

THE BUILDING UNDERSTANDING STANDARD

BUS-2

Time Series

Part 2 of the Building Understanding Standard

Version 0.3 · Working Draft · August 2026

The vendor keeps the historian. The owner keeps the history.

THE BOUNDARY TEST, APPLIED TO HISTORY

WORKING DRAFT FOR IMPLEMENTER REVIEW – NOT FOR PROCUREMENT USE

Status of this document

This is version 0.3 of BUS-2, the second part of the Building Understanding Standard. It is a working draft published for implementer review, and it is the first version of this part: the time-series domain was named by BUS-1 v0.1, declared in every completeness declaration since, and specified by nothing until now.

BUS-2 specifies the last of the twelve substrate domains, which is the fact that matters most about it. Through v0.2 the `full` conformance profile could be claimed only under a standing dispensation — declare `timeSeries` absent, cite `deferredToLaterVersion`, and claim the profile anyway. That dispensation is withdrawn (2.2). A `full` claim is now a claim about all twelve domains, and every one of them has a specification to be tested against.

This part is short because most of what it needs already exists. Datatypes, identity, the package container, digests, the completeness declaration, canonical form, and the conformance machinery are all BUS-1's, used by reference. What BUS-2 adds is exactly two things: an entity that declares a series, and a file format the samples travel in. The design decisions behind both — and the formats rejected, with reasons — are recorded in the clauses that make them, because a reviewer can argue with a decision and has nothing to argue with in an omission.

Maturity labels, editorial conventions, requirement keywords, and the boundary test are BUS-1's and are not restated here. Where this part cites a clause number without a part name, the clause is in this part; clauses of Part 1 are cited as BUS-1.

Contents

- 1 Scope and relationship to BUS-1
- 2 Conformance
- 3 The series model
- 4 The sample format
- 5 Packaging and integrity
- 6 Completeness and honesty
- 7 Canonical form and comparison
- 8 Security considerations
- 9 References

1 Scope and relationship to BUS-1

PROVISIONAL First version of this part. Every construct here is new and expected to be challenged by the first real historian export.

1.1 Purpose

This document specifies how historical time series travel in an exchange package: a declaration entity, `bus:Series`, and a sample format, `bus-csv-1`. Its object is the same as BUS-1's — continuity. A building accumulates years of what it did, and that record is the evidence behind every retro-commissioning study, every energy baseline, every fault investigation, and every dispute about whether the chiller was always this bad. Today it lives in the vendor's historian and leaves with the vendor.

Both mature semantic models punt exactly here, and in the same direction. Project Haystack treats history as a capability of the point — the `his` marker — addressed by the point's own identifier through a live `hisRead` operation against a running server. Brick's `ref:TimeseriesReference` is an external database pointer: a timeseries identifier and the name of the store it lives in. Both answers assume the system of record is still running and still cooperative, which is precisely the dependency this standard exists to remove. Neither carries the data. This part carries the data.

1.2 Relationship to BUS-1

BUS-2 is not free-standing. It uses BUS-1's datatypes (BUS-1 7), identity rules (BUS-1 6), entity envelope (BUS-1 8.1), package container and integrity machinery (BUS-1 16), completeness declaration (BUS-1 16.6), and conformance targets and grading (BUS-1 2, Annex A). A conformance claim against this part is a claim about a BUS-1 package, and nothing here modifies a BUS-1 requirement.

The seam between the parts is BUS-1 8.8. A point declares *that* durable history exists and over what extent; this part specifies how the history itself travels. The two statements are made by the same exporter in the same package and are checked against each other (test TS-7).

1.3 The division of labor: why samples are not entities

The single structural decision in this part is that bulk samples are NOT JSON entities. The declaration of a series is an entity; the samples are a file inside the package, digested in the manifest like every other document and resource (BUS-1 16.5).

The decision is measured, not aesthetic. At 344,285 entities, JSON Schema validation took roughly a hundred seconds of a hundred-and-two second reference-implementation import, while every other stage took seconds. A single trended point at a fifteen-minute interval produces about 35,000 rows a year; a modest building trending two thousand points for a decade holds several hundred million. History is orders of magnitude more rows than the model is entities, and a format that pushed those rows through the entity envelope, the schema validator, and the canonicalizer would make

every package with real history unimportable in practice. The wire format would be correct and the standard would be ignored.

Requirement 1.3-1

Bulk samples **MUST NOT** be carried as entities in a model document. They travel as package resources, in the format of clause 4, declared by a `bus:Series` entity per clause 3. An importer **MUST NOT** schema-validate sample files as JSON; they are not JSON, and clause 4 defines their entire grammar.

NOTE The same measurement is why the package inspector runs schema validation as a separate opt-in stage. This part applies the lesson at the format level rather than the tooling level, which is where it stops being advice and starts being load-bearing.

1.4 What is out of scope

- **Analytics and forecasting** — a series is a record of what happened. What it implies belongs to whoever analyzes it, and the internal shape of an analytic result remains deferred (BUS-11.3).
- **Live streaming and query** — this is a package standard. Protocols for subscribing to, querying, or trickling telemetry exist and are not its business.
- **Compression beyond the container** — the ZIP container already offers DEFLATED (BUS-116.2), and repetitive timestamped text compresses well under it. A columnar or binary encoding is a MINOR-version question for a future `format` token, to be raised only with scale evidence in hand.

2 Conformance

AGREED The grading rule and the withdrawal of the v0.2 dispensation are settled. The tests themselves are as new as the part.

2.1 Requirement keywords and targets

Requirement keywords are used per BCP 14 [RFC2119] [RFC8174], capitalized exactly as in BUS-12.1. The conformance targets — package, exporter, importer, round-trip — are BUS-12.2's, and profiles remain floors, not ceilings (BUS-12.3).

2.2 The full profile

The `full` profile names the `timeSeries` domain, and through v0.2 that domain was specified by nothing, so BUS-12.3 carried a dispensation: declare it absent with reason `deferredToLaterVersion` and claim `full` anyway. This part withdraws it. The domain is specified, `deferredToLaterVersion` is no longer a true statement about it, and an absent `timeSeries` domain now requires a real reason under BUS-116.6 like any other absent domain — clause 6 says which reasons are honest where.

A claim of `full` is tested against the group TS tests of this clause in addition to the Annex A groups of BUS-1. An implementation already claiming a lower profile is unaffected: `timeSeries` is not a domain of `core`, `core-intent`, or `core-intent-knowledge`, and carrying series at those profiles is permitted and simply untested.

2.3 Group TS

Group TS tests a package, statically, with neither system present — the same discipline as group PK. The grading rule is BUS-1 Annex A's: a contradiction visible in the package is **MUST**, a judgment about what the exporter knew is **SHOULD**, and an importer never refuses on a **SHOULD**.

ID	GRADE	TEST AND PASS CRITERION
TS-1	MUST	Sample files conform to bus-csv-1 (BUS-2 4). Parse every file a bus:Series names against the bus-csv-1 grammar: encoding, line endings, the exact header, timestamp form and ordering, the value grammar, the quality vocabulary. <i>Pass:</i> Every file parses with no violations. UTF-8 without a byte order mark, LF line endings, the header row exactly 'timestamp,value,quality', RFC 3339 UTC timestamps with the Z designator in non-decreasing order, values within the grammar for the point's dataType, quality tokens from the fixed vocabulary or empty, and the value field empty exactly where quality is missing.
TS-2	MUST	Series references resolve (BUS-2 3.2, 5.2). For every bus:Series, resolve pointRef against the package's entities and resourceRef against manifest.resources and the archive. <i>Pass:</i> pointRef names a bus:Point in the package. resourceRef names a manifest resource whose path exists in the archive, so its digest was verified by PK-4 before any row was read. A series naming a point or a file the package does not contain is a package error.
TS-3	MUST	Row count is accurate (BUS-2 3.3). Count the data rows of every sample file and compare with the declaring series. <i>Pass:</i> Series.rowCount equals the rows in the file, excluding the header. A mismatch is the signature of a truncated or padded export, and the declared count is what makes it detectable — the same mechanism as the actuation declaration (PK-15).
TS-4	MUST	Coverage encloses the data (BUS-2 3.3). Compare the first and last timestamps of every sample file with the declaring series' coverage interval. <i>Pass:</i> Every row's timestamp lies within [coverage.start, coverage.end]. Coverage MAY be wider than the data — an exported period that held no samples — and MUST NOT be narrower.

ID	GRADE	TEST AND PASS CRITERION
TS-5	MUST	Series unit names the point's quantity (BUS-2 3.4). For every series whose point carries a unitCode, resolve both codes against the quantity families of bus-units.json. <i>Pass</i> : The series unit and the point unit name the same quantity. A series in kPa on a temperature point is a mis-wired export, visible to any receiver, and stating the unit in both places is what makes it detectable.
TS-6	SHOULD	Series unit code equals the point's (BUS-2 3.4). For every series whose point carries a unitCode, compare the two codes for equality. <i>Pass</i> : They are equal. Where they differ, the samples are in the series' unit, the disagreement is a true fact about the source system — a historian that kept degF after the point was re-configured to degC — and it is reported, never repaired by conversion (BUS-1 7.4-1). Graded SHOULD because a true statement about the source is not a contradiction in the package.
TS-7	MUST	Declared history agrees with carried series (BUS-2 6.2). Where the timeSeries domain is declared complete, compare each point's history declaration (BUS-1 8.8) with the series that carry it. <i>Pass</i> : Every point declaring history available has at least one series, and the union of its series' coverage encloses the declared from/to extent. Both statements are the exporter's own, in one package; a package that declares completeness its own point declarations contradict is internally inconsistent, which is what MUST grades. A domain declared partial or absent is not graded here.

Two tests live in BUS-1's own groups because that is where their targets are: RT-28 (**MUST**) requires that series survive a round trip byte-identical — clause 7 states the comparison rule it applies — and EX-10 (**SHOULD**) grades the honesty of the completeness declaration against the source system, which only certification can see.

NOTE A machine-readable form of every test in this clause is distributed in bus-conformance-tests.json, the same file that carries BUS-1's suite. One artifact, one grading vocabulary, one profile table.

3 The series model

PROVISIONAL

3.1 What a series is

A bus:Series declares one exported time series: the history of one point, in one unit, under one sampling discipline, in one sample file. It is the receiving system's reading order — everything it needs to know before opening the file, and everything it needs afterward to say what the file was.

A point **MAY** be declared by several series: successive coverage intervals of a long history split for size, or raw records alongside the rollups computed from them. Two series for the same point with the same sampling and aggregation **SHOULD NOT** overlap in coverage, because overlapping duplicates of the same records are a merge problem this part declines to create.

NOTE A series names its point through `pointRef`, intrinsically, and BUS-15.3.1 lists the constructs permitted to do that. A series joins them for the same reason an event does: it exists in order to describe its point's past, and would be meaningless without the reference. No `bus:hasSeries` relationship is created.

3.2 What a series declares

FIELD	REQUIREMENT
<code>pointRef</code>	REQUIRED . The <code>bus:Point</code> whose history this is. MUST resolve within the package (test TS-2).
<code>format</code>	REQUIRED . The sample format of the named file. This version defines <code>bus-csv-1</code> (clause 4); further formats are a MINOR change.
<code>resourceRef</code>	REQUIRED . Package-relative path of the sample file, exactly as listed in <code>manifest.resources</code> (5.2). <code>resources/series/</code> is the RECOMMENDED home.
<code>coverage</code>	REQUIRED . The closed interval the export covers, both ends stated, both UTC. 3.3.
<code>sampling</code>	REQUIRED . A kind (3.5) and, where the kind is <code>sampling</code> , the recording interval.
<code>aggregation</code>	Present where the rows are rollups rather than source records: the <code>function</code> and the <code>period</code> . Absence asserts the rows are what the source system recorded. 3.6.
<code>unitCode</code>	REQUIRED where the point carries one. The unit of every value in the file. 3.4.
<code>sourceTimeZone</code>	RECOMMENDED where the source system stored local time. The IANA zone converted from. 6.5.
<code>rowCount</code>	REQUIRED . The number of data rows in the file, excluding the header. 3.3.
<code>provenance</code>	RECOMMENDED , via the entity envelope. Which historian, read when, by what — the same questions BUS-114 asks of every assertion.

A series is an entity, so it also carries an identifier, a name, external identifiers — the historian's own series or trend-log id belongs in `externalIds`, and is frequently the only bridge back to an archive tier this package does not carry — classifications, and everything else in BUS-18.1. The schema requires `pointRef`, `format`, `resourceRef`, `coverage`, `sampling`, `rowCount`.

3.3 Coverage and row count are declared twice on purpose

Both `coverage` and `rowCount` restate something the file itself shows, and the redundancy is the mechanism. A declared value that must match the content is how truncation, padding, and tampering become detectable rather than plausible — the same pattern as the actuation declaration (BUS-116.7) and the reference dimension on a drawing (BUS-113.3). Tests TS-3 and TS-4 do the comparing.

Coverage **MAY** be wider than the data: an export of 2020 through 2025 for a point that was offline through 2021 declares the interval it exported, and the empty year is visible as the absence of rows. Coverage **MUST NOT** be narrower than the data, because rows outside the declared interval are rows a reader was told not to expect.

3.4 Units

unitCode states the unit of every value in the file, in the vocabulary of BUS-1 Annex C. It is stated on the series, once, rather than on each row — a row-level unit column would repeat one fact hundreds of millions of times, which is 5.3.1’s argument at data scale.

Requirement 3.4-1

The series unit **MUST** name the same quantity as the point’s unit (test TS-5), and **SHOULD** equal it (test TS-6). Where the two differ, the samples are in the series’ unit, the disagreement travels verbatim, and an implementation **MUST NOT** repair it by converting the samples.

The rule when they disagree was decided deliberately. A historian that kept recording in degF after the point was re-configured to degC in 2019 is a real system, and its history is really in degF. Forcing agreement would force conversion, and BUS-17.4-1 forbids conversion precisely because it is a silent, compounding loss that destroys the ability to tell a measured value from a computed one. So exact agreement is reported when absent, never manufactured; only a cross-quantity mismatch — pressure samples on a temperature point — is a package error, because that is not a fact about a historian, it is a mis-wired export.

3.5 Sampling kinds

sampling.kind states how rows relate to time: `sampled`, `cov`, `totalized`, `event`. The vocabulary follows the one piece of real prior art, Project Haystack’s `hisMode` — `sampled`, `cov`, and `consumption`, here renamed `totalized` — with `event` added for observations that are neither periodic nor change-driven.

KIND	WHAT A ROW IS
sampled	A reading taken at the regular interval the series declares. interval is REQUIRED , and a gap longer than it is visible as missing rows.
cov	A reading taken at the instant the value changed. The value holds until the next row. No interval exists to declare.
totalized	An accumulated quantity for the period ending at the row’s timestamp — meter consumption. The unit is the accumulated quantity’s, kWh not kW.
event	An irregular observation: a manually logged reading, a spot measurement. The timestamps are the only statement about timing.

KIND

WHAT A ROW IS

Requirement 3.5-1

An exporter **MUST** declare the kind the source system actually used and **MUST NOT** resample change-of-value or event data onto a regular grid in order to declare `sampled`. A resampled series is a computation presented as an observation, and a receiver has no way to recover what was actually recorded. A system that can only produce resampled output **MUST** declare it as a rollup under 3.6.

3.6 Rollups

An `aggregation` block states that the rows are computed rollups: a `function` — `mean`, `min`, `max`, `sum`, `first`, `last`, `count`, `integral`, `other` — and the `period` each row summarizes. `integral` is the honest name for most "average demand" rollups, which are time-weighted integrals rather than arithmetic means, and `other` requires the method be named in the series' description.

Absence of the block is itself an assertion: these rows are what the source system recorded. An exporter holding only rollups exports them as rollups, honestly declared, and clause 6 says what its completeness declaration owes. It does not pretend uniformity that the historian never had.

3.7 `Point.history`, and what this part changes about it

BUS-18.8's point-level declaration is unchanged: `Point.history` still states that durable history exists, over what extent, and at what nominal interval. It is the summary a receiver reads while deciding what it is missing, and it remains meaningful in packages that carry no series at all.

One field of it is superseded. `history.resourceRef` pointed at a resource with no statement of format, coverage, unit, or count — a locator without a declaration. It is deprecated in this version in favor of `bus:Series.resourceRef`, and per BUS-115.5 it **MUST** still be accepted; where both are present, the series entity is the statement and the point field is a label. The declaration (`available`, `from`, `to`, `interval`) is not deprecated, and test TS-7 holds it and the carried series to the same story.

4 The sample format

PROVISIONAL The grammar is normative and closed. The machine-readable form of every fixed fact in this clause is the `sampleFormat` member of `bus-timeseries.schema.json`.

4.1 What the format must survive

The samples are the part of a package most likely to be read in 2056 by someone with none of today's software: an energy baseline is evidence, and evidence outlives tools. So

the format is judged against four requirements. Readable with no library — a person with a text editor can verify what a tool asserts. Streamable — a receiver processes a decade row by row without holding it in memory. Appendable — an exporter writes a large series in one pass without seeking backward. And outside the schema-validation bottleneck that 1.3 measured.

`bus-csv-1` is CSV per [RFC4180], profiled hard enough to have exactly one form. Two candidate classes were considered and rejected, on the record:

- **Parquet and HDF5 — rejected** — both are binary formats whose readability depends on a maintained library, and both fail the 2056 test outright: a format that needs this decade’s toolchain to open is a vendor dependency with the vendor generalized. Their compression advantage is real and mostly recovered by DEFLATE over repetitive timestamped text; their query advantage is irrelevant to a package. A future binary `format` token remains a MINOR change if scale evidence ever demands one.
- **Inline JSON arrays — rejected** — samples as entity fields or document arrays would put hundreds of millions of rows through the JSON parser, the schema validator, and the canonicalizer. 1.3 is the measurement; this is the decision it forces.
- **One multi-series file with a series column — rejected** — a `seriesRef` column makes the manifest digest, the byte length, and the row count meaningless per series, obliges a receiver to partition the file before using it, and turns appending by independent writers into a merge. One series per file keeps every declared quantity checkable against exactly one artifact.

4.2 Container and encoding

- A sample file is plain text, UTF-8, no byte order mark. Media type `text/csv` with the `charset=utf-8` parameter, stated in the manifest resource entry.
- Line endings are LF. This is a deliberate deviation from [RFC4180]’s CRLF, and the reason is canonical form: a format with two permitted line endings has two encodings of the same series, and clause 7 needs there to be one. LF is what current tools write; the carriage return buys nothing.
- The final row ends with a newline, so files append and concatenate cleanly and the row count is the line count minus one.

4.3 The header

The first line is **REQUIRED** and is exactly `timestamp,value,quality` — these names, this order, lowercase, no spaces, no extra columns. A fixed header on a fixed schema is redundant, and deliberately so: a sample file separated from its package is still self-describing to the person who finds it.

4.4 Timestamps

Requirement 4.4-1

Every timestamp **MUST** be an [RFC3339] date-time in UTC with the uppercase Z designator. A numeric offset, including +00:00, **MUST NOT** be used. Fractional seconds are permitted where the source records them; trailing fractional zeros **MUST NOT** be written.

Storage is UTC; display is the building's. The building already declares its IANA time zone (BUS-18.4), so any reader can render wall-clock time correctly across every daylight-saving rule that applied over the series' life — which per-row offsets cannot do, and which per-row zone suffixes would pay for in bytes on every one of a hundred million rows. A source system that stored local time converts on export and records what it converted from (6.5).

Rows are in non-decreasing timestamp order. Equal timestamps **SHOULD NOT** occur; where a source system genuinely holds two records at one instant — change-of-value data at coarse clock resolution does this — they are preserved in source order rather than invented apart.

4.5 Values

- For a **number** point: an optional minus sign, decimal digits, an optional **.** fraction. No exponent, no leading +, no thousands separator, and **.** as the only decimal mark regardless of locale. 55.4, -3.25, 0.5 — never 5.54e1, never 55,4.
- For a **boolean** point: 0 or 1.
- For an **enum** point: the member code verbatim, as declared by the point's enumeration.
- For a **string** point: [RFC4180] quoting where the value contains a comma, a double quote, or a newline; otherwise unquoted.
- The value field is empty exactly where quality is **missing** (4.6). An empty value with any other quality, or a **missing** row carrying a value, is a grammar violation (test TS-1).

A gap is the absence of rows, visible against the declared sampling interval, and an exporter **MUST NOT** invent rows to fill one. A **missing** row and no row are different statements — *the historian recorded that it failed to record* versus *the historian recorded nothing* — and both age better than a plausible interpolation.

4.6 Quality

The quality column carries one of **good**, **uncertain**, **bad**, **missing**, or is empty.

TOKEN	MEANING
good	The source system recorded the sample as good.

TOKEN	MEANING
uncertain	Recorded, flagged doubtful: out of range, sensor fault suspected, manually entered.
bad	Recorded and known bad. Carried because deleting evidence is worse than flagging it.
missing	The source recorded that no value exists at this timestamp. The value field is empty.

An empty quality field means the source system recorded no quality, which is unstated — NOT good. This is the same rule as provenance confidence (BUS-1 14.1): writing `good` for a source that never assessed quality asserts something nobody knows, which is the failure Requirement 9.4-3 of BUS-1 names for ports and this clause refuses for samples.

4.7 Worked example

The supply air temperature of AHU-1 from the Cedar Street worked example (BUS-1 Annex D), one hour of it, with one failed scan and one doubtful reading. First the file:

```
resources/series/ahu1-sat-2026h1.csv

timestamp,value,quality
2026-06-18T05:00:00Z,13.3,good
2026-06-18T05:15:00Z,13.0,good
2026-06-18T05:30:00Z,,missing
2026-06-18T05:45:00Z,13.9,uncertain
2026-06-18T06:00:00Z,13.4,good
```

Then the series that declares it, in a `timeSeries` model document. Note that `rowCount` is 5 — the `missing` row is a row — and that the unit is the point's:

```
{ "id": "urn:uuid:...0a1",
  "type": "bus:Series",
  "localId": "AHU1-SAT-TREND",
  "pointRef": "urn:uuid:...061",
  "format": "bus-csv-1",
  "resourceRef": "resources/series/ahu1-sat-2026h1.csv",
  "coverage": { "start": "2026-06-18T05:00:00Z", "end": "2026-06-18T06:00:00Z" },
  "sampling": { "kind": "sampled",
    "interval": { "value": 15, "unitCode": "min" } },
  "unitCode": "degC",
  "rowCount": 5,
  "externalIds": [ { "scheme": "com.example.historian.trendId",
    "value": "TL-201-14" } ],
  "provenance": { "assertedBy": { "kind": "system", "name": "busref 0.3.0" },
    "assertedAt": "2026-06-18T06:10:00Z",
    "method": "measured",
    "source": { "kind": "system", "label": "Historian PRD" } } }
```

And the manifest entry that makes the file verifiable before a single row is read (5.2):

```
"resources": [
  { "path": "resources/series/ahu1-sat-2026h1.csv",
    "mediaType": "text/csv; charset=utf-8",
    "digest": "sha-256-k7r7wZyhDvFH70aMBcb13cfiRWXXQnPC4SDvSiLhTqU=",
    "byteLength": 183 } ]
```

5 Packaging and integrity

AGREED Nothing in this clause is new machinery. That is its content.

5.1 The series document

`bus:Series` entities travel in a model document with `documentType` `timeSeries`, conventionally `model/timeseries.json`, listed in `manifest.documents` and schema-validated like any other model document (test PK-3). Everything BUS-1 16.4 says about model documents applies, including splitting across files with `partOf`. The entities are few — one per series, not one per row — so the scale concern of 1.3 does not reach them.

5.2 Sample files are resources

Requirement 5.2-1

Every sample file **MUST** be listed in `manifest.resources` with its media type, its digest computed over the exact stored bytes, and its byte length, and every `Series.resourceRef` **MUST** equal the listed path. Sample files live under `resources/`, and `resources/series/` is **RECOMMENDED**.

This is clause 16 of BUS-1 doing its existing job. A receiver verifies every digest before ingesting any content (BUS-1 16.5, test PK-4), so a corrupted or truncated series is refused before a row of it is parsed; an undeclared file in the archive is a package error (PK-5); and a series naming a file the manifest does not list fails TS-2. No new integrity mechanism was needed, and none was invented.

5.3 Size, splitting, and exclusion

A decade of one point at fifteen minutes is roughly 350,000 rows and some ten megabytes of text before the container's DEFLATE, which compresses repetitive timestamped rows well. History still dominates the byte size of any package that carries it, and three honest tools exist for that: split a long history into several series over successive coverage intervals; carry rollups alongside — or instead of — raw records, declared as rollups (3.6); or exclude history by size, declared under BUS-1 16.6 with reason

`excludedForSize` and the content available by another route. What is not available is silence.

6 Completeness and honesty

AGREED This clause is the reason the part exists. The formats are means.

6.1 The domain declaration

The `timeSeries` domain of BUS-116.6 now declares against a specification. `complete` means every series the exporting system holds for the model's points is present. `partial` means some are, and the reason code says why the rest are not. `absent` with a reason remains an honest package at any profile below `full`. `deferredToLaterVersion` is no longer a valid reason for this domain — the version arrived.

6.2 Reason codes for partial history

Real historian exports fail partially, in recurring ways, and the reason codes are how each way is said in a fixed vocabulary an owner can read before purchase. Two codes are added to BUS-116.6 in this version because partial history needed statements the prior vocabulary could not make honestly:

THE SITUATION	THE DECLARATION
The retention window purged it. 2015 was collected and is gone.	<code>partial</code> with <code>retentionExpired</code> . Distinct from <code>notHeldBySystem</code> because <i>collected and purged</i> and <i>never collected</i> age into different answers — the same distinction that makes provenance method REQUIRED (BUS-114.1).
The historian was unreachable at export time — offline, an archive tier unmounted.	<code>partial</code> or <code>absent</code> with <code>sourceUnavailable</code> . The data may exist; this package does not carry it, and neither of the older codes says that.
The owner excluded an interval — a tenancy dispute, a confidential period.	<code>partial</code> with <code>excludedByOwner</code> , and the interval named in a note.
The export would be too large and the history is available by another route.	<code>partial</code> or <code>absent</code> with <code>excludedForSize</code> (5.3).
The system trends the points; the export function cannot read the historian.	<code>partial</code> or <code>absent</code> with <code>notSupportedByExporter</code> — the code an owner should read as a product gap.
The system never trended these points.	<code>notApplicable</code> , or <code>absent</code> with <code>notHeldBySystem</code> . Nothing was lost; nothing existed.

Test TS-7 checks a `complete` declaration against the package's own point declarations, and EX-10 grades the rest against the source system during certification. Nothing in this clause can compel completeness. Like BUS-116.6, it converts an unbounded quality

problem into a bounded disclosure problem and makes the disclosure uncomfortable to write.

6.3 Change-of-value data

A change-of-value historian records instants of change, not a grid, and 3.5-1 requires it exported that way. The temptation this clause exists to name: resampling COV data onto fifteen-minute rows produces a file that looks like every other trend and is a fabrication — each invented row asserts an observation nobody made. A receiver interpolating COV data for display is doing analysis, which is its business; an exporter interpolating it into the package is manufacturing evidence.

6.4 Systems that hold only rollups

Some systems discard raw records on a schedule and keep hourly summaries forever. The honest export is the rollup series with its `aggregation` declared (3.6), plus a domain declaration that says what happened to the raw records: `retentionExpired` where they were collected and purged, `notHeldBySystem` where summaries are all that was ever stored. What 3.5-1 and this clause together forbid is the third option — shipping the summaries undeclared, as if the building were observed once an hour.

6.5 Sources that stored local time

Some historians store wall-clock local time with no offset. Requirement 4.4-1 still applies: the exporter converts to UTC using the IANA rules of the zone the source ran in, and records that zone in `sourceTimeZone` so the conversion is auditable and wall-clock time is reconstructable.

One hour a year cannot be converted with certainty: during a backward daylight-saving transition each local time occurs twice, and a bare local timestamp does not say which. An exporter **MUST** resolve the ambiguity by the earlier UTC instant, consistently, and **SHOULD** record the limitation in `completeness.knownLimitations`. This is a real loss, it is the source system's loss rather than this format's, and the honest response is to state the rule and the residue rather than pretend the conversion is exact.

7 Canonical form and comparison

AGREED

7.1 The canonical form of a series file is the file

BUS-117.5 needs a canonicalization step because JSON has permitted variation — key order, whitespace, number spelling — and comparison requires removing it. `bus-csv-1` was profiled until it has none: one encoding, one line ending, one header, one timestamp form, one number grammar, one row order. Two conforming exporters of the same records produce the same bytes, so the canonical form of a conforming sample file is

itself, and a canonicalization step would be a place for two implementations to disagree about nothing.

Requirement 7.1-1

Two series files are compared for equality as byte sequences, with no canonicalization applied. Row order is significant and **MUST** be preserved on re-export, including the source order of rows with equal timestamps. The comparison rule for the declaring `bus:Series` entities is BUS-117.5's, unchanged.

Test RT-28 applies this across a round trip: an importer stores samples in whatever internal form it likes, and what it re-emits is the same bytes. This is stricter than semantic equivalence and deliberately so — for evidence-grade data, "equivalent" is an argument and "identical" is a checksum, and the digests of BUS-116.5 mean the check costs nothing.

7.2 What byte equality does not require

It does not require that two *different* exports of the same historian agree — coverage, retention, and export time all legitimately differ, and package chaining (BUS-116.9) is the tool for relating successive exports. It constrains re-emission of what was received, which is the reconstruction guarantee's concern, and nothing else.

8 Security considerations

AGREED Escalated in the same spirit as BUS-119: history is evidence, and evidence attracts adversaries.

8.1 A sample file is inert

A `bus-csv-1` file has no elements, no references, no styles, no scripts, no external fetches, and no execution model — three columns of text. Compare BUS-119.3's handling of figures: SVG is a document format with live semantics, so figure mode needed a closed profile, forbidden-element lists, and refusal rules to make it safe to render. CSV needed none of that, because there is no behavior to constrain. That asymmetry is an argument for the format, not an accident of it, and it is why this clause is short.

Inert is not free. An importer **MUST** bound the resources it will spend parsing sample files — nothing in this part constrains row count or decompressed size — and the path-safety and digest-before-ingestion rules of BUS-119.3 apply to sample files exactly as to every other package entry.

8.2 History is personal data at fine grain

BUS-119.4 notes that occupancy points can reveal the movements of identifiable people. A time series is that risk at full resolution: months of occupancy, badge-driven lighting, or single-tenancy meter data is a record of when particular humans were where. An

exporter **MUST** provide a means to withhold such series and **MUST** declare the withholding with reason `excludedForConfidentiality` — declared, not silent, for BUS-1 19.4’s reason: an undeclared redaction is indistinguishable from an assertion that was never made.

8.3 History as evidence

An energy baseline dispute, a warranty claim, a performance-contract settlement — series data is what the argument is about, which makes tampering profitable. The digests of BUS-1 16.5 make tampering detectable within a package; signatures over the manifest (BUS-1 16.3) extend that to the package as an artifact; and the byte-equality rule of 7.1 means a re-export cannot silently launder an altered file as an equivalent one. Where series data is relied on as evidence, the package **SHOULD** be signed, and the signature-format gap BUS-1 Annex F.1 records applies here with the same weight.

9 References

Normative except where marked. BUS-1’s references apply wherever this part uses BUS-1 machinery.

The list is short because the part is: everything this document builds on that is not listed here reaches it through [BUS-1], whose clause 3 carries the full normative set — the container, the digest algorithms, the timestamp grammar, the schema dialect. A reader implementing BUS-2 alone is implementing it wrong, and the reference list is shaped to say so.

REFERENCE	TITLE AND NOTE
[BUS-1]	<i>BUS-1: Core Information Model, Portable Presentation, and Exchange Package</i> , version 0.3. Part 1 of this standard. This part is not free-standing (1.2).
[RFC2119]	Bradner, S., <i>Key words for use in RFCs to Indicate Requirement Levels</i> , BCP 14, RFC 2119, March 1997.
[RFC8174]	Leiba, B., <i>Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words</i> , BCP 14, RFC 8174, May 2017.
[RFC3339]	Klyne, G., Newman, C., <i>Date and Time on the Internet: Timestamps</i> , RFC 3339, July 2002.
[RFC4180]	Shafranovich, Y., <i>Common Format and MIME Type for Comma-Separated Values (CSV) Files</i> , RFC 4180, October 2005. Profiled by clause 4; the LF deviation is stated in 4.2.
[HAYSTACK]	Project Haystack, version 4.0.0. Informative. The <code>hisMode</code> vocabulary 3.5 follows, and the live-service history model 1.1 declines.
[BRICK]	Brick Schema, version 1.4.4, the <code>ref:</code> namespace. Informative. The external-pointer history model 1.1 declines.

Thirty years of what the building did, in a file the owner holds.

Not a live service. Not a database pointer. Rows a text editor can open in 2056, with a digest that says nobody touched them.

END OF BUS-2 · VERSION 0.3 · WORKING DRAFT