Analyzer Fact Contract v1.1

Media Audit Pro™ / Asset Intelligence Engine™ — 2026-08-11

What changed since v1.0

Two changes. Neither requires any modification to an analyzer written against v1.0.

The return envelope is now documented. v1.0 described the fact record but not the structure that carries it. The Engine has always expected an envelope with a `label` and a `facts` array; an analyzer returning a bare array of fact records falls into the legacy compatibility path and produces no attributed results. This was an omission in v1.0, not a change in behaviour — the correction is to the document.

Run metrics are now accepted. A new optional `metrics` key allows an analyzer to report what it examined, not only what it found. See *Run metrics* below for why this could not be implemented in the Engine.

Rule

Analyzers return observed asset-reference facts, including one or more candidate paths and relevant source context. The Asset Intelligence Engine™ resolves those facts to inventoried assets and determines what they mean.

The Asset Intelligence Engine™ owns the interpretation of every fact returned by an Analyzer.

This preserves the platform's governing principle: Analyzers provide facts. The Engine provides intelligence. Analyzers answer "Where?" The Asset Intelligence Engine™ answers "So what?"

Transport

No change to the plugin mechanism. Analyzers remain ordinary Joomla plugins responding to the existing event, `onMediaauditttoolCollectReferences()` (with the legacy `onMatCollectReferences()` alias retained for compatibility).

Design goals

- Keep analyzers small.
- Keep analyzers stable.
- Keep analyzers extension-specific.
- Keep intelligence centralized.
- Allow the Engine to evolve without requiring analyzer updates.

The return envelope

An Analyzer returns an associative array — the envelope — not a bare list of fact records.

```
return [
    'label'   => 'jDownloads',
    'facts'   => [ /* fact records, see below */ ],
    'metrics' => [ /* optional, see below */ ],
];
```

Envelope keys

`facts` — array of fact records. May be empty. **The presence of this key is what identifies the return as a Fact Contract payload.** An envelope containing neither `facts` nor `unattributed_paths` is treated as a legacy return and its records are not attributed to owning objects.

`label` — human-readable name for the Analyzer, used wherever results are attributed in the interface. Optional; an Analyzer that omits it is displayed as unknown, which is never what you want. Supply it.

`unattributed_paths` — optional array of plain asset paths the Analyzer observed but could not tie to an owning record. These are counted as references but carry no relationship detail, because there is no owner to name. Use this rather than inventing a placeholder owner: a fact with a fabricated `owner_id` is worse than an honest unattributed path.

`metrics` — optional. See below.

Unknown envelope keys are ignored. An Analyzer may safely emit keys a future contract version defines; an older Engine will not fail on them.

Fact records

Each entry in `facts`:

```
[
    'analyzer'        => 'jdownloads',
    'owner_type'      => 'download',
    'owner_id'        => 123,
    'owner_title'     => 'Annual Report',
    'published'       => true,
    'public_url'      => '/downloads/annual-report',

    'candidate_paths' => [
        'downloads/reports/annual-report.pdf',
        'reports/annual-report.pdf',
        'annual-report.pdf',
    ],

    'metadata' => [
        'catid'             => 7,
        'raw_category_path' => 'Reports/Annual Reports',
        'source_field'      => 'url_download',
        'raw_value'         => 'annual-report.pdf',
    ],
```

```
]
```

Required fields

`analyzer` — string identifying the source analyzer (e.g. `'jdownloads'`).

`owner_type` — the type of object that owns this reference, as understood by the source extension (e.g. `'download'`).

`owner_id` — the source extension's internal ID for the owning object.

`candidate_paths` — one or more asset paths the Analyzer observed or derived from the source extension's data and storage conventions. The Analyzer does not choose among them. If the Analyzer only found one plausible path, this array has one entry; it is still called "candidate," never "resolved."

Optional fields

`owner_title` — human-readable title of the owning object.

`published` — publication state of the owning object, as reported by the source extension. This is a fact about the owner, not a renderability determination.

`public_url` — front-end URL, if the Analyzer can determine one directly from the source extension.

`metadata` — an opaque, analyzer-defined context bag. Carries whatever source-specific detail helped the Analyzer generate its `candidate_paths` (category IDs, category path text, which database field the value came from, the raw stored value before any transformation). Shape may vary by analyzer. It exists primarily for tracing and debugging resolution decisions.

A third-party Analyzer that supplies only the required fields is a valid Analyzer. Everything optional the Engine can work around; nothing required, the Engine cannot.

Run metrics

New in v1.1. Optional. An Analyzer may describe the work it did, alongside what it found:

```
'metrics' => [
    'records_examined'   => 45,
    'records_with_value' => 12,
    'duration_ms'        => 118,
    'error'              => null,
]
```

`records_examined` — how many source records the Analyzer inspected.
`records_with_value` — of those, how many held anything in the field the Analyzer reads. Optional even within `metrics`.

`duration_ms` — elapsed milliseconds.

`error` — a short string when the Analyzer could not complete, or `null`. An Analyzer that catches an exception and returns zero facts should say so here.

All keys are optional and all are scalars. The Engine renders whichever it receives and requires none of them.

Why this is in the contract rather than the Engine

The v1.0 amendment policy requires that any proposed change first be tested against a single question: *can this be accomplished without changing the Analyzer Fact Contract?*

It cannot. Zero facts from an Analyzer is ambiguous — the site may genuinely hold no matching assets, or the Analyzer may be reading a column that has been renamed or moved. Only the Analyzer that ran the query knows how many records it looked at. The Engine cannot observe it, infer it, or reconstruct it after the fact.

The distinction matters because the two cases call for opposite responses, and because the error is asymmetric: an Analyzer that misses references makes assets appear unreferenced, which is the direction that ends in deleting a file still in use.

`error` is included for the same reason. An Analyzer that catches a database exception and returns an empty result is indistinguishable, from outside, from one that ran perfectly against an empty table.

Analyzer responsibilities

An Analyzer must not normalize paths, deduplicate references, choose a single canonical path among candidates, determine renderability, generate labels or display formatting, generate notes or recommendations, or contain business logic. Generating multiple candidate paths from known storage and category conventions is fact-gathering, not a violation of this rule — the Analyzer is reporting what it observed, not deciding what is true.

Metrics do not change this. `records_examined` is an observation about the Analyzer's own run, not a judgement about the site. An Analyzer must not decide that its own results look suspicious, or suppress them; it reports the numbers and the Engine draws the conclusion.

Engine responsibilities

The Engine resolves `candidate_paths` against its own asset inventory to determine the actual matching asset, deduplicates across Analyzers and across an Analyzer's own candidates, evaluates renderability and public accessibility, and generates Rendered Intelligence, notes, recommendations, and reports.

The Engine must not depend upon any analyzer-specific `metadata` key unless that key has been promoted into the formal contract. Metadata may be used for diagnostics only.

The Engine must tolerate a missing `metrics` key, and any subset of its members, without degradation. An Analyzer written against v1.0 remains fully supported.

Compatibility

An Analyzer written against v1.0 requires no changes. It will not report metrics, and the Engine will not present metric-derived observations for it — that is the only difference.

Additions to this contract are always optional. A breaking change would be issued as a new contract, not as a revision to this one.

Future considerations (out of scope)

A stable per-fact identifier (e.g. `fact_id` or `reference_id`, such as `jdownloads:123:url_download`) would make caching, incremental rescans, diagnostics, and diff reports easier. Not needed now — noted here so it isn't lost, and so the contract stays minimal.

Static declaration of what an Analyzer targets — extension element, supported version range, certification — is deliberately **not** part of this contract. It does not change between runs and must be readable without performing a scan. It will be issued as a separate sibling contract, in the same way Inventory Hints and Cache Locations were.

Amendment policy

Before proposing any change to this contract, ask: can this be accomplished without changing the Analyzer Fact Contract? If yes, it belongs in the Engine. Only if the answer is no should a contract revision be considered.

v1.1 was issued after applying that test to run metrics. The answer was no: the Engine cannot observe how many records an Analyzer examined. The test is the amendment process, not an obstacle to it — but the burden of proof sits with the proposal.