shared profile. This makes it possible to increase interoperability as new needs emerge — something that is not possible with fixed standards.

A critical advantage of this approach is that software can recognize profiles by name, including their base profiles, and automatically respond with preconfigured functionality. This is fundamentally impossible in a world of copy/paste duplication, where similar standards contain subtly different descriptions of the same concepts. Comparing complex standards and determining whether their intent is equivalent cannot be done easily, or at all, by automated systems. Formal, machine-readable profiling makes this recognition trivial.

Where transformation between profiles is needed, registered transformation engines provide the bridge. A transformation from the German profile to the French profile would be registered as an asset, linked to both source and target profiles, and potentially executable on demand. This is the mechanism for treating transformations as first-class citizens. They are not ad hoc scripts but governed, registered, testable assets.

4.3 Building Block Composition

4.3.1 Building Block Anatomy

Each building block is a self-contained specification component. Every building block has:

- An identity — a registered URI within the OGC Standards Building Blocks Register.
- A scope definition — a clear statement of what the building block specifies and what it does not.
- Dependencies on other building blocks, with version constraints.
- Tested examples demonstrating correct usage, validated automatically.
- An extension point specification — how this building block can be extended by other building blocks or profiles.

In addition, a building block must carry at least one of the following specification artifacts (most carry several):

- One or more schema fragments in applicable schema languages (JSON Schema, XML Schema, SHACL, etc.).
- Constraint rules applicable to this building block in isolation.
- Mapping of schema elements to semantic descriptions (for example, a JSON-LD context).
- Transformation specifications.

This separation of concerns, between schemas, semantic mappings, and transformation specifications, is deliberate and architecturally significant. It allows, for example, two different building blocks to map the same schema to different ontologies for different communities, or for building blocks to contain only transformation specifications without schemas. This modularity maximizes reuse and avoids forcing false coupling between concerns that may evolve independently.

4.3.2 Composition Patterns

Standards and profiles are composed from building blocks using several defined patterns:

- Aggregation assembles multiple building blocks side by side. A feature standard might aggregate a geometry building block, a temporal building block, and a metadata building block. Each retains its identity and constraints; the standard adds the binding logic that connects them.
- Refinement takes an existing building block and narrows it. A domain-specific feature type building block might refine the generic geometry building block to

mandate a specific geometry type (e.g., Polygon only) and a specific coordinate reference system.

- Extension takes an existing building block and adds new elements. A provenance building block might extend the generic metadata building block with lineage and processing step information.
- Substitution replaces one building block with another that provides the same interface but different behavior. A high-performance geometry building block using DGGS encoding might replace the standard WKT geometry building block in performance-critical contexts.

All composition operations are declared in the machine-readable dependency graph, enabling automated tools to resolve, validate, and visualize the full composition structure of any standard or profile.

4.4 The Register Infrastructure

4.4.1 Register Types

The ecosystem operates multiple interconnected register types, each governed and curated according to its purpose:

Register	Contents and Purpose
Standards Building Blocks Register	All reusable specification components: schema fragments, constraint rules, extension point definitions, code lists, and conceptual model elements. This includes both formally adopted components and candidate or community-contributed patterns. This is the primary library from which standards and profiles are composed, supporting federated views across governance domains.
Standards Register	Published standards with their full machine-readable descriptions, dependency graphs, and links to constituent building blocks. Includes version history and normative status.
Profile Register	Community-specific profiles organized by domain and jurisdiction. Includes the profiling relationship chain and links to the specific constraints, vocabulary bindings, and extensions each profile introduces.
Implementation Register	Software products, libraries, and services that implement standards or building blocks. Linked to conformance test results and deployment metadata. Enables discovery of available implementations for any standard or profile.
Transformation Register	Mappings and transformation engines that convert data between standards, profiles, or encodings. Each entry specifies source and target, transformation logic (or a reference to an executable engine), and quality/fidelity metadata. Architecturally, the Transformation Register may function as a linked view into the Feature Type Catalog and Profile Register, though governance

	and security considerations may warrant maintaining it as a distinct but federated register. Transformation testing is best conducted within the building block context.
Validation Register	Conformance test suites, validators, and test harnesses. Linked to the standards and profiles they test. Enables automated conformance testing as a service.
Feature Type Catalog	A federated, DNS-like catalog of real-world feature types. Enables semantic discovery: finding data about buildings, rivers, or soil types regardless of the standard or encoding used. Organized hierarchically with domain authorities managing their own namespaces.
Vocabulary Register	Code lists, controlled vocabularies, and classification systems used by standards and profiles. Supports multilingual labels, hierarchical relationships, and mapping between equivalent vocabularies from different communities.

4.4.2 Federation and the DNS Analogy

The register architecture is federated rather than centralized. OGC acts as the root authority, defining the structural patterns that all registers must follow, maintaining the root-level registers, and operating the resolution protocol that allows traversal across federated nodes.

Domain authorities — national mapping agencies, thematic communities, regional bodies, industry consortia — operate their own register nodes. These nodes conform to the structural patterns defined by OGC but are independently governed and maintained by the authorities who understand their respective domains.

The resolution mechanism works hierarchically, following the DNS pattern: a query for a specific asset starts at the local node, which resolves it directly if the asset is within its own domain. If the asset is unknown locally, the query escalates upward — potentially reaching the root — which redirects to the appropriate domain authority. At each level, the register API is consistent, enabling clients to traverse the federation without special knowledge of individual nodes. This is precisely the DNS pattern applied to standards assets.

The architecture is designed to support caching and replication at every level. Frequently accessed assets can be cached locally, reducing load on authoritative sources. Full or partial register replicas can be maintained for resilience, offline access, air-gapped operational contexts, or performance. The provenance metadata ensures that consumers always know the authoritative source, regardless of which replica they accessed.

4.4.3 Register Governance

Each register operates under a governance framework that defines who may submit entries, what review process applies, how conflicts are resolved, and how entries are versioned, deprecated, and retired. The governance framework is itself described in machine-readable terms, enabling automated compliance checking.

The OGC root registers follow OGC's established governance processes, which should be updated to more formally and consistently align with the ISO 19135 model for the management of registers — a well-established and sensible framework for register governance. Federated registers operated by domain authorities follow their own governance processes, provided they conform to the structural and interoperability requirements defined by the root. This respects the autonomy of domain authorities while ensuring that the federated ecosystem remains coherent.

5. Operational Workflows

5.1 Publishing a Standard

The workflow for publishing a standard in this ecosystem differs significantly from traditional document-centric processes. Achieving this workflow requires investment — particularly in the document generation tooling — and should be co-designed with practicing standards developers to ensure that the process genuinely simplifies their work rather than adding complexity:

1. The standard is authored in a machine-readable canonical form. Schema fragments are written in JSON Schema, XML Schema, or an equivalent format. Constraints are expressed in SHACL, Schematron, or equivalent rule languages. Dependencies on building blocks and other standards are declared explicitly.
2. Examples are written and automatically validated against the schemas and constraint rules in a continuous integration pipeline. Any failure in validation halts publication.
3. Human-readable documentation is generated from the canonical forms, ensuring consistency. Authors may enrich the generated documentation with explanatory text, but the normative content is derived rather than independently authored.
4. The standard and all its constituent building blocks are registered in the appropriate registers, with full metadata, dependency declarations, and provenance information.
5. The standard's URI becomes resolvable, pointing to a landing page that serves machine-readable metadata (via content negotiation) and human-readable documentation.
6. Automated conformance validators are published in the Validation Register, linked to the standard, enabling immediate testing of implementations.

5.2 Creating a Profile

A community needing to specialize a standard follows a parallel workflow:

7. The community identifies the base standard(s) or profile(s) to be profiled and defines the required specializations: restrictions, constraints, extensions, and vocabulary bindings.
8. Each specialization is expressed as a machine-readable delta against the base. New constraints could be, for example, additional SHACL shapes or Schematron

rules. Extensions are additional building blocks composed into the profile. Vocabulary bindings link generic fields to specific code lists from the Vocabulary Register. (The vocabulary binding mechanism itself needs to be standardized to ensure consistent behavior across the ecosystem, particularly to support integration with external vocabulary services operated by other organizations.)

9. The profile's validation rules are assembled automatically by combining the base standard's rules with the profile's additional rules. Examples are validated against the combined rule set.
10. The profile is registered in the appropriate Profile Register, with its profiling chain explicitly declared. The registration triggers updates to the dependency graph, making the profile visible to consumers querying for specializations of the base standard.

5.3 Discovering and Using Assets

A data publisher, software developer, or AI agent needing to work with standards assets follows a discovery workflow enabled by the register infrastructure:

11. The consumer queries the register infrastructure for assets matching their needs. This might be a query for "building feature types in Germany" (hitting the Feature Type Catalog), "JSON Schema validators for OGC API Features" (hitting the Validation Register), or "transformations from INSPIRE Buildings to CityGML" (hitting the Transformation Register).
12. The register returns matching assets with their full metadata, dependency chains, and links to related assets.
13. The consumer retrieves the machine-readable artifacts: schemas for implementation, constraint rules for validation, examples for testing, and transformation engines for data conversion.
14. The consumer uses these artifacts programmatically: validating data against schemas and constraints, transforming data using registered engines, or implementing APIs against registered interface specifications.
15. The consumer's own implementation can itself be registered in the Implementation Register, closing the loop and enriching the ecosystem.

5.4 AI Readiness

The machine-readable ecosystem is inherently AI-ready, but specific design considerations maximize its utility for AI agents:

- API definitions and schemas provide the structural information AI agents need to formulate valid queries and interpret responses.
- Constraint rules serve as guardrails, enabling AI agents to validate their own outputs before submitting them.
- The register infrastructure provides a discovery mechanism that AI agents can traverse programmatically, enabling them to find the assets they need without human guidance.

- Provenance metadata enables AI agents to assess the trustworthiness and currency of the assets they discover.
- Examples serve as few-shot learning material, enabling AI agents to understand expected patterns by inspecting validated instances.

The DGGS AI pilot demonstrated both the potential and the challenges of AI agents consuming OGC APIs. Servers were overwhelmed by the volume of AI-generated requests, and agents lacked sufficient context to formulate effective queries. The machine-readable ecosystem addresses this by providing rich, structured metadata that enables AI agents to understand what an API offers, which constraints apply, and how to interact efficiently, before making their first request.

A key principle emerging from this work is that AI readiness is not a feature to be added after the fact. It must be tested continuously as standards evolve. The ecosystem should include automated AI consumption testing as part of its continuous integration pipeline, ensuring that every standard, profile, and building block remains usable by AI agents as it evolves. This does imply an additional category of conformance tests, distinct from schema and instance validation: tests that assess whether an AI agent, given only the machine-readable artifacts of a standard, can correctly discover, interpret, and apply them without human intervention.

6. Integrity, Provenance, and Trust (IPT)

The register-based ecosystem provides the foundation for a comprehensive Integrity, Provenance, and Trust framework that permeates every layer of the architecture.

Integrity is assured through machine-readable constraints and automated validation. Every asset in the ecosystem — from individual building blocks to complete profiles — includes validation rules that can be automatically executed. Data, implementations, and derived products can be checked for conformance at any point, providing continuous assurance that the standards ecosystem's guarantees are being honored.

Provenance is intrinsic to the register architecture. Every asset has a documented origin, governance chain, version history, and dependency graph. When a consumer encounters a dataset claiming conformance to a profile, they can trace that profile back through its profiling chain to the base standard, verify who maintains each layer, check when it was last updated, and understand exactly what the conformance claim entails. For data, provenance extends to the observation methods, sampling strategies, and processing steps that produced it — all of which are expressible using registered building blocks for provenance metadata.

Trust emerges from the combination of integrity and provenance with the governance framework. A consumer can assess trust based on the authority of the register, the governance rigor of the domain authority, the testing history of implementations, and the completeness of the provenance chain. The register infrastructure does not impose trust — it makes information available to consumers so they can make informed trust decisions.

The IPT pattern applies universally across a wide range of operational scenarios: whether a German mapping agency is aggregating intelligence from diverse sources, a weather service is managing hundreds of variables, a biodiversity monitoring system is combining local and global datasets, or a risk assessment platform is computing indices from heterogeneous inputs, the fundamental requirement is the same — knowing what the data represents, where it came from, how it was processed, and whether it can be trusted. The machine-readable standards ecosystem provides the infrastructure to answer these questions systematically rather than on an ad hoc basis.

7. Relationship to OGC Policy and Procedure Changes

The architecture described in this document does not exist in a vacuum. For it to succeed, OGC's policies and procedures will need to evolve alongside it. This section outlines a set of suggested policy directions for OGC members to consider, discuss, and refine. These are not prescriptive mandates; they are proposals intended to open a conversation about what the policy framework could look like if the membership decides to move toward a machine-readable standards ecosystem. The specific wording, scope, and phasing of any eventual policy changes are for the membership to determine through the established governance process.

The suggestions below are also informed by existing OGC policies regarding the use of URI identifiers and the OGC Modular Specification model (aka "*ModSpec*"), and should be read as building on rather than replacing those foundations.

7.1 Suggested Meta-Principles for Policy Design

Before considering specific policy initiatives, the membership may wish to agree on a set of meta-principles that govern how policies themselves are designed. The following four constraints are suggested as a starting point for discussion:

- **Policies should not specify software or toolchains.** Policies should define outcomes and requirements, not the tools used to achieve them. The difficulties encountered with INSPIRE's technology-specific mandates illustrate the risks of coupling policy to particular software stacks.
- **Policies should be backed by at least one implementation option.** No policy should be adopted unless a viable, demonstrated implementation pathway exists. This avoids aspirational policies that create compliance burdens with no practical route to fulfilment.
- **Implementation options need to be fully backed and resourced by the organisation.** Where OGC offers or endorses an implementation pathway, the organisation should ensure it is adequately maintained and supported so that adoption does not rely solely on goodwill.
- **Testing of policy and the feasibility of implementation should be embedded in the standards programme.** There is ample precedent for this approach within OGC (for example, ShapeChange and similar tooling validation), and it ensures that policy requirements remain grounded in practical experience.

The architecture presented in this document has been designed with these meta-principles in mind. It does not mandate specific toolchains; the OGC Building Blocks infrastructure serves as a reference implementation to demonstrate feasibility, and the continuous integration approach ensures compliance is tested rather than merely asserted. Members may wish to adopt some or all of these meta-principles as an overarching framework before turning to the specific initiatives below.

7.2 Suggested Policy Initiatives

Subject to the meta-principles above, the following eight policy initiatives are proposed for the membership's consideration. Each is presented as an area where policy could be strengthened, not a finalised policy statement. The membership should evaluate these against OGC's strategic priorities, resource constraints, and the practical realities faced by standards working groups.

1. **URI-addressable standards components with landing pages.** Standards components at any degree of granularity should be supported by URIs and landing pages that allow all supporting documentation to be accessed in both human-readable and machine-readable forms.
2. **Collaborative, version-controlled publication environments.** Publication of standards should be done in an environment that supports collaboration, versioning, proposals, and automation of processing toolchains.
3. **Documents derived from canonical machine-readable forms.** Human-readable documents should be derived directly from canonical machine-readable forms of specification to the greatest extent possible. Where toolchain improvements are required to support this, they should be formally documented and added to a development roadmap, including incorporation into initiative designs and calls for sponsorship as resources become available. The scope of canonical machine-readable expression should focus on the normative parts of a standard, where formal semantics deliver the clearest return. Informative content may continue to be authored primarily for human readers, with building blocks serving as the granular mechanism for individually formalizing, referencing, and reusing normative elements.
4. **Automatically tested examples.** Standards elements should be supported by examples, and these examples should be automatically tested for compliance with canonical specifications.
5. **Formal machine-readable constraint rules.** Where possible, specification constraints should be supported by one or more formal machine-readable constraint rules. For example, any JSON-based schema could be mapped to JSON-LD and have rules defined in SHACL, though this need not be the only option now or in the future. Automated testing against these rules, as per the previous initiative, is the key requirement.
6. **Machine-readable dependency declarations.** Dependencies between standards must be declared in a machine-readable manner to support FAIR principles and enable automated reasoning about compatibility and impact.

7. **Clearly tested and documented profiling and extension patterns.** The patterns and mechanisms for downstream use of standards by communities specialising them (through profiles or extensions) should be clearly tested and documented. This may have two flavours: OGC-published shared profiles on one hand, and local application usage on the other, provided that additional constraints are clearly identified for OGC candidate standards.
8. **Registration of all standards components.** All standards components must be registered in an appropriate register to support FAIR principles, with defined interoperability requirements for submission.

7.3 How the Architecture Maps to These Suggestions

The architecture described in the preceding sections was designed so that each of the suggested policy initiatives above could be concretely realised. If the membership decides to adopt some or all of these directions, the architecture provides a coherent technical pathway. The following mapping illustrates this relationship:

- **URI-addressable components with landing pages** → the canonical URI and content-negotiated landing pages in the register infrastructure (Section 4.4), where every registered asset resolves to both human-readable and machine-readable representations.
- **Collaborative, version-controlled publication** → the Git-based authoring and continuous integration pipeline for machine-readable canonical forms (Section 5.1), which provides the collaborative infrastructure for standards development.
- **Documents derived from canonical forms** → the principle that human documentation is generated rather than independently authored (Sections 3.1 and 5.1), ensuring consistency between the machine-readable canonical source and what practitioners read.
- **Automatically tested examples** → the continuous integration pipeline that validates all examples against schemas and constraint rules (Section 5.1), with validation failure halting publication.
- **Formal machine-readable constraint rules** → SHACL shapes, Schematron rules, and equivalent constraint languages as core components of every standard and building block (Sections 3.1, 4.1, and 4.3.1).
- **Machine-readable dependency declarations** → the explicit dependency graphs linking standards, profiles, and building blocks (Sections 3.1, 4.1, and 4.2), enabling automated compatibility and impact analysis.
- **Documented profiling and extension patterns** → the formal profiling mechanism with its defined specialization categories and composition patterns (Sections 3.2, 4.2, and 5.2), supporting both OGC-published shared profiles and local application usage.
- **Registration of all standards components** → the federated register infrastructure (Sections 3.4 and 4.4), with its defined register types, governance framework, and interoperability requirements for submission.

It is worth emphasizing that the OGC Building Blocks work provides one reference implementation that meets these criteria, but it need not be the only option. Instead, the ecosystem should be defined by principles and interoperability requirements, not by a single technical solution.

That is why the meta-principles above deliberately separate the policy goals from any particular implementation. The concepts should be evaluated on their own merits; the Building Blocks infrastructure demonstrates their feasibility and provides a concrete starting point for adoption. Alternative implementations that meet the same principles and interoperability requirements are welcomed — the register infrastructure itself provides the mechanism for coexistence. The membership should consider defining the available registers (e.g., for building blocks, profiles, vocabularies, schemas, APIs, and transformers) and the interoperability requirements for submission (e.g., required metadata and identifiers).

The strategy recommended for discussion is to advance these principles first, demonstrate their value through working implementations, and allow the membership to converge on policy language once the practical implications are well understood. None of these suggestions requires adoption as a complete package; they are designed to be individually valuable and incrementally adoptable.

8. Path Forward

This report presents the results of several years of research into what a machine-interpretable standards ecosystem could look like in practice. The architecture, workflows, and worked examples draw on repeated analysis of a wide range of real-world use cases and operational scenarios. Taken together, they indicate that the transition from document-centric standards to a machine-interpretable, register-based ecosystem is both feasible and worthwhile.

These results are offered to the OGC community as concrete suggestions for modernizing how standards are authored, published, profiled, discovered, validated, and governed. They are also offered to the broader geospatial community as a practical framework for reducing ambiguity and enhancing semantic interoperability and AI readiness across standards, data, and services. The intention is not to prescribe a single implementation path, but to provide a coherent architectural direction, supported by worked examples and reference implementations, that communities can adapt to their own operational and governance contexts.

The most valuable next step is community feedback grounded in concrete priorities and implementation needs. Which parts of the architecture are ready for near-term adoption, which areas would benefit most from shared infrastructure or policy support, and which additional use cases should be further developed are questions best answered in collaboration with standards developers, implementers, domain experts, and operational users.

The research team stands ready to continue this work with OGC members and partners. Further collaboration could include additional pilots, more worked examples, refinement of governance and security patterns, support for implementation and policy design, and

focused work in priority domains. Feedback on what is most useful, what is missing, and what is most urgently needed will help shape the next phase of the work.

Annex: Worked Examples

This annex presents detailed examples that illustrate how the machine-interpretable standards architecture described in this document applies to real-world problems. Each example follows a common structure: the real-world problem, why current approaches fail, how the architecture's building blocks, profiles, registers, and transforms address the problem, and what becomes possible — especially for AI agents — that was not possible before. The examples are deliberately chosen to span different domains, scales, and stakeholder communities, demonstrating the generality of the architectural approach.

Each example describes the building block decomposition in terms of the OGC Building Blocks framework: the *bblock.json* metadata, *schema.yaml* for JSON Schema definitions, *context.jsonld* for JSON-LD semantic uplift, SHACL shapes for content and logical constraint validation, *semantic-uplift.yaml* for pre- and post-processing transforms, and the continuous integration pipeline that automatically validates examples and test cases. For full documentation of the Building Blocks framework, see https://ogcincubator.github.io/bblocks-docs/.

A note on terminology. The examples below use a small number of recurring technical terms from the OGC Building Blocks framework. For readers unfamiliar with these, a one-line gloss may help:

- schema.yaml — declares the structure of the data (what fields exist, what types they are).
- context.jsonld — declares the meaning of the data (what each field corresponds to in a shared vocabulary), allowing data from different sources to be compared on an equal footing.
- SHACL shape — a machine-checkable rule that validates whether data meets the building block's requirements (for example, "the coordinate reference system must be declared").
- semantic-uplift.yaml — declares the transformation steps that convert raw data from its original form into the shared semantic form.
- JQ pre-processing — a way of restructuring JSON data before it is interpreted; used here to reshape data from various source formats into the shape the building block expects.
- SPARQL CONSTRUCT — a query that produces new semantic data from existing semantic data; used here for mappings and derived results.
- CI pipeline — continuous integration; the automated process that checks every building block, profile, and example against its rules whenever anything changes, so that problems are caught immediately.

A.1 Evaluating the Impact of Offshore Wind Farm Policies

A.1.1 The Problem

A national government has enacted policies promoting offshore wind energy. Five years in, policymakers need to evaluate cross-sectoral impacts: energy output versus targets, effects on commercial fishing yields within exclusion zones, changes in coastal tourism

revenue, displacement of bird migration corridors, and effects on seabed ecology. This requires defining composite indicators — such as "net economic impact per GW installed capacity" — that aggregate measurements from completely different domains: energy production telemetry, AIS vessel tracking data, species observation records, hotel booking statistics, and bathymetric surveys. Each domain uses its own feature types, observation models, spatial reference conventions, and temporal granularity. A marine biologist's "survey area" and a fisheries authority's "fishing zone" may overlap spatially but are defined in incompatible schemas. An AI agent asked to compute a composite indicator would have to reverse-engineer the semantics of each source.

Some relationships between variables—such as how wind speed influences electricity output—are already well understood from past research. Other relationships are uncovered through simulations or machine-learning models, which can reveal new patterns or help validate existing assumptions.
Whenever a new composite indicator is created, it must come with clear information about:

- which environmental or social factors were included,
- how each factor was defined,
- what units of measurement were used, and
- how reliable and complete the underlying data is.

This level of context and digital model boundaries helps policymakers—and AI systems—interpret the results correctly and confidently. It also enables several important capabilities:

- Combining the outputs of different sub-models into more general frameworks that can be applied in other regions or policy settings.
- Reusing digital-twin models across scenarios, adjusting them depending on what data is available, which transformations are needed, and which variables are included.

A.1.2 Building Block Decomposition

The architecture addresses this by decomposing the problem into reusable specification components, each implemented as an OGC Building Block with the standard directory structure (*bblock.json, schema.yaml, context.jsonld, examples.yaml, tests/)*.

An Observation Building Block, reused from OGC Observations, Measurements and Samples (OMS), provides the semantic core. Its *schema.yaml* defines the JSON structure for any observation: observed property, result (with value and unit of measure), feature of interest (with geometry), and *phenomenonTime*. Its *context.jsonld* performs semantic uplift, mapping *observedProperty* to *sosa:observedProperty*, result to *sosa:hasResult*, and *unitOfMeasure* to *qudt:unit* — each linked to a URI in a registered vocabulary. This mapping means that an energy production reading in MWh and a fish catch weight in tonnes both carry machine-interpretable unit declarations through the same structural pattern.

A Wind Farm Feature Type Building Block defines the schema for an offshore wind farm installation: identifier, geometry (polygon for the exclusion zone, point for individual turbines), installed capacity, commissioning date, and operator. Its *context.jsonld* maps these properties to URIs in an energy domain ontology. A SHACL shape (*shapes.shacl*) declares constraints: installed capacity must be a positive number with unit *qudt:MegaWatt*; geometry must be in EPSG:4326; commissioning date must be ISO 8601.

A Composite Indicator Building Block defines a schema for a policy indicator that formally declares its inputs (references to other building blocks by URI), its computation method (a formula or algorithm reference), its output unit, and its temporal aggregation rule. For example, the indicator "tourism revenue change per wind farm" declares inputs from the tourism observation profile and the wind farm feature type, specifies a spatial join (tourism observations within a 50 km buffer of wind farm geometry), a temporal alignment rule (annual aggregation), and an output unit (EUR/year). The *context.jsonld* maps all of this to an indicator ontology, enabling SHACL validation that every declared input actually exists in the register and that the computation references valid building blocks.

A.1.3 Profile Layering

A Marine Ecology Observation Profile constrains the generic observation building block: it binds *observedProperty* to a registered marine species vocabulary (via *shaclClosures* referencing the vocabulary's Turtle file), requires *featureOfInterest* to carry a marine habitat classification code, and mandates that the CRS for underwater observations includes a vertical datum declaration. This profile inherits all validation from the base observation building block — JSON Schema for structure, JSON-LD uplift for semantics, SHACL for content constraints — and adds its own SHACL shapes on top. The CI pipeline (tests/) includes conformant examples (a valid species count observation) and failure tests (an observation missing the habitat classification, expected to fail SHACL validation).

Similarly, a Fisheries Observation Profile, a Tourism Statistics Profile, and an Energy Production Profile each constrain the same base observation building block with their own vocabulary bindings, unit requirements, and domain-specific SHACL shapes.

A Wind Farm Policy Evaluation Profile sits at the top, declaring dependencies on all four domain profiles plus the composite indicator building block. Its *bblock.json* lists these as *dependsOn* entries (URIs in the OGC Building Blocks register). The framework automatically assembles the combined JSON Schema (via $ref composition), the merged JSON-LD context (combining all domain-specific *context.jsonld* files), and the aggregated SHACL shapes. A test case for this profile provides a complete policy evaluation dataset — energy observations, fisheries data, tourism statistics, ecological surveys, and composite indicator results — and the CI pipeline validates the entire chain in a single pass.

A.1.4 Semantic Uplift and Transforms

The critical capability is semantic uplift: raw JSON data from each domain agency is transformed into RDF using the building block's JSON-LD context, validated with SHACL, and made queryable via SPARQL. An AI agent tasked with computing the composite indicator "spawning ground displacement" can issue a SPARQL query against the uplifted

data, joining marine ecology observations (species counts before and after wind farm commissioning) with wind farm geometries (exclusion zone polygons), because both have been uplifted to the same semantic model through their respective building blocks.

Where a data source does not natively conform — for example, a legacy fisheries database that reports catch weights without explicit unit declarations — a registered transform (defined in *semantic-uplift.yaml* as a JQ pre-processing step) adds the missing unit annotation before the JSON-LD context is applied. This transform is itself a registered, versioned building block in the Transformation Register, testable and traceable.

A.1.5 What Becomes Possible

A policy analyst (or an AI agent acting on their behalf) can discover all relevant data sources for a wind farm impact assessment, verify that they share compatible semantics through common building blocks, compute composite indicators applying composite-indicator definitions — whether these are formally declared formulas, shared conventions, or documented methodologies — that are themselves registered and referenceable, and trace every result back to its source measurements with full provenance.

Because every source, transform, intermediate result, and final indicator is linked back to its registered building blocks, the analyst can also audit the assessment itself: verify the credibility of each data source, inspect which indicators and assumptions define the assessment boundary, and understand the uncertainty introduced at each step of the chain. A socio-ecological system analyst wishing to extend the model — by adding new indicators, connecting additional workflows, or tuning existing components — can do so by registering new assets against the same baseline building blocks, with human-readable documentation (property listings derived from the schema, and semantic definitions derived from the JSON-LD context) generated automatically from the canonical machine-interpretable forms.

All building blocks are published in federated registers: the base observation building block in the OGC Standards Building Blocks Register; the domain profiles in domain-specific registers operated by the relevant agencies; the composite indicator definitions in a policy evaluation register. An AI agent discovering “wind farm impact assessment” through the Feature Type Catalog traverses to the policy evaluation profile, follows its dependency chain to the domain profiles, and retrieves everything needed — schemas, contexts, SHACL shapes, transforms, and validated examples — to understand, validate, and compute the assessment.

A.2 Digital Building Permits

A.2.1 The Problem

A developer in Stuttgart submits a building permit application as a BIM model (IFC format). The municipality must check it against zoning regulations, setback requirements, height limits, floor-area ratios, fire safety clearances, environmental constraints (flood zones, protected trees), and utility easement zones. Currently this process takes weeks or months because it requires human experts to interpret regulations and compare them against submitted plans. Worse, the software vendor building the permit-checking

application must create bespoke integrations for each of Germany's approximately 11,000 municipalities, as they use different terminology, data models, and regulatory structures. One municipality talks about "fences," another about "parcel boundaries," a third about "lot lines" — for overlapping or identical concepts.

A.2.2 Building Block Decomposition

A Parcel Boundary Building Block defines the schema for a cadastral parcel: parcel identifier (national format), geometry (polygon), CRS declaration (mandatory, with SHACL constraint requiring an EPSG code), and area. Its *context.jsonld* maps *parcelId* to *inspire:localId* and geometry to *geo:hasGeometry*, ensuring that whether a municipality calls it a "parcel boundary," "lot line," or "*Grundstuecksgrenze*," the semantic identity is the same URI. This is exactly the problem the Building Blocks semantic uplift solves: different JSON property names across municipal schemas map to the same RDF predicate via their respective JSON-LD contexts.

A Building Footprint Building Block defines the schema for a building's ground-level outline extracted from the BIM model: geometry (polygon), height above ground (with mandatory vertical reference system declaration via SHACL — precisely the ambiguity problem from section 2, number of floors, and gross floor area.

A Setback Constraint Building Block is a rule building block rather than a data building block. Its *schema.yaml* defines the structure of a setback requirement: *minimumDistance* (value and unit), *measuredFrom* (feature type URI, for example the parcel boundary building block's URI), *measuredTo* (feature type URI, for example the building footprint building block's URI), and *applicableZone* (land use classification code). Its *context.jsonld* maps these to a building regulation ontology. A SHACL shape validates that the *measuredFrom* and *measuredTo* references resolve to registered building blocks, ensuring referential integrity. Height Limit, Floor-Area Ratio, and Fire Safety Clearance Constraint Building Blocks follow the same pattern. Any checking software — whether linked-data-based or operating on native geometric models — can use this information as an input to identify and inform the required geometrical analysis and checks.

A.2.3 Profile Layering

A German National Building Regulation Profile comprises all the constraint building blocks and binds them to the national regulatory vocabulary. It defines national-level defaults: CRS must be ETRS89/UTM (zone determined by location), vertical datum DHHN2016, and land use classification according to the Baunutzungsverordnung (BauNVO). The CI pipeline tests a complete permit application dataset against the full SHACL shape set.

A Baden-Württemberg State Profile profiles the national profile, adding the Landesbauordnung's additional fire safety rules, hillside building regulations for the Black Forest region, and specific vegetation setback requirements. It inherits all national constraints and adds its own SHACL shapes.

A Stuttgart Municipal Profile profiles the state profile, binds the land-use classification to Stuttgart's specific Bebauungsplan zones, adds local code lists for building types, and incorporates Stuttgart's flood zone designations derived from the Neckar River flood model. Because profiles inherit validation from their bases, a software vendor implementing the national profile gets all the common constraint-checking logic for free;

adapting to Stuttgart requires only implementing the delta — the local code list bindings and the flood zone extension.

A.2.4 Transforms

The IFC-to-geospatial transform is critical. The developer submits a BIM model in IFC; the permit system must check it against geospatial regulations defined in terms of parcels, setbacks, and zones. A registered transform building block (defined in semantic-*uplift.yaml*) converts IFC building geometry to CityJSON with declared CRS, extracts the building footprint polygon, and maps IFC property sets to the building footprint building block's schema. This transform uses the Building Blocks framework's support for JQ pre-processing (to restructure the JSON) followed by JSON-LD context application (to perform the semantic uplift). The transform is tested in the CI pipeline against reference IFC models with known correct outputs.

A.2.5 What Becomes Possible

An AI-powered permit system receives a submission. It queries the Profile Register for "Stuttgart building permit," retrieves the municipal profile, follows its *dependsOn* chain to the state and national profiles, and assembles the complete SHACL shape set. It uses the BIM-to-GIS converter to extract the building footprint. It retrieves the parcel geometry from Stuttgart's cadastral OGC API, which serves data conforming to the parcel boundary building block. The chosen geometric engine or other algorithm (flexibly chosen based on the specific building authority's needs and preferences) runs the geometric checks — is the building footprint within the parcel? Does it satisfy the setback distance? Does the height comply with the zoning limit? — using the formally declared constraints. It produces a structured compliance report that references each constraint by its building block URI, including pass/fail results and the specific rule evaluated. Software vendors build once against the national profile and adapt incrementally for each municipality. Future work may extend this example into areas not yet mature in production use — for instance, AI-assisted interpretation of complex geometric features (such as balconies or irregular envelopes) or automated geo-to-BIM enrichment — where the same ecosystem of profiles, building blocks, and registers would provide the semantic anchoring for those capabilities.

A.3 Climate Data for Local Impact Assessment in Urban Planning and Forestry

A.3.1 The Problem

Climate models used for European impact assessment — such as those from the EURO-CORDEX ensemble — typically produce data at approximately 12.5 km resolution, encoded in CF-NetCDF with domain-specific conventions: rotated pole grids, 360-day calendars, and precipitation expressed in kg m^{-2} s^{-1}. These conventions are well-suited to the climate science community but unusable to the downstream communities that most urgently need climate information. Urban planners need sub-kilometer climate indices that can be linked to municipal data on buildings, vegetation, and population exposure. Foresters need species growth projections at 50 × 50 m stand resolution, fused with soil, terrain, and satellite-derived canopy observations. Neither community can consume the

raw science outputs directly, and the adaptation work currently falls to bespoke projects that rebuild the same pipeline each time.

A.3.2 Why Current Approaches Fall Short

The gap between climate science data and downstream use is semantic, not merely syntactic. CF Standard Name tokens such as *tas* (near-surface air temperature) or pr (precipitation flux) are opaque to planning software that expects labeled properties. Coordinate reference systems, calendar conventions, units, and classification vocabularies differ across climate science, Earth observation, municipal GIS, and forest inventory systems, and they are rarely declared in a form that machines can reconcile. INSPIRE compliance alone does not bridge this gap, because INSPIRE harmonizes exchange formats but does not formally link climate variables to the downstream vocabularies that planners and foresters actually use.

A.3.3 Building Block Decomposition

Five building blocks decompose the problem. A Climate Variable Building Block maps each CF Standard Name to a URI, binds units to QUDT, and declares the calendar type. A Climate Index Building Block defines the schema for a computed indicator, with explicit declarations of the algorithm used, the ensemble statistics, and the baseline period. A Statistical Downscaling Building Block registers downscaling models as first-class assets, with references to conditioning data, training provenance, and uncertainty quantification. A Forest Site Assessment Building Block carries species taxonomy URIs, soil classification, and historical growth observations with temporal validity. An Urban Climate Vulnerability Building Block combines urban morphology, population, and exposure indicators into a composite index with declared aggregation logic. Each is accompanied by its *schema.yaml*, *context.jsonld*, SHACL shapes, and test cases.

A.3.4 Profile Layering

The profiles layer ranges from the general to the specific. A European Climate Assessment Profile sits at the base, binding the climate variable and climate index building blocks to European reference vocabularies and ensembles. An Urban Climate Services Profile and a Forest Climate Services Profile each constrain the base profile for their respective communities. At the bottom, municipal and national forest profiles — for example, a Stuttgart urban heat profile or a Czech Republic forest growth profile — bind local vocabularies and constraints. Each layer inherits all validation from below, so transferring the workflow to a new city or region requires only a new local profile, not a rebuilt pipeline.

A.3.5 Semantic Uplift and Transforms

Three transforms do the heavy lifting. A CF-to-Planning transform converts climate model outputs from their native conventions into planning-ready CRS, units, and calendars, implemented as a JQ pre-processor followed by JSON-LD context application and, where aggregation is required, a SPARQL CONSTRUCT step. A Sentinel-2-to-Forest-Health transform performs spectral band extraction and computes vegetation indices suitable for fusion with forest site assessments. A Municipal GIS Transform maps local classification vocabularies to the shared building block vocabulary via SKOS. Each transform is a registered, versioned building block with its own test cases in the CI pipeline.

A.3.6 What Becomes Possible

An urban planner asking about future heat vulnerability in a specific neighborhood, or a forester asking about species suitability on a specific stand over an eighty-year horizon, no longer needs to commission a bespoke data preparation project. An AI agent acting on their behalf discovers the applicable profile through the register infrastructure, assembles the dependency chain, applies the registered transforms, and returns results with fully traceable provenance. Climate outputs are published once, transforms are registered once, and new municipalities or forest regions are onboarded by adding a local profile rather than by rebuilding the pipeline.

A.4 Biodiversity Reporting under the EU Deforestation Regulation

A.4.1 The Problem

The EU Deforestation Regulation is a strong example of a domain where semantic interoperability is essential. At first sight, the task seems simple: identify where a commodity was produced, link it to a plot of land, and demonstrate that production is not associated with deforestation. In reality, this requires a much richer chain of meaning that connects supply-chain actors, production units, geospatial boundaries, temporal baselines, forest interpretations, biodiversity indicators, evidence packages, and due diligence statements.

The difficulty increases further when biodiversity-related reporting is added. Terms such as habitat extent, ecosystem condition, forest degradation, or high-biodiversity area may sound concrete, but they are often not defined precisely enough for consistent operational use. The result is that organizations may all claim to report on the same thing while actually using different geographies, baselines, classifications, and processing assumptions.

A.4.2 Why Current Approaches Fall Short

The current approach often focuses on data submission rather than on explicit meaning. Coordinates may be exchanged, documents may be attached, and reports may be generated, but the underlying semantics are frequently left implicit. A plot may be represented differently by different providers, linked inconsistently to farms, lots, or suppliers, and assessed against different forest maps or biodiversity interpretations without those differences being made visible.

This creates a situation in which outputs may look comparable while in fact resting on very different assumptions. Downstream users — whether regulators, auditors, banks, or buyers — cannot easily determine whether a difference in results comes from the source data, the baseline year, the classification method, the supply-chain linkage, or the reporting logic. The process therefore remains difficult to trust, difficult to compare, and difficult to automate safely.

A.4.3 How the New Architecture Helps

The architecture described in this document helps by turning the entire reporting chain into a set of explicit, machine-interpretable assets. Instead of treating EUDR reporting as a mixture of spreadsheets, GIS layers, PDFs, and proprietary workflows, the architecture

makes the relevant elements referenceable, structured, and connected. Geolocation, supply-chain linkages, baseline definitions, indicator semantics, transformations, validation rules, and provenance can all be described explicitly and made available through a governed ecosystem.

This shifts the focus from merely exchanging data to exchanging meaning. It allows every reportable result to be linked to its underlying definitions, assumptions, dependencies, and evidence. In doing so, it becomes possible to understand not only what a reported value is, but also how it was derived and under which interpretation framework it should be understood.

A.4.4 Building Block Decomposition

Building block decomposition is especially valuable because EUDR biodiversity reporting is not one single problem. It is a combination of smaller problems that recur across many use cases: identifying production plots, linking them to supply-chain assets, defining a forest baseline, expressing a biodiversity indicator, recording uncertainty, capturing provenance, and validating results. Treating all of this as one monolithic specification would make the system rigid, opaque, and hard to reuse.

By contrast, a building block approach allows each of these concerns to be treated as a reusable specification component. One building block can define the plot geometry, another the business linkage, another the baseline reference, another the habitat indicator, another the uncertainty statement, and another the validation result. This modularity makes assumptions visible, allows individual elements to evolve independently, and supports reuse across commodities, jurisdictions, and reporting contexts. It also means that the same building blocks can be assembled differently for EUDR compliance, broader biodiversity disclosure, or future domain-specific reporting needs.

A.4.5 Profile Layering

Profiles are needed to adapt the shared architecture to specific regulatory, sectoral, and organizational contexts without breaking interoperability. A first profile would be an EUDR compliance profile that defines the mandatory elements required for due diligence, such as geolocation, commodity identification, evidence references, and statement structure.

On top of that, commodity-specific profiles would likely be needed for sectors such as cocoa, coffee, soy, rubber, cattle, or wood, because their supply chains and practical reporting constraints differ. Biodiversity interpretation profiles would also be needed to define which forest or habitat baseline is used, what ecosystem concepts are in scope, at what scale analysis is performed, and how uncertainty is reported. Finally, jurisdictional or corporate profiles may be required to capture national implementation practices, sector conventions, or company-specific reporting commitments while preserving traceability back to the common base.

A.4.6 What Becomes Possible

Once this architecture is applied, biodiversity reporting under EUDR becomes much more transparent, comparable, and extensible. A company can report not only a conclusion, but also the exact baseline, interpretation profile, indicator definition, transformation

chain, and provenance that produced it. Regulators and auditors can inspect the logic behind a result instead of receiving only a final statement. Different providers can coexist without locking customers into a single proprietary interpretation, because their assumptions are explicit and therefore comparable.

The architecture also creates a path from narrow compliance to broader biodiversity intelligence. New indicators can be added incrementally. Alternative habitat or forest definitions can coexist in a governed way. Results from different providers can be compared with a clear understanding of where they differ. AI systems can support explanation, validation, and analysis because they no longer need to guess what the data means. In that sense, the architecture does not simply improve reporting efficiency; it makes the reporting domain itself more coherent, trustworthy, and operational.

A.5 Defense Geointelligence: Coalition Data Integration

A.5.1 The Problem

A NATO-allied nation with limited indigenous intelligence, surveillance, and reconnaissance (ISR) capabilities is contributing to a multinational coalition operation. Due to budget constraints and national strategy, it relies on a combination of sources for its geospatial intelligence picture: commercial very-high-resolution satellite imagery procured from allied commercial providers, terrain and elevation data provided by a partner nation's national geospatial-intelligence agency, feature data extracted by a third ally's automated AI-based exploitation systems, and the nation's own ground-truth observations collected by deployed forces. All of this data must be fused into a coherent, trusted common operational picture — in a secure, often air-gapped environment where internet-based discovery services are unavailable.

Each contributing nation and commercial provider uses different feature type catalogs, different classification vocabularies (both data classification and information security classification), different coordinate reference systems, different metadata profiles, and different levels of positional accuracy. A commercial imagery provider delivers orthorectified imagery with metadata in a proprietary JSON format; a partner nation provides vector feature data using its national adaptation of the NATO Geospatial Information Framework (STANAG 2592) with feature types from its national feature data dictionary; a third ally provides AI-extracted objects tagged with a machine-learning model's confidence scores but no formal link to any standardized feature type catalog.

A.5.2 Why It Fails Today

The Defence Geospatial Information Working Group (DGIWG) has been working on these interoperability challenges since 1983, developing defense profiles of ISO and OGC standards. Yet coalition interoperability exercises such as NATO's Coalition Warrior Interoperability Exercise (CWIX) consistently reveal that data shared between allies is often structurally exchangeable but semantically opaque. A feature labeled "building" in one nation's dataset may include only permanent structures, while another nation's "building" includes temporary shelters. An elevation value may be orthometric in one dataset and ellipsoidal in another, with no machine-readable declaration of which. Positional accuracy metadata may be present but expressed in different statistical

conventions (circular error probable versus 90% confidence ellipse versus root mean square error).

The problem intensifies with commercial data. Commercial providers are not bound by NATO STANAGs. Their imagery metadata may follow STAC with provider-specific extensions or a proprietary format. AI-extracted features from commercial analytics platforms arrive with labels derived from the provider's training data taxonomy — "vehicle," "aircraft," "vessel" — with no formal mapping to NATO's standardized target categories (STANAG 3596) or to any ally's national feature data dictionary.

Air-gapped operation adds a further constraint. The register infrastructure, schemas, vocabularies, and transformation engines that would normally be discovered via network services must be available locally, cached, and verifiable. If a building block definition or vocabulary has been updated since the local cache was provisioned, the system must know whether its cached version is still authoritative — a provenance and trust challenge that is architectural, not merely operational.

A.5.3 Building Block Decomposition

A Feature Observation Building Block provides the semantic foundation for any reported geospatial feature — whether extracted by a human analyst, an AI model, or derived from a partner nation's dataset. Its schema.yaml defines: feature type (URI from a registered feature type catalog), geometry (with mandatory CRS declaration via SHACL), temporal validity (observation time versus information currency time — a critical distinction in intelligence), source characterization (sensor type, platform, collection geometry), and positional accuracy (with mandatory declaration of the statistical convention used). The context.jsonld performs semantic uplift, mapping featureType to a predicate that resolves against the Feature Type Catalog, positionalAccuracy to a QUDT-compatible quality structure, and source to a provenance ontology.

A Confidence and Trust Metadata Building Block addresses a problem that the commercial and civilian worlds share but that defense makes existentially critical: how much should the consumer trust this data element? The schema defines: source reliability rating (adapting the NATO Admiralty Code's A–F scale for source reliability and 1–6 for information credibility, expressed as machine-readable URIs rather than free-text codes), AI model confidence score (a numeric value with declared confidence interpretation — probability, logit, or softmax), provenance chain (a linked list of processing steps, each referencing a registered building block or transformation), and information security classification (expressed as a structured object with classification level, caveats, releasability markings, and policy reference). The context.jsonld maps the classification structure to an information security ontology, enabling SHACL validation that releasability markings are internally consistent.

A National Feature Data Dictionary Mapping Building Block defines the mapping between a national feature type and the coalition's shared feature type catalog. Its schema declares: national feature type code, national feature type definition URI, coalition feature type URI, mapping confidence (exact match, close match, broad match, narrow match — using SKOS mapping predicates), and any semantic caveats. The context.jsonld maps these to SKOS, enabling SPARQL queries such as "find all features that are an exact match to coalition feature type bridge" across all contributing nations' datasets. A SHACL

shape validates that every national feature type used in a contributed dataset has a registered mapping, preventing the insertion of unmapped features that downstream consumers cannot interpret.

A Commercial Data Ingest Profile profiles the feature observation building block for data arriving from commercial providers. Its semantic-uplift.yaml defines JQ pre-processing steps that restructure common commercial formats — STAC item metadata, Maxar catalog JSON, Planet API responses — into the feature observation building block's schema. For AI-extracted features from commercial analytics platforms, a specific JQ transform maps the provider's taxonomy labels to the coalition Feature Type Catalog URIs, using a registered mapping table. Where no exact mapping exists, the transform inserts a skos:closeMatch relationship and flags the element for analyst review.

A Positional Accuracy Harmonization Building Block addresses the fact that different sources express positional accuracy differently: the US uses Circular Error Probable (CEP) at 90%, some European nations use root mean square error, and commercial providers may report CE90 or LE90. This building block defines a canonical accuracy model declaring the error value, the error metric (each as a URI from a registered accuracy methodology vocabulary), the confidence level, and the applicable dimension (horizontal, vertical, 3D). A registered SPARQL CONSTRUCT transform can convert between conventions where the mathematical relationship is defined and flags conversions requiring assumptions about error distribution for analyst review.

A.5.4 Profile Layering for Coalition Operations

A DGIWG Coalition Interoperability Profile sits at the top of the profile hierarchy. It composes the feature observation, confidence and trust metadata, feature dictionary mapping, and positional accuracy harmonization building blocks. It binds feature types to the DGIWG Feature Data Dictionary (or its successor), CRS to the EPSG register, and units to QUDT. Its bblock.json declares dependencies on all constituent building blocks, and the CI pipeline assembles the combined JSON Schema, merged JSON-LD context, and aggregated SHACL shapes. A test case provides a complete multi-source dataset — imagery-derived features from a commercial provider, vector data from an ally, and ground-truth observations from deployed forces — and validates that every element carries the required semantic declarations.

Beneath this coalition profile, each contributing nation creates a National Contribution Profile that maps its own national conventions to the coalition profile. A US National Contribution Profile maps NFDD feature types to the coalition Feature Type Catalog URIs, declares that US-sourced elevation data uses the EGM2008 geoid model, and binds the Admiralty Code source reliability ratings to the confidence and trust building block's registered vocabulary. A European ally's National Contribution Profile maps its national feature dictionary, declares EVRF2007 as the vertical reference, and maps its national classification markings to the coalition's information security ontology. These national profiles inherit all validation from the coalition profile and add their nation-specific SHACL shapes.

A.5.5 Air-Gapped Operation and Register Caching

The register infrastructure supports the air-gapped operational context through its caching and replication architecture. Before deployment, the coalition's register infrastructure is replicated to a secure, air-gapped environment. This includes all building block definitions (schemas, JSON-LD contexts, SHACL shapes, transforms), all vocabulary registers (feature types, units, accuracy methodologies, classification markings), and all registered transforms. Each cached asset carries provenance metadata: its authoritative source URI, its version, its retrieval timestamp, and a cryptographic hash.

In the air-gapped environment, an AI exploitation agent operates against the local register cache exactly as it would against the networked register. When it receives a new commercial imagery product, it queries the local cache for the commercial data ingest profile, retrieves the appropriate JQ transform, restructures the data, applies the JSON-LD context, validates against the SHACL shapes, and produces a coalition-conformant feature set — all without network access. The provenance chain records that the transform was applied from the cached version (with hash), enabling post-operation audit to verify that the cached transform was current at the time of use.

A.5.6 What Becomes Possible

An AI-powered intelligence fusion system receives three data packages: a commercial satellite imagery product with AI-extracted features in the provider's proprietary format, a vector feature dataset from an allied nation's geospatial-intelligence agency, and ground-truth observation reports from deployed forces. For the commercial data, the AI agent retrieves the commercial data ingest profile, applies the JQ transform, performs semantic uplift via JSON-LD, and validates via SHACL. Features that fail validation (missing CRS, unmapped feature types, no accuracy declaration) are quarantined for analyst review. For the allied data, the AI agent retrieves the ally's National Contribution Profile, applies its feature dictionary mapping (converting national feature codes to coalition Feature Type Catalog URIs via the SKOS mappings), and harmonizes the positional accuracy declarations. For the ground-truth data, observations pass validation directly because the tactical reporting system was built against the coalition profile.

The AI agent fuses the three validated, semantically harmonized datasets into a common operational picture. For each feature, it records the full provenance chain: which source contributed it, which profile and transform were applied, what the original and harmonized accuracy declarations are, what the confidence and trust rating is, and whether any semantic mapping involved approximation. An analyst reviewing the fused product can query this provenance — "show me all features where the feature type mapping was approximate" or "show me all elevation values that were transformed from a different vertical datum" — and make informed decisions about which elements require additional verification.

The architecture transforms coalition geointelligence from a labor-intensive manual reconciliation process into a semantically mediated, machine-assisted fusion pipeline. Contributing nations publish their data against declared national profiles; the building block inheritance chain ensures that every contribution carries the semantic declarations needed for automated fusion; registered transforms handle the mechanical aspects of harmonization; and the confidence and trust metadata building block ensures that the

uncertainty introduced at every step is tracked and traceable. AI agents can operate on this fused picture with known, declared semantics rather than inferred assumptions — exactly the requirement for well-performing, well-behaving AI described in section 1.

A.6 Integrated Cadastre, Land Use, and Deed System

A.6.1 The Problem

A country wants to unify its cadastral records (who owns which parcel, with survey-grade geometry), land use regulations (what activities are permitted on each parcel), and deed registry (the legal chain of ownership transfers) into an integrated system. Citizens, notaries, banks, tax authorities, and planning agencies all need different views of the same underlying reality. Today, a human must manually cross-reference three systems with different identifiers, different coordinate systems, and no machine-readable links between them.

A.6.2 Building Block Decomposition

A Parcel Identifier Building Block serves as the semantic anchor. Its schema.yaml defines a composite identifier: country code (ISO 3166-1), administrative unit code, parcel register number, and optional subdivision suffix. Its context.jsonld maps this to a nationally recognized URI pattern, so that the same parcel referenced in the cadastral database, the land use plan, and the deed registry resolves to a single, unambiguous identity. A SHACL shape validates the identifier format and requires the country code to match a registered code list.

A Parcel Geometry Building Block defines survey-grade polygon geometry with mandatory CRS declaration, positional accuracy metadata (expressed as a confidence interval building block), survey date, and surveyor reference. Its SHACL shapes enforce that accuracy must be declared and CRS must be from the national geodetic register. If the parcel includes a 3D boundary (for example, for subterranean or airspace rights), the vertical reference system must be declared — directly addressing the height ambiguity from section 2.

An Ownership Record Building Block defines the legal ownership information: owner identification (natural person or legal entity), share of ownership (fraction), encumbrances (mortgage, easement, usufruct — each as a nested building block with its own schema and semantic mapping), and temporal validity (from-date, to-date, with ISO 8601 format enforced by SHACL, preventing the date format ambiguity from section 2 by construction). Its context.jsonld maps encumbrance types to a legal ontology, enabling AI agents to reason about whether a mortgage exists on a parcel without understanding country-specific legal terminology.

A Land Use Designation Building Block defines the permitted use: land use category (bound to a registered classification vocabulary via shaclClosures), maximum building height, floor-area ratio, specific restrictions (environmental protection, heritage protection), and the regulatory reference (which law or plan establishes the designation). A SHACL shape validates that every land use category value appears in the registered vocabulary.

A.6.3 Cross-Domain Profiles and Constraint Rules

A National Integrated Land Administration Profile composes all four building blocks and adds cross-domain SHACL constraints that enforce referential integrity across the three systems. Every deed record must reference a parcel identifier that exists in the cadastral register (a SHACL shape cross-referencing a shaclClosure loaded from the cadastral register's RDF export). Every land use designation must reference a parcel geometry that is temporally current (a SHACL shape checking that the geometry's survey date is within an acceptable age). Every ownership transfer must reference both a valid deed and a corresponding parcel. These cross-domain constraints are tested in the CI pipeline with examples that deliberately violate each rule — a deed referencing a non-existent parcel, a land use designation on an expired geometry — confirming that the SHACL validation catches each case.

A.6.4 Semantic Uplift for Legacy Systems

The deed registry is a legacy text database. A semantic-uplift.yaml for the deed profile defines a JQ pre-processing step that restructures the legacy JSON export (which uses flat key-value pairs) into the nested structure expected by the ownership record building block's schema, followed by the JSON-LD context application that maps the restructured data to RDF. A SPARQL CONSTRUCT post-processing step then generates the cross-reference links between deed records and parcel identifiers, materializing the relationships that were implicit in the legacy data. All of this is tested and versioned as part of the building block's CI pipeline.

A.6.5 What Becomes Possible

A notary processing a property sale can query a single integrated view: the parcel geometry from the cadaster, the applicable zoning from the land use register, and the full ownership chain from the deed registry — all linked through shared building blocks and validated by cross-domain constraint rules. An AI agent assisting a bank with mortgage underwriting can automatically verify that the parcel exists, is correctly delineated, has no conflicting encumbrances, and conforms to the declared land use. The Feature Type Catalog registers "parcel" as a feature type with links to all three domain profiles, enabling an AI agent to discover that a query about "land ownership in district X" requires joining cadastral geometry, deed records, and land use designations — and to retrieve the transformation rules needed to align them.

A.7 Cross-Border Flood Risk Assessment

A.7.1 The Problem

The Rhine flows through Switzerland, France, Germany, and the Netherlands. An extreme rainfall event in the Swiss Alps requires a coordinated flood model that seamlessly combines terrain data, land cover, hydrological measurements, and critical infrastructure locations from all four countries. Each country publishes data using national coordinate reference systems, national height systems, national land cover classification vocabularies, and national hydrological observation conventions.

A.7.2 Building Block Decomposition

A Terrain Elevation Building Block requires: elevation value, horizontal CRS (EPSG code), vertical CRS (EPSG code), resolution (ground sample distance), and accuracy metadata. The SHACL shape enforces that both horizontal and vertical CRS are explicitly declared — directly addressing the ellipsoidal versus orthometric height ambiguity from section 2. The context.jsonld maps elevation to qudt:QuantityValue with unit qudt:M (meters) and CRS references to the EPSG register URIs.

A Hydrological Observation Building Block profiles the OGC OMS observation building block for discharge rate (cubic meters per second), water level (meters above a declared gauge datum), and precipitation intensity (mm/h). Each observation profile constrains the unit, temporal resolution (minimum 15-minute intervals for real-time flood forecasting), and requires the gauge's position to be declared in a specified CRS.

A Land Cover Building Block defines: geometry (polygon), classification code, classification system URI, and temporal validity. The context.jsonld maps classificationCode to a generic landcover:class predicate, and a SHACL shape requires the classificationSystem to be a URI from the registered vocabulary register — so the Swiss NOAS04 code "forest" and the German ATKIS code "Wald" are both valid but carry their system declaration, enabling machine-readable vocabulary mapping.

A.7.3 National Profiles and Registered Transforms

A Swiss Terrain Profile constrains the elevation building block: horizontal CRS = EPSG:2056 (LV95), vertical CRS = EPSG:5728 (LN02). A German Terrain Profile constrains to EPSG:25832 (ETRS89/UTM32) and EPSG:7837 (DHHN2016). A French Terrain Profile constrains to EPSG:2154 (RGF93/Lambert-93) and EPSG:5720 (NGF-IGN69). A Dutch Terrain Profile uses EPSG:28992 (Amersfoort/RD New) and EPSG:7415 (NAP). Each profile's bblock.json declares it profiles the base elevation building block, and the CI pipeline tests with actual terrain tiles.

CRS transformations between national systems are registered as transform building blocks. Each declares its source profile, target profile, transformation method (EPSG transformation code), and accuracy bounds. For example, the Swiss-to-German terrain transform declares: source = Swiss Terrain Profile, target = German Terrain Profile, horizontal transformation = EPSG:1676 (LV95 to ETRS89), vertical transformation via EVRF2007, accuracy not exceeding 0.05 m. The CI pipeline tests this transform against known control points.

A vocabulary mapping transform converts between the four national land cover classification systems. This is implemented as a SPARQL CONSTRUCT query (defined in semantic-uplift.yaml) that maps each national code to a shared European EAGLE land cover classification, using a registered mapping table loaded as a shaclClosure. The mapping is imperfect — some national categories have no exact equivalent — and these cases produce a skos:closeMatch rather than skos:exactMatch, preserving information about the mapping's confidence level.

A.7.4 What Becomes Possible

When heavy rainfall is forecast in the Swiss Alps, an AI-powered early warning system can automatically discover the relevant national datasets through the register infrastructure, apply the registered transformations to create a unified cross-border terrain and land cover model, retrieve real-time hydrological observations from all four countries' OGC API endpoints, validate them against the hydrological observation building block's SHACL shapes (confirming that units and gauge datums are correctly declared), apply temporal alignment to a common 15-minute grid, and feed the unified dataset into the flood model. The entire provenance chain — which national datasets were used, which transforms were applied, and which validation checks passed — is traceable through the building-block URIs and register entries. The result is seamless flood risk maps across all four countries, produced in minutes rather than the weeks of manual harmonization currently required.