THE BUILDING UNDERSTANDING STANDARD

BUS-1

Core Information Model, Portable Presentation, and Exchange Package

Part 1 of the Building Understanding Standard

Version 0.3 · Working Draft · August 2026

> The vendor keeps everything a user sees and feels. The owner keeps everything a draftsman or engineer decided.
>
> THE BOUNDARY TEST, APPLIED TO GRAPHICS

WORKING DRAFT FOR IMPLEMENTER REVIEW — NOT FOR PROCUREMENT USE

Status of this document

This is version 0.3 of BUS-1, the first part of the Building Understanding Standard. It is a working draft published for implementer review. It has not been ratified by any standards body, and no governance body has yet been constituted to hold it.

Version 0.3 closes four of the gaps its predecessor recorded against itself in Annex F.1, each of them measured against the two mature semantic models in this field rather than against opinion. Ports are declared by the asset that has them, so a package can now say which side of a valve, which branch, and that a coil has a supply and a return (9.4). A network is an entity and a protocol is a registered token, so the fact that two devices share a trunk is asserted once about the pair rather than repeated as a string on every point (8.10). `bus-vector-1` is specified as a closed profile of SVG, which makes figure mode implementable and gives the security requirement of 19.3 a format to attach itself to (13.4.3). And a sequence can state whether it is design intent or what a controller is actually running (10.5) — the narrow and honest part of a gap whose wide part, extraction of deployed control logic, remains open and has no precedent in any standard.

What this version declines is recorded with its reasons rather than passed over: no point-function taxonomy, no equipment classes for network hardware, no prototype templates, no control language, and no growth of the symbol role vocabulary until an implementer reports hitting its ceiling. A reviewer can argue with a decision. An omission gives them nothing to argue with. Annex F holds the declines; the companion Decision Record holds the reasoning behind every design decision, including the alternatives rejected.

New in this version are crosswalks in both directions. Annex G maps inward, from Project Haystack 4.0.0 and Brick Schema 1.4.4: it fixes the rules by which a foreign vocabulary reaches the `classifications` field, so that two implementers exporting the same source model produce the same package. Annex H maps outward, to object stores and property graphs: it fixes the four decisions a consuming ontology platform would otherwise make differently from the next one — link realization, denormalization, open vocabularies in closed stores, and where quality and provenance land. Neither maps a taxonomy onto anything; this document has none by design.

A reference implementation accompanies this draft and passes the Annex A suite in self-round-trip. Building it fed three clarifications back into this text — the newer-MINOR schema leniency noted in 18.2, the permitted envelope differences of 17.5, and the typed-model warning of 17.4 — which is the direction of travel a working draft should have.

Schema identifiers, namespace URIs, and the media type used in this draft are **provisional**. They resolve to a placeholder authority and will be reassigned when a governance body exists. Implementations **SHOULD** treat them as opaque strings and **MUST NOT** hard-code retrieval from them.

Reading the maturity labels

Every clause carries a maturity label. A draft that asserts uniform normativity across eighty pages overclaims, and it makes review harder because a reviewer cannot tell where an opinion is wanted.

- **AGREED** — settled in working session and stable enough to build against. Reversing it would invalidate work already done.
- **PROVISIONAL** — decided, but on thin evidence. Expected to be challenged by the first real implementation, and the place review effort is most valuable.
- **OPEN** — named because the decision is pending and its absence is load-bearing. A reader who assumes it was considered and settled would be wrong.

New in this version, from external review: a PROVISIONAL clause **SHOULD** end with a one-sentence *promotion criterion* — the concrete evidence that would move its label to AGREED. The label without the criterion invites opinion; the criterion turns it into a request for a specific experiment, and reports against those criteria are the most useful form of feedback this draft can receive.

Editorial conventions

This document and its companions use **US English** throughout, in prose and in identifiers. Contributors should not introduce British forms; a specification that spells the same word two ways invites the reader to wonder what else is inconsistent.

Two deliberate exceptions are retained as terms of art, and both appear in schema identifiers as well as prose:

- `storey` — kept rather than *story*, following `IfcBuildingStorey` in ISO 16739-1, and because *story* is ambiguous in a document that also discusses narrative records. Appears as a `spaceKind` value and in `storeysAboveGround`.
- `.buspkg`**,** `bus:` **and all schema field names** — lowerCamelCase and case-sensitive. These are identifiers, not words, and are never re-spelled to match prose conventions.

Requirement keywords are printed in a distinguishing color as a review aid; color carries no meaning and capitalization does. Inline code, schema field names, and enumeration values are set in a monospaced face.

Relationship to the Founding Charter

The Founding Charter states the commitments. This document specifies how they are met. Where the two disagree, the charter states the intent and this document is at fault.

Four charter commitments carry directly into normative requirements here, and are the ones most likely to be quietly broken by an implementation that otherwise validates:

- **The reconstruction guarantee** — a compliant package must be sufficient on its own — 2.4.

- **Import and export symmetry** — a system may not export less than it needs to operate — 2.4 and test EX-4.
- **No lowest-common-denominator dump** — omission is permitted, silent omission is not — 16.6.
- **Portability of representation** — geometry, semantic element identity, and bindings are owner assets and travel; appearance does not — clause 13.

Contents

1 Scope
2 Conformance
3 Normative references
4 Terms and definitions
5 Architecture and design rules
6 Identity and addressing
7 Datatypes, properties, and semantics
8 The information model
9 Relationships
10 Operational intent
11 Alarms
12 Events
13 Portable presentation
14 Provenance and governance
15 Extensions
16 The exchange package
17 The JSON binding
18 Versioning and compatibility
19 Security, privacy, and integrity
20 Relationship to existing work

A Conformance test suite outline (normative)
B Symbol role vocabulary, tier one (normative)
C Unit codes (normative)
D Worked example (informative)
E Schema files (informative)
F Open issues and decisions taken (informative)
G Mapping to Project Haystack and Brick (informative)
H Mapping to object stores and property graphs (informative)

1 Scope

AGREED

1.1 Purpose

This document specifies a vendor-neutral information model for the operational understanding of a building, a portable model for the graphics through which that building is operated, and a package format for moving both between systems without loss of meaning.

Its object is not interoperability at run time. Systems that exchange live values already have protocols for that. Its object is **continuity**: the ability of a building owner to take the accumulated understanding of a building out of one system and stand it up in another, years or decades later, without the cooperation of the system it came from.

1.2 What this version specifies

- An encoding-neutral information model for building identity, spatial structure, systems and assets, points, typed relationships, operational intent, alarms, events, and provenance (clauses 6 to 14).
- A portable presentation model: drawings, coordinate systems and georeferencing, layers, elements with symbol roles or embedded figures, and bindings including command actions (clause 13).
- A neutral vector figure format, `bus-vector-1`, defined as a closed profile of SVG with a fixed element list (13.4.3).
- A requirement that every property carry a definition declaring its datatype, unit, and meaning (7.8).
- A governed extension mechanism, including the rules that keep an extension from being silently discarded (clause 15).
- An exchange package: container, manifest, integrity, an actuation declaration, and a mandatory completeness declaration (clause 16).
- A JSON binding with schema documents and a canonical form (clause 17), and conformance profiles and round-trip test methodology (clause 2, Annex A).

1.3 What is deferred

Time-series history, deferred by both prior versions, is deferred no longer: it is specified by **BUS-2: Time Series**, Part 2 of this standard. Within this document, points declare whether durable history exists and over what extent (8.8); BUS-2 specifies how the history itself travels, and 8.8 states where the two parts meet. What follows is what remains deferred.

DOMAIN	IN V0.3	DEFERRED TO
Structured knowledge	`bus:KnowledgeNote` carries free text with a kind, an author, a review state, and typed attachment predicates (8.9).	BUS-3, structured knowledge types
Deployed control logic	A sequence declares whether it is design intent or what a controller is running, and carries the controller, the program identifier, and a checksum (10.5). The logic itself is referenced, not extracted.	unscheduled; no prior art exists
Analytic results	The attachment interface is normative (clause 15). The internal shape of an insight, diagnosis, or forecast is not.	BUS-4, analytic results

> NOTE The vector figure format was the one place v0.2 knowingly named something it did not define. It is defined in this version as a closed profile of SVG (13.4.3), which is what makes an element in figure mode portable in practice rather than only in principle. Where a symbol role exists, still prefer it: a role lets the receiving vendor apply its own design system and a figure cannot.

1.4 What is out of scope

- **Field and transport protocols** — BUS-1 records how a point is reached (8.6). It does not define how to reach it.
- **Proprietary method** — How a vendor arrives at a conclusion is not specified and not disclosed. Only the durable result is portable.
- **Rendering and interaction** — No requirement constrains artwork, color, animation, transitions, typography, responsive layout, navigation design, or interaction model. Clause 13 is explicit about where this line falls.
- **Three-dimensional geometry** — Presentation is two-dimensional. A 3D or BIM model is referenced as a resource. IFC already exists and is better at this.
- **Equipment taxonomy** — A small structural class set, with external classifications preserved verbatim (7.7). Not intended to compete with the ontologies that define taxonomies.

2 Conformance

AGREED Profile names and the reconstruction requirement are stable. The test suite of Annex A is normative as to what must be tested; fixtures are not yet published.

2.1 Requirement keywords

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **NOT RECOMMENDED**, **MAY**, and **OPTIONAL** are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals.

> NOTE In this draft the keywords are printed in a distinguishing color as an aid to review. Color carries no meaning; capitalization does.

2.2 Conformance targets

TARGET	WHAT IT IS	HOW IT IS TESTED
Package	A single artifact.	Statically, with neither system present. Annex A group PK.
Exporter	A system that produces packages.	By the packages it produces, and by whether the owner can produce one unaided. Group EX.
Importer	A system that consumes packages.	By what it retains, what it refuses, and what it reports. Group IM.
Round-trip	An exporter and importer as a pair.	Export, import, re-export, compare in canonical form. Group RT.

A system that both imports and exports **MUST** be evaluated as all three, and in round-trip against itself. Round-trip against itself is the weakest of the four and **MUST NOT** be reported alone.

2.3 Conformance profiles

PROFILE	TOKEN	DOMAINS
Core	`core`	identity, spaces, assets, points, relationships, resources, provenance
Core with intent	`core-intent`	core plus operational intent, alarms, and events
Core with intent and knowledge	`core-intent-knowledge`	the above plus human knowledge
Full	`full`	the above plus portable presentation and time-series history

All four profiles are normatively defined. The time-series domain required by `full` is specified by BUS-2, and a claim of `full` is tested against group TS of the suite as well as the groups of Annex A. The dispensation this clause carried through v0.2 — claiming `full` for presentation while declaring `timeSeries` absent with reason `deferredToLaterVersion` — is withdrawn: the domain is specified, that reason code is no longer a true statement about it, and an absent `timeSeries` domain needs a real reason under 16.6 like any other.

Property definitions are not a domain. They are infrastructure required of every profile by 7.8, and a package at any profile that emits a property without its definition is invalid.

Profiles are floors, not ceilings. An implementation claiming `core` **MAY** carry presentation, and **SHOULD** do so; it simply does not have its presentation handling tested.

2.4 The reconstruction requirement

Requirement 2.4-1

A conforming exchange package **MUST** be sufficient, on its own, for a conforming importer to reconstruct the portable understanding of the building it describes, without recourse to the exporting vendor, to undocumented behavior, or to any service that the exporting vendor controls.

1. **Self-sufficiency.** Every reference **MUST** resolve within the package, or **MUST** be a durable external locator with a digest (16.5). A reference only the exporting system can dereference does not satisfy this requirement, whatever its syntax. Test EX-1.
2. **Symmetry.** An exporter **MUST NOT** omit any construct that the exporting system itself relies upon to operate the building, unless the omission is declared under 16.6. Test EX-4.
3. **Preservation over comprehension.** An importer **MUST** preserve what it does not understand. Understanding is optional; retention is not. Tests RT-6, RT-7, RT-12.

For presentation specifically, reconstruction has a stronger practical test: an owner-role user **MUST** be able to export a floor plan and an equipment graphic, with geometry, element identity, and bindings intact, without vendor assistance (test EX-6). That test exists because the rebuild cost of graphics is the largest single line item in a real migration.

2.5 Claiming conformance

An implementation claiming conformance **MUST** publish a conformance statement giving the targets and profiles claimed, the document and schema versions, the result of every applicable test in Annex A including tests not passed, every extension namespace it emits with its must-understand flag, and the substrate domains it cannot carry with the reason codes it will emit.

A conformance statement that reports only passes is not a conformance statement. An owner reading it **MUST** be able to determine, before purchase, what the product can and cannot preserve.

3 Normative references

AGREED

The following documents are referred to in such a way that some or all of their content constitutes requirements of this document. For dated references, only the cited edition applies.

REFERENCE	TITLE AND NOTE
[RFC2119]	Bradner, S., *Key words for use in RFCs to Indicate Requirement Levels*, BCP 14, RFC 2119, March 1997.
[RFC8174]	Leiba, B., *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*, BCP 14, RFC 8174, May 2017.
[RFC3986]	Berners-Lee, T., et al., *Uniform Resource Identifier (URI): Generic Syntax*, STD 66, RFC 3986, January 2005.
[RFC8141]	Saint-Andre, P., Klensin, J., *Uniform Resource Names (URNs)*, RFC 8141, April 2017.
[RFC9562]	Davis, K., Peabody, B., Leach, P., *Universally Unique IDentifiers (UUIDs)*, RFC 9562, May 2024.
[RFC3339]	Klyne, G., Newman, C., *Date and Time on the Internet: Timestamps*, RFC 3339, July 2002.
[RFC9557]	Sharma, U., Ruiz, C., *Timestamps with Additional Information*, RFC 9557, April 2024. Applies where 7.3 permits an IXDTF suffix.
[RFC5545]	Desruisseaux, B., ed., *iCalendar*, RFC 5545, September 2009. Only the RECUR value type of section 3.3.10 is used.
[RFC4648]	Josefsson, S., *The Base16, Base32, and Base64 Data Encodings*, RFC 4648, October 2006.
[RFC8785]	Rundgren, A., Jordan, B., Erdtman, S., *JSON Canonicalization Scheme (JCS)*, RFC 8785, June 2020.
[RFC8259]	Bray, T., ed., *JSON*, STD 90, RFC 8259, December 2017.
[RFC6838]	Freed, N., Klensin, J., Hansen, T., *Media Type Specifications and Registration Procedures*, BCP 13, RFC 6838.
[BCP47]	Phillips, A., Davis, M., *Tags for Identifying Languages*, BCP 47.
[JSONSCHEMA]	*JSON Schema*, draft 2020-12, identified by meta-schema URI `https://json-schema.org/draft/2020-12/schema`.
[FIPS180-4]	NIST, *Secure Hash Standard (SHS)*, FIPS PUB 180-4, August 2015.
[ISO8601-1]	ISO 8601-1:2019 including Amendment 1:2022. Depended upon only through [RFC3339].
[ISO21320-1]	ISO/IEC 21320-1:2015, *Document Container File — Part 1: Core*. Constrains the ZIP container of 16.2.
[ISO3166-1]	ISO 3166-1 alpha-2 country codes.

REFERENCE	TITLE AND NOTE
[ISO19111]	ISO 19111:2019, *Geographic information — Referencing by coordinates*. Cited for the coordinate reference system identifier of 13.3.
[TZDB]	IANA Time Zone Database.
[BUSUNITS]	Annex C of this document. The unit code vocabulary.

> NOTE [ISO19111] is new in v0.2 and is cited only so that a georeference names its coordinate reference system in a recognized way. This specification does not otherwise depend on geographic information standards.
>
> NOTE Unit codes are defined in Annex C rather than adopted from an external registry. Annex C.3 publishes a mapping to the Unified Code for Units of Measure for implementations that need one; that mapping is informative and no conformance requirement depends on it.

4 Terms and definitions

AGREED

Terms are defined in the sense used here, which may be narrower than industry usage.

building understanding

The complete set of assertions about a building that a competent operator would need in order to run it as it is currently run: what exists, how it is arranged, how it is connected, what it is for, how it is operated through, what is known about it, and where each assertion came from.

portable understanding

That subset of building understanding which this standard requires to survive transfer between conforming systems.

entity

Any addressable object in the model. Every entity has a durable identifier and a type. Relationships, bindings, and presentation elements are entities.

intrinsic

True of a thing in itself, independently of anything else. The test that decides whether a fact is a property or a relationship (5.3.1).

point

An addressable item of operational data: a measurement, an instruction, a state, a configured parameter, a calculated value, or a meter register. A point is not a value; it is the thing a value is about.

property definition

A declaration of what a property name means, its datatype, and its unit. Required for every property an exporter emits (7.8).

operational intent

What a building or system is expected to achieve, as distinct from what it is doing and what it has done.

alarm definition

The durable definition of a condition, its limits and timing, its severity, and how it is to be routed, acknowledged, and shelved.

event

A timestamped occurrence attached to a subject: an alarm instance, an override, a mode change, a command, a service action. Not a time-series sample.

drawing

A page of portable presentation: a floor plan, equipment graphic, schematic, or one-line.

element

A placed thing on a drawing, identified by a symbol role, an embedded figure, or geometry.

symbol role

A hierarchical dotted path naming what an element depicts, independently of how it is drawn.

binding

A declaration that an element is driven by, or writes to, a building entity through a named channel.

channel

The aspect of an element that a bound value drives: its displayed value, its state, its fill, its rotation, and so on. A closed vocabulary (13.6.1).

telemetry binding

The protocol-specific information needed to reach a point in a live system. Distinct from a presentation binding.

exchange package

A single artifact conveying a portable understanding of one or more buildings, with a manifest that declares and verifies its contents.

reconstruction

Building, in a receiving system, a portable understanding semantically equivalent to the one in a package, without assistance from the system that produced it.

semantic equivalence

The relation holding between two models when, after canonicalization, there is a mapping between their entities preserving identity, type, normative properties, and the labeled relationship graph. Defined operationally in 17.5.

completeness declaration

The mandatory statement, in the manifest, of which substrate domains the package carries in full, in part, or not at all, and why.

actuation

The capability, carried by a write binding, to command plant from a graphic.

5 Architecture and design rules

AGREED The design rules of 5.3 are the generative principles of the model. Every entity-level decision in clauses 6 to 15 follows from them.

5.1 Three layers

LAYER	WHAT IT FIXES	LIFETIME
Information model (6 to 15)	Classes, properties, relationships, and the rules relating them. Normative and encoding-neutral.	Decades. Changes only through clause 18.
Binding (17)	How the model is written down. v0.2 defines one binding, in JSON.	Years. Further bindings may be added without touching the model.
Package (16)	How bound documents and resources travel and are verified.	Years. Container conventions change more often than the model.

An implementation that has encoded the JSON binding into its internal design, rather than the model, will find the second binding expensive. Implementers are warned.

5.2 The boundary

> **The boundary test**
>
> If it describes the building, it belongs in the standard. If it describes the software experience, it belongs to the vendor.

Applied to a graphic, the same test divides four layers rather than two, and the line falls between the third and the fourth. Clause 13.1 sets this out; it is the most consequential application of the boundary in this document.

5.3 Design rules

Six rules generate the model. Where a later clause looks arbitrary, it is usually one of these applied.

5.3.1 Objects own intrinsic properties only

Requirement 5.3-1

An entity **MUST** carry only facts intrinsic to itself. Location, membership, service, control, measurement, and every other contextual fact **MUST** be expressed as a typed relationship. An implementation **MUST NOT** introduce convenience properties that duplicate a relationship.

This is the only generative principle in the model. For any candidate field it answers object-or-graph without further argument, and it catches the mistake every building schema makes: treating location as a property of a thing rather than a fact about a pair of things. The cost is verbosity. The benefit is that any given fact lives in exactly one place, so the two cannot diverge.

> NOTE Four constructs carry a reference intrinsically and are deliberate exceptions: `Relationship` names its subject and object; `Binding` names its element and target; `Element` names what it represents; `Event` names its subject. In each case the entity exists in order to connect, and would be meaningless without the reference. The exception is not available to ordinary entities.

5.3.2 Relationships are first-class entities

A relationship has its own identifier, type, endpoints, qualifiers, validity period, provenance, and extensions (clause 9). This costs roughly a third of package size in the worked example. It buys the ability to say who asserted a connection and on what evidence, to end a relationship without deleting either endpoint, and to let an extension say something about a connection rather than about its endpoints.

5.3.3 Structure is specified; taxonomy is preserved

The model defines a small structural class set and a controlled predicate set. It does not define what kinds of equipment exist. External classifications travel verbatim with their source vocabulary and version (7.7), and the shared vocabulary that does emerge grows bottom-up through property definitions (7.8) rather than top-down from a committee.

5.3.4 Intent is separate from implementation and from state

What a building is meant to do is modeled separately from what it is doing and what it did. Sequences implement intent and are connected to it by `bus:implements`; they are not part of it. Conflating the three is the most common way intent is lost in migration.

5.3.5 Provenance is not metadata

Any assertion may carry who made it, how, and how confident they were. An assertion whose origin is unknown is worth materially less thirty years later, and the model makes that visible rather than hiding it behind a uniform surface.

5.3.6 Omission is allowed; silence is not

No exporter can carry everything. An exporter that carries less **MUST** say so, in a fixed vocabulary, in the manifest (16.6). This converts an unbounded quality problem into a bounded disclosure problem.

6 Identity and addressing

AGREED

Identity is the load-bearing wall of this standard. Every other guarantee — relationships, provenance, events, bindings, replacement — rests on the proposition that the thing referred to today is the thing referred to in twenty years.

6.1 Identifier form

Every entity **MUST** carry an `id` that is an absolute URI conforming to [RFC3986]. The `urn:uuid` form is **RECOMMENDED**, and version 7 UUIDs are **RECOMMENDED** within it because they sort by creation time and make large packages cheaper to diff.

An implementation **MAY** use `https` URIs under a domain it controls. If it does, it **MUST NOT** rely on those URIs being dereferenceable: a receiving system **MUST** treat every identifier as an opaque string and **MUST NOT** attempt retrieval in order to interpret the model. An identifier that must be fetched to be understood is a dependency on the exporting vendor and defeats 2.4.

6.2 Uniqueness

Within a package, an `id` **MUST NOT** occur more than once. Across packages describing the same building, an `id` **MUST** refer to the same entity. An exporter **MUST NOT** mint an identifier unique only within its own database unless the scoping is explicit in the form.

6.3 External identifiers

An entity **SHOULD** carry, in `externalIds`, every identifier by which other systems know it. This is how a receiving system re-establishes correspondence with a CMMS, a BIM model, a controller, or a spreadsheet this standard never touches.

```
"externalIds": [
  { "scheme": "org.bacnet.deviceInstance", "value": "201001",
    "authority": "BACnet/IP network 2001" },
  { "scheme": "org.buildingsmart.ifc.guid", "value": "1Xh2LmQz9E8vRk3TbYpQaB" },
  { "scheme": "com.example.cmms.assetId", "value": "AST-118240",
    "authority": "Maximo PRD" }
]
```

An importer **MUST** preserve every external identifier it receives, including those whose scheme it does not recognize. They are frequently the only bridge back to a system that has since been replaced.

6.4 Stability and identity mapping

An exporter **MUST NOT** change the `id` of an entity between exports because of an internal event: a database migration, a rename, a re-commissioning, a change of asset tag, or a change of product version. Tests EX-2 and EX-3.

Requirement 6.4-1

An importer that reassigns identifiers **MUST** emit, on subsequent export, a `bus:IdentityMapping` for every reassignment, with `equivalence` set to `same`. Reassignment without a recorded mapping is a conformance failure, not a quality issue, because it silently severs every relationship, provenance chain, event, and binding that pointed at the old identifier.

`IdentityMapping` also carries `narrower`, `broader`, and `related` for cases where correspondence is real but not exact — a merge, a split, a partial match against a foreign system. These preserve the fact that a human once decided the two were connected, which is otherwise lost entirely.

6.5 Local identifiers

An entity **SHOULD** carry a `localId`: the name the building actually goes by. `AHU-1`. `VAV-2-01`. A `localId` **MUST** be unique within a package and **MUST NOT** be used as a reference target. It exists because a durable identifier is unreadable, and because in thirty years the tag on the physical unit may be the only surviving link between the model and the machine.

6.6 Revision

PROVISIONAL New in v0.2. The semantics of revision under concurrent editing are not yet specified.

An entity **SHOULD** carry a `versioning` block with a `revision` that increases monotonically with each change to that entity. Revision supports synchronization and merge, which history alone does not: a receiving system can determine whether the copy it holds is behind the one it has been sent without comparing every field.

An importer **MUST** preserve or advance a revision. It **MUST NOT** reset one. An entity re-exported at revision 1 having arrived at revision 4 is a conformance failure (test RT-24), because it asserts that three changes never happened.

```
"versioning": {
  "revision": 4,
  "schemaVersion": "0.2",
  "effectiveFrom": "2009-09-14T00:00:00-05:00"
}
```

6.6.1 Single-writer assumption

This version assumes a single authoritative writer for any given entity at a point in time. An exporter that receives concurrent updates from multiple systems **MUST** do one of two things: serialize them under a single writer and advance `revision` monotonically, or emit distinct packages with distinct `packageId`s and record the succession under 16.9. Doing neither — advancing one revision counter under two authorities — produces a package whose revision numbers assert an ordering nobody enforced, and is a conformance failure (test EX-11).

A future MINOR version is expected to define a minimal multi-writer merge rule: identity mapping for structural divergence, and last-writer-wins or an explicit `ChangeRecord` for property divergence. Until that rule is published, an implementer holding concurrent writes **SHOULD** treat them as requiring a new `packageId`. The assumption is stated here rather than left implicit so that an implementer who correctly assumes single-writer today does not discover the boundary by crossing it.

Promotion criterion — at least two independent implementations report successful round-trip of revision under concurrent package exchange, or the multi-writer merge rule is ratified.

7 Datatypes, properties, and semantics

AGREED Property definitions (7.8) are new in v0.2 and are the change most likely to be felt by implementers.

7.1 Primitive types

Five primitives: string, number, boolean, timestamp, and null. A binding **MUST** specify its numeric range and precision limits and **MUST NOT** silently narrow a value.

7.2 Text and language

Any property typed `Text` accepts either a plain string or a language map keyed by [BCP47] tags. An importer supporting one language **MUST** preserve the full map and **MUST NOT** collapse it. Buildings outlive the tenancy that set their working language.

7.3 Timestamps

A timestamp **MUST** be an [RFC3339] `date-time` with an explicit offset. A bare local time is not permitted anywhere in this standard.

An [RFC9557] IXDTF suffix **MAY** be appended to carry the originating time zone, and **SHOULD** be used where the civil time zone is operationally significant — schedule boundaries, occupancy transitions, event timestamps, and anything a human will later read as a wall-clock time. An importer **MUST** accept and preserve such a suffix even if it does not interpret it.

```
"assertedAt": "2019-04-11T14:00:00-05:00"
"at":         "2017-05-09T11:20:00-05:00[America/Chicago]"    (preferred for civil time)
```

> NOTE Offset alone is not a time zone. A package recording only offsets loses the ability to reason about a daylight-saving transition that happened after export, which is exactly the reasoning an operator needs when reading a ten-year-old event log.

7.4 Quantities and unit codes

A dimensional value **MUST** be expressed as a `Quantity`: a magnitude and a `unitCode`.

```
"grossArea":   { "value": 90000, "unitCode": "ft2" }
"designFlow":  { "value": 12500, "unitCode": "cfm" }
"lowLimit":    { "value": 50.0,  "unitCode": "degF" }
"filterAlarm": { "value": 1.0,   "unitCode": "inH2O" }
```

Unit codes are drawn from the vocabulary of Annex C. The codes are the ones an engineer writes on a drawing: `degF`, `cfm`, `gpm`, `inH2O`, `kWh`, `tonRef`. An operator reading a package should not have to consult an external registry to know what a value means, and should never meet an unfamiliar rendering of their own working units.

A `unitSystem` field **MAY** name a different vocabulary — `ucum`, `haystack`, `bacnet`, or `qudt` — for a unit Annex C does not cover. Absent the field, the vocabulary is `bus`. This is the same pattern the standard uses for classifications: define the small thing that is needed, and name the external vocabulary where one is used rather than forcing everything through it.

Requirement 7.4-1

An importer **MUST NOT** convert a quantity to a different unit as a condition of storing it. Where an implementation converts for internal use, it **MUST** retain the original magnitude and unit code and re-export them unchanged. Round-tripped conversion is a silent, compounding form of loss, and it destroys the ability to tell a measured value from a computed one.

> NOTE Temperature differences use the delta forms. A two degree rise is `deltaDegF`, not `degF`. Conflating the two is the most common unit error in building data and the model makes it expressible rather than relying on context.

7.5 Enumerations

Where a point or property carries a discrete state, the enumeration **MUST** be declared rather than assumed from a protocol. Each member has a `code`, an **OPTIONAL** `label`, and an **OPTIONAL** `isAlarm` flag.

Enumerations defined by this standard are open. An importer encountering an unrecognized member **MUST** preserve the value and **MUST NOT** coerce it to a default (test IM-4). This is what allows the vocabulary to grow in a MINOR version without breaking deployed importers.

7.6 Digests

A digest is written `<algorithm>-<base64>`. Only `sha-256`, `sha-384`, and `sha-512` are defined, per [FIPS180-4], encoded per [RFC4648] section 4.

7.7 Classifications

A `classification` records that an entity corresponds to a term in a taxonomy defined elsewhere: a `system`, an **OPTIONAL** `version`, a `code`, and an **OPTIONAL** `label`. An entity **MAY** carry any number, including several from the same system and several a receiving implementation considers inconsistent. This standard does not adjudicate between them.

Requirement 7.7-1

An importer **MUST** preserve every classification verbatim, including `system` tokens and `version` values it does not recognize, and **MUST NOT** normalize, translate, deduplicate, or drop them.

> NOTE The `version` field matters more than it appears to. A Brick 1.3 class URI and a Brick 1.5 class URI may be the same string with different meanings. Recording the version is what makes a mapping possible later; omitting it is what makes it guesswork.

7.8 Property definitions

AGREED New in v0.2. Replaces the free-form property map of v0.1.

Every intrinsic fact about an entity that is not modeled by a named field is carried as a property value, and every property value references a `bus:PropertyDefinition`.

Requirement 7.8-1

An exporter **MUST** include, in the package, a `bus:PropertyDefinition` for every property it emits, declaring the property name, its datatype, its unit where dimensional, and its meaning in prose. Bare key-value property data **MUST NOT** appear anywhere in a conforming package. An importer **MUST** reject a package containing a property value whose definition is absent.

```
{ "type": "bus:PropertyDefinition",
  "propertyName": "filterClass",
  "datatype": "string",
  "meaning": "Particulate filtration efficiency rating of an air filter, stated in
              the rating scheme named by the value, for example a MERV rating
              under ASHRAE 52.2.",
  "origin": "manufacturer" }

"properties": [ { "definition": "urn:uuid:...076", "value": "MERV 13" } ]
```

The requirement exists because a free-form property map interacts badly with the completeness declaration, which is the honesty mechanism this standard leans on. An exporter could place most of what it knows in bare key-value pairs, declare the domain complete, and pass every other test — technically true, materially false.

The cost falls in the right place. A vendor with two hundred distinct property names writes two hundred definitions once per product and ships them with every export, whether the package covers fifty assets or fifty thousand. That is a one-time engineering cost on the vendor rather than a recurring interpretation cost on the owner.

7.8.1 Consequences

- **Quantity kinds resolve here** — a point's `quantityKind` is a reference to a property definition rather than a free token (8.5). Point semantics and asset properties use one mechanism instead of two.
- **Package quality becomes measurable** — the proportion of property values that are defined against those that are not is a figure an owner can read before purchase. It converts "is this export any good" from a judgment into a number.
- **Shared vocabulary grows bottom-up** — definitions that many independent vendors emit identically are evidence of real shared meaning, and become promotion candidates into the core through the mechanism of 15.5. The standard never has to guess a taxonomy in advance.

Definitions are entities, so they carry provenance and revision. Who defined `filterClass`, when, and what changed when the definition was revised are all answerable.

A definition **MUST NOT** be silently merged with another of the same name but different meaning (test RT-21). Two vendors using `capacity` to mean different things is a fact

about the world, and flattening it destroys the evidence that a human needs to reconcile them.

> NOTE An implementation that claims conformance **SHOULD** be prepared to report, on request, two figures for any package it produces: the number of distinct `bus:PropertyDefinition` entities emitted, and the share of property values carrying `provenance`. These are not pass/fail criteria — a package thin on both remains valid — and the first draft of this note proposed a third figure, the share of values referencing a definition, which this clause makes constitutionally one hundred percent and therefore worthless as a signal. The two that remain vary, and they convert *is this export any good* from a judgment into numbers an owner can compare across vendors before purchase.

8 The information model

AGREED The class set was reduced from v0.1 in two places and extended in four. See Decision Record group B.

8.1 The entity envelope

PROPERTY	REQUIREMENT
`id`	**REQUIRED**. Clause 6.
`type`	**REQUIRED**. A core class prefixed `bus:`, or an extension class in a declared namespace.
`abstractTypes`	**RECOMMENDED**. Ancestor classes, most specific first, so a receiver can place an entity whose leaf class it does not recognize. 8.2.
`localId`	**RECOMMENDED**. Unique within the package. Not a reference target.
`name`, `description`	**RECOMMENDED**. Text, per 7.2.
`externalIds`	**RECOMMENDED** where other systems know the entity. 6.3.
`classifications`	**RECOMMENDED**. 7.7.
`properties`	**OPTIONAL**. Property values, each referencing a definition. 7.8.
`lifecycle`	**RECOMMENDED** for physical entities. Status and the dates bounding service life.
`versioning`	**RECOMMENDED**. Revision, schema version, effective period, deprecation. 6.6.
`provenance`	**RECOMMENDED**. Clause 14.
`extensions`	**OPTIONAL**. Clause 15.

An entity carrying nothing beyond `id` and `type` is valid. It asserts existence and nothing more, which is occasionally the honest thing to assert.

> NOTE v0.1 carried a `References` field group in the envelope. It was removed: it violates 5.3.1 and duplicates the `bus:documentedBy` relationship.

8.2 Abstract types and graceful degradation

PROVISIONAL New in v0.2.

Promotion criterion — an importer that does not understand a leaf class correctly places the entity via `abstractTypes` in a published conformance run.

An entity **SHOULD** declare its ancestor classes. A receiver meeting `bus:Asset` with `assetKind` of a value it does not recognize, or an extension class it has never seen, can then still place it correctly rather than treating it as opaque.

```
"type": "bus:Asset",
"assetKind": "equipment",
"abstractTypes": ["bus:Asset", "bus:PhysicalEntity", "bus:Entity"]
```

The same principle governs symbol roles (13.4.2), and the two **SHOULD** behave alike: unknown leaf, known ancestor, correct handling.

8.3 Parties

`bus:Party` is a person or organization with a role in the building: owner, operator, occupant, tenant, service provider, vendor, integrator, consultant, or authority.

The owner **SHOULD** be identified and named in the manifest subject. A standard whose purpose is owner rights, in which the owner is not addressable, cannot state whose asset a package is.

> NOTE Provenance may name an acting party inline as an `Agent`. Where the party is addressable, the agent **SHOULD** carry a `partyRef` so that the two are connected.

8.4 Buildings, spaces, and zones

Three classes carry structure, on a deliberate cut.

- `bus:Building` — the unit of ownership and transfer. The manifest subject points at buildings. Carries address, geographic position, time zone, and gross figures that let a receiving system sanity-check a model before trusting it. `timeZone` **MUST** be an identifier from [TZDB]; a fixed offset is not acceptable.
- `bus:Space` — a bounded region, with a `spaceKind` of site, storey, room, open area, circulation, shaft, plant room, exterior, parking, or roof. Spaces nest through `bus:contains`, so a site contains a building's spaces and a storey contains rooms. Where `spaceKind` is storey, a signed `ordinal` gives an orderable value that survives translation, with 0 the principal ground storey.
- `bus:Zone` — a set of spaces behaving as one for some purpose, named by `zoneKind`. A zone is not a region and has no boundary. Modeling a zone as a space makes containment lie.

Spatial containment is expressed with relationships, not parent pointers, so a space **MAY** be contained by more than one thing — the usual case for a space that is both on a storey and in a fire compartment.

> NOTE v0.1 had five classes here. Site, storey, and room differ in attributes rather than in kind, and were collapsed. Building and Zone were kept for the reasons above.

8.5 Systems, asset types, and assets

`bus:System` groups assets that together deliver something, named by a coarse `systemKind` and, where relevant, a `medium`. Systems are not physical; they are how the building is reasoned about.

`bus:Asset` is a single class with an `assetKind`:

ASSETKIND	WHAT IT IS	EXAMPLE
`equipment`	A functional unit reasoned about as a whole.	An air handling unit, a chiller, a VAV terminal.
`component`	A separately identified, maintained, or replaced part.	A filter bank, a belt, a reheat valve.
`device`	A thing that hosts points.	A field controller, a gateway, a meter.
`virtual`	An asset existing only in software but reasoned about as one.	A calculated whole-building meter.

The distinction between equipment and device is functional, not physical. A packaged unit with an integral controller is one physical object and **SHOULD** be modeled as two entities related by `bus:hasPart`, because the controller will be replaced several times during the life of the unit and history should follow the right one.

An asset **MAY** declare the ports at which connections land — which side of a valve, which of a coil's two water tappings. Ports are declared by the asset and named by the relationship; 9.4 specifies them.

`bus:AssetType` carries what is true of a product rather than of a unit: manufacturer, model, product line, and rated properties. An asset references it through `assetTypeRef`. Four hundred identical terminal units then carry one copy of the product facts rather than four hundred, and replacing every actuator with a new model becomes a single assertion.

PROVISIONAL The type and instance split is new in v0.2 and adds an indirection every reader must resolve. It is the change most likely to irritate implementers.

8.6 Points

PROPERTY	REQUIREMENT
`pointKind`	**REQUIRED**. `sensor`, `setpoint`, `command`, `status`, `parameter`, `derived`, or `meter`.

PROPERTY	REQUIREMENT
`dataType`	**REQUIRED**. `number`, `boolean`, `string`, `enum`, or `timestamp`.
`unitCode`	**REQUIRED** where `dataType` is `number` and the quantity is dimensional. Annex C.
`quantityKind`	**REQUIRED** for numeric points. A reference to a `bus:PropertyDefinition`. Its unit code **MUST** agree with the point's unit code.
`range`, `resolution`	**RECOMMENDED**. The operating envelope, not the datatype limits.
`enumeration`	**REQUIRED** where `dataType` is `enum`.
`meterScope`	**RECOMMENDED** where `pointKind` is `meter`. `site`, `submeter`, or `virtual`. A virtual meter is computed from other meters rather than measured, and a receiver **MUST NOT** re-bill from one without saying so.
`writable`	**REQUIRED** where the point is `command` or `setpoint`, and **MUST** be `true` for those kinds.
`commandPriority`	**RECOMMENDED** where the point is written through a prioritized mechanism. Configuration only, never live state.
`binding`	**REQUIRED** where the point is acquired from a live system. 8.7.
`history`	**RECOMMENDED**. Declares whether durable history exists and over what extent. 8.8.

`pointKind` is the durable distinction between an observation, an instruction, a state, and a configured value. It is separate from `dataType` because the two are independent, and collapsing them is how systems lose the ability to tell a setpoint from a reading.

> NOTE A writable point is not necessarily a command. A `parameter` may be writable and is not an instruction to the plant.

8.7 Telemetry binding

A `binding` on a point records how it is reached: `protocol`, `address`, an **OPTIONAL** `deviceRef`, an **OPTIONAL** `networkRef`, and `parameters`. `protocol` is a token from the registry of 8.10.2. The free-string `network` of v0.2 is deprecated in favor of `networkRef` and **MUST** still be accepted; 8.10.3 sets out the transition.

```
"binding": {
  "protocol": "bacnet",
  "address": "analog-input,3",
  "deviceRef": "urn:uuid:...053",
  "network": "BACnet/IP 2001",
  "parameters": { "objectName": "AHU1_SAT", "covIncrement": 0.2 }
}
```

`address` and `parameters` are opaque to this standard. An importer **MUST** preserve them verbatim (test RT-9) and **MUST NOT** normalize them, even where it recognizes the protocol and believes it knows a better form.

This is the least glamorous part of the model and among the most valuable. A migration preserving every semantic relationship but losing the addresses obliges an integrator to rediscover and rebind tens of thousands of points by hand.

> NOTE A telemetry binding is not a presentation binding. The two share a word and nothing else: one says how a value is acquired, the other says what a value drives on a page.

8.8 History at the interface

BUS-1 does not carry time-series payloads, and as of this version that is a boundary between parts rather than a deferral: **BUS-2: Time Series** specifies how history travels — a `bus:Series` entity declaring each series, and a sample file carried as a package resource under the digest machinery of 16.5. What this clause carries is the point-level declaration that durable history exists, its extent, and its nominal interval, which is what a receiving system needs in order to know what it is missing. An exporter holding history and not including it **MUST** declare the `timeSeries` domain as partial or absent (16.6).

> NOTE `history.resourceRef` is deprecated in this version in favor of `bus:Series.resourceRef` (BUS-2 clause 3): it was a locator with no statement of format, coverage, unit, or count. It **MUST** still be accepted, per 15.5. Where both are present, the series entity is the statement and the point field is a label.

8.9 Knowledge

`bus:KnowledgeNote` carries a `noteKind`, a `body` of free text, an **OPTIONAL** `observedAt`, an **OPTIONAL** `reviewState`, and provenance. It is attached by `bus:explains`, `bus:supports`, `bus:justifies`, or the generic `bus:appliesTo`.

Requirement 8.9-1

An importer at profile `core-intent-knowledge` or above **MUST** preserve the full text of every knowledge note, unaltered and untruncated, together with its kind, its author, and its attachment. Knowledge notes are the part of the model with no other copy: the engineer who wrote one has, by the time it matters, retired.

> NOTE Structured knowledge types are deferred to BUS-3. A premature structure would be worse than free text, because it would encourage implementers to discard what does not fit.

8.10 Networks and drivers

PROVISIONAL New in v0.3. The network entity and the protocol registry close the first half of the gap Annex F.1 recorded; driver configuration is declined, in 8.10.4, on the record.

Promotion criterion — three independent exporters produce network entities for MS/TP or BACnet/IP trunks, and importers preserve `networkRef` without normalization.

8.10.1 The network entity

`bus:Network` is a communications network that devices sit on: an MS/TP trunk, an IP subnet, a radio mesh, a driver's list of devices. It carries a **REQUIRED** `networkKind`, an **OPTIONAL** `protocol`, and the `identifier`, `segment` and `parameters` by which the source system names it.

Through v0.2 the network was a free string on each telemetry binding. That put the same fact in as many places as there were points, and it put it in the wrong place. Sharing a trunk is a fact about a set of devices, not an intrinsic property of any one of them, which is precisely what 5.3.1 forbids. It also failed in practice: two controllers whose bindings read `BACnet/IP 2001` and `BACnet IP 2001` are on the same wire and no receiving system can tell, because there is nothing for the two strings to be the same as.

A network entity is that thing. It is asserted once, every binding on the trunk points at it, and `bus:hostedBy` and `bus:connectedTo` relate devices to it in the ordinary way.

PROPERTY	REQUIREMENT
`networkKind`	**REQUIRED**. One of `fieldbus`, `ip`, `wireless`, `serial`, `virtual`, `other`.
`protocol`	**RECOMMENDED**. A token from the registry of 8.10.2.
`identifier`	**RECOMMENDED**. How the network is named in the source system — a BACnet network number, a subnet, a trunk label. Opaque.
`segment`	**OPTIONAL**. A subdivision within the network, where the source system distinguishes one.
`parameters`	**OPTIONAL**. Vendor network settings. Opaque, preserved verbatim.

NETWORKKIND	WHAT IT IS
`fieldbus`	A wire the devices share. An MS/TP trunk, a LON segment, an RS-485 loop.
`ip`	A routed network. A subnet, a VLAN.
`wireless`	A radio network.
`serial`	A point-to-point serial link.
`virtual`	Exists only in software: a driver's device list, an aggregation with no wire behind it.
`other`	Anything else, and what an importer writes when the source does not say.

> NOTE An importer whose source carries no network kind **MUST** write `other` and **MUST NOT** infer `fieldbus` from the presence of a device list. Project Haystack, whose root `^network` entity type this follows, carries no kind at all.

8.10.2 The protocol token registry

`protocol` is a token from the registry published as `bus-protocols.json`. It holds 22 tokens: 18 seeded from the protocol defs of Project Haystack 4.0.0, and the remainder field buses that implementers reported and for which Haystack has no def.

The registry is **registered but open**, in the sense of 7.5. A token outside it is valid and **MUST** be preserved verbatim. An exporter for which a registered token exists **MUST** use it rather than a variant spelling.

This does not close the field and is not meant to. It fixes the spelling of the protocols that actually recur, which is the whole of the problem: through v0.2 an exporter writing `bacnet` and one writing `BACnet/IP` both validated, and a receiver could not determine whether two bindings reached the same kind of device.

TOKEN	WHAT IT NAMES	TYPICAL NETWORK KIND
`bacnet`	ASHRAE building automation and control protocol.	`fieldbus`
`bluetooth`	Short range wireless communication protocol.	`wireless`
`coap`	Constrained Application Protocol.	`ip`
`dali`	Digital Addressable Lighting Interface protocol for lighting.	`fieldbus`
`ftp`	File Transfer Protocol.	`ip`
`haystack`	Haystack HTTP protocol for exchanging tagged data.	`ip`
`http`	Hypertext Transfer Protocol which is foundation of the web.	`ip`
`imap`	Internet Message Access Protocol for retrieving email.	`ip`
`knx`	KNX protocol commonly used for lighting systems.	`fieldbus`
`lonworks`	ISO/IEC 14908 control network protocol, still widespread in installed plant.	`fieldbus`
`modbus`	Register based communication protocol used with industrial devices.	`fieldbus`
`mqtt`	Message Queuing Telemetry Transport publish/subscribe protocol.	`ip`
`mstp`	BACnet MS/TP over EIA-485. Distinguished from bacnet because the trunk is the network and the addressing differs.	`fieldbus`
`n2`	Johnson Controls N2 bus. Legacy, and the reason an exporter meets undocumented addressing.	`fieldbus`
`obix`	XML based Open Building Information eXchange protocol.	`ip`

TOKEN	WHAT IT NAMES	TYPICAL NETWORK KIND
`opc-ua`	IEC 62541 OPC Unified Architecture.	`ip`
`pop3`	Post Office Protocol version 3 for retrieving email.	`ip`
`smtp`	Simple Mail Transfer Protocol for sending email.	`ip`
`snmp`	Simple Network Management Protocol for managing IP devices.	`ip`
`sox`	Sedona Framework UDP based communication protocol.	`ip`
`zigbee`	Low power wireless communication protocol for home automation.	`wireless`
`zwave`	Low power wireless communication protocol for home automation.	`wireless`

8.10.3 Binding a point to a network

`TelemetryBinding.networkRef` names the `bus:Network` over which a point is reached. The free-string `network` of v0.2 is deprecated. An importer **MUST** continue to accept it, per the deprecation rule of 15.5, and an exporter **SHOULD** write both while both are read. Where both are present, `networkRef` is the statement and the string is a label. A future MINOR version will set a sunset after which the free-string form **MAY** be omitted; until it does, dual-writing is what keeps a package readable by importers on either side of the transition.

`identifier` and `parameters` on a network are opaque to this standard for exactly the reasons `address` and `parameters` on a telemetry binding are (8.7). An importer **MUST** preserve them verbatim and **MUST NOT** normalize them. A BACnet network number that arrives as `2001` and leaves as `2001` is worth more than one a receiver has tidied into a form it prefers.

```
{ "id": "urn:uuid:...071",
  "type": "bus:Network",
  "name": "MS/TP trunk 2, mechanical room 2",
  "networkKind": "fieldbus",
  "protocol": "mstp",
  "identifier": "2001",
  "segment": "MSTP-2" }

"binding": {
  "protocol": "bacnet",
  "address": "analog-input,3",
  "deviceRef": "urn:uuid:...053",
  "networkRef": "urn:uuid:...071",
  "parameters": { "objectName": "AHU1_SAT", "covIncrement": 0.2 }
}
```

8.10.4 Driver configuration is out of scope

This version does not model driver configuration, and the omission is a decision rather than an oversight.

Poll rates, retry counts, COV subscription lifetimes, scan intervals, thread pools, license terms, a vendor's device template: these describe how a piece of integration software is set up. They fail the boundary test of 5.2. A building does not have a poll rate; a product does. Carrying them would mean this standard tracking the configuration surface of every driver any vendor ships, which ages at the speed of the fastest-moving product in the field and describes none of the building.

What BUS carries instead is **which network, which device, which address**. That is enough for an integrator to rediscover the driver, which is a day of work. It is not enough to rediscover the points, which is the multi-month exercise this standard exists to prevent. The line falls where the labor is irreplaceable.

> NOTE The gap table of 20.3 names integration and driver configuration as a row nothing serializes. v0.3 does not close that row and does not claim to. It closes the part of the row that describes the building and says plainly that the rest is not the standard's to hold.

Brick's ICT equipment classes were considered and declined. Brick 1.4.4 carries classes for the boxes — BACnet and Modbus controllers, gateways, routers, switches, access points — and carries no network entity and no network-membership predicate at all, so adopting them would import an equipment taxonomy without importing the thing that was missing. 5.3.3 refuses equipment taxonomy. A BACnet router is a `bus:Asset` with `assetKind: "device"` and a `brick:` classification, and it loses nothing by being one; Annex G states the classification rule.

9 Relationships

AGREED The predicate set grew from 27 tokens in v0.1 to 44: 21 inverse pairs, symmetric connectedTo, and the deliberately weak references. v0.3 adds the submeter pair; earlier additions are listed in Decision Record group C.

9.1 Relationships are entities

A relationship is an addressable entity with its own identifier, provenance, and validity period. It has a `predicate`, a `subject`, and an `object`, and **MAY** carry a `medium`, port qualifiers, and arbitrary `qualifiers`.

```
{ "id": "urn:uuid:...2014",
  "type": "bus:Relationship",
  "predicate": "bus:feeds",
  "subject": "urn:uuid:...051",         // AHU-1
  "object":  "urn:uuid:...052",         // VAV-2-01
  "medium": "air",
  "subjectPort": "supply", "objectPort": "inlet",
  "qualifiers": { "designFlow": { "value": 0.42, "unit": "m3/s" } } }
```

9.2 The normative predicate set

An implementation **MUST NOT** use a `bus:`-prefixed predicate outside this set. The set, the meanings, and the cardinality hints below are one source, `bus-predicates.json`; this table renders from it, and the build fails if the registry and the schema enumeration disagree. Through v0.3 this table was typed by hand beside the enumeration — the arrangement whose failure mode the front matter of this document describes.

PREDICATE	INVERSE	CARDINALITY	MEANING, SUBJECT TO OBJECT
`bus:contains`	`bus:containedIn`	many-to-many	Spatial or organizational containment. Transitive.
`bus:hasPart`	`bus:partOf`	one-to-many	Compositional part of an asset. Transitive. Equipment to component.
`bus:hasMember`	`bus:memberOf`	many-to-many	Grouping without containment. Not transitive. System to equipment, zone to space.
`bus:locatedIn`	`bus:hasLocation`	many-to-one	Physical placement of an asset in a space.
`bus:serves`	`bus:servedBy`	many-to-many	Delivery of a service to a space, zone, or asset.

PREDICATE	INVERSE	CARDINALITY	MEANING, SUBJECT TO OBJECT
`bus:feeds`	`bus:fedBy`	many-to-many	Directed flow of a medium. Requires `medium`.
`bus:connectedTo`	(symmetric)	many-to-many	Undirected physical connection, qualified by medium and port.
`bus:powers`	`bus:poweredBy`	many-to-many	Electrical supply. What an operator traces during an outage.
`bus:measures`	`bus:measuredBy`	many-to-one	Observation. Point to the thing observed.
`bus:commands`	`bus:commandedBy`	many-to-one	Actuation by a point. Point to the thing acted upon.
`bus:controls`	`bus:controlledBy`	many-to-many	Governance by a controller, sequence, or strategy.
`bus:hosts`	`bus:hostedBy`	one-to-many	Provision of an addressable home for a point. Device to point.
`bus:appliesTo`	`bus:hasApplicable`	many-to-many	Attachment of intent, an alarm definition, or a result to what it concerns.
`bus:implements`	`bus:implementedBy`	many-to-many	A sequence or schedule realizes an intent.
`bus:explains`	`bus:explainedBy`	many-to-many	Knowledge accounts for why something is as it is.
`bus:supports`	`bus:supportedBy`	many-to-many	Knowledge or evidence supports an assertion.
`bus:justifies`	`bus:justifiedBy`	many-to-many	A rationale justifies a decision or change.
`bus:derivedFrom`	`bus:derives`	many-to-many	Dependence of a calculated value on its inputs.
`bus:documentedBy`	`bus:documents`	many-to-many	Attachment of a resource to what it describes.
`bus:represents`	`bus:representedBy`	many-to-one	A drawing or element depicts a building entity.

PREDICATE	INVERSE	CARDINALITY	MEANING, SUBJECT TO OBJECT
`bus:replaces`	`bus:replacedBy`	one-to-one	Lifecycle succession.
`bus:hasSubMeter`	`bus:subMeterOf`	one-to-many	Metering subdivision. The object meters a share of what the subject meters. New in v0.3; energy allocation was inexpressible without it.
`bus:references`	(none)	many-to-many	A link that is real but whose nature is not modeled. Deliberately weak.

The cardinality column is ADVISORY, and the grading is deliberate. It states the ordinary shape of the relationship, subject to object, so that a consumer building an index — a property graph, an object store, a triple store — knows what to expect and what to flag, instead of guessing and guessing differently from the next consumer. A package exceeding a hint is NOT invalid and **MUST NOT** be refused: dual electrical feeds through a transfer switch, a check meter observing two parents, and an asset consolidation replacing two things with one are all real, all legal, and all recorded as notes in the registry. The building is the authority on its own topology.

> NOTE The inverse of a hint is its mirror — where `bus:hosts` is one-to-many, `bus:hostedBy` is many-to-one — and is derived from the registry, never stored in it.

Both directions of a pair are defined so an exporter can write whichever is natural. An implementation **MUST NOT** assert both directions of the same pair for the same subject and object; that is redundancy, and it diverges under editing. A receiving system **MAY** substitute the inverse on re-export, and test RT-3 accepts that.

9.3 Predicate semantics

- `bus:feeds` — **MUST** carry a `medium`. A flow relationship with no medium is not interpretable, and the medium is the property most often lost when a topology is flattened into a hierarchy.
- `bus:connectedTo` — is symmetric and **MUST NOT** be asserted twice for the same pair. A duct has a direction and a pipe joint does not; one predicate cannot carry both, which is why there are two.
- `bus:measures` **and** `bus:commands` — the subject **MUST** be a point. This is what separates a point from a value: a point is the thing that stands in a relationship to what it observes or acts upon.
- `bus:replaces` — both endpoints **MUST** remain in the model. Replacement does not delete.

Requirement 9.3-1

An implementation **MUST NOT** delete an entity in order to record that it was replaced, superseded, or removed from service. Deletion is reserved for the case where the entity should never have existed. Everything else is a lifecycle transition, and lifecycle transitions are the mechanism by which a building keeps its identity across thirty years of component turnover.

9.4 Ports

PROVISIONAL New in v0.3. This closes the gap v0.2 called the largest remaining one in the information model. The shape is taken from ASHRAE 223P rather than invented, and the encoding is deliberately not Haystack's.

Promotion criterion — an exporter from a system holding surveyed port structure round-trips ports, `pairedWith` and `mapsTo`; an exporter holding only topology omits ports and is accepted.

A port is a named place on an asset where a connection lands: the supply outlet of an air handler, the A side of a two-way valve, the return tapping of a coil. **The asset declares its ports. The relationship names them.**

Through v0.2 `subjectPort` and `objectPort` were free strings that referred to nothing. A receiver could read the word `supply` and could not check it, resolve it, or ask an asset what ports it has, so it was not possible to say which branch, which side of a valve, or that a coil has a supply and a return. That is the gap this clause closes.

9.4.1 A port is not an entity

5.3.2 makes relationships entities because a relationship carries provenance and a validity period, can be asserted by one party and denied by another, and needs to be ended without deleting either endpoint. **A port carries none of that.** It has no identity apart from the asset that has it, no lifetime apart from that asset, and nothing to say that the asset does not already say. It is a name for a place, and a name for a place is intrinsic to the thing that has the place.

The cost of the alternative is concrete. Relationships are already roughly a third of package size (5.3.2). A port entity for every inlet and outlet would add one entity per opening on every pump, valve, coil, terminal unit and heat exchanger in a plant, for a fact the asset can carry in one array. Entity inflation is not a theoretical objection in a format an owner has to store, transfer and verify for decades.

> NOTE The port declaration **MAY** carry its own `provenance` where the assertion needs attribution — a port structure surveyed in the field rather than taken from a submittal. That does not make a port addressable: nothing can reference one except by naming it on its asset.

9.4.2 What a port declares

PROPERTY	REQUIREMENT
`name`	**REQUIRED**. Unique among the ports of one asset, and case-sensitive. This is the string a relationship names.
`direction`	**REQUIRED**. `inlet`, `outlet`, `bidirectional`, from the point of view of the asset that declares the port.
`medium`	**REQUIRED**. What moves through the port, from the same vocabulary a relationship uses (9.3).
`label`	**RECOMMENDED**. What the port is called on the drawing or the nameplate.
`ordinal`	**OPTIONAL**. Distinguishes otherwise identical ports: a valve's A and B sides, an air handler's two return-air inlets.
`pairedWith`	**OPTIONAL**. Names the port on the same asset carrying the other half of one circuit — a coil's supply and its return.
`mapsTo`	**OPTIONAL**. Names an asset and a port on it that are the same physical opening as this one: an air handler's supply outlet is really its supply fan's outlet.
`provenance`	**OPTIONAL**. Clause 14.

Direction and medium are **REQUIRED** because a port that states neither is the free string again under another name. Direction is what makes a connection checkable — an outlet feeds an inlet — and medium is what makes it interpretable at all, for the same reason `bus:feeds` requires one.

```
"ports": [                                        // AHU-1
  { "name": "chwIn",   "direction": "inlet",  "medium": "chilledWater",
    "label": "CHW supply", "pairedWith": "chwOut" },
  { "name": "chwOut",  "direction": "outlet", "medium": "chilledWater",
    "label": "CHW return", "pairedWith": "chwIn" },
  { "name": "returnA", "direction": "inlet",  "medium": "air", "ordinal": 1 },
  { "name": "returnB", "direction": "inlet",  "medium": "air", "ordinal": 2 },
  { "name": "supply",  "direction": "outlet", "medium": "air",
    "mapsTo": { "assetRef": "urn:uuid:...058",   // the supply fan
                "port": "discharge" } } ]
```

`ordinal` is what Haystack's encoding cannot express and what 9.4 was written to demand: two ports of the same medium and direction on one asset, told apart. `pairedWith` is what turns two openings into one circuit, so that a receiver knows a coil's supply and return are the two ends of the same water path rather than two unrelated tappings. `mapsTo` is what keeps a containment hierarchy from lying: the air leaving an air handler leaves through the supply fan, and both statements are true at once.

9.4.3 Naming a port from a relationship

Requirement 9.4-1

A `subjectPort` or `objectPort` **MUST** match a declared `Port.name` on the asset at that end of the relationship. A port name matching no declared port is a package validity failure. Free-string port names, valid in v0.2, are not valid in this version.

Requirement 9.4-2

Where a relationship carries a `medium` and a port it names carries one, the two **MUST** agree. A `bus:feeds` whose medium is `air` naming a chilled-water port is a mis-wired export, and stating the medium in both places is what makes it detectable rather than plausible.

Requirement 9.4-3

An exporter that knows an asset's connection topology but not its port structure **MUST** omit `ports` rather than inventing them. A synthesized port is indistinguishable from a surveyed one, and no later evidence will separate them. Omission is declarable under 16.6; invention is not.

> NOTE An exporter that synthesizes port names or structure from a coarser topology assertion — expanding a source system's single output-substance fact into a concrete supply port, say — produces a package indistinguishable from one that was surveyed. That synthesis is the 5.3.5 failure in its most durable form, and it is reported as a failure of EX-8. Omission remains declarable; invention is not.

The third of those is the one implementers will be tempted by, because a plausible port structure is easy to generate and looks like better data. It is not better data. A chiller given exactly one chilled-water outlet because the source said `hasOutputSubstance` may have four, and the receiver has no way to know that the number came from a guess.

9.4.4 Prior art, and what was declined

- **ASHRAE 223P** — the factoring is taken from `s223:ConnectionPoint` and its three subclasses, which give inlet, outlet and bidirectional, together with the two relations 9.4 needed most: `s223:pairedConnectionPoint` becomes `pairedWith` and `s223:mapsTo` becomes `mapsTo`. 223P is the only model in this field that has thought the problem through, and it is not credited anywhere in v0.2 because v0.2 had nothing to credit it for.
- **223P's `Connection` entity** — declined. In 223P the duct, pipe or conductor is itself an entity with its own class, which is right for a physics-grade model and wrong here: it multiplies entity count across an entire hydronic plant to carry a fact that `bus:feeds`

with a medium and two ports already carries. A duct **MAY** be a `bus:Asset` with `assetKind: "component"` where it matters. It is never required.

- **Brick 1.4.4** — has no port model at all. This was verified against the shipped ontology rather than assumed: there is no `brick:hasPort`, no `brick:Port` and no `brick:Connection`. `brick:connectedTo` exists as a bare symmetric predicate with no domain, no range, no direction and no medium, and the only Brick terms with Port in the name are 2 equipment classes. An import from a plain Brick model **MUST** leave `ports` absent (Annex G).
- **Project Haystack 4.0.0** — encodes a port as a supertype: 28 conjuncts of the form *substance*-input and *substance*-output, which 34 defs declare in their own `is` list. The input/output factoring is right and the encoding is why BUS did not follow it. Because the port is a supertype, an asset can declare at most **one port per medium per direction**, so a valve's A and B sides on the same medium, an air handler's second return-air inlet, and a coil's supply and return are all inexpressible — which is exactly the list of things 9.4 exists to say. Declaring ports on the asset costs one array and says all three.

9.5 Extension predicates

An implementation **MAY** define additional predicates in a declared namespace, and **SHOULD** do so rather than overloading `bus:references`. An importer **MUST** preserve relationships whose predicate it does not recognize.

10 Operational intent

AGREED Intent is decomposed rather than composite, and Sequence sits outside it. Both differ from the earlier technical specification; see Decision Record D-1 and D-2.

10.1 Why intent is modeled separately

A building has three tenses. What it is meant to do. What it is doing. What it did. Systems routinely store the first inside the third — an intent becomes a setpoint value in a database, indistinguishable from any other stored number — and when the system is replaced, the intent is what does not survive.

Modeling intent separately is what lets a receiving system answer the question an operator asks in the first week: not what is this value, but what was this supposed to achieve.

10.2 The intent classes

CLASS	WHAT IT CARRIES
`bus:Objective`	What is to be achieved, its rationale, a category, and a priority. Prose, deliberately.

CLASS	WHAT IT CARRIES
`bus:Target`	A quantified expectation attached to a point or quantity kind: a value or range, a tolerance, a condition, and a schedule reference.
`bus:Constraint`	A limit that holds regardless of objectives, with a `constraintKind` of safety, limit, deadband, interlock, regulatory, warranty, or operational.
`bus:Schedule`	When intent applies. Segments carry an [RFC5545] recurrence rule, local clock bounds, and a value. Exceptions carry explicit dates.

Each is separately addressable, which is what makes it possible to say twenty years later that *this specific target* was relaxed on a date, by a person, for a reason. A composite intent object is easier to author and blurs exactly that audit trail; authoring convenience is a tooling problem and addressability is not.

`bus:Objective` is prose because objectives are not reducible without loss. The rationale field carries what a later engineer most needs and most rarely gets: why the number is what it is.

10.3 Sequences

`bus:Sequence` is the logic that implements an intent, connected by `bus:implements`. It is not part of intent. A sequence names the `language` it is written in — `prose`, `ashrae-231-cdl`, `ashrae-guideline-36`, `iec-61131-3`, `iec-61499`, `sedona`, `niagara-wiresheet`, `function-block`, `vendor` — and carries either inline source or a resource reference. The vocabulary is open per 7.5: an unlisted token is valid and **MUST** be preserved.

> NOTE ANSI/ASHRAE Standard 231-2026, Control Description Language, published February 2026, is the first citable machine-readable form for control sequences. It is design-time: it describes what logic should be, not what a deployed controller is running. BUS-1 references sequences rather than extracting them for that reason, and extraction remains open (Annex E).

10.4 Constraints and safety

Requirement 10.4-1

An importer **MUST** preserve every constraint and **MUST NOT** relax, widen, or discard a constraint whose expression it cannot evaluate. Where a constraint cannot be enforced by the receiving system, the importer **MUST** report it as unenforced rather than treating it as satisfied. A safety limit that survives migration as an uninterpreted note is recoverable; one that survives as an assumption is not.

A constraint **MAY** carry a machine-readable `expression`, which **MUST** name its `language`. This standard does not define an expression language and does not require a receiver to evaluate one. Naming the language is what lets a receiver decide honestly whether it can.

Constraints are also referenced from write bindings (13.7), so that a limit enforced at the graphic is the same limit recorded in the model rather than a second copy that will diverge.

10.5 Design intent and deployed logic

PROVISIONAL New in v0.3. The envelope is specified. Extraction is not, and Annex F.1 keeps it as an open gap.

A sequence in a package is one of two very different things, and through v0.2 there was no way to say which. It is either the sequence of operation from the submittal — what the building was specified to do — or it is what the controller is actually running.

Those two diverge, and they diverge in the direction nobody documents. An override becomes permanent. A loop is retuned during a hot week. A vendor patches a program on site at two in the morning and writes nothing down. Ten years later the submittal is a description of a building that no longer exists, and it is indistinguishable in a package from a description of one that does.

Which of the two a sequence is, is the single most valuable fact about it in a migration, and it was inexpressible.

10.5.1 The deployment block

`bus:Sequence` carries an **OPTIONAL** `deployment` block. Where it is present, `state` is **REQUIRED**.

FIELD	REQUIREMENT
`state`	**REQUIRED**. `asDesigned` — from the specification, submittal, or design intent. `asDeployed` — extracted from the controller that runs it. `unknown` — the exporter cannot tell.
`controllerRef`	**RECOMMENDED** where the state is `asDeployed`. The `bus:Asset` that executes the program.
`programIdentifier`	**RECOMMENDED** where the state is `asDeployed`. How the controller names the program. Opaque, and preserved verbatim.
`extractedAt`, `extractedBy`	**RECOMMENDED** where the state is `asDeployed`. When the logic was taken off the controller, and by whom, as an agent per 14.1.
`checksum`	**OPTIONAL**. A digest of the program as extracted, so that a later export can be compared against this one without the controller being available.

FIELD	REQUIREMENT

```
"deployment": {
  "state": "asDeployed",
  "controllerRef": "urn:uuid:...053",
  "programIdentifier": "AHU1_ECON_v7",
  "extractedAt": "2026-06-18T14:02:00-05:00[America/Chicago]",
  "extractedBy": { "kind": "system", "name": "busref 0.3.0" },
  "checksum": { "algorithm": "sha-256", "value": "9f2c..." }
}
```

Requirement 10.5-1

An exporter that cannot determine whether a sequence is design intent or running code **MUST** declare `state` as `unknown`. It **MUST NOT** declare `asDesigned` by default. `unknown` is an honest answer and is the correct one; `asDesigned` is an assertion about where the sequence came from, and asserting it in ignorance is the exact failure this block exists to make impossible.

Requirement 10.5-2

An exporter whose system holds deployed control logic and which exports only design-time sequences **MUST** declare the `intent` domain as partial with reason `notSupportedByExporter` (16.6).

Without the second of those, an exporter can be technically conforming and materially false. It ships the submittal's control description, declares intent complete, and the receiving system is told the building runs logic it stopped running in 2019. Nothing in the package is untrue on its face and the whole of it is misleading. This is the same failure 7.8 was written to close for properties, appearing again one clause over.

10.5.2 What this version does not do

v0.3 does NOT define control-program extraction, and it does NOT define a control language.

There is no prior art to follow. Project Haystack 4.0.0 has no sequence, program, or control-logic construct anywhere in its 719 defs; Brick 1.4.4 has none in its 1,433 classes. The nearest constructs in either model describe what produced a value, not what logic runs the plant. ANSI/ASHRAE 231-2026 is design-time by construction — it describes what logic should be — and IEC 61131-3 is a programming standard, not an exchange format for a program already running in a controller. Annex F.1 is right that extraction has no precedent in any standard, and a specification that invented one here would be guessing in public and asking implementers to pay for the guess.

What v0.3 does instead is make the gap visible and measurable. A package can now say that its sequences are design intent, or that nobody knows, or that a named agent read a named controller on a date and here is the digest of what came back. An owner can count how many of their sequences are `asDeployed`. That number is unknowable today, and it is the number that decides whether a migration is a transcription or a re-engineering.

> NOTE The `language` vocabulary was extended in the same pass, with the formats a deployed program is actually held in rather than the formats a design is written in. Naming the format is not reading it; it is what lets a receiver decide honestly whether it can. The full token list is in 10.3.

10.5.3 Explicit non-goal and opaque attachment

BUS-1 does not define extraction of proprietary controller programs, and does not define a control language. A sequence carrying state `asDeployed` asserts exactly this much: an agent read a named controller on a named date and obtained a program whose checksum is recorded. It does not assert that any receiving system can execute, interpret, or diff that program, and a reader who takes the deployment block as a claim that extraction is solved has been misled by nothing in this document.

An exporter **MAY** attach the opaque program bytes, or a resource reference to them, under the existing resource mechanism — together with the `language` token 10.3 already requires and the checksum in the deployment block. The attachment is a byte-preserving convenience for the integrator who must re-commission; it is not a portable semantic representation of control logic. A receiver that cannot interpret the language token **MUST** still preserve the resource and the deployment metadata, which is 2.4 applied and not a new obligation.

This subsection does not close the extraction gap recorded in Annex F.1. It fixes the boundary of what is claimed, and provides a non-destructive place for later standards or vendor tools to land extracted artifacts without breaking the reconstruction guarantee.

Promotion criterion — packages from at least two controller platforms emit `asDeployed` with checksum, and the distinction survives a third-party import.

11 Alarms

AGREED New in v0.2. v0.1 represented alarms as a point kind and a boolean flag, which carried almost none of the configuration.

11.1 Alarm definitions are intent

An alarm definition states a condition under which the building is to draw attention to itself, and what is to happen then. Limits, delays, priorities, routing, and acknowledgment policy are engineered decisions about how the building should behave, which places them in intent rather than in configuration.

`bus:AlarmDefinition` is attached to its subject by `bus:appliesTo`.

11.2 Content

FIELD	REQUIREMENT
`alarmKind`	**REQUIRED**. `limit`, `deviation`, `rateOfChange`, `status`, `digital`, `calculated`, or `communication`.
`severity`, `priority`	**RECOMMENDED**. `priority` is the source system value, preserved verbatim, not normalized to a BUS scale.
`limits`	For limit and deviation alarms: `highHigh`, `high`, `low`, `lowLow`, `deviation`, and a `referenceRef` for deviation from a setpoint.
`triggerStates`	For status and digital alarms: the enumeration members that raise the alarm.
`deadband`, `onDelay`, `offDelay`	**RECOMMENDED**. The timing that separates a real alarm from a nuisance one, and the most common cause of a migrated system flooding.
`requiresAcknowledgement`	**RECOMMENDED**. Operator-safety policy, not interface behavior.
`shelvable`, `maxShelveDuration`	**OPTIONAL**. Whether an operator may suppress the alarm and for how long.
`enabledByScheduleRef`	**OPTIONAL**. Alarms that apply only during occupancy.
`routing`	**RECOMMENDED**. Ordered destinations with channel, party, schedule, and escalation interval.
`message`	**RECOMMENDED**. What the operator is told.

11.3 Preservation

Requirement 11.3-1

An importer **MUST** preserve every field of an alarm definition, including routing it cannot execute and priorities it cannot represent natively. Where it cannot enforce a field, it **MUST** report that field as unenforced. Silently dropping routing or renormalizing priority is a conformance failure (test RT-22).

> NOTE Priority renormalization is the specific failure to guard against. A source priority of 120 on a 0-to-255 scale mapped onto a receiver's 1-to-5 scale is not recoverable, and the resulting alarm order is subtly wrong in a way nobody notices until an incident.

12 Events

AGREED New in v0.2. An event is not a time-series sample and does not belong in the timeSeries domain, which BUS-2 specifies.

12.1 What an event is

A `bus:Event` is a timestamped occurrence attached to a subject: an alarm raised, cleared, acknowledged, or shelved; an override applied or released; a mode change; a command; a setpoint change; a service action; a commissioning activity; a fault; a configuration change; an outage.

The subject is intrinsic. An event without a subject is meaningless, so unlike ordinary contextual facts it is carried on the entity rather than as a relationship (5.3.1).

12.2 Why events are core

Folding events into time-series history and deferring both would mean the core cannot express the sentence that matters most in a migration: *someone overrode this in 2019 and never took it out*. That fact appears on no drawing, in no database field, and in no tag. It is precisely the class of understanding this standard exists to preserve, and it is cheap to carry because events are sparse where trend data is dense.

```
{ "type": "bus:Event",
  "eventKind": "override",
  "subject": "urn:uuid:...067",                    // VAV-2-01 damper command
  "at": "2017-05-09T11:20:00-05:00[America/Chicago]",
  "by": { "kind": "person", "name": "D. Okafor", "partyRef": "urn:uuid:...004" },
  "priorValue": 18.0, "value": 24.0, "priority": 8,
  "message": "Minimum damper command raised to 24 percent to clear the binding band.
              Override left in place deliberately; see KN-AHU1-OAD." }
```

12.3 Completeness

An event log is rarely complete and completeness is not required. An exporter carrying a subset **MUST** declare the `events` domain as partial with a reason (16.6). A partial event history is materially better than none, and a standard demanding completeness here would be widely ignored.

An importer **MUST** preserve an event's subject reference. An override event that loses its subject is a failure (test RT-23), because it becomes an anecdote rather than a fact about a point.

13 Portable presentation

AGREED The layering, the role-and-library mechanism, and the binding model are settled. The neutral vector format used by figure mode is specified in v0.3 as a closed profile of SVG; see 13.4.3.

Owners pay substantial sums to create floor plans, equipment graphics, point bindings, and navigation structures. Those assets describe the building and embody real

engineering labor. At present they are rebuilt at every migration, and the rebuild is among the largest single costs an owner incurs in changing platforms.

The evidence that this is structural rather than incidental is set out in 20.4. In short: no standard anywhere defines operator graphics with live bindings; the one vendor format published as a specification keeps its bindings in a database rather than in the file; vendors' own conversion utilities drop dynamics and require manual rebinding; and the third-party graphics industry is organized by platform with no conversion service between them.

13.1 The four layers of a graphic

The reason this has never been standardized is that a graphic is treated as one thing. It is four, and the ownership line falls cleanly between the third and the fourth.

LAYER	WHAT IT IS	WHOSE
Geometry and structure	Coordinate system, space boundaries, linework, placement, rotation, layers, grouping, z-order.	Owner. Survey and drafting labor.
Semantic element identity	That the thing at a location is a supply fan, not that it is a blue rectangle with an arc.	Owner.
Bindings	Which element is driven by which entity, through which channel; command actions; alarm annunciation; behavior on stale data.	Owner. The largest engineering cost.
Appearance	Artwork, color, animation curves, transitions, hover behavior, typography, responsive layout, interaction model.	Vendor. Entirely.

The boundary test, applied to graphics

The vendor keeps everything a user sees and feels. The owner keeps everything a draftsman or engineer decided.

This standard specifies the first three layers and says nothing whatever about the fourth. A conforming receiving system is free to render a package in a visual language entirely unlike the one it came from, and remains conforming.

13.2 Drawings

A `bus:Drawing` is a page: a floor plan, site plan, equipment graphic, schematic, one-line, riser, detail, or dashboard. It carries a coordinate system, an extent, an **OPTIONAL** nominal scale and reference dimension, and an **OPTIONAL** `subject` naming the space or asset it is of.

A drawing **MUST** contain at least one element. Layers and elements name their drawing; an element's layer **MUST** belong to the same drawing as the element (test PK-14).

13.2.1 Layers

A `bus:Layer` carries a declared `purpose`: background, architecture, structure, spaces, equipment, ductwork, piping, electrical, controls, annotation, or navigation.

Purpose is declared rather than left to naming convention so that a receiving system can drop the architectural context and keep the semantic content — which is what a system that already has its own base plan needs to do, and what a system rebuilding a plan from scratch cannot do at all today.

13.3 Coordinate systems and georeferencing

A drawing **MUST** declare its coordinate system: the length unit code its coordinates are expressed in, the direction of increasing y, and an **OPTIONAL** origin.

A drawing of physical space **SHOULD** carry a **georeference**: a latitude and longitude, the drawing coordinate they correspond to, a rotation from true north, an **OPTIONAL** elevation, and an **OPTIONAL** coordinate reference system identifier per [ISO19111].

```
"coordinateSystem": {
  "unitCode": "m", "yAxis": "up", "origin": { "x": 0.0, "y": 0.0 },
  "georeference": {
    "latitude": 30.267105, "longitude": -97.723940,
    "atX": 0.0, "atY": 0.0, "rotation": 14.5,
    "elevation": { "value": 4.2, "unit": "m" }, "crs": "EPSG:4326" }
}
```

Georeferencing matters more than it appears to. An anchored plan survives into any CAD, BIM, or GIS tool indefinitely, and two plans produced by different vendors at different times will register against each other without manual alignment. An unanchored plan is a picture whose relationship to the world has to be re-established by hand every time it is reused.

New in this version, and additive: a georeference **MAY** also carry a precomputed **`affine`** — six numbers mapping drawing coordinates directly into the named CRS. The anchor form above, plus the drawing unit, already places a true-scale plan completely; the affine exists for the two cases the anchor form cannot express — the scale error of a scanned or traced plan, and shear — and for consumers that apply a transform rather than derive one, which is every GIS and every object store. Where both forms are present they **MUST** describe the same placement within the RT-18 tolerance, and a transform whose linear part is singular is a package error (test PK-27): two permitted encodings of one fact are tolerable only while the package proves they are the same fact.

A drawing **SHOULD** also carry a `referenceDimension`: a known real-world distance between two drawing coordinates. This makes a scale error detectable rather than silent, which matters because a plan imported at the wrong scale looks entirely correct until someone measures something (test PK-17).

13.4 Elements and symbol roles

A `bus:Element` is a placed thing. It names its drawing and layer, and carries a transform, an **OPTIONAL** z-order and group, an **OPTIONAL** label, and an **OPTIONAL** `represents` naming the building entity it depicts.

An element **MUST** carry at least one of a `role`, a `figure`, or a `geometry`.

13.4.1 Role mode

In role mode the package states that at a location, rotated, on a layer, there is an element of a declared symbol role. **The receiving system supplies the artwork.**

```
{ "type": "bus:Element",
  "localId": "AHU-SF",
  "drawing": "urn:uuid:...602", "layer": "urn:uuid:...617",
  "role": "fan.centrifugal.supply",
  "represents": "urn:uuid:...051",
  "transform": { "x": 1120, "y": 380, "rotation": 0 },
  "zOrder": 5 }
```

This is what preserves the owner's survey, placement, and bindings in full while leaving the vendor's design system entirely untouched. Vendor A draws a flat icon; Vendor B draws a rendered one; both are conforming, and neither has been asked to adopt the other's visual language.

> NOTE The mechanism is not speculative. IEC 61970-453, the CIM Diagram Layout Profile, exchanges power-system diagram layout with links to modeled objects and explicitly leaves the symbol library to the receiver. It has been in production use for over a decade.

13.4.2 Role fallback

A symbol role is a hierarchical dotted path. Roles degrade by truncation: `fan.centrifugal.supply` to `fan.centrifugal` to `fan`.

Requirement 13.4-1

An importer encountering a symbol role it does not implement **MUST** truncate the role at successive dots and render at the deepest ancestor it does implement. It **MUST NOT** refuse the package or drop the element. It **MUST** re-export the original leaf role unchanged; substituting the ancestor into the exported data is a conformance failure (tests IM-6 and RT-19).

This is what allows the vocabulary to grow indefinitely without breaking deployed importers, and it is the same property that abstract types give the entity model (8.2). Annex B defines the first tier: 52 roots with 75 named leaves.

13.4.3 Figure mode and `bus-vector-1`

PROVISIONAL New in v0.3. v0.2 named the format and did not define it. The profile below is defined; the element list is where review is most wanted.

Promotion criterion — a graphics-bearing package using figure mode and dictionaries passes the full RT suite between two independent implementations.

Sometimes the owner paid for the artwork itself: a bespoke plant schematic, a one-line an engineer drew, a symbol no vocabulary contains. In figure mode an element carries a neutral vector payload, inline or as a resource, that a receiver **MUST** render substantially as given rather than replacing it with a library symbol (test RT-20).

An element **MAY** carry both a role and a figure. Where it does, the role is a hint for receivers that prefer their own library and the figure is the fallback. An exporter **SHOULD** prefer role mode where a role fits, because role mode is what lets the receiving vendor apply its own design system.

`bus-vector-1` is **a closed profile of SVG 1.1 Second Edition, Static processing mode**. It is not a new format, and a new format was rejected: there is no second implementer to share the cost of one, no tooling for one, and no prior art to draw on — neither Project Haystack nor Brick has any vector, drawing, symbol or figure construct of any kind. Every plausible receiver already has an SVG parser. What this version adds is the constraint list that makes the security requirement of 19.3 true by construction rather than by vigilance.

Naming the format and not defining it was worse than either alternative. A package using figure mode was portable only between implementations that had privately agreed what the payload was, which is to say between none; and 19.3's requirement that a receiver not execute script, resolve external references, or fetch remote content had no format to attach itself to.

A figure **MUST** use only these 15 elements: `svg`, `g`, `defs`, `title`, `desc`, `use`, `path`, `rect`, `circle`, `ellipse`, `line`, `polyline`, `polygon`, `text`, `tspan`.

The 16 elements below are forbidden. The reasons are not stylistic and are worth reading, because each one is a rule this document states elsewhere arriving in a new place.

ELEMENT	WHY IT IS EXCLUDED
`script`	19.3 — a figure **MUST NOT** execute.
`foreignObject`	19.3 — escapes the profile entirely.
`image`	Raster content is not vector geometry and is usually an external reference.
`a`	Navigation is the receiver's, not the figure's.
`style`	A receiver must be able to strip appearance and keep shape without parsing a stylesheet.
`switch`	Conditional rendering makes 'render substantially as given' untestable.

ELEMENT	WHY IT IS EXCLUDED
`filter`	Appearance, and expensive; 13.9 puts appearance on the vendor's side.
`animate`	A figure is a drawing, not a behavior.
`animateTransform`	As animate.
`animateMotion`	As animate.
`set`	As animate.
`marker`	Renders differently across parsers; use explicit geometry.
`pattern`	Appearance, and a common vehicle for external references.
`mask`	Appearance; changes geometry a receiver reads.
`clipPath`	Appearance; hides geometry that is nonetheless present.
`symbol`	Overlaps bus:SymbolRole in name and in function, and confuses the two mechanisms.

`script` and `foreignObject` are excluded because 19.3 forbids a figure from executing anything or escaping into a document; a neutral vector payload is data, not a program. `style` is excluded because a receiver must be able to strip appearance and keep shape without parsing a stylesheet, which is the whole basis on which a figure is owner property under 13.1. `filter`, `mask` and `clipPath` claim to be appearance and change the geometry a receiver reads, so a receiver that stripped appearance would silently change the drawing.

ID	GRADE	CONSTRAINT
`VEC-1`	**MUST**	The root element is svg and carries a viewBox. It **MUST NOT** carry width or height in absolute units: a figure composes with the drawing's coordinate system (13.3) and does not assert a physical size of its own.
`VEC-2`	**MUST**	Every element is drawn from the permitted list. Any other element, in any namespace, makes the figure invalid.
`VEC-3`	**MUST**	No attribute contains an href, xlink:href or url() reference to anything other than a fragment within the same figure. No external entity, no DOCTYPE, no @import.
`VEC-4`	**MUST**	Appearance is carried by presentation attributes only. No style attribute, no style element, no CSS of any kind.

ID	GRADE	CONSTRAINT
VEC-5	**MUST**	Element nesting **MUST NOT** exceed 16 levels and a figure **MUST NOT** contain more than 4096 elements. This gives 19.3's requirement that a receiver bound the size of an inline figure payload a number to bound against.
VEC-6	**SHOULD**	Text carries font-family and font-size where they matter to legibility. A receiver **MAY** substitute any font: fonts are the vendor's under 13.9.
VEC-7	**MUST**	The document is well-formed XML in UTF-8, with no processing instructions other than the XML declaration.

VEC-5 is what gives 19.3's requirement that a receiver bound the size of an inline figure payload a number to bound against: 16 levels of nesting and 4096 elements. A constraint graded **SHOULD** is a recommendation and nothing more: a figure that does not satisfy one is a valid figure, and an importer **MUST NOT** refuse it.

v0.2 named two format tokens, `bus-vector-1` and `svg-profile-1`, for one thing that neither of them defined. Two undefined names for one undefined format is worse than one, so they are collapsed. `bus-vector-1` is the name of the profile. A receiver **MUST** accept `svg-profile-1` as an alias for one MINOR version; an exporter **MUST NOT** write it.

Requirement 13.4-2

A receiver renders a figure **substantially as given** when geometry is preserved within the tolerance of test RT-18 — one part in ten thousand of the drawing extent — and topology, meaning element count and nesting, is preserved. Colors, strokes, fills, line weights and fonts are NOT guaranteed, and a receiver **MAY** replace them wholesale.

That definition is load-bearing, and it is why this clause needed a version of its own. Read loosely, 13.4.3 and 13.9 contradict each other: one requires a receiver to render a figure faithfully, the other says appearance belongs entirely to the vendor and that no conforming package carries a color at all. An implementer reading both in good faith could not tell which obligation won.

A figure is portable shape, not portable artwork. Both clauses stand once *substantially* means shape. The owner who paid a draftsman for a plant schematic gets the schematic back, with its geometry and its structure intact. They do not get the receiving vendor's palette overruled, and they were never entitled to.

13.5 Geometry

An element **MAY** carry `geometry`: a point, polyline, polygon, rectangle, circle, arc, or path, expressed as a flat array of alternating x and y values in drawing units, with **OPTIONAL** holes for polygons.

The single most valuable use of geometry is a **space boundary polygon carrying the identifier of the space it bounds**. That is what turns a plan from a picture into a view of

the model: every downstream capability — coloring a floor plan by temperature, locating an alarm, computing an area, finding which zone a complaint came from — depends on it, and none of it is possible from a drawing file.

```
{ "type": "bus:Element", "localId": "L2-SP-201",
  "layer": "<spaces layer>", "represents": "<Open Office 201>",
  "geometry": { "kind": "polygon", "closed": true,
                "coordinates": [2, 18, 34, 18, 34, 32, 2, 32] } }
```

Geometry fidelity is tested to one part in ten thousand of the drawing extent (test RT-18), which is tighter than any plausible rendering difference and looser than floating-point representation noise.

13.6 Bindings

A `bus:Binding` declares that an element is driven by, or writes to, a building entity. Both endpoints are intrinsic to it, exactly as a relationship names its subject and object, and no separate relationship is created.

> NOTE This is a deliberate departure from 5.3.2. A binding carries a value-mapping table, which is more structure than relationship qualifiers should hold; and naming endpoints intrinsically avoids generating eight thousand relationship entities for a plan with four thousand bindings. Making it an entity rather than a property means bindings carry provenance and revision, so the fact that an element was repointed in 2019 becomes visible — bindings drift constantly and at present nobody can prove it.

13.6.1 Channels

A binding names a `channel`: the aspect of the element the value drives. The vocabulary is closed and may be extended only by a MINOR version.

CHANNEL	DRIVES
`value`	A displayed numeric or text value.
`state`	Selection among named element states.
`visibility`	Whether the element is shown.
`fill`, `stroke`	Selection among named fill or stroke states. Not a color: the receiver chooses the color.
`rotation`, `position`, `scale`	Geometric transformation, for dampers, valves, sliders, and level indicators.
`opacity`	Transparency.

CHANNEL	DRIVES
`label`	Displayed text distinct from the value.
`alarm`	Annunciation of an alarm definition on the element.
`enable`	Whether an interactive element accepts input.

A channel names an intent, never an appearance. `fill` selects among named states; it does not carry `#FF0000`. This is what keeps the binding on the owner's side of the boundary and the rendering on the vendor's.

13.6.2 Mapping

A binding **MAY** carry a `mapping` translating source values into element states: discrete `states` for enumerated targets, numeric `ranges`, and a `scale`, `offset`, `format`, and `displayUnitCode`. A `displayUnitCode` is a code from the Annex C vocabulary, exactly as `unitCode` is; a receiver formats the value, it does not invent a unit notation.

```
"mapping": { "ranges": [
  { "max": 20.0, "select": "cold", "label": "Below band" },
  { "min": 20.0, "max": 24.5, "select": "ok", "label": "In band" },
  { "min": 24.5, "select": "hot", "label": "Above band" } ] }
```

Where the target is an enumerated point, every `when` value in a state mapping **SHOULD** correspond to a member of the target's enumeration. A mapping referring to a state the point cannot take is a modeling error and **SHOULD** be reported by a validator.

13.6.3 Quality behavior

A binding **SHOULD** declare what the element does when its value is stale, unavailable, or out of service: a `staleAfter` interval and an action of indicate, gray, hide, hold, show a question mark, or show the last good value.

> **Requirement 13.6-1**
>
> Quality behavior is engineered behavior and **MUST** be preserved (test RT-17). It is not styling and **MUST NOT** be treated as vendor territory.

This is the field implementers are most likely to consider cosmetic and is not. A graphic that displays the last good value indefinitely, rather than graying out, tells an operator that a dead sensor is reading normally. That is a safety failure, and the decision about which behavior applies was made by an engineer who is unlikely to still be available.

13.7 Actuation

AGREED Command actions travel. The consequence is a materially sharper security posture; see 19.2.

A binding whose `direction` is `write` or `readWrite` carries a `write` block. The ability to act on plant from a graphic is engineered, owner-funded behavior and meets the boundary test; excluding it would mean every migration rebuilds the command layer by hand, which is the most safety-critical part to rebuild.

```
"direction": "readWrite",
"write": {
  "priority": 10,
  "requiresConfirmation": true,
  "confirmationText": "Change the AHU-1 supply air temperature setpoint?",
  "constraintRef": "urn:uuid:...203",       // the safety low limit
  "releaseOnExit": false
}
```

13.7.1 Priority

Requirement 13.7-1

A write binding **MUST** declare the command priority at which it writes, preserved verbatim from the source system. A receiving system that cannot honor the source priority model **MUST** report the binding as unenforced rather than substituting its own priority (tests RT-16 and IM-7).

If a source binding writes at priority 8 and a receiver writes at priority 16, the graphic looks identical and the plant behaves differently. Nobody discovers this until an override fails to hold, typically during an incident. This is the presentation-layer analog of the unit-conversion rule of 7.4.

A declared priority **MUST** be within the target point's declared priority model where one exists (test PK-16).

13.7.2 Limits and interlocks

A write binding that restricts the permitted value range **MUST** reference a `bus:Constraint` through `constraintRef` rather than carrying its own minimum and maximum. Two places to state the same limit guarantees they diverge, and the graphic is the copy that gets edited without the model being updated.

An `interlockRef` names a constraint that must hold before the write is permitted — a fan that may not start while a fire alarm is active, for instance. Interlocks that the receiver cannot enforce **MUST** be reported, not assumed satisfied (10.4).

13.7.3 Confirmation

`requiresConfirmation` records that an operator must confirm before the write is issued. The requirement travels; the design of the confirmation dialog does not. The same boundary test that gives the vendor the dialog gives the owner the requirement, because

the decision that this particular action needs a second thought was an operator-safety judgment.

13.8 Navigation

An element **MAY** carry `navigatesTo` naming another drawing. This preserves the structure by which an operator moves from a floor plan to an equipment graphic — which is engineered information architecture, not menu design.

How navigation is presented, whether as a click, a hover, a breadcrumb, or a search result, is entirely the receiving vendor's.

13.9 What presentation does not carry

For the avoidance of doubt, nothing in this clause constrains, and no conforming package carries:

- Colors, gradients, shadows, textures, or any other appearance value.
- Symbol artwork, except in figure mode where the owner funded the artwork itself.
- Animation curves, easing, timing, or transitions.
- Fonts, type sizes, or text styling.
- Hover, focus, selection, or gesture behavior.
- Responsive breakpoints, layout adaptation, or device-specific presentation.
- Menu structure, search, personalization, or role-based presentation.

A receiving system that renders a package in an entirely different visual idiom, with different symbols, different colors, and different interaction, is fully conforming. That is the intended outcome, not a tolerated one.

14 Provenance and governance

AGREED

14.1 The provenance block

FIELD	REQUIREMENT
`assertedBy`	**REQUIRED**. An agent with a kind of person, organization, system, or device, and a `partyRef` where the party is addressable.
`assertedAt`	**REQUIRED**. Timestamp per 7.3.
`method`	**REQUIRED**. `measured`, `manual`, `commissioned`, `imported`, `derived`, `inferred`, `asserted`, `surveyed`, or `unknown`.
`source`	**RECOMMENDED**. A document, system, person, survey, device, drawing, or prior package.

FIELD	REQUIREMENT
`confidence`	**OPTIONAL**. 0 to 1. Absence means unstated, NOT certain.
`approvals`	**OPTIONAL**. Who signed off, when, in what role.
`changeReason`, `supersedes`	**OPTIONAL**. Why the current assertion replaced its predecessor, and which one.

`method` is required because it is the field that ages best. Thirty years on, knowing that a value was measured at commissioning rather than typed in from a drawing is often the difference between trusting it and re-surveying.

14.2 Change records

A `bus:ChangeRecord` states that something changed: its subject, an operation, the time, the agent, a reason, and the revision it moved to. An exporter holding change history and omitting it **MUST** declare the `provenance` domain as partial.

> NOTE Change records are not an audit log and need not be complete or contiguous. A partial change history is materially better than none, and a standard demanding completeness here would be ignored.

14.3 Provenance is not overwritten

Requirement 14.3-1

An importer **MAY** add its own provenance to any assertion it receives. It **MUST NOT** replace, rewrite, or discard the provenance that arrived with it. An import is an event in a building's history, not a reset of it.

A package that has passed through three systems in twenty years carries three layers of provenance, and an operator can see which assertions have been touched by whom. That accumulation is the point.

15 Extensions

AGREED

15.1 Purpose

The extension mechanism exists so that innovation does not wait for a version of this document, and so that a vendor is never forced to choose between conformance and a capability. It is also the mechanism most likely to be abused, and this clause is correspondingly strict.

15.2 Namespaces

Extension data is carried in an `extensions` map whose keys are absolute URIs or reverse-DNS tokens. The prefix `bus` is reserved and **MUST NOT** be redefined.

15.3 Declaration

Every namespace used anywhere in a package **MUST** be declared in the manifest with a prefix, a namespace URI, a must-understand flag, and **SHOULD** carry a description and specification URI. An undeclared namespace is a package validity failure (test PK-9). The rule exists so an owner can read one file and see every proprietary construct their building has accumulated.

15.4 Must-understand and preservation

Requirement 15.4-1

An importer **MUST** preserve every extension it receives, verbatim, whether or not it understands the namespace, and **MUST** re-emit it on export with its declaration intact. Preservation is unconditional and does not depend on the must-understand flag.

Requirement 15.4-2

Where an extension is declared with `mustUnderstand` true and the importer cannot interpret it, the importer **MUST NOT** report the import as successful. It **MUST** report the import as incomplete and name the namespace. It **MUST** still preserve the data.

`mustUnderstand` **SHOULD** be set only where acting on the model without interpreting the extension would be wrong — a safety interlock expressed in a proprietary form, for example. Setting it routinely, so that no competitor can claim a clean import, is an abuse of the mechanism and governance should treat it as a conformance matter rather than a technical one.

15.5 Promotion

An extension implemented by several independent parties, that expresses something about the building rather than about a product, is a candidate for promotion into the core in a later MINOR version. On promotion the core construct becomes normative and the extension namespace is deprecated but **MUST** continue to be accepted.

The same mechanism applies to property definitions (7.8.1), and is the route by which shared vocabulary is expected to emerge without a committee defining one in advance.

16 The exchange package

AGREED Dictionaries (16.8) are PROVISIONAL and the compact wire format remains OPEN.

16.1 Overview

```
cedar-street-building-a.buspkg
  mimetype                    first entry, stored uncompressed
  manifest.json               the entry point
  model/
    identity.json             parties and buildings
    properties.json           property definitions
    spaces.json               spaces and zones
    assets.json               systems, asset types, assets
    points.json               points and telemetry bindings
    relationships.json        the graph
    intent.json               objectives, targets, constraints, schedules, sequences
    alarms.json               alarm definitions
    events.json               events
    knowledge.json            knowledge notes
    resources.json            resource descriptors
    presentation.json         drawings, layers, elements, bindings
    provenance.json           change records
  resources/                  included artifacts
  dictionaries/               optional compaction (16.8)
  extensions/                 unattached extension payloads
```

16.2 The container

A package **MUST** be a ZIP archive conforming to [ISO21320-1]. Only STORED and DEFLATED are permitted, and encryption **MUST NOT** be used at container level.

The first entry **MUST** be named `mimetype`, stored uncompressed, with no extra field, containing exactly the package media type with no trailing newline. This places the media type at a fixed byte offset so a file can be identified without parsing the archive, following the convention of ODF and EPUB.

The file extension is `.buspkg`. The media type is `application/vnd.bus.package`.

> NOTE The media type is in the vendor tree because no recognized standards development organization holds this specification. On adoption, registration **MUST** move to the standards tree per [RFC6838]. The vendor-tree name `application/vnd.bus.package` **MUST** continue to be accepted indefinitely; new packages **SHOULD** emit the standards-tree name once registration is complete; and a receiver **MUST** accept both, because packages written during the transition outlive it.

16.3 The manifest

FIELD	REQUIREMENT
`busVersion`, `packageId`, `createdAt`	**REQUIRED**.
`profile`	**REQUIRED**. The conformance profile claimed. 2.3.
`subject`	**REQUIRED**. Buildings, sites, owner, and the instant the model state was captured.
`exporter`	**REQUIRED**. Vendor, product, version, and a URI for the published conformance statement.
`documents`	**REQUIRED**. Path, type, media type, digest, byte length, entity count.
`resources`	**REQUIRED** where resources are included.
`extensions`	**REQUIRED** where extensions are used. 15.3.
`actuation`	**REQUIRED**. 16.7.
`completeness`	**REQUIRED**. 16.6.
`dictionaries`	**OPTIONAL**. 16.8.
`predecessor`	**RECOMMENDED**. The package this one succeeds, forming a chain of custody.
`signatures`	**OPTIONAL**, but **SHOULD** be present where actuation is present. 19.2.

16.4 Layout and partitioning

Model documents **MUST** be under `model/`, resources under `resources/`, dictionaries under `dictionaries/`, and unattached extension payloads under `extensions/`.

A document type **MAY** be split across files; each part is listed separately with the same `partOf` value, and an importer **MUST** treat the parts as one logical document without assuming ordering. Every archive entry other than `mimetype` and `manifest.json` **MUST** appear in the manifest. No entry may be absolute, contain a `..` segment, or resolve outside the package root.

16.5 Integrity

Every document and included resource **MUST** carry a digest computed over the exact bytes stored in the archive. An importer **MUST** verify every digest before ingesting any content and **MUST** refuse a package in which any digest does not match. Partial ingestion of a package that fails verification is NOT permitted.

A resource referenced rather than included **MUST** carry a durable `uri` and **SHOULD** carry a digest. A reference with neither is not durable and does not satisfy 2.4.

16.6 The completeness declaration

Requirement 16.6-1

A manifest **MUST** declare, for each of the twelve substrate domains, whether the package carries it in full, in part, or not at all. A domain declared partial or absent **MUST** carry a reason code. Omitting a domain is a package validity failure, and no other conformance result is meaningful without it.

The twelve domains are `identity`, `spaces`, `assets`, `points`, `relationships`, `intent`, `events`, `timeSeries`, `knowledge`, `resources`, `representation`, and `provenance`.

> NOTE v0.1 had eleven. The `history` domain was split into `events` and `timeSeries` because events are now core and time series are specified in BUS-2, the second part of this standard; an exporter can be complete on one and absent on the other, which is the honest statement either way.
>
> NOTE Document types map onto domains where the names differ: the `alarms` document declares under `intent`, because alarm definitions are operational intent (11.1); the `presentation` document declares under `representation`; and the `properties` and `identityMap` documents are infrastructure accompanying every profile rather than domains in their own right.

REASON CODE	MEANING
`notHeldBySystem`	The exporting system never held this. Not a limitation of the exporter.
`notSupportedByExporter`	The system holds it; the export function does not carry it. The code an owner should read as a product gap.
`deferredToLaterVersion`	This version of the standard does not specify the domain. As of v0.3 every domain is specified — `timeSeries` by BUS-2 — so a v0.3 exporter has no domain to write this against.
`retentionExpired`	The system held it and no longer does: the retention window purged it. Distinct from `notHeldBySystem` because *collected and purged* and *never collected* age into different answers. New in v0.3, with BUS-2.
`sourceUnavailable`	The store that holds it could not be reached at export time — a historian offline, an archive tier unmounted. The data may exist; this package does not carry it. New in v0.3, with BUS-2.
`excludedByOwner`	Withheld at the owner's instruction.
`excludedForConfidentiality`	Withheld for confidentiality, including personal data. 19.3.

REASON CODE	MEANING
`excludedForSafety`	Withheld because its disclosure would create operational risk. Typically actuation. 19.2.
`excludedForSize`	Withheld for size, with the content available by another route.
`other`	Anything else. A `note` is **REQUIRED**.

> NOTE The declaration is deliberately uncomfortable to write. An exporter that cannot carry graphics must say so, in a fixed code, in a file the owner reads before purchase. That discomfort is the enforcement mechanism, since no technical measure can compel completeness.

16.7 The actuation declaration

AGREED New in v0.2, following the decision that command actions travel.

Requirement 16.7-1

A manifest **MUST** declare whether the package contains write bindings, and if so how many and at what highest priority. A package containing write bindings and declaring actuation absent is invalid (test PK-15).

```
"actuation": {
  "present": true,
  "bindingCount": 2,
  "highestPriority": 8,
  "note": "Supply air setpoint at priority 10 and supply fan command at priority 8.
           Both require confirmation. The fan command is interlocked against the
           Level 2 fire alarm zone."
}
```

The declaration exists so that a recipient knows what they are holding before they open the model. A package with actuation is not an ordinary document, and the difference should not have to be discovered by reading thirteen thousand entities.

16.8 Dictionaries

PROVISIONAL

A package **MAY** carry dictionaries that replace repeated strings — entity types, predicates, property definition identifiers, units, classifications, symbol roles, entity identifiers — with compact tokens. Where present they **MUST** be declared in the manifest, and a conforming importer **MUST** be able to expand every dictionary declared.

Canonical JSON remains the form against which conformance is measured (17.5). The model must never become hostage to a compression scheme.

> NOTE The volume driver in a graphics-bearing package is bindings, not points: thousands of near-identical records differing only in a target identifier. That is the ideal case for dictionary coding, and it is the reason dictionaries appear at all. The compact wire format itself is open; see Annex E.

16.9 Package chaining

A package **SHOULD** carry a `predecessor` naming the identifier, digest, and creation time of the package it succeeds. Over several transfers this forms a chain of custody: an owner can demonstrate what was handed over, by whom, and when, without depending on any party in the chain still existing.

16.10 Scale and progressive disclosure

Informative.

This standard deliberately imposes no hard limit on entity count or decompressed package size, and the scale measurement (test RT-14) measures rather than bounds. Implementers will nevertheless meet practical limits when exporting whole portfolios or long event histories. When an exporter withholds material solely because of size, it **MUST** use the reason code `excludedForSize` and **SHOULD** supply a durable external locator with a digest (16.5), so that the omitted content remains recoverable and the omission remains checkable. Exhaustive event logs and time-series payloads carried under BUS-2 are the domains most likely to need the code.

Profiles remain floors, not ceilings: an implementation claiming `core` **MAY** still carry `representation`; it simply does not have its presentation handling tested. A large export **MAY** be split across multiple predecessor-linked packages (16.9), provided each package's completeness declaration is accurate for its own contents — a declaration that is true of the set but false of the member is false.

17 The JSON binding

AGREED

17.1 Status

This clause defines the first normative binding. Additional bindings **MAY** be defined later; an RDF binding is anticipated. A conformance claim **MUST** name the binding it was tested against. Documents are JSON per [RFC8259], UTF-8, no byte order mark.

17.2 The document envelope

```
{ "busVersion": "0.2",
  "documentType": "presentation",
  "packageId": "urn:uuid:...",
  "generatedAt": "2026-07-26T09:15:00-05:00[America/Chicago]",
  "namespaces": { "ex": "https://example.com/bus-ext/v1" },
  "entities": [ ... ] }
```

`documentType` **MUST** match the type declared for that path in the manifest. Entity order within the array is NOT significant and **MUST NOT** be relied upon.

17.3 Naming and encoding rules

- Property names are lowerCamelCase and case-sensitive.
- Class names are UpperCamelCase and always prefixed. Predicates are lowerCamelCase and always prefixed.
- Symbol roles are lowerCamelCase segments separated by dots.
- A property whose value is unknown **MUST** be omitted. An explicit `null` asserts the value is known to be absent, which is a different statement and **MUST NOT** be used to mean unknown.
- An empty array asserts the collection is known to be empty. Where that is not the intent, omit the property.

The distinction between omission and `null` is not pedantry. In a model read by people who cannot ask the author what they meant, the difference between "we did not record this" and "there is none" is frequently the whole content of the field.

17.4 Schema validation

SCHEMA	VALIDATES
`bus-core.schema.json`	Datatypes, the entity base, and every core class including the presentation model.
`bus-document.schema.json`	The document envelope and the classes permitted for each `documentType`.
`bus-manifest.schema.json`	The manifest, including completeness, actuation, and dictionary declarations.
`bus-timeseries.schema.json`	The `bus:Series` entity of BUS-2, and the machine-readable facts of the `bus-csv-1` sample format. Sample files themselves are not JSON and are never schema-validated (BUS-2 1.3).

> NOTE Implementation experience: an importer whose internal model is a set of typed classes will silently drop fields those classes do not declare, which violates preservation over comprehension (2.4) in a way no schema check detects. The reference implementation holds entities verbatim and layers interpretation on top; an implementer who maps onto typed models must round-trip the unmapped remainder explicitly.

The schemas use [JSONSCHEMA] draft 2020-12 and resolve cross-references by `$id`; a validator **MUST** be configured with all four as a registry. Schema validation is necessary and not sufficient: it cannot detect a dangling reference, a mismatched digest, an undeclared namespace, an undefined property, an inaccurate actuation declaration, or a false completeness declaration. Annex A group PK exists for those.

17.5 Canonical form

A model document in canonical form is its [RFC8785] canonicalization with two rules applied first:

4. The `entities` array is sorted ascending by `id`, compared as UTF-8 code point sequences.
5. Within each entity, `externalIds` is sorted by scheme then value, `classifications` by system then code, and `properties` by definition. All other arrays retain their order, which is significant — notably `mapping.ranges` and `routing`.

Semantic equivalence, as used in 2.4 and Annex A, is: after canonicalization and after applying the identity mapping of test RT-1, the two documents are equal except where a difference is explicitly permitted by the applicable test.

Two envelope fields are permitted differences under every round-trip test: `generatedAt` and `packageId`. They identify the act of serialization, not the model, and requiring them to survive a round trip would make honest re-export impossible. Everything else in the envelope, including `namespaces`, is compared.

> NOTE Digests in the manifest are computed over stored bytes, not canonical form, because a receiver must verify integrity before parsing.

17.6 JSON-LD compatibility

The binding is designed to be interpretable as JSON-LD without modification. This version does not define a normative context document and an implementation **MUST NOT** depend on one. The intent is that an RDF binding can be added later without changing the JSON on the wire.

18 Versioning and compatibility

AGREED

18.1 Version identifiers

This document and its schemas are versioned `MAJOR.MINOR`. Packages declare the version they conform to in `busVersion`. Entities separately carry a `revision` (6.6), which is unrelated.

18.2 Compatibility rules

CHANGE	IMPACT	IMPORTER OBLIGATION
Adding an **OPTIONAL** property	MINOR	Preserve it. Ignoring is permitted; discarding is not.
Adding an enumeration member	MINOR	Preserve the value. **MUST NOT** coerce to a default.
Adding a class, predicate, channel, or symbol role	MINOR	Preserve entities using it. Symbol roles degrade per 13.4.2.
Promoting an extension or property definition to the core	MINOR	Accept both forms. **SHOULD** emit the core form on re-export.
Deprecating a construct	MINOR	Continue to accept it until a MAJOR change.
Removing or narrowing a construct	MAJOR	Not permitted within a MAJOR version.
Changing the meaning of a construct	MAJOR	Not permitted within a MAJOR version.

An importer **MUST** refuse a package whose `busVersion` has a higher MAJOR number and **MUST** report the version it supports. It **MUST** accept a package with the same MAJOR and a higher MINOR, preserving what it does not understand and reporting which constructs it ignored.

> NOTE Acceptance of a newer MINOR constrains schema validation: a package written under 0.3 may carry enumeration members and properties that the 0.2 schemas reject, and those rejections are unknown constructs to be preserved and reported, never grounds for refusal. Structural checks — digests, referential closure, identifier uniqueness, the completeness declaration — remain strict at every version. The reference implementation initially got this wrong, and test IM-3 exists to catch exactly that mistake.

18.3 Backward interpretability

The obligation is stronger than backward compatibility of software. It is backward interpretability of packages: a package written correctly under v0.2 **MUST** remain interpretable under every subsequent version with the same MAJOR number, without the original exporter and without a migration tool only one party can run.

Where a MAJOR change is unavoidable, the governance body **MUST** publish an open-source, deterministic migration from the previous MAJOR version before the new one takes effect.

> NOTE v0.2 is not backward compatible with v0.1 — the property model, the space classes, and the party class all changed. This is permitted only because no v0.1 package exists outside this project. It will not be permitted again.

18.3.1 Two v0.3 requirements that read as narrowing, and why they are not

Two additions in this version constrain fields that were previously unconstrained, and 18.2 classifies narrowing a construct as a MAJOR change. Both are scoped so that no v0.2 package becomes invalid:

- **Declared port names** — a `subjectPort` or `objectPort` **MUST** name a port its endpoint declares. A v0.2 package carries no `ports` array, so it cannot fail the check — the requirement applies where an asset declares ports, which no package written before this version does. A receiver reading a v0.2 package **MUST** continue to treat the string as the opaque label it was.
- **The figure profile** — a figure **MUST** conform to `bus-vector-1`. A v0.2 figure was written against a format that was named and never defined, so there was nothing for it to conform to and nothing it can be said to have violated. A receiver meeting a v0.2 figure that would fail the profile **MUST** refuse to *render* it under 19.3, and **MUST NOT** refuse the package.

Stated plainly: this version narrows what a *v0.3 exporter* may write. It does not narrow what a receiver **MUST** accept, which is the thing 18.3 actually protects. A requirement that made existing packages invalid would be a MAJOR change however it were graded, and neither of these does.

> NOTE This subsection exists because the first draft of v0.3 stated both requirements without scoping them, which put them in direct conflict with 18.2 and 18.3 — and the document said nothing about the conflict. A specification that contradicts itself and does not notice is worse than one that admits a hard case.

19 Security, privacy, and integrity

AGREED Escalated in v0.2 because command actions now travel.

19.1 What a package is

A conforming package is a complete operational description of a building: its plant, topology, control points, protocol addresses, setpoints, safety limits, alarm routing, occupancy schedule, floor plan geometry — and, where actuation is present, the map by which plant can be commanded, at what priority.

That is a complete operational control kit. This standard makes packages portable. It does not make them safe to publish, and nothing here should be read as encouraging owners or vendors to treat them as ordinary documents.

19.2 Actuation

Requirement 19.2-1

A package carrying write bindings **MUST** declare actuation in the manifest (16.7), **SHOULD** be signed, and **SHOULD** be transferred only under an agreement that names who may hold it. When signed, the signature **MUST** cover the manifest bytes — the manifest already lists the digest of every document and resource and contains the actuation declaration, so signing it binds the whole package with nothing to enumerate and nothing to enumerate wrongly. The envelope **SHOULD** be a detached [JWS] or CMS signature carried in the manifest's `signatures` slot; algorithm choice and key management are outside this standard. An exporter **MUST** be able to exclude command bindings on request, declaring the exclusion with reason `excludedForSafety`.

An unsigned package that carries write bindings remains valid. A conformance statement claiming a security posture **SHOULD** report whether actuation packages are signed in normal operation — the registry holds statements, and *we sign what we ship* is a statement (test PK-28 checks the envelope where one is present).

Excluding actuation degrades the package: the receiving system will have to rebuild the command layer, which is the most safety-critical part to rebuild. The exclusion exists so that the decision is the owner's and is visible, not so that it becomes routine.

19.3 Handling

- A package **SHOULD** be encrypted in transit and at rest. Container-level encryption is not used (16.2), so protection is the responsibility of the transport and the store.
- An implementation **MUST NOT** include live credentials, private keys, shared secrets, or authentication tokens. Telemetry bindings carry addresses, not credentials.
- An importer **MUST** treat every path as untrusted input and **MUST** reject absolute paths and traversal segments before extraction. The ZIP container is a well-known vector for path traversal.
- An importer **MUST** verify digests before ingestion and **MUST** bound the resources it will expend, since neither entity count nor decompressed size is constrained by this standard.
- An importer **MUST** bound the size of an inline `figure` payload it will parse, and **MUST NOT** execute script, resolve external references, or fetch remote content from a figure. A neutral vector payload is data, not a document.

19.4 Personal data

A building model is not intended to contain personal data, but four constructs commonly do: parties name individuals; provenance names who asserted what; knowledge notes record what individuals observed and decided; and events name who acted, at what time.

Occupancy-related points and schedules can also reveal the movements of identifiable people in small tenancies.

An exporter **MUST** provide a means to withhold personal data and **MUST** declare any withholding with reason `excludedForConfidentiality`. It **MUST NOT** silently redact, because an undeclared redaction is indistinguishable to a receiver from an assertion that was never made.

> NOTE This creates real tension with clause 14. Provenance is most valuable when it names a person and least compliant when it does. The standard resolves the tension by making withholding visible rather than by preferring either side; an owner configuring an export is making a judgment this document cannot make for them.

19.5 Regulatory context

This standard is voluntary and confers no rights. Regulation (EU) 2023/2854, the Data Act, has since 12 September 2025 given users of connected products a right of access to data those products generate and a right to direct it to a third party; commentary treats building management equipment as within scope. A conforming exporter is a practical means of satisfying such a request. It is not evidence of compliance with it, and the two should not be conflated in marketing.

20 Relationship to existing work

This clause is informative. Statuses were verified in July 2026 and will age.

20.1 Position

BUS-1 sits above the semantic modeling standards. It does not replace them, compete with them, or profile them. It specifies what an owner must be handed and what a receiving system must be able to reconstruct, which is a question none of them asks.

Formal mappings are deliberately not part of v0.2. Publishing a mapping to a specification that is itself unpublished or pre-release would bind this standard to another body's schedule, and 7.7 preserves external classifications losslessly without one.

20.2 The semantic modeling standards

STANDARD	STATUS, JULY 2026	RELATIONSHIP
ASHRAE SPC 223P	Project committee, unpublished. Second advisory public review closed August 2025. Models type, topology, composition, telemetry linkage, characteristics, in RDF.	Complementary. No intent, knowledge, geometry, graphics, alarms, events, or exchange package. Its classes are a natural `classification` value.
Brick Schema	v1.4.4 stable, v1.5.0-rc1 pre-release. Brick Consortium, not a standards development organization.	Complementary. Equipment and point taxonomy, carried in `classifications`.

STANDARD	STATUS, JULY 2026	RELATIONSHIP
Project Haystack	Version 4.0.0 is the current published defs release. Haystack 5 with Xeto announced 2025, unreleased.	Complementary. Tag vocabulary in `classifications`. Five serializations, two explicitly lossy, and no package concept.
RealEstateCore	v4.0.0, distributed inside Brick releases. Harmonized, not merged.	Complementary. Spatial and real-estate concepts, plus the only native geometry among these.
ISO 16739-1:2024 (IFC)	Second edition, March 2024, schema IFC 4.3.2.0.	Adjacent. Design and handover with 3D geometry BUS-1 does not carry. No telemetry, no point addressing; `ControlElementId` was removed in IFC4.
ANSI/ASHRAE 135-2024 (BACnet)	Protocol Revision 31. Addendum 135-2012ba added a `Tags` property in 2016.	Below. Its tag mechanism defines a container and no vocabulary, and it has no bulk export or configuration portability.
ANSI/ASHRAE 231-2026 (CDL)	Published February 2026.	Adjacent. Design-time control description. Referenced by `bus:Sequence`, not extracted.

20.3 The portability gap

No standard, in any body, requires a building technology vendor to hand over a reconstitutable package of a building's operational system, and none defines what such a package would contain.

COMPONENT OF AN EXIT PACKAGE	CAN BE SERIALIZED BY	COMPELLED BY
Point and asset model	Brick, Haystack, 223P	nothing
Historical time series	**BUS-2 of this standard**	EU Data Act, partially, for device-generated data
Graphics with bindings	**nothing — clause 13 of this document**	nothing
Control logic	ASHRAE 231, at design time	nothing
Schedules, alarm configuration, analytics rules	nothing	nothing
Integration and driver configuration	nothing	nothing

ASHRAE Guideline 13, the natural home for such a requirement, covers architecture, input and output structure, communication, configuration, testing, and documentation,

and contains no section on owner data rights or vendor exit. Software escrow has no ISO, ANSI, or BSI standard and appears rarely in building automation contracting. Independent data layer architectures are widely discussed and have no specification, governance body, or exit mechanism.

20.4 Portable presentation: the evidence and the precedents

Point semantics are standardized, geometry is standardized, transport is standardized. The symbol on a page bound to a live point is standardized nowhere, and every effort in this field has explicitly placed it out of scope.

Three pieces of evidence establish that the rebuild cost is structural rather than incidental. Schneider published TGML as a format specification in 2010 and its own knowledge base states that artwork travels but "all of the binding and linking information would be lost," because bindings live in the database. Johnson Controls' own N1-to-Metasys conversion utility drops color-change, animation, and sizing dynamics and requires manual rebinding. And QA Graphics, the dominant third-party graphics house, organizes its service line by platform with no conversion service between them — its business model is the missing format.

Two precedents outside buildings show the problem is tractable, and clause 13 is measured against them:

- **IEC 61970-453** — the CIM Diagram Layout Profile exchanges diagram geometry in which each diagram object links to the underlying modeled power-system object. Real bindings, machine-readable, in production. It does not exchange symbol libraries or styling, and carries static layout rather than rendering rules — which is precisely the division clause 13 adopts.
- **IEC 61804-3 (EDDL)** — a standardized, vendor-portable description language including menus, graphical representations, and a graphing system bound to device parameters. The one place a standards body shipped portable graphics with data bindings. Its scope is device faceplates, not plant graphics.

A Conformance test suite outline

Normative as to what must be tested. A fixture kit of twelve tampered packages, each declaring the behavior a conforming implementation must exhibit, is published with the reference tooling, together with two implementations — Python and TypeScript — that pass the suite and reconstruct each other's exports canonically. Both share an author; results from an independent party remain self-declared until one exists.

A.1 Structure

The groups match the four conformance targets of 2.2, with one addition: group TS, defined by BUS-2 clause 2 and listed in the same suite so that the standard tests remain one artifact. Each test carries a grade:

- **MUST** — failure means the implementation does not conform at the claimed profile.
- **SHOULD** — failure is reported in the conformance statement but does not defeat conformance.
- **INFO** — measured and reported; no pass or fail.

A machine-readable form is distributed as `bus-conformance-tests.json`. Where the two differ, this annex governs. Group RT is the heart of the suite: each test exports package A, imports it, re-exports package B, and compares in canonical form per 17.5.

A.2 Group PK — Package validity

Target: package.

ID	GRADE	TEST AND PASS CRITERION
PK-1	MUST	**Container form** (BUS-1 16.2). Open the package as a ZIP archive. *Pass:* The archive opens, the first entry is named 'mimetype', it is stored uncompressed, and its content is exactly the registered media type string with no trailing newline.
PK-2	MUST	**Manifest present and valid** (BUS-1 16.3). Read manifest.json at the package root and validate against bus-manifest.schema.json. *Pass:* The manifest exists and validates with no errors.
PK-3	MUST	**Document validity** (BUS-1 17.4). Validate every file listed in manifest.documents against bus-document.schema.json. *Pass:* Every listed document exists and validates with no errors.
PK-4	MUST	**Digest integrity** (BUS-1 16.5). Recompute the digest of every entry in manifest.documents and manifest.resources. *Pass:* Every recomputed digest equals the declared digest, and every declared byteLength matches.
PK-5	MUST	**No undeclared payload** (BUS-1 16.4). Enumerate all archive entries. *Pass:* Every entry other than 'mimetype' and 'manifest.json' appears in manifest.documents or manifest.resources.

ID	GRADE	TEST AND PASS CRITERION
PK-6	MUST	**Path safety** (BUS-116.4). Inspect every archive entry name. *Pass:* No entry is absolute, contains a '..' segment, or resolves outside the package root.
PK-7	MUST	**Identifier uniqueness** (BUS-16.2). Collect every entity id in the package. *Pass:* No id occurs twice.
PK-8	MUST	**Referential closure** (BUS-19.2). Resolve every reference in the package: relationship subject and object, deviceRef, networkRef, targetOf, appliesToRef, scheduleRef, deployment.controllerRef, port mapsTo.assetRef, ChangeRecord subject, and every other IdRef-typed field defined by the core schema. Derive the field list from the schema rather than from this sentence: v0.3 added three reference fields that a hand-maintained list did not catch. *Pass:* Every reference resolves to an entity present in the package, or to an entity explicitly declared external in the manifest.
PK-9	MUST	**Extension declaration** (BUS-115.3). Collect every namespace prefix used in type names, predicates and extension keys. *Pass:* Every non-bus prefix is declared in manifest.extensions.
PK-10	MUST	**Completeness declaration** (BUS-116.6). Inspect manifest.completeness.domains. *Pass:* All twelve substrate domains are present, and every domain in state partial or absent carries a reason.
PK-11	SHOULD	**Profile support** (BUS-12.3). Compare the declared profile against the domains actually populated. *Pass:* Every domain required by the declared profile is in state complete or partial, not absent.
PK-12	MUST	**Property definition coverage** (BUS-17.8). Collect every property value in the package and resolve its definition. *Pass:* Every property value references a bus:PropertyDefinition present in the package. No bare key-value property data appears anywhere.
PK-13	MUST	**Property value agrees with its definition** (BUS-17.8). For each property value, compare its shape against the declared datatype and unit. *Pass:* Quantity values carry the unit declared by the definition; string, number and enum values match their declared datatype.
PK-14	MUST	**Presentation referential integrity** (BUS-113). Resolve every element drawing and layer, every layer drawing, every represents and navigatesTo, and every binding element and target. *Pass:* All resolve. Every element's layer belongs to the same drawing as the element. Every navigatesTo names a drawing.
PK-15	MUST	**Actuation declaration is accurate** (BUS-116.7). Count write and readWrite bindings and compare against manifest.actuation. *Pass:* present matches whether any write binding exists; bindingCount and highestPriority match the content. A package containing write bindings and declaring actuation absent fails.

ID	GRADE	TEST AND PASS CRITERION
PK-16	MUST	**Write bindings are well formed** (BUS-1 13.7). For each write binding, inspect its write block and its target. *Pass:* A command priority is declared, the target point is writable, the priority is within the target's declared priority model, and any constraintRef or interlockRef resolves to a bus:Constraint.
PK-17	SHOULD	**Drawing scale is checkable** (BUS-1 13.3). Inspect each drawing for a reference dimension. *Pass:* A reference dimension is present and its stated length agrees with the distance between its two drawing coordinates under the declared units.
PK-18	MUST	**Port names are unique within an asset** (BUS-1 9.4.2, Requirement 9.4-1). For every bus:Asset carrying ports, compare the port names in the array. *Pass:* No asset declares two ports with the same name. JSON Schema cannot express uniqueness over a field of an array item, so this is the only check that enforces it — and PK-21 is undecidable without it, because a subjectPort could match two ports at once. Note that direction and medium are required by the schema and therefore already graded by PK-3.
PK-19	MUST	**Paired and mapped ports resolve** (BUS-1 9.4.2). For every port carrying pairedWith or mapsTo, resolve the target. *Pass:* pairedWith names another port on the same asset. mapsTo names an asset that exists in the package and a port that asset declares. An unresolvable pairing is reported, not repaired.
PK-20	MUST	**Port medium agrees with relationship medium** (BUS-1 9.4.3, Requirement 9.4-2). For every relationship naming a port at either end, compare the relationship's medium with the medium of the named port. *Pass:* They are equal. A relationship carrying chilled water into a port declared for air is reported: it is the signature of a mis-wired export and is invisible without ports.
PK-21	MUST	**Named ports are declared ports** (BUS-1 9.4.3, Requirement 9.4-1). For every relationship carrying subjectPort or objectPort, look for a matching Port.name on the endpoint asset. *Pass:* Every named port is declared by the asset at that end of the relationship. A port name matching nothing is a package error. Scoped per 18.3.1: the check applies to an endpoint that declares ports, so a v0.2 package — which cannot declare any — neither fails nor is exempted by special case.
PK-22	MUST	**Figures conform to the bus-vector-1 profile** (BUS-1 13.4.3). Parse every figure source. Compare its elements and attributes against bus-vector-profile.json. *Pass:* Every figure is well-formed XML, declares a viewBox, contains no element outside the permitted list, contains no reference resolving outside the figure, carries no CSS, and is within the nesting and element ceilings of VEC-5. Scoped per 18.3.1: a package declaring busVersion 0.2 is not graded on a profile that did not exist when it was written. A receiver still refuses to render a figure that violates 19.3, which is a rendering refusal and not a package refusal.
PK-23	MUST	**Network references resolve** (BUS-1 8.10.3). For every TelemetryBinding carrying networkRef, resolve it. *Pass:* It names a bus:Network in the package. A binding pointing at a network the package does not contain is reported.

ID	GRADE	TEST AND PASS CRITERION
PK-24	**SHOULD**	**Protocol tokens are spelled from the registry** (BUS-1 8.10.2). Compare every protocol token against bus-protocols.json. *Pass:* A token that matches a registered token case-insensitively but not exactly is reported — 'BACnet/IP' where 'bacnet' is registered. An unregistered token is NOT a failure: the registry is open and an unknown protocol **MUST** be preserved.
PK-25	**SHOULD**	**Sequences declare whether they are designed or deployed** (BUS-1 10.5). For every bus:Sequence, look for a deployment block. *Pass:* The block is present and its state is asDesigned, asDeployed or unknown. Absence is reported: it is the case where a reader cannot tell whether they are looking at the submittal or at what the controller runs.
PK-26	**MUST**	**Deployment controller references resolve** (BUS-1 10.5.1). For every bus:Sequence carrying deployment.controllerRef, resolve it. *Pass:* It names an entity in the package. PK-8 enumerates the reference fields it closes over and was not extended when this field was added, so without this check a sequence could name a controller that does not exist and pass the suite.
PK-27	**MUST**	**Georeference forms agree and are non-degenerate** (BUS-1 13.3). For every georeference carrying the v0.3 affine form: compute the determinant of its linear part (*ae* - *bd*); where the anchor fields are also present, apply the affine to (atX, atY) and compare against the anchor's position, and compare the affine's rotation against the declared rotation. *Pass:* The determinant is nonzero, every coefficient is finite, and where both forms are present they describe the same placement within the RT-18 tolerance. A singular transform maps the whole drawing onto a line, and two forms that disagree are a package contradicting itself — both package-visible, neither a matter of exporter knowledge. Applies only where the affine is present; both forms are **OPTIONAL** and an exporter that does not know a transform omits it (13.3).
PK-28	**SHOULD**	**Signature envelopes parse and declare their coverage** (BUS-1 19.2, Requirement 19.2-1). For every signature present in the manifest's signatures slot, parse the envelope. *Pass:* It is a structurally valid detached JWS or CMS envelope and it covers the manifest bytes. Cryptographic verification is out of scope for a package test — the receiver holds no keys — but an envelope that does not parse, or that covers something other than the manifest, is decoration wearing a signature's clothes, and is reported. Absence of any signature is NOT a failure; 19.2-1 grades signing **SHOULD**.

A.3 Group RT — Round-trip reconstruction

Target: roundTrip. Each test exports package A from the source system, imports A into the receiving system, re-exports package B from the receiving system, and compares A with B after canonicalization per BUS-1 14.5.

ID	GRADE	TEST AND PASS CRITERION
RT-1	**MUST**	**Identity preservation** (BUS-16.4). Compare the id of every entity in A with its counterpart in B. *Pass:* Every id in A appears unchanged in B, or B contains a bus:IdentityMapping with equivalence 'same' linking the new id to the original. No entity is silently reidentified.
RT-2	**MUST**	**Object meaning** (BUS-18). Compare type, pointKind, dataType, unit, range, enumeration, classifications and attributes for every entity. *Pass:* All are preserved. Unit codes are preserved without conversion, or converted with the original code recorded in provenance.
RT-3	**MUST**	**Relationship graph** (BUS-19). Construct the labeled directed graph of relationships in A and in B and test for isomorphism under the identity mapping from RT-1. *Pass:* The graphs are isomorphic, including predicate, direction, medium and qualifiers. Inverse-form substitution is permitted; loss of a relationship is not.
RT-4	**MUST**	**Intent distinguishability** (BUS-110). For each Objective, Target, Constraint, Schedule and SequenceReference in A, locate it in B. *Pass:* The receiving system can still distinguish what should happen from what is happening and what happened. Constraints retain their constraintKind and bounds; schedules retain their recurrence and exceptions.
RT-5	**MUST**	**Provenance retention** (BUS-114). Compare provenance blocks and ChangeRecords. *Pass:* assertedBy, assertedAt, method, source, confidence and approvals survive. The receiving system **MAY** add its own provenance; it **MUST NOT** overwrite the original.
RT-6	**MUST**	**Unknown extension preservation** (BUS-115.4). Include in A an extension namespace the receiving system does not implement, with mustUnderstand false, attached to at least one entity of each core class. *Pass:* Every extension value reappears in B byte-identical after canonicalization, and the namespace is still declared in B's manifest.
RT-7	**MUST**	**Must-understand refusal** (BUS-115.4). Include in A an extension with mustUnderstand true that the receiving system does not implement. *Pass:* The importer reports the import as incomplete and names the namespace. It **MUST NOT** report success, and it **MUST** still preserve the data.
RT-8	**MUST**	**Resource integrity** (BUS-116.5). Compare resource payloads and digests between A and B. *Pass:* Included resources are byte-identical and digests match. Externally referenced resources retain their URI and digest.
RT-9	**MUST**	**Telemetry binding retention** (BUS-18.6). Compare the binding object of every point. *Pass:* protocol, address, network and parameters are preserved verbatim, so acquisition can be re-established without rediscovery.
RT-10	**MUST**	**Knowledge retention** (BUS-18.9). Compare every KnowledgeNote and its attachment relationship. *Pass:* Body text, noteKind, observedAt and the appliesTo relationship survive unaltered.

ID	GRADE	TEST AND PASS CRITERION
RT-11	**MUST**	**Lifecycle continuity** (BUS-1 9.3). In A, replace an asset and record bus:replaces and bus:replacedBy. Round-trip. *Pass:* B retains both the superseded and the superseding entity and the replacement relationship, so history remains attributable across the change.
RT-12	**MUST**	**Omission honesty** (BUS-1 16.6). Compare B's completeness declaration against what B actually contains, and against A. *Pass:* Anything present in A but absent from B is declared in B's completeness block with a reason. Silent loss is a failure of this test even when every other test passes.
RT-13	**SHOULD**	**Idempotence** (BUS-1 17.5). Import B and re-export as C. *Pass:* C and B are identical after canonicalization. Drift between successive round trips indicates unstable identity or unstable ordering.
RT-14	INFO	**Scale** (BUS-1 16.4). Round-trip a package with at least 50000 points and 250000 relationships. *Pass:* Report wall-clock export time, import time and peak memory. No pass or fail threshold is set in v0.1.
RT-15	**MUST**	**Binding preservation** (BUS-1 13.6). Compare every binding in A with its counterpart in B: element, target, channel, direction, mapping, quality. *Pass:* All survive. A binding lost, repointed, or reduced to a different channel is a failure, even where the graphic still renders.
RT-16	**MUST**	**Command priority is not substituted** (BUS-1 13.7). Round-trip a write binding whose priority the receiving system does not natively support. *Pass:* The priority is re-exported unchanged and the importer reported that it could not honor it. Silent substitution is a failure even though the graphic appears identical.
RT-17	**MUST**	**Quality behavior preservation** (BUS-1 13.6). Compare the quality block of every binding. *Pass:* staleAfter, onStale, onUnavailable and onOutOfService survive unchanged.
RT-18	**MUST**	**Geometry fidelity** (BUS-1 13.5). Compare drawing coordinate systems, extents, layer purposes, and every element geometry and transform. *Pass:* Coordinates agree within one part in 10000 of the drawing extent. The coordinate system, georeference, and layer purposes are unchanged.
RT-19	**MUST**	**Symbol role fallback** (BUS-1 13.4). Include an element whose leaf symbol role the receiving system does not implement but whose ancestor it does. *Pass:* The receiver renders using the ancestor role and re-exports the original leaf role unchanged. Substituting the ancestor into the exported data is a failure.
RT-20	**MUST**	**Bespoke figure preservation** (BUS-1 13.4). Round-trip an element carrying a figure payload for which no symbol role applies. *Pass:* The figure survives byte-identical after canonicalization and is rendered substantially as given rather than replaced by a library symbol.

ID	GRADE	TEST AND PASS CRITERION
RT-21	MUST	**Property definitions survive** (BUS-17.8). Compare every property definition and every property value. *Pass:* Definitions survive with their meaning text intact. No property is re-exported without its definition, and no definition is silently merged with another of the same name but different meaning.
RT-22	MUST	**Alarm configuration survives** (BUS-111). Compare every alarm definition: limits, deadband, delays, priority, acknowledgment, shelving and routing. *Pass:* All survive. Routing that the receiver cannot execute is preserved and reported as unenforced, not dropped.
RT-23	MUST	**Events survive with their subjects** (BUS-112). Compare every event and its subject reference. *Pass:* Event kind, timestamps, actor, values and subject survive. An override event that loses its subject is a failure.
RT-24	MUST	**Revision continuity** (BUS-16.6). Compare the versioning block of every entity across the round-trip. *Pass:* Revision numbers are preserved or advanced, never reset. An entity re-exported at revision 1 having arrived at revision 4 is a failure.
RT-25	MUST	**Ports survive a round trip** (BUS-19.4). Export, import, re-export. Compare ports on every asset. *Pass:* Every port's name, direction and medium is unchanged, and every relationship still names the same ports. Reordering the array is permitted; renaming a port is not.
RT-26	SHOULD	**Network structure survives a round trip** (BUS-18.10). Export, import, re-export. Compare bus:Network entities and the bindings referencing them. *Pass:* Networks appear with the same identifier and protocol, and the same bindings reference them. Collapsing two networks into one is a failure even where every binding still resolves.
RT-27	MUST	**Deployment state is not upgraded** (BUS-110.5.1). Export, import, re-export a sequence whose deployment state is unknown. *Pass:* The state is still unknown. A round trip that turns unknown into asDesigned has manufactured a claim about the building, which is the specific failure 10.5.1 names.
RT-28	MUST	**Series survive a round trip byte-identical** (BUS-27). Export, import, re-export a package carrying series. Compare every bus:Series and every sample file between A and B. *Pass:* Each sample file reappears byte-identical — the format has no permitted variation, so byte equality is the comparison rule — and each bus:Series survives with pointRef, coverage, sampling, aggregation, unitCode and rowCount unchanged. An importer stores samples however it likes; what it re-emits is the same bytes.

A.4 Group IM — Importer behavior

Target: importer.

ID	GRADE	TEST AND PASS CRITERION
IM-1	MUST	**Rejects a corrupt digest** (BUS-1 16.5). Present a package in which one document digest does not match. *Pass:* The importer refuses the package and names the failing path. Partial silent ingestion is a failure.
IM-2	MUST	**Rejects an unknown major version** (BUS-1 18.2). Present a package whose busVersion has a higher major number. *Pass:* The importer refuses and reports the version it supports.
IM-3	MUST	**Accepts an unknown minor version** (BUS-1 18.2). Present a package whose busVersion has the same major and a higher minor number. *Pass:* The importer accepts it, preserves what it does not understand, and reports which constructs it ignored.
IM-4	MUST	**Tolerates unknown enumeration members** (BUS-1 18.3). Present a package using an enumeration member added after the importer was built. *Pass:* The importer preserves the value and does not coerce it to a default.
IM-5	SHOULD	**Reports a coverage summary** (BUS-1 2.5). Import a valid package. *Pass:* The importer produces a machine-readable summary of entities ingested, entities preserved but not interpreted, and constructs not understood.
IM-6	MUST	**Unknown symbol role degrades rather than fails** (BUS-1 13.4). Present an element with a symbol role several levels deeper than the importer implements. *Pass:* The importer walks the dotted path upward, renders at the deepest ancestor it knows, and preserves the original role. Refusing the package or dropping the element is a failure.
IM-7	MUST	**Reports unenforceable actuation** (BUS-1 13.7). Present write bindings whose priority model, interlock, or confirmation requirement the importer cannot enforce. *Pass:* Each is named in the import report as unenforced. Importing them as ordinary writes without report is a failure.
IM-8	MUST	**Refuses undefined properties** (BUS-1 7.8). Present a package containing a property value whose definition is absent. *Pass:* The importer reports the package as invalid and names the property. Accepting it as an untyped string is a failure.

ID	GRADE	TEST AND PASS CRITERION
IM-9	MUST	**Preserves unknown protocol tokens** (BUS-18.10.2). Import a package whose TelemetryBinding carries a protocol token outside the registry, then re-export. *Pass:* The token survives byte-for-byte. An importer that normalizes an unknown protocol to a known one, or drops the binding, has failed.
IM-10	SHOULD	**Preserves ports it does not model** (BUS-19.4). Import a package declaring ports into a system with no port concept, then re-export. *Pass:* The ports reappear, or the re-export declares the relevant domain partial with a reason. Silent loss is the failure this test exists for.
IM-11	MUST	**Refuses an unsafe figure** (BUS-119.3). Present a package containing a figure with a script element, a foreignObject, or an href resolving outside the figure. *Pass:* The importer refuses the figure and says why. Silently stripping the offending element is a failure: the exporter is not told its package is invalid, and the next receiver may not strip it.

A.5 Group EX — Exporter behavior

Target: exporter.

ID	GRADE	TEST AND PASS CRITERION
EX-1	MUST	**Exports without a live vendor dependency** (BUS-12.4). Produce a package and inspect it for references that can only be resolved by the exporting vendor. *Pass:* Every reference is either resolvable inside the package or is a durable external locator with a digest. No reference requires a proprietary service to interpret.
EX-2	MUST	**Identifier stability across exports** (BUS-16.4). Export the same building twice with no intervening change. *Pass:* Every entity id is identical in both exports.
EX-3	MUST	**Identifier stability across change** (BUS-16.4). Rename an asset, then export again. *Pass:* The id is unchanged and the rename appears as a ChangeRecord.
EX-4	MUST	**Export is not narrower than operation** (BUS-12.4). Compare the exported model against the model the system uses to operate the building. *Pass:* No construct the system relies on to operate is omitted without a completeness declaration naming it.
EX-5	SHOULD	**Owner-initiated export** (BUS-12.2). Attempt an export as an operator-role user without vendor assistance. *Pass:* The export completes without vendor involvement, professional services, or a support ticket.
EX-6	MUST	**Graphics export is unaided** (BUS-12.4). Export a floor plan and an equipment graphic as an operator-role user. *Pass:* Both export with geometry, roles and bindings intact, without vendor involvement or professional services. This is the test the rebuild tax is measured against.

ID	GRADE	TEST AND PASS CRITERION
EX-7	**MUST**	**Property definitions are published, not invented per export** (BUS-1 7.8). Export the same building twice and compare property definition identifiers. *Pass:* Definition identifiers are stable between exports. Regenerating identifiers each time defeats RT-21 and makes cross-package comparison impossible.
EX-8	**SHOULD**	**Does not invent ports** (BUS-1 9.4.3, Requirement 9.4-3). Export from a system that records connection topology but not port structure. *Pass:* Ports are absent. An exporter that synthesizes port names to fill the field has asserted something nobody knows, which 5.3.5 exists to prevent.
EX-9	**SHOULD**	**Declares intent partial when only design-time logic is exported** (BUS-1 10.5.1, Requirement 10.5-2). Export from a system holding both design sequences and running control programs, exporting only the former. *Pass:* The manifest declares intent partial with reason notSupportedByExporter. An exporter that ships the submittal, declares intent complete, and leaves the running logic behind is technically conforming and materially false. Graded **SHOULD** despite the requirement being a **MUST**, because nothing in the package reveals what the exporting system also held: this is checked against the source system during certification, not by a receiver, and a receiver **MUST NOT** refuse on it.
EX-10	**SHOULD**	**Declares time series honestly when the historian is out of reach** (BUS-2 6.2). Export from a system whose historian is unreachable, retention-limited, or partially excluded by the owner. *Pass:* The timeSeries domain is declared partial or absent with the reason that is true: sourceUnavailable for a store that could not be reached, retentionExpired for data the window has purged, excludedByOwner for an interval withheld on instruction. Graded **SHOULD** because nothing in the package reveals what the exporter's historian held; this is checked against the source system during certification, and a receiver **MUST NOT** refuse on it.
EX-11	**MUST**	**Serializes or forks under concurrent authority** (BUS-1 6.6.1). Present concurrent updates for one entity from two systems to an exporter, then inspect what it emits. *Pass:* Either one package whose revisions advanced under a single serialized writer, or two packages with distinct packageIds and the succession recorded per 16.9. One package advancing one revision counter under two authorities asserts an ordering nobody enforced, and fails. This test exists so that the single-writer assumption is a tested boundary rather than an escape hatch: an exporter cannot claim concurrency to exempt itself from the reconstruction tests, because handling concurrency correctly is itself the requirement.

A.6 Group TS — Time series

Target: package. Defined by BUS-2 and listed here so the suite remains one artifact. The tests apply to a package carrying the timeSeries domain, and they are what makes the full profile claimable without the dispensation 2.3 carried through v0.2.

ID	GRADE	TEST AND PASS CRITERION
TS-1	**MUST**	**Sample files conform to bus-csv-1** (BUS-2 4). Parse every file a bus:Series names against the bus-csv-1 grammar: encoding, line endings, the exact header, timestamp form and ordering, the value grammar, the quality vocabulary. *Pass:* Every file parses with no violations. UTF-8 without a byte order mark, LF line endings, the header row exactly 'timestamp,value,quality', RFC 3339 UTC timestamps with the Z designator in non-decreasing order, values within the grammar for the point's dataType, quality tokens from the fixed vocabulary or empty, and the value field empty exactly where quality is missing.
TS-2	**MUST**	**Series references resolve** (BUS-2 3.2, 5.2). For every bus:Series, resolve pointRef against the package's entities and resourceRef against manifest.resources and the archive. *Pass:* pointRef names a bus:Point in the package. resourceRef names a manifest resource whose path exists in the archive, so its digest was verified by PK-4 before any row was read. A series naming a point or a file the package does not contain is a package error.
TS-3	**MUST**	**Row count is accurate** (BUS-2 3.3). Count the data rows of every sample file and compare with the declaring series. *Pass:* Series.rowCount equals the rows in the file, excluding the header. A mismatch is the signature of a truncated or padded export, and the declared count is what makes it detectable — the same mechanism as the actuation declaration (PK-15).
TS-4	**MUST**	**Coverage encloses the data** (BUS-2 3.3). Compare the first and last timestamps of every sample file with the declaring series' coverage interval. *Pass:* Every row's timestamp lies within [coverage.start, coverage.end]. Coverage **MAY** be wider than the data — an exported period that held no samples — and **MUST NOT** be narrower.
TS-5	**MUST**	**Series unit names the point's quantity** (BUS-2 3.4). For every series whose point carries a unitCode, resolve both codes against the quantity families of bus-units.json. *Pass:* The series unit and the point unit name the same quantity. A series in kPa on a temperature point is a mis-wired export, visible to any receiver, and stating the unit in both places is what makes it detectable.
TS-6	**SHOULD**	**Series unit code equals the point's** (BUS-2 3.4). For every series whose point carries a unitCode, compare the two codes for equality. *Pass:* They are equal. Where they differ, the samples are in the series' unit, the disagreement is a true fact about the source system — a historian that kept degF after the point was re-configured to degC — and it is reported, never repaired by conversion (BUS-1 7.4-1). Graded **SHOULD** because a true statement about the source is not a contradiction in the package.
TS-7	**MUST**	**Declared history agrees with carried series** (BUS-2 6.2). Where the timeSeries domain is declared complete, compare each point's history declaration (BUS-1 8.8) with the series that carry it. *Pass:* Every point declaring history available has at least one series, and the union of its series' coverage encloses the declared from/to extent. Both statements are the exporter's own, in one package; a package that declares completeness its own point declarations contradict is internally inconsistent, which is what **MUST** grades. A domain declared partial or absent is not graded here.

A.7 Reporting

A conformance statement **MUST** report every applicable test by identifier with its result and **MUST NOT** omit failures (2.5). A statement reporting RT-12 as passed while declaring domains absent without reasons is internally inconsistent and **SHOULD** be treated as no statement at all.

B Symbol role vocabulary, tier one

Normative. This is a first tier, deliberately small. Extension is a MINOR change and requires no importer update, because roles degrade by truncation (13.4.2).

Roles are hierarchical dotted paths in lowerCamelCase segments. A receiver truncates at successive dots until it reaches a role it implements. An implementation **MUST** support every root in this table; support for deeper leaves is **RECOMMENDED** and never required.

The tier holds 52 roots and 75 named leaves: small enough to ratify, large enough to cover the overwhelming majority of building graphics. The failure mode to avoid is a three-hundred-role taxonomy project that never ships, and the fallback rule is what makes that avoidable: an implementation that supports only the roots is conforming, and one that supports every leaf is better.

> NOTE Anything outside this vocabulary uses figure mode (13.4.3). An exporter **SHOULD** prefer a role where one fits, because a role is what lets the receiving vendor apply its own design system; a figure is rendered as given and therefore looks foreign in the receiving interface.

FAMILY	ROOT	LEAVES AND NOTES
Air movement	`fan`	`fan.centrifugal`, `fan.axial`, `fan.plug`; and by service `fan.centrifugal.supply`, `.return`, `.exhaust`, `.relief`
	`damper`	`damper.modulating`, `damper.twoPosition`, `damper.fireSmoke`; by service `damper.modulating.outsideAir`, `.mixed`, `.bypass`
	`louver`	Fixed intake or discharge louver.
	`duct`	`duct.segment`, `duct.segment.supply`, `.return`, `.exhaust`, `.outsideAir`
	`plenum`, `mixingBox`	Air paths that are not ducts.
	`diffuser`, `grille`	Terminal air devices.
	`terminal`	`terminal.vav`, `terminal.vav.reheat`, `terminal.cav`, `terminal.fanPowered`
Wet side	`pump`	`pump.centrifugal`, `pump.inline`; by service `pump.centrifugal.chilledWater`, `.hotWater`, `.condenser`
	`valve`	`valve.twoWay`, `valve.threeWay`, `valve.modulating`, `valve.isolation`, `valve.check`
	`coil`	`coil.cooling`, `coil.heating`, `coil.preheat`, `coil.reheat`; by medium `coil.cooling.chilledWater`, `coil.cooling.directExpansion`
	`pipe`	`pipe.segment` with the medium carried by the binding or the represented entity.

FAMILY	ROOT	LEAVES AND NOTES
	`heatExchanger`, `tank`, `humidifier`	Plant items that are neither a coil nor a vessel.
Plant	`chiller`	`chiller.centrifugal`, `chiller.screw`, `chiller.absorption`, `chiller.airCooled`
	`boiler`	`boiler.condensing`, `boiler.steam`, `boiler.electric`
	`coolingTower`	`coolingTower.openCircuit`, `coolingTower.closedCircuit`
	`compressor`, `filter`	`filter.panel`, `filter.bag`, `filter.hepa`
Electrical	`breaker`, `switch`, `transformer`, `panel`, `busway`, `generator`, `ups`, `meter.electrical`	One-line and riser primitives.
Controls	`controller`, `actuator`, `vfd`, `relay`, `thermostat`	Control devices shown on a graphic.
	`sensor`	`sensor.temperature`, `sensor.pressure`, `sensor.flow`, `sensor.humidity`, `sensor.co2`, `sensor.occupancy`, `sensor.current`
Interface	`readout.value`, `readout.text`	Displayed values distinct from the thing that produces them.
	`indicator.status`, `indicator.alarm`	State annunciation.
	`gauge`, `trend`, `bar`	Value presentation forms.

FAMILY	ROOT	LEAVES AND NOTES
	`control.input.numeric`, `control.input.enum`, `control.button.toggle`, `control.button.momentary`, `control.slider`	Interactive elements. A write binding attaches to one of these.
	`label.text`, `label.dimension`, `legend`, `navigation.link`, `group`	Annotation and structure.
Space	`space.boundary`, `wall`, `door`, `window`, `core`, `stair`, `column`	Architectural context on a plan. Usually geometry rather than symbol.

C Unit codes

Normative. C.3 is informative.

> NOTE Errata, v0.2 to v0.3: the claim that no external unit registry carried a code for the ton of refrigeration was wrong. Project Haystack does, through the Fantom unit database it delegates to — `tons_refrigeration`, symbol `tonref`. The BUS code `tonRef` is retained for legibility, and the informative mapping in C.3 records the correspondence. The incorrect claim is corrected rather than deleted so that an implementer who memorized it is not surprised twice.

C.1 Why this vocabulary exists

A dimensional value needs a canonical unit representation or values cannot be compared, converted, or checked. This document defines one rather than adopting an external registry, for two reasons.

The first is legibility. A package is read by the engineers who operate the building, and the codes they meet should be the codes they write. The second is coverage: at least one unit in daily use across the industry — the ton of refrigeration — has no code at all in the registry this draft previously cited, which is a coverage failure rather than a matter of taste.

> NOTE This is the same pattern used for classifications (7.7). Define the small thing that is needed, name the external vocabulary where one is used, and do not force everything through a foreign one.

C.2 The vocabulary

Codes are case-sensitive and follow SI capitalization: `kW`, not `KW`. Compound codes that appear here are atomic tokens, not a grammar; `m3/s` is one code, not a construction. A unit not in this table is expressed by setting `unitSystem` and carrying that system's code (7.4), and is NOT invented as a new `bus` code.

Temperature

CODE	UNIT	QUANTITY
`degF`	degrees Fahrenheit	temperature
`degC`	degrees Celsius	temperature
`K`	kelvin	temperature
`deltaDegF`	delta degrees Fahrenheit	temperature difference
`deltaDegC`	delta degrees Celsius	temperature difference

Pressure

CODE	UNIT	QUANTITY
`Pa`	pascal	pressure
`kPa`	kilopascal	pressure
`inH2O`	inches of water column	pressure
`inHg`	inches of mercury	pressure
`psi`	pounds per square inch	pressure
`bar`	bar	pressure

Flow and velocity

CODE	UNIT	QUANTITY
`cfm`	cubic feet per minute	volumetric flow
`m3/s`	cubic meters per second	volumetric flow
`m3/h`	cubic meters per hour	volumetric flow
`L/s`	liters per second	volumetric flow
`gpm`	US gallons per minute	volumetric flow
`kg/s`	kilograms per second	mass flow
`lb/h`	pounds per hour	mass flow
`fpm`	feet per minute	velocity
`m/s`	meters per second	velocity

Power and energy

CODE	UNIT	QUANTITY
`W`	watt	power
`kW`	kilowatt	power
`MW`	megawatt	power
`Btu/h`	BTU per hour	power
`MBtu/h`	thousand BTU per hour	power

CODE	UNIT	QUANTITY
`tonRef`	tons of refrigeration. One ton of refrigeration is 12000 Btu/h exactly. UCUM has no code for it. Project Haystack does, via the Fantom unit database it delegates to: tons_refrigeration, symbol tonref. The v0.2 claim that no external registry had an equivalent was wrong.	power
`hp`	horsepower	power
`Wh`	watt hour	energy
`kWh`	kilowatt hour	energy
`MWh`	megawatt hour	energy
`J`	joule	energy
`MJ`	megajoule	energy
`Btu`	British thermal unit	energy
`therm`	therm	energy
`tonHours`	ton hours of refrigeration	energy

Electrical

CODE	UNIT	QUANTITY
`V`	volt	electric potential
`A`	ampere	electric current
`kVA`	kilovolt-ampere	apparent power
`kVAR`	kilovolt-ampere reactive	reactive power
`Hz`	hertz	frequency
`ohm`	ohm	resistance

Length, area, volume, and mass

CODE	UNIT	QUANTITY
`m`	meter	length
`mm`	millimeter	length

CODE	UNIT	QUANTITY
`ft`	foot	length
`in`	inch	length
`m2`	square meters	area
`ft2`	square feet	area
`m3`	cubic meters	volume
`ft3`	cubic feet	volume
`L`	liter	volume
`gal`	US gallon	volume
`kg`	kilogram	mass
`lb`	pound	mass

Time

CODE	UNIT	QUANTITY
`ms`	millisecond	time
`s`	second	time
`min`	minute	time
`h`	hour	time
`day`	day	time
`yr`	year	time

Ratio, humidity, and concentration

CODE	UNIT	QUANTITY
`pf`	power factor	ratio
`pct`	percent	ratio
`ratio`	dimensionless ratio	ratio
`pctRH`	percent relative humidity	relative humidity
`gPerKg`	grams of water per kilogram of dry air	humidity ratio

CODE	UNIT	QUANTITY
grPerLb	grains of water per pound of dry air	humidity ratio
ppm	parts per million	concentration
ppb	parts per billion	concentration
mg/m3	milligrams per cubic meter	mass concentration
ug/m3	micrograms per cubic meter	mass concentration
count	count	count

Rotation, light, sound, and angle

CODE	UNIT	QUANTITY
rpm	revolutions per minute	rotational speed
lux	lux	illuminance
fc	foot-candle	illuminance
dB	decibel	sound level
dBA	A-weighted decibel	sound level
deg	degree of arc	angle
rad	radian	angle

C.3 Mapping to the Unified Code for Units of Measure

Informative. Provided for implementations that exchange with systems using that registry. No conformance requirement depends on this table.

> NOTE Three codes have no equivalent in that registry: tonRef, tonHours, and kVAR. All three are in daily use in building operations, which is part of why this document defines its own vocabulary.

BUS CODE	EQUIVALENT	BUS CODE	EQUIVALENT
degF	[degF]	degC	Cel
K	K	deltaDegF	[degF]
deltaDegC	Cel	Pa	Pa
kPa	kPa	inH2O	[in_i'H2O]
inHg	[in_i'Hg]	psi	[psi]

BUS CODE	EQUIVALENT	BUS CODE	EQUIVALENT
bar	bar	cfm	[cft_i]/min
m3/s	m3/s	m3/h	m3/h
L/s	L/s	gpm	[gal_us]/min
kg/s	kg/s	lb/h	[lb_av]/h
fpm	[ft_i]/min	m/s	m/s
W	W	kW	kW
MW	MW	Btu/h	[Btu_IT]/h
MBtu/h	k[Btu_IT]/h	tonRef	(none defined)
hp	[HP]	Wh	W.h
kWh	kW.h	MWh	MW.h
J	J	MJ	MJ
Btu	[Btu_IT]	therm	[Btu_IT]
tonHours	(none defined)	V	V
A	A	kVA	kV.A
kVAR	(none defined)	Hz	Hz
pf	1	ohm	Ohm
m	m	mm	mm
ft	[ft_i]	in	[in_i]
m2	m2	ft2	[sft_i]
m3	m3	ft3	[cft_i]
L	L	gal	[gal_us]
kg	kg	lb	[lb_av]
ms	ms	s	s
min	min	h	h
day	d	yr	a
pct	%	pctRH	%

BUS CODE	EQUIVALENT	BUS CODE	EQUIVALENT
ppm	[ppm]	ppb	[ppb]
ratio	1	count	1
mg/m3	mg/m3	ug/m3	ug/m3
gPerKg	g/kg	grPerLb	[gr]/[lb_av]
rpm	{rpm}	lux	lx
fc	[ftc]	dB	dB
dBA	(none defined)	deg	deg
rad	rad		

D Worked example

Informative. The package described here is distributed with this draft and validates against the normative schemas.

D.1 The building

Cedar Street Building A is a four-storey office building in Austin, Texas, built in 2009, with variable air volume and hydronic reheat. The example models one air system end to end — an air handling unit, one VAV terminal, the controller hosting their points, the zone they serve, a Level 2 floor plan, and the AHU-1 equipment graphic. It is small enough to read and large enough to exercise every core class.

DOCUMENT	ENTITIES	CONTENTS
`identity.json`	5	Owner, commissioning authority, engineer, vendor, and the building. The engineer's personal name is withheld; D.2.
`properties.json`	9	Five quantity kinds and four asset property definitions, each with prose meaning.
`spaces.json`	7	Site, two storeys, three spaces, one HVAC zone.
`assets.json`	6	One system, one asset type, four assets across equipment, component and device.
`points.json`	9	Sensors, setpoints, commands, a status enumeration, and a meter.
`relationships.json`	48	Containment, composition, flow with ports, service, power, hosting, measurement, command, control, implementation, knowledge attachment, representation.
`intent.json`	6	An objective, a target, a safety constraint, an interlock constraint, a schedule, a sequence.
`alarms.json`	1	Filter differential pressure, with limits, timing, escalation and routing.
`events.json`	2	A 2026 alarm instance and the 2017 override that is still in place.
`knowledge.json`	2	Two field notes that exist nowhere else.
`resources.json`	2	A sequence of operation and an operations manual, both included.

DOCUMENT	ENTITIES	CONTENTS
`timeseries.json`	1	One BUS-2 series declaring an hour of supply air temperature; the samples travel as a `bus-csv-1` resource.
`presentation.json`	38	Two drawings, eight layers, seventeen elements, eleven bindings of which one is a write.
`provenance.json`	2	Change records for the 2017 economizer modification and the 2019 rebalance.

The package declares profile `core-intent` and declares nine domains as less than complete, including `representation` as partial because the source system can export two drawings and not the rest.

D.2 What the example demonstrates

- **The bound graphic** — the AHU-1 drawing carries a fan, coil, filter, damper and sensor as symbol roles, a bespoke nameplate in figure mode, and eleven bindings including a supply-air setpoint write at priority 10 referencing the safety constraint. The floor plan carries a georeferenced coordinate system and a space boundary polygon bound to zone temperature for color.
- **The knowledge notes** — a damper linkage that binds between 18 and 24 percent, and a VAV box rebalanced in 2019 whose record drawings were never revised. Neither exists in any drawing, database field, or tag, and both are what disappears when an engineer retires.
- **The 2017 override event** — still in place nine years later, linked to the knowledge note that explains it. This is the sentence the event model exists to carry.
- **The two declared exclusions** — the operating engineer's personal name is withheld: the party keeps its kind and roles, a generic label stands wherever provenance names the person, and the `identity` and `provenance` domains declare the withholding with reason `excludedForConfidentiality` (19.4). The supply fan command path is excluded: the point's telemetry binding and the graphic's write channel are gone, the actuation declaration counts what remains, and the `points` domain declares `excludedForSafety` (19.2). Both are visible in the manifest before a single entity is read, which is the design.
- **The carried hour of history** — one `bus:Series` declares an hour of the AHU-1 supply air temperature as a `bus-csv-1` file — five rows, one failed scan, one doubtful reading, in the point's own `degC` — and the `timeSeries` domain declares partial with `excludedForSize`, because seventeen years of archive exist and one representative hour travels. BUS-2 clause 4.7 walks through this series.
- **The completeness declaration** — nine domains declared less than complete. An example in which everything is complete would teach implementers the wrong lesson, because no real export is complete.

D.3 Validation

The distributed example is accompanied by a validator running the structural checks of Annex A group PK: schema validation of the manifest and all fourteen documents, digest and byte-length verification, entity counts, identifier uniqueness, referential closure across relationships and every typed reference, declaration of all twelve substrate domains, property definition coverage and datatype agreement, point consistency, presentation referential integrity, binding channel and write-block validity, priority within the target's priority model, membership of every unit code and display unit code in the Annex C vocabulary, and accuracy of the actuation declaration. The example passes every such check.

> NOTE The validator tests a package, which is one of four targets. The round-trip tests that carry the reconstruction guarantee require two implementations and cannot be run against a file.

E Schema files

Informative. The schema files themselves are normative for the JSON binding, per 17.4.

FILE	CONTENTS
`bus-core.schema.json`	Datatypes, entity base, versioning, property definitions, party, spaces, assets, points, relationships, intent, alarms, events, knowledge, resources, and the full presentation model.
`bus-document.schema.json`	The document envelope with per-`documentType` class constraints, across 16 document types.
`bus-manifest.schema.json`	The manifest, including completeness across twelve domains, the actuation declaration, extension and dictionary declarations.
`bus-timeseries.schema.json`	BUS-2: the `bus:Series` entity, its sampling and aggregation vocabularies, and the fixed facts of the `bus-csv-1` sample format, held as data so the prose of BUS-2 reads them rather than retyping them.
`bus-symbol-roles.json`	Machine-readable form of Annex B. 52 roots and 75 leaves.
`bus-units.json`	Machine-readable form of Annex C, including the informative mapping of C.3.
`bus-conformance-tests.json`	Machine-readable form of Annex A. 85 tests across 5 groups.
`bus-protocols.json`	The protocol token registry of 8.10.2. 22 registered tokens; the list is open.
`bus-predicates.json`	The predicate registry of 9.2: 23 entries covering the full token set, each with its inverse, its meaning, and an ADVISORY cardinality; the schema enumeration and clause 9.2 both derive from it. New in v0.3, from the first consumer review.
`bus-vector-profile.json`	The `bus-vector-1` figure profile of 13.4.3: permitted elements, forbidden elements with reasons, and the constraints.
`bus-haystack-map.json`, `bus-brick-map.json`	Machine-readable form of Annex G. Generated from the source ontologies, not hand-written.

All four schema documents use the [JSONSCHEMA] 2020-12 dialect and resolve cross-references by `$id`. A validator **MUST** be configured with all four as a resource registry; they are not self-contained individually.

> NOTE The `$id` values resolve to a placeholder authority and will be reassigned when a governance body exists. Implementations **MUST** treat them as opaque and **MUST NOT** fetch them at validation time.

F Open issues and decisions taken

Informative. Recorded so review can be directed where it is wanted. The companion Decision Record gives the reasoning behind each.

F.1 Known gaps

Four gaps recorded here against v0.2 are closed, in whole or in part, by this version: ports (9.4), networks (8.10), the neutral vector format (13.4.3), and the design-versus-deployed distinction that was the narrow part of control-logic extraction (10.5). A fifth closes outside this document: time-series history, deferred since v0.1, is specified by BUS-2, and the `full` profile is claimable without a dispensation for the first time (2.3). What follows is what remains.

- **Extraction of deployed control logic** — v0.3 makes the distinction between design intent and running code expressible and auditable (10.5). It does not extract anything. A sequence is still referenced rather than read out of the controller, ASHRAE 231 is still design-time, and extraction still has no precedent in any standard — neither Project Haystack nor Brick has a control-logic construct of any kind. This remains the largest gap between this document and the charter's reconstruction promise.
- **Analytic rule portability** — the extension interface attaches a result. It cannot carry the rule that produced it, and the boundary between a portable rule and a proprietary method is genuinely unclear.
- **Signature formats** — narrowed in this version to the minimum necessary surface: 19.2-1 fixes what is signed — the manifest bytes, which bind every digest and the actuation declaration — and names detached [JWS] or CMS as the envelope. Algorithm agility and key management remain outside the standard, deliberately; a building standard that adjudicates PKI will be wrong about it within the decade.
- **Revision under concurrent editing** — narrowed in this version from open to a stated assumption with a forward path: 6.6.1 names the single-writer assumption, requires serialize-or-fork under concurrent authority (test EX-11), and sketches the minimal merge rule a future MINOR version is expected to define. The merge rule itself remains open.
- **Package scale** — nothing bounds entity count or decompressed size. Test RT-14 measures scale without setting a threshold. A real portfolio export will find the limit before the standard does.

Three constructs were considered for this version against the two mature semantic models and declined. They are recorded because a reviewer can argue with a decision and has nothing to argue with in an omission.

- **Weather stations, occupancy vocabulary, EV charging, synthetic and machine-learned models, point groups — declined** — each is expressible today as a `bus:Asset` with relationships, property definitions and verbatim classifications, and none meets the promotion bar of 15.5: implemented by several independent parties, and about the building rather than about a product. Where an implementer finds one of these

genuinely inexpressible rather than merely unnamed, that is the report 15.5 is waiting for.

- **Prototype and expected-content templates — declined** — Project Haystack carries prototype structure on 47 defs — what an instance of a kind is expected to contain — and it is the best idea in Haystack that this model lacks. It is still declined. It is a conformance-profiling mechanism rather than an information model: it states what a good export contains, which is Annex A's job. Adopting it would put this document in the position of asserting that an air handler without a discharge temperature sensor is defective, which is an equipment-taxonomy judgment and 5.3.3 forbids it. The right home is a future test in group EX, expressed against the completeness declaration rather than against a class.
- **Quantity dimension vectors — adopted after this document first declined them.** Every one of the 29 quantities in Annex C now carries an SI dimension vector in `bus-units.json`, with a kind field so that an angle is not compatible with an efficiency and real power is not compatible with reactive. This entry originally recorded the decline; the vectors were adopted in the same version because test TS-5 needed them — a series' unit and its point's unit are compared by quantity family, and a comparison with no algebra behind it is a spelling check.

F.2 Decisions that merit challenge

- **No equipment taxonomy** — defended in 1.4 and 5.3.3. The counter-argument is that a package whose classifications are all in vocabularies the receiver does not implement is portable in form and not in substance. Property definitions (7.8) are the mitigation; whether they are sufficient is untested.
- **Relationships as entities** — roughly a third of package size. A compact form for relationships carrying no provenance is worth considering.
- **The size of the symbol role vocabulary — 52 roots and 75 named leaves** — small enough to ratify, possibly too small to be useful. The fallback rule means the cost of being wrong is low, but the first implementer to hit the ceiling should say so.

> NOTE This entry said "forty symbol roles" through v0.2, when the vocabulary file already held 52 roots. It was one of five places where the prose had drifted from the data it describes. Both figures above are now read from `bus-symbol-roles.json` when this document is built, which is the only fix that stays fixed.

- **No point-function taxonomy** — Brick 1.4.4 carries 938 point classes and Haystack composes the meaning of a point from tags. This document has 7 `pointKind` values and will not grow them into a taxonomy. 1.4 and 5.3.3 forbid it; the 7 kinds already span Brick's branches with `derived` and `meter` besides; and the meaning of a point is carried by `quantityKind` against a property definition (7.8.1), by `bus:measures` naming the thing observed, and by verbatim classifications (7.7). Adopting a point taxonomy would duplicate two mature vocabularies, require this document to adjudicate between them — which 7.7 explicitly refuses to do — and age at the speed of the faster-moving of the two: Brick has deprecated 185 of its own classes, 112 of them into another organization's namespace. The counter-argument is in the first entry of this subclause and is not dismissed.

- **No growth of the symbol role vocabulary without implementer evidence** — 32 of the 52 roots have no leaves at all, so the right answer to "possibly too small" is leaves under existing roots rather than more roots — and Requirement 13.4-1's truncation rule means a leaf costs a receiver nothing, since an implementation supporting only roots is conforming. F.3 asks implementers how many roles were unavailable. Until one answers, growth is guesswork, and Annex B names the failure mode it would produce: a three-hundred-role taxonomy project that never ships. Figure mode is the designed escape for anything outside the vocabulary, and it was unusable until this version specified `bus-vector-1` — which is why the correct response to role pressure in v0.3 was 13.4.3 and not more roles.
- **Free-text knowledge notes** — defended in 8.9 on the grounds that structure encourages discarding what does not fit. The opposite argument, that free text cannot be reasoned over and will be ignored, is also strong.
- **Vendor-tree media type** — correct today and wrong as soon as a governance body exists. The transition needs specifying before adoption, not after.

F.3 What review is asked for

Implementers are asked to attempt an export from a real system at profile `core-intent`, and separately to attempt a graphics export, and to report: what the exporting system holds that the model cannot express; which completeness reason codes were needed and which were missing; how many symbol roles were unavailable; and what the export could not do without vendor assistance.

The last question is the one the charter cares about most and the one a schema cannot answer.

Implementers attempting a graphics export are asked for four figures in particular: the total number of distinct symbol roles the source system's drawings require; how many of those had no match in the tier-one vocabulary even after truncation; the five to ten most frequently needed missing leaves, grouped under existing roots where possible; and whether figure mode was an acceptable substitute for each miss, or whether the loss of design-system control was material. Growth under existing roots is preferred over new roots, and a root with no leaves is not a defect — it is a placeholder that keeps the truncation rule usable. Answers in this form are a dataset; answers in any other form are a mood.

G Mapping to Project Haystack and Brick

Informative, except that the rules stated for the classifications field, for ports on import, and for the medium of a flow relationship are normative. The machine-readable forms are published as bus-haystack-map.json and bus-brick-map.json, and every count in this annex is computed from them. Where a rule here and a rule there differ, the file governs: it is generated from the ontologies and this text is not.

G.1 What this annex is and is not

This annex maps no taxonomy onto any other. This document defines no equipment taxonomy (5.3.3) and preserves external classifications verbatim (7.7), so a Haystack def and a Brick class reach a package as a `classification` and never as a `type`. Nothing here reduces Brick's 1,433 classes to this model's 26, because that is not the shape of the problem.

The problem it solves is narrower and more consequential. 7.7 requires classifications to be preserved verbatim and does not say *which* classifications. Two implementers exporting the same Haystack model can both obey it and produce different packages: one emits a tag per marker, another emits the composed name it displays in its own interface, a third emits the equipment def and drops the rest. Verbatim needs a referent. This annex supplies one.

	PROJECT HAYSTACK	BRICK SCHEMA
Version this annex is written against	4.0.0	1.4.4
`classification.system`	`haystack`	`brick`, `brick-tag`, `rec`
`classification.version`	`4.0.0`	`1.4.4`, and omitted on a `rec:` code
Vocabulary size	719 defs, of which 302 markers and 162 declared conjuncts	1,433 classes, of which 185 deprecated
Entity types the crosswalk places	142, of which 7 reach no BUS type	56 branches covering 1,431 classes

	PROJECT HAYSTACK	BRICK SCHEMA
Predicates mapped	33, of which 5 reach no BUS predicate	37, of which 23 reach no BUS predicate
Ports recoverable	28	0 — the model has none (9.4)
Constructs that reach BUS in no form	11	9
BUS constructs the source cannot express	10	12

G.2 Classifications from Project Haystack

An exporter from Project Haystack **MUST** use `system: "haystack"` and `version: "4.0.0"`. The code is the def symbol exactly as authored, without the leading caret: `ahu`, `chilled-water-plant`, `discharge`. Not the library-qualified form, not the URI, not a display name.

Haystack's meaning lives in a tag set rather than in a name, and that is what makes the rule necessary. There is no def for a discharge air temperature sensor. The four applied tags each name a def, and the largest declared conjunct they match is `air-temp`, which covers two of them. An exporter that synthesizes a code for the composition invents a term the source vocabulary does not contain, which is exactly what Requirement 7.7-1 forbids.

- R1. An exporter **MUST** emit one classification per applied marker tag that names a def. Tags that name no def are not classifications; they are property values (7.8) or named fields.
- R2. An exporter **MUST** additionally emit one classification for the maximal matched conjunct, where one exists: the declared conjunct whose parts are all present in the applied tag set and which has the most parts. Where two conjuncts tie at the maximum, all tied conjuncts **MUST** be emitted, because the match is genuinely ambiguous and picking one silently invents a fact.
- R3. The maximal-conjunct match **MUST** exclude the port conjuncts. A port is an interface assertion, not a class: emitting 'air-input' as a classification asserts that an air handler is a kind of air inlet. Ports go to Asset.ports via portMap.
- R4. An exporter **MUST NOT** synthesize a code that is not a def symbol. There is no def for a composed point such as 'discharge air temp sensor' — the maximal match for those four tags is a two-part conjunct — and inventing one violates 7.7-1's verbatim requirement.
- R5. Classifications **MUST NOT** be deduplicated against each other or against the entity's type. An entity may carry the marker code and the conjunct code that contains it; 7.7 permits several from one system and does not adjudicate between them.

- R6. The choice-slot options a point carries (pointFunction, ductSection, and the other choice slots) are marker tags and are emitted under R1 like any other. BUS does not model the slot, only the option.

APPLIED TAGS	MAXIMAL MATCHED CONJUNCT	CLASSIFICATIONS EMITTED
`discharge` `air` `temp` `sensor`	`air-temp`	`air`, `air-temp`, `discharge`, `sensor`, `temp`
`chilled` `water` `plant`	`chilled-water-plant`	`chilled`, `chilled-water-plant`, `plant`, `water`
`elec` `meter`	`elec-meter`	`elec`, `elec-meter`, `meter`
`air` `input`	(none)	`air`, `input`

> NOTE Haystack's authored form is not its normalized form. contains, tags and quantities are computedFromReciprocal and never appear in .trio source; inherited tags are derivable but not materialized. An exporter reading a Haystack server through op:defs therefore sees a larger model than one reading source. A conformance claim **MUST** state which was consumed.

G.3 Classifications from Brick

An exporter from Brick **MUST** use `system: "brick"` and `version: "1.4.4"`, and the code is the CURIE exactly as Brick spells it: `brick:Chilled_Water_System`, `rec:Room`, `tag:Discharge`. The prefix is part of the code, because one crosswalk emits 3 namespaces.

The rule that matters is deprecation. 185 of Brick's 1,433 classes are deprecated, 184 of them carrying a replacement, 112 of which point into the `rec:` namespace. The whole of the Location branch — 109 classes — is deprecated in favor of `rec:`, which is to say that Brick has delegated its spatial vocabulary to another ontology mid-life.

- R1. An exporter **MUST** emit one classification per asserted rdf:type in the brick: namespace, with system 'brick' and version '1.4.4'.
- R2. Where the class is deprecated AND carries brick:isReplacedBy, the exporter **MUST** emit the replacement as a second classification, with system 'rec' where the replacement is in the rec: namespace and 'brick' where it is in brick:. It **MUST NOT** emit the replacement instead of the original: the original is what the source graph actually said, and 7.7-1 forbids translating.
- R3. classification.version **MUST** be omitted on a rec: code. The shipped Brick distribution states no REC version, and inventing one is worse than omitting it.
- R4. Superclasses **MUST NOT** be emitted as additional classifications. The asserted type is the assertion; the ancestry belongs in abstractTypes (8.2), which is BUS's mechanism for exactly this and is not a classification.
- R5. Tags asserted with brick:hasTag are emitted with system 'brick-tag'. Tags reachable only through the class, via brick:hasAssociatedTag, **MUST NOT** be emitted:

they are ontology content, not instance data, and emitting them would put 1249 classes' worth of derived tags into packages that never asserted them.

- R6. An importer **MUST NOT** deduplicate the brick: code against the rec: code. They are different vocabularies making the same claim, and a receiver may recognize either.

Emitting both codes is not belt and braces. In the shipped distribution the 268 `rec:` classes are typed `rdfs:Class` and `sh:NodeShape`, not `owl:Class`. A tool that discovers Brick's hierarchy by asking for `owl:Class` — which is precisely how the 1,433 figure above is produced — sees 109 deprecated location classes and none of their replacements. A crosswalk targeting `rec:` alone is therefore invisible to a conventional Brick consumer, and one targeting `brick:` alone hands the receiver a vocabulary its own publisher has abandoned. 7.7 permits an entity to carry several classifications from one system and does not adjudicate between them, so both travel and the receiver decides.

> NOTE This is the clearest vindication of 5.3.3 in the whole exercise. Because this model never adopted a spatial taxonomy, Brick's abandonment of its own costs it nothing. A standard that had adopted one would be republishing.

G.4 Ports on import

A Haystack equipment def declares its ports in its own `is` list, as conjuncts of the form *substance*-input and *substance*-output. An importer **MUST** partition that list into genus and ports before emitting anything. It **MUST** emit each port conjunct as an entry in `Asset.ports`, with `name` equal to the conjunct symbol so that two independent importers produce the same name, and it **MUST NOT** emit a port conjunct as a `classification` or as an `abstractType` (8.2).

The hazard is concrete and easy to walk into. 34 defs declare a port in their `is` list, and all 34 of them owe their multiple inheritance entirely to it. A translator that turns every `is` edge into a classification asserts that an air handler is a kind of air outlet, which is not a statement Haystack makes and not one a receiver can undo.

Haystack's encoding cannot supply `ordinal` or `pairedWith`, because a port is a supertype there and a def can carry at most one port per medium per direction. An importer **MUST** leave both absent. Synthesizing an ordinal invents topology. This is the one place where v0.3 is strictly more expressive than the source and the crosswalk cannot fill the gap; it can only decline to fabricate.

The Brick port rule

An import from a Brick model **MUST** leave `Asset.ports` absent. Brick 1.4.4 has no port model to populate it from, and an importer **MUST NOT** synthesize a port from `hasInputSubstance` or `hasOutputSubstance`: those give a substance and no count, no name, no ordinal and no pairing, so a chiller would acquire exactly one chilled-water outlet whether it has one or four.

> NOTE The port prior art in the Brick distribution is `extensions/223p.ttl`, which is ASHRAE 223P and is not in Brick's own imports. A model carrying 223P alongside Brick can populate `Asset.ports`; a plain Brick model cannot. 9.4 credits 223P for the shape.

G.5 The medium of a flow relationship

Brick puts substance on the node rather than on the edge: `hasInputSubstance` and `hasOutputSubstance` have SHACL domain `brick:Equipment`. Requirement 9.3 requires a `medium` on every `bus:feeds`. A chiller with chilled water out one side and condenser water out the other therefore has two `feeds` edges in the source and no statement anywhere of which carries which.

The medium fallback rule

Where the subject asset has exactly one output substance and that substance maps to a BUS medium, an importer **MAY** use it. In every other case the importer **MUST** emit `medium: "other"`, **MUST** carry the source substance terms verbatim in `Relationship.qualifiers`, and **MUST** declare the `relationships` domain partial with reason `notSupportedByExporter` (16.6). It **MUST NOT** omit the medium: 9.3 requires one, and a `bus:feeds` without a medium is an invalid package.

This is the only route by which a conforming import from Brick could otherwise produce an invalid package, and closing it needs no schema change: `qualifiers` is already an open map and `medium` already has `other`. 14 of Brick's 60 substances reach this model only as `other` in any case, among them glycol, oil, ice and several gases, and the source term in `qualifiers` is what a later reader has to work with.

Brick also draws positional distinctions inside the substance — supply air, return air, entering and leaving chilled water. Here the position is a property of the port and not of the medium (9.4), so those widen onto the base medium and survive in the classification. A source that carried port structure would let a receiver recover the position; Brick does not carry one.

G.6 Identity, provenance, and what does not cross

Two obligations catch every import from either model, because neither model has the construct this document requires.

- **Identity** — a Haystack ref and a Brick IRI are not BUS identifiers under 6.1. An importer mints a BUS identifier and carries the source identifier in `externalIds` with its scheme (6.3). Discarding it is the one thing that makes a second import impossible to reconcile with the first.
- **Provenance** — neither model carries provenance on an assertion, and `method` is **REQUIRED** (14.1). An importer **MUST** assert `method: "imported"`, naming itself as the agent and the source model as the source, rather than leaving provenance absent. It is

a thin claim and it is an honest one: the receiver can see that nothing behind this assertion was surveyed.

Beyond that, 10 constructs of this model cannot be expressed in Haystack and 12 cannot be expressed in Brick, and the two lists are largely the same list: operational intent, sequences, alarm definitions, events, knowledge notes, the presentation model of clause 13, provenance on every assertion, and the completeness declaration itself. An export from this model into either **MUST** declare the affected domains partial with reason `notSupportedByExporter`.

> NOTE That is not a criticism of either model. Both are semantic models of what a building is; this is a portability standard for what an owner is owed when they leave. The overlap is what makes a crosswalk possible and the non-overlap is the reason this document exists.

Take my building with me.

Not a data dump. Not a proprietary backup. A portable, durable, reconstructable understanding of the building the owner already paid to create.

END OF BUS-1 · VERSION 0.3 · WORKING DRAFT

H Mapping to object stores and property graphs

Informative as an annex; the MUST and SHOULD rules within it are normative for a consumer that claims conformant import into an object or graph store. New in v0.3, from the first consumer review of this standard — an ontology engineer asked whether a package would hydrate a platform ontology cleanly, and the four decisions this annex fixes are the four that review found left open.

H.1 Why these rules are fixed here

A package maps onto an object store almost mechanically: model documents are collections, entities are objects, `id` is the primary key, and the closed predicate set is the link vocabulary. But four decisions fall outside what the schemas can express, and a consumer that decides them differently from the next consumer produces a different store from the same package — which forfeits, on the last step, the determinism every earlier clause pays for. Annex G fixed the rules for reading foreign vocabularies in; this annex fixes them for writing BUS out.

> NOTE Nothing here obliges an exporter. These rules bind the consumer, and only a consumer that wants two independent imports of one package to agree.

H.2 Links

A store with typed links **SHOULD** realize every relationship twice: as a direct link of its predicate, for traversal, and — where the relationship carries provenance, qualifiers, a validity interval, or ports — as a link object holding that metadata, since most stores cannot attach properties to an edge. A consumer **MUST NOT** silently discard relationship metadata to fit an edge-only model: dropping the validity interval turns *was connected until the 2019 retrofit* into *is connected*, which is a false statement about the building.

Link cardinality comes from the registry (`bus-predicates.json`, 9.2), and its ADVISORY grading governs: a consumer **SHOULD** index to the hinted shape and **MUST NOT** refuse or truncate instances that exceed it. The registry's notes name the known cases — dual feeds, check meters, consolidations — and a store that cannot hold them is misrepresenting the buildings where topology matters most.

H.3 Denormalization

5.3.1 keeps every fact in one place, and stores tuned for interactive query will want some facts copied — a located-in space key on the asset object, not a two-hop traversal. Materialized properties of that kind are a cache, and **MUST** behave like one: derived from relationships by rule, regenerated on every synchronization, and never exported back as intrinsic properties of the entity. A re-export that turns a cached copy into an asserted fact manufactures an assertion nobody made, which is the 5.3.5 failure in a new coat.

H.4 Open vocabularies in closed stores

Enumerations are open (7.5) and store enum types are usually closed. A consumer **MUST** land open-enumeration values as strings — or an equivalent type that admits unknown

members — rather than a closed enum that would refuse the next MINOR version's token, which is the IM-4 failure relocated into the store. Relationship `qualifiers` land as structured values preserved whole; a consumer that indexes some qualifier keys **MUST** still retain the ones it does not index.

H.5 Time series and quality

Stores whose series type is timestamp-and-value have nowhere for the quality column of BUS-2. A consumer **MUST NOT** silently drop rows whose quality it cannot carry. It **SHOULD** carry quality as a parallel series per point; where it cannot, it **MUST** record per-series counts of `uncertain`, `bad` and `missing` rows on the series object, so that *the store holds less than the package* is a stated fact rather than a discovery.

H.6 Provenance

Entity-level provenance lands as structured properties on the object. Property-level provenance (7.8) usually has no native home; a consumer **SHOULD** spill it to provenance records linked to the object, and where it drops it instead **MUST** say so in whatever completeness statement its import produces. Losing who-asserted-this silently is the one loss this standard was built to make impossible, and an import that reintroduces it has not imported the package — it has imported the data and discarded the understanding.

H.7 What not to import

The presentation model does not map to a store and is not meant to: drawings, elements and figures are rebuilt in the receiving system's own display layer (13.9). Two parts of it are data a consumer should still take. Bindings carry the display logic an owner paid for — state maps and range maps from element to point — and translate directly into widget configuration. And a georeferenced drawing (13.3) places every element with geometry into the named CRS, which makes the drawing's contents spatial data even where its artwork is ignored.