Export manifest consumer interface with stable game IDs

Specification document version: **1.1.0**. JSON interface version: **1**.

Version 1.1.0 documents the optional `split_from` field on change records. The JSON interface version remains `1`; the specification document version is not serialized in the release files.

This document describes how an external application reads exported PGN releases, verifies their integrity, and applies incremental changes. It covers only the published files and their interpretation.

1. Published files

Each PGN stream has three accompanying files:

```
games.pgn
games.pgn.manifest.json
games.pgn.records.jsonl
games.pgn.changes.jsonl
```

File	Consumer purpose
`.pgn`	Complete exported games for this stream.
`.manifest.json`	Stream identity, predecessor, counts and integrity information.
`.records.jsonl`	Complete inventory of the games in the PGN.
`.changes.jsonl`	Changes relative to the stream's predecessor release.

A complete release can contain several streams, such as OTB and Online. A final **`.release.json`** marker lists all streams belonging to that release. Use this marker when those streams must be installed together.

The marker location is supplied with the export. Typical locations are:

- Single stream: `games.pgn.release.json`.

- OTB/Online pair: the OTB PGN basename followed by `.release.json`.
- Partitioned collection: `<prefix>export.release.json` in the output directory.

Only the published files listed here are part of the consumer interface. An export without these manifests does not provide this incremental contract.

Manifest generation can be disabled per export file type (for example year partitions or conflict files) in the producer's export file overview. Such PGN files still carry `DeduplicatorId` tags, but have no `.manifest.json`, `.records.jsonl` or `.changes.jsonl`; existing sidecars of these files are removed on the next export, and they are not listed in `.release.json`. If no stream of a release keeps manifests, the `.release.json` marker is removed. Re-enabling manifests later starts a new lineage for that stream: its first manifest has no predecessor and lists every game as `added`.

2. Game identity and scope

Each exported game participating in this interface contains one identity tag:

```
[DeduplicatorId "4294967297"]
```

`DeduplicatorId` is a **stable game ID** within its namespace. It remains unchanged across successive exports and replacement of a game's representative, provided the logical identity is unchanged. This lets consumers recognize an existing game without relying on its position in the PGN or its current metadata.

The value is a positive decimal string in `1..=9223372036854775807`. **Treat IDs as strings**, particularly in JavaScript, where ordinary numbers cannot represent the entire range exactly. Do not derive meaning from an ID or attempt to recreate it.

A game's identity is scoped by the manifest's `namespace`. The same numeric ID in different namespaces does not establish that the games are related.

An ID identifies a logical game rather than a particular exported version. Metadata, moves or provenance may change while the ID remains the same. When identities are consolidated, the change feed can provide a redirect.

Some streams intentionally contain multiple copies of one logical game. For these streams, inventory and change records use `(id, occurrence)`:

- `occurrence` is a one-based integer assigned separately for each ID.

- It follows the order of that ID's appearances in the final PGN.
- It is not another PGN tag or a persistent identity for a source copy.
- Reordering copies can produce occurrence-level changes.

To match inventory records to PGN games, read the PGN in order and count occurrences separately for each `DeduplicatorId`.

3. Complete-release marker

The `.release.json` marker is the entry point for consuming a complete release. It is published after the member files. The following examples use abbreviated hashes for readability; actual SHA-256 values are 64 lowercase hexadecimal characters.

```
{
  "version": 1,
  "namespace": "example-namespace",
  "release_id": "...",
  "predecessor_id": null,
  "stream_count": 2,
  "total_count": 123,
  "members": [
    {
      "filename": "otb.pgn",
      "profile": "example-otb-profile",
      "release_id": "...",
      "count": 100,
      "pgn": {"sha256": "...", "bytes": 1000},
      "records": {"sha256": "...", "bytes": 400},
      "changes": {"sha256": "...", "bytes": 700},
      "manifest": {"sha256": "...", "bytes": 800}
    },
    {
      "filename": "online.pgn",
      "profile": "example-online-profile",
      "release_id": "...",
      "count": 23,
      "pgn": {"sha256": "...", "bytes": 500},
      "records": {"sha256": "...", "bytes": 200},
      "changes": {"sha256": "...", "bytes": 300},
      "manifest": {"sha256": "...", "bytes": 800}
    }
  ]
}
```

Field	Meaning
`version`	JSON interface version; currently the integer `1`.
`namespace`	Identity scope shared by the members.
`release_id`	Opaque identifier for this complete release.
`predecessor_id`	Previous complete-release ID, or `null` for a baseline.
`stream_count`	Number of members.
`total_count`	Sum of member record counts, including repeated occurrences.
`members`	Streams in the release, sorted by relative filename.

Each member supplies:

- `filename`: PGN path relative to the marker's directory.
- `profile`: opaque identifier for that stream's export/filter contract.
- `release_id`: member's per-stream release ID.
- `count`: number of PGN game occurrences.
- `pgn`, `records`, `changes`, `manifest`: expected checksum and byte length.

The companion filenames are derived by appending `.records.jsonl`, `.changes.jsonl`, and `.manifest.json` to `filename`.

Resolve member paths relative to the marker, not the application's working directory. Paths may contain `../`; enforce an allowed base directory before reading files from an untrusted distribution.

`total_count` is **not** a count of distinct logical games across streams. Streams can overlap.

Complete-release IDs and per-stream release IDs belong to different chains. Compare a predecessor only with an ID from the corresponding chain. A new complete-release marker may have a different ID even when the member game-data release IDs have not changed.

4. Per-stream manifest

```
{
  "version": 1,
  "namespace": "example-namespace",
  "profile": "example-otb-profile",
  "release_id": "...",
  "predecessor_id": "...",
  "predecessor_profile": "example-otb-profile",
  "count": 100,
  "change_count": 2,
  "pgn": {"sha256": "...", "bytes": 1000},
  "records": {"sha256": "...", "bytes": 400},
  "changes": {"sha256": "...", "bytes": 700}
}
```

Field	Meaning
`version`	JSON interface version; currently the integer `1`.
`namespace`	Scope of the game IDs.
`profile`	Opaque stream/export compatibility identifier.
`release_id`	Opaque identifier for this stream's game-data release.
`predecessor_id`	Release to which the change feed applies, or `null`.
`predecessor_profile`	Predecessor's profile, or `null`.
`count`	Number of game occurrences in the PGN and inventory.
`change_count`	Number of JSON objects in the change feed.
`pgn`, `records`, `changes`	Integrity information for the corresponding files.

Both predecessor fields are `null` for a baseline.

Namespace and profile must match the consumer's installed stream before an incremental update is applied. Treat a different namespace or profile as a separate baseline, not as a continuation of the existing stream.

Profiles and release IDs are opaque strings: compare them exactly rather than parsing or recalculating them.

An unchanged republication retains the per-stream game-data release ID and has an empty change feed. Empty streams are valid: their PGN and inventory can have zero records, and their changes can remove previously exported records.

5. Complete inventory

The `.records.jsonl` file contains one JSON object per line:

```
{
  "id": "4294967297",
  "occurrence": 1,
  "revisions": {
    "moves": "...",
    "content": "...",
    "provenance": "...",
    "structural_master_id": "8589934593"
  }
}
```

Records are sorted numerically by `(id, occurrence)`, not by PGN order. The inventory describes exactly the games in this exported stream. It is the authoritative full snapshot for installing a baseline or recovering from a missing predecessor.

Revision fields

Field	Consumer interpretation
`moves`	Revision token for the rendered movetext, including comments and variations.
`content`	Revision token for exported game content other than provenance and the identity tag; includes movetext changes.
`provenance`	Revision token for exported provenance and representative-selection changes.
`structural_master_id`	Opaque representative identifier, or `null` if unavailable.

Revision tokens are lowercase SHA-256 hexadecimal strings. Compare them for equality; consumers do not need to reproduce their generation.

The `moves` revision is **textual**, not proof of a changed played mainline. Comments, variations or formatting can change it. A consumer requiring semantic move equivalence must inspect the PGN.

Provenance covers the exported `Source`, `SourceQuality`, `ImportDate`, `IdentityEvidenceWhite`, and `IdentityEvidenceBlack` values, as well as representative changes. A provenance-only update does not by itself require rebuilding a positional index.

`structural_master_id` allows a consumer to observe replacement of the selected representative, even when the game ID stays unchanged. It is not necessarily the ID of a separately exported game or the row supplying every exported value. Do not use it as a PGN lookup key or infer its internal structure.

6. Incremental change feed

The `.changes.jsonl` file contains one JSON object per change:

```
{
  "id": "4294967297",
  "occurrence": 1,
  "kind": "changed",
  "reason": "revision_changed",
  "fields": ["provenance"],
  "old": {
    "moves": "...",
    "content": "...",
    "provenance": "...",
    "structural_master_id": "4294967297"
  },
  "new": {
    "moves": "...",
    "content": "...",
    "provenance": "...",
    "structural_master_id": "8589934593"
  }
}
```

`kind`	`reason`	Required action
`added`	`new_id`	Add the occurrence using the current PGN; `old` is `null`.

`changed`	`revision_changed`	Update the occurrence using its new revisions and current PGN.
`removed`	`missing_id`	Remove the occurrence from this stream; `new` is `null`.
`redirect`	`identity_redirect`	Retire the old identity in this stream and record its successor.

`old` and `new`, when non-null, have the same shape as the inventory's `revisions` object. `fields` names the affected revision tokens: `moves`, `content`, and/or `provenance`. Additions and removals list all three.

A redirect additionally contains:

```
{
  "target_id": "4294967297"
}
```

This field belongs to the same change object, not to a separate JSONL record. A redirect is emitted only when the old ID has disappeared entirely and its successor exists in the current stream. `new` contains the successor's occurrence 1 revisions. Several old occurrences can redirect to the same successor; do not insert multiple successor copies solely because of redirects.

If a successor is not present in this stream, the disappearance is reported as a removal instead. A removal means **absent from this export**, not a claim that the logical game was deleted from every collection.

Added/changed records follow current inventory order. Removed/redirect records follow afterward in predecessor inventory order. Apply the feed as a single transaction against the final inventory, not as an unverified sequence of independent edits.

The feed contains revisions and identity mappings, **not complete replacement PGN games, byte offsets, or patches**. Read added/changed game payloads from the current PGN by `(DeduplicatorId, occurrence)`. A single streaming PGN pass can collect the required payloads.

The first baseline reports all occurrences as additions. Do not invent successors from similar names, moves, or numeric IDs.

Master links created by Exact, Subsumption and Join Lines can form chains. Export resolves these links to the final current master before checking for identity splits. Intermediate master replacements are not splits and do not change the stable ID. Cyclic current master links stop export before publication.

Automatic identity splits (NEW v4.3.13). When deduplication separates games that previously shared one ID (for example after a newly quarantined identity conflict), export migrates the registry automatically instead of stopping. The part containing the ID's representative source keeps the ID; if that source no longer exists, the part containing the previous current master keeps it, otherwise the part with the smallest source ID. Every other part receives the smallest source ID of its members as new ID. The choice is deterministic and a rerun is a no-op. The change feed stays within version 1: a split-off ID appears as `added` / `new_id` with an optional `"split_from": "<previous ID>"`; the retained ID appears as `changed` if its revisions changed. Consumers may ignore `split_from`. If more than 0.5% of all logical identities (and more than 100) would split in one export, export stops before publication and leaves the registry unchanged, because such a mass split indicates a deduplication regression rather than a correction.

7. Integrity verification

Every integrity object has this shape:

```
{"sha256": "<64 lowercase hexadecimal characters>", "bytes": 12345}
```

`sha256` is SHA-256 of the **exact distributed file bytes**. `bytes` is the byte length, not a character count. Do not normalize line endings, reformat JSON, or modify PGN text before checking the checksum.

For a complete release:

1. Validate marker `version`, namespace and member paths.
2. Verify each member manifest against the marker's `manifest` integrity object.
3. Match each manifest's namespace, profile, release ID and count to its member.
4. Verify PGN, inventory and change files against both declarations.
5. Check record counts, change counts and complete-release totals.

For standalone stream consumption, begin with its manifest and verify its three files. Do not combine streams from unrelated complete-release markers.

Checksums detect corruption and mismatched files. They do **not** authenticate the publisher: a trusted delivery channel or independent signature is required when authenticity matters.

8. Recommended installation workflow

1. Read the complete-release marker, or a standalone stream manifest.
2. Copy/download the referenced files into an isolated staging location.
3. Verify all integrity information and compatibility fields.
4. Choose an installation mode:
   - **Incremental:** the installed namespace/profile match and `predecessor_id` equals the installed per-stream release ID.
   - **Baseline:** no matching predecessor is installed; use the complete inventory and PGN instead of applying the change feed.
   - **No game-data change:** the per-stream release ID is already installed; no game update is needed, but verify a newly received complete release.
5. For incremental updates, confirm that affected local revisions agree with `old`; on disagreement, stop and recover from a verified baseline.
6. Apply payload updates, removals and redirects transactionally. The resulting stream must match the complete inventory.
7. Commit the installed release IDs only after successful installation.

For OTB/Online or other multi-stream installations, stage and verify every member before making the complete release visible locally.

A release marker indicates publication completion, but files can still be unavailable or inconsistent after an interrupted transfer or replacement. Do not install a partial release when any checksum, count or scope check fails. Keep the currently installed release and obtain a complete, valid copy.

Consumers should retain their installed namespace, profile, per-stream release IDs, complete-release ID when applicable, and occurrence/revision inventory. The change feed covers only its declared predecessor, not arbitrary older releases.

9. Format and compatibility rules

- This specification document is version **1.1.0**; the JSON interface version remains **1**. The document version is not serialized in release files and does not redefine the meaning of their `version` fields.
- JSON and JSONL are UTF-8; JSONL contains one complete object per nonempty line.
- IDs are decimal strings; counts, byte lengths and occurrences are integers.
- Missing optional values represented in the schema use JSON `null`, not the string `"null"`.
- Reject unsupported interface versions or unrecognized change kinds before applying updates. Additional object fields may be ignored.
- Treat separate namespaces/profiles as separate streams unless a future documented migration explicitly connects them.
- Stream large inventories and PGNs; the interface does not require loading the entire release into memory.

Changelog

- **1.1.0 — 2026-10-24:** Documents the optional `split_from` field for automatically split identities. The JSON interface version remains `1`.