THE BUILDING UNDERSTANDING STANDARD

BUS-1

Core Information Model, Portable Presentation, and Exchange Package

Part 1 of the Building Understanding Standard

Version 0.2 · Working Draft · July 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

Status of this document

This is version 0.2 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 0.2 consolidates two independently drafted v0.1 specifications and the working sessions that reconciled them. The substantive change from v0.1 is that **portable presentation is now part of the core** rather than deferred, together with alarm definitions, events, a party model, and a requirement that every property carry a definition. Every design decision, including the alternatives rejected, is recorded in the companion Decision Record.

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.

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

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 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

Domain	In v0.2	Deferred to
Time-series history	Points declare whether durable history exists, its extent, and its nominal interval (8.8). Events, which are not time series, are core (clause 12).	BUS-2, time series
Structured knowledge	`bus:KnowledgeNote` carries free text with a kind, an author, a review state, and typed attachment predicates (8.9).	BUS-2, structured knowledge types
Vector figure format	Elements may carry a `figure` payload. The neutral vector format `bus-vector-1` is named but not specified.	BUS-2, figure format

Domain	In v0.2	Deferred to
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 figure format is the one place v0.2 knowingly names something it does not define. An element in figure mode is portable in principle and not yet portable in practice. Where a symbol role exists, use it.

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.

Target	What it is	How it is tested
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

v0.2 normatively defines `core`, `core-intent`, and `core-intent-knowledge`. An implementation **MAY** claim `full` for presentation only if it also declares `timeSeries` as absent with a reason of `deferredToLaterVersion`; the time-series domain is not specified in this version and cannot be tested.

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.

Reference	Title and note
[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.
[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"
}
```

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.

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.

Property	Requirement
`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.*

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.

`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`.
`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`.
`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`, `network`, and `parameters`.

```
"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

v0.2 does not carry time-series payloads. It carries the declaration that history exists, which is what a receiving system needs 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).

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-2. A premature structure would be worse than free text, because it would encourage implementers to discard what does not fit.

9 Relationships

AGREED *The predicate set grew from 27 tokens in v0.1 to 42: twenty inverse pairs, symmetric connectedTo, and the deliberately weak references. 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.

Predicate	Inverse	Meaning, subject to object
`bus:contains`	`bus:containedIn`	Spatial or organizational containment. Transitive.
`bus:hasPart`	`bus:partOf`	Compositional part of an asset. Transitive. Equipment to component.
`bus:hasMember`	`bus:memberOf`	Grouping without containment. Not transitive. System to equipment, zone to space.
`bus:locatedIn`	`bus:hasLocation`	Physical placement of an asset in a space.
`bus:serves`	`bus:servedBy`	Delivery of a service to a space, zone, or asset.
`bus:feeds`	`bus:fedBy`	Directed flow of a medium. Requires `medium`.
`bus:connectedTo`	(symmetric)	Undirected physical connection, qualified by medium and port.
`bus:powers`	`bus:poweredBy`	Electrical supply. What an operator traces during an outage.
`bus:measures`	`bus:measuredBy`	Observation. Point to the thing observed.
`bus:commands`	`bus:commandedBy`	Actuation by a point. Point to the thing acted upon.
`bus:controls`	`bus:controlledBy`	Governance by a controller, sequence, or strategy.
`bus:hosts`	`bus:hostedBy`	Provision of an addressable home for a point. Device to point.
`bus:appliesTo`	`bus:hasApplicable`	Attachment of intent, an alarm definition, or a result to what it concerns.
`bus:implements`	`bus:implementedBy`	A sequence or schedule realizes an intent.
`bus:explains`	`bus:explainedBy`	Knowledge accounts for why something is as it is.
`bus:supports`	`bus:supportedBy`	Knowledge or evidence supports an assertion.
`bus:justifies`	`bus:justifiedBy`	A rationale justifies a decision or change.
`bus:derivedFrom`	`bus:derives`	Dependence of a calculated value on its inputs.

Predicate	Inverse	Meaning, subject to object
`bus:documentedBy`	`bus:documents`	Attachment of a resource to what it describes.
`bus:represents`	`bus:representedBy`	A drawing or element depicts a building entity.
`bus:replaces`	`bus:replacedBy`	Lifecycle succession.
`bus:references`	(none)	A link that is real but whose nature is not modeled. Deliberately weak.

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

OPEN *The port model is unresolved.* `subjectPort` *and* `objectPort` *are free strings and are a placeholder, not a model.*

Without a port model it is not possible to say which branch, which side of a valve, or that a coil has a supply and a return. This is the largest remaining expressiveness gap in the information model. It is also the change most likely to turn a clean specification into an ontology, and it should be decided against a real hydronic plant rather than in the abstract.

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.
`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`, or `vendor` — and carries either inline source or a resource reference.

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.

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.

Field	Requirement
`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 deferred history domain.*

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 named and not yet specified; 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.

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: fifty-two roots with seventy-five named leaves.

13.4.3 Figure mode

PROVISIONAL *The neutral vector format is named and not specified. An element in figure mode is portable in principle and not yet portable in practice.*

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 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.

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.
`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.
`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] and the vendor-tree name **MUST** continue to be accepted.

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 remain deferred; an exporter can therefore be complete on one and absent on the other, which is the honest statement.

> **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.

Reason code	Meaning
`excludedByOwner`	Withheld at the owner's instruction.
`excludedForConfidentiality`	Withheld for confidentiality, including personal data. 19.3.
`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.

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.

> **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 three 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.

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. An exporter **MUST** be able to exclude command bindings on request, declaring the exclusion with reason `excludedForSafety`.

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`.
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

Component of an exit package	Can be serialized by	Compelled by
Historical time series	nothing, as a package	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. Fixtures and a reference runner are deferred to the reference tooling release, and until they exist all results are self-declared.

A.1 Structure

Four groups, matching the four conformance targets of 2.2. 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.
PK-6	MUST	**Path safety** (BUS-1 16.4). 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-1 6.2). Collect every entity id in the package. *Pass:* No id occurs twice.
PK-8	MUST	**Referential closure** (BUS-1 9.2). Resolve every subject, object, deviceRef, targetOf, appliesToRef, scheduleRef and ChangeRecord subject. *Pass:* Every reference resolves to an entity present in the package, or to an entity explicitly declared external in the manifest.

ID	Grade	Test and pass criterion
PK-9	MUST	**Extension declaration** (BUS-1 15.3). 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-1 16.6). 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-1 2.3). 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-1 7.8). 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-1 7.8). 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-1 13). 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-1 16.7). 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.
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.

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-1 6.4). 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-1 8). 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-1 9). 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-1 10). 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-1 14). 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-1 15.4). 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-1 15.4). 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-1 16.5). 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-1 8.6). 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-1 8.9). Compare every KnowledgeNote and its attachment relationship. *Pass:* Body text, noteKind, observedAt and the appliesTo relationship survive unaltered.
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.

ID	Grade	Test and pass criterion
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.
RT-21	MUST	**Property definitions survive** (BUS-1 7.8). 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-1 11). 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.

ID	Grade	Test and pass criterion
RT-23	MUST	**Events survive with their subjects** (BUS-1 12). 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-1 6.6). 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.

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.

A.5 Group EX — Exporter behavior

Target: exporter.

ID	Grade	Test and pass criterion
EX-1	MUST	**Exports without a live vendor dependency** (BUS-1 2.4). 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-1 6.4). 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-1 6.4). 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-1 2.4). 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-1 2.2). 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-1 2.4). 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.
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.

A.6 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 fifty-two roots and seventy-five 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.

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.
	`heatExchanger`, `tank`, `humidifier`	Plant items that are neither a coil nor a vessel.

Family	Root	Leaves and notes
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.
	`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.

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.

C Unit codes

Normative. C.3 is informative.

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

Code	Unit	Quantity
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
tonRef	tons of refrigeration. No equivalent exists in any external registry surveyed. One ton of refrigeration is 12000 Btu/h exactly.	power
hp	horsepower	power
Wh	watt hour	energy

Code	Unit	Quantity
`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
`ft`	foot	length
`in`	inch	length
`m2`	square meters	area
`ft2`	square feet	area
`m3`	cubic meters	volume

Code	Unit	Quantity
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
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	concentration

Code	Unit	Quantity
ug/m3	micrograms per cubic meter	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.

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]
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

BUS code	Equivalent	BUS code	Equivalent
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	%
ppm	[ppm]	ppb	[ppb]
ratio	1	count	1
mg/m3	mg/m3	ug/m3	ug/m3

BUS code	Equivalent	BUS code	Equivalent
`gPerKg`	`g/kg`	`grPerLb`	`[gr]/[lb_av]`
`rpm`	`{rpm}`	`lux`	`lx`
`fc`	`[ftc]`	`dB`	`dB`
`dBA`	(none defined)	`deg`	`deg`
`rad`	`rad`		

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.

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.
`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.
`presentation.json`	38	Two drawings, eight layers, seventeen elements, eleven bindings of which two are writes.
`provenance.json`	2	Change records for the 2017 economizer modification and the 2019 rebalance.

The package declares profile `core-intent` and declares five 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, and a fan command write at priority 8 referencing the fire interlock. 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 completeness declaration** — five 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 thirteen 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 751 such checks.

> **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 fifteen document types.
`bus-manifest.schema.json`	The manifest, including completeness across twelve domains, the actuation declaration, extension and dictionary declarations.
`bus-symbol-roles.json`	Machine-readable form of Annex B. Fifty-two roots and seventy-five 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. Fifty-six tests across four groups.

All three schema documents use the [JSONSCHEMA] 2020-12 dialect and resolve cross-references by `$id`. A validator **MUST** be configured with all three 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 for version 0.3

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

F.1 Known gaps

- **Ports and connection points** — the largest remaining expressiveness gap. `subjectPort` and `objectPort` are free strings. Without a model it is impossible to say which branch, which side of a valve, or that a coil has a supply and a return. Should be decided against a real hydronic plant.
- **Networks and drivers** — `binding.network` is a free string. Integration and driver configuration appears in this document's own gap table (20.3) as something no standard carries — and this one does not carry it either.
- **The neutral vector format** — `bus-vector-1` is named in 13.4.3 and not specified. Figure mode is portable in principle and not in practice until it is.
- **Extraction of deployed control logic** — sequences are referenced, not extracted. ASHRAE 231 is design-time. Extraction has no precedent in any standard and is 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** — the manifest admits signatures and constrains nothing. This matters more now that actuation travels, and should be settled before any package is relied on as evidence.
- **Revision under concurrent editing** — revision is defined for a single writer. Two systems editing the same building and exchanging packages is not yet specified.
- **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.

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.
- **Forty symbol roles** — 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.
- **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.

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 0.2 · WORKING DRAFT