SVG Embedded Semantic Metadata (SESM)

Specification and tooling for embedding structured, rich metadata directly inside SVG assets, converting raster images to SVG, and integrating semantic graphics with Aptlantis.

APTLANTIS / SESM | DRAFT V0.2 | PYTHON 100%

What Is SESM?

Core Concept

SESM treats SVGs not merely as visual graphic files, but as **self-describing semantic capsules** — carrying context about provenance, UI layout hints, themes, LLM prompt hints, and archival rules inside a valid `<metadata id="sesm">` tag.

In Aptlantis

SESM acts as a bridge between compiled SVG components and automated pipelines. An SVG can represent a dataset card, a pipeline status panel, a theme board, or a status badge.

Even if no system understands SESM, the image remains a valid SVG. Agents that do understand SESM can extract, parse, validate, and summarize the asset intelligently.

Metadata Block Structure

A SESM block is a single JSON object wrapped inside an XML `CDATA` block inside `<metadata id="sesm">`. Only `sesm_version` is required. The block can carry fields like `asset`, `theme`, `provenance`, `llm`, and more.

```
<metadata id="sesm"><![CDATA[
{
 "sesm_version": "0.2.0",
 "asset": { "id": "apt-caddy-logo", "role": "logo" },
 "theme": { "id": "neon-ink", "mode": "dark" },
 "provenance": { "generated": true, "generated_at": "2026-05-27T22:21:44Z" }
}
]]></metadata>
```

Key Metadata Fields

`asset`

Describes what the SVG represents: `id`, `role`, `title`, `ecosystem`, `tags`. Common roles: `logo`, `icon`, `dataset-card`, `status-badge`.

`theme`

Styling context: modes (`dark`/`light`), color contract, accent values, semantic states (`running`, `warning`, `error`, `verified`), and NIPC color tokens.

`llm`

Natural language explanations, prompt hints, and interpretations intended for LLMs to understand graphical content without visual chart reading.

`provenance & integrity`

Audit logs of how the SVG was created (generator tool, version, timestamp, git commit) plus hash mappings (BLAKE3/SHA-256) linking assets to release snapshots.

Embedding Tool: `Embed-SESM.py`

Automatically processes SVG dimensions, strips editor-specific namespace junk (Inkscape/Sodipodi), scans for visual themes, deep-merges override files, and embeds compliant JSON metadata.

NIPC Theme Detection

Scans SVGs for NeonInk Palette Contract colors. If found, populates `theme.tokens`, sets visual mode, and maps accents automatically.

Deep Merge Overrides

Recursively merges metadata overrides from `svg-metadata.overrides.json` keyed by asset slug.

Legacy Mapping

Harvests legacy metadata formats, mapping `ai` summaries/tags to `llm.summary` and `asset.tags`.

Automated Validation

Validates all metadata blocks using `jsonschema`. Falls back to a robust manual structural validator if the library is unavailable.

`Embed-SESM.py` CLI Options

Option	Default	Description
`--input-dir / -i`	`./svg`	Directory containing SVG assets
`--overrides / -o`	`./svg-metadata.overrides.json`	Path to overrides database
`--schema / -s`	`./svg_asset.schema.json`	Path to validation JSON Schema
`--validate-only`	—	Validate metadata without rewriting files
`--verbose / -v`	—	Print verbose debug output

Conversion Tool: `Convert-to-SVG.py`

Converts traditional raster formats (PNG, JPG, WEBP, BMP, etc.) into clean SVGs via two modes.

Mode 1: Base64 Embed

`--mode embed` (default fallback). Loads the raster image, encodes it as base64 PNG, and wraps it inside an SVG `<image>` tag. Guarantees 100% identical appearance for complex visuals.

Mode 2: Vector Trace

`--mode trace` performs monochrome contour vectorization using OpenCV. Supports options: `--threshold` (Otsu auto or 0–255), `--simplify` (Ramer-Douglas-Peucker), `--blur`, `--invert`, and `--min-area` for noise filtering.

Advanced Conversion Features

Hole Carving (RETR_CCOMP)

Unlike naive tracers that fill holes inside letters (like 'O', 'A', 'B'), `Convert-to-SVG.py` traverses contour hierarchies to pair external boundaries with internal child boundaries, generating a single path with `fill-rule="evenodd"`.

Graceful Fallbacks

If OpenCV is unavailable, or if tracing fails to detect contours, the tool automatically falls back to base64 raster embedding — ensuring the conversion pipeline **never breaks**.

Testing & Aptlantis Integration

🧪 Running Tests

The project includes an integration and unit test suite built on Python's standard `unittest` framework. Run with `python tests/run_tests.py`. Covers:

- Overrides merging (recursive deep merges)
- Color heuristics (NIPC contract token detection)
- Structural validation
- Raster conversion, blur, min-area, and hole-carving logic

🎨 Studio Integration Patterns

- **Static Web Generation**: Site compilers parse `<metadata id="sesm">` to populate search, dashboards, and index pages without querying an external DB.
- **LLM Context Injection**: Scripts extract `llm.summary` and `llm.prompt_hints` to help agents understand graphical dataset cards.
- **Archive Integrity Check**: Archivers read the `integrity` object to confirm content hashes match original database snapshots.

SESM: SVGs That Speak for Themselves

SESM transforms SVG files from passive graphics into intelligent, self-describing semantic assets — carrying provenance, theme, LLM hints, and integrity data wherever they go.

Spec

Draft v0.2 — open standard, JSON inside CDATA

Tools

Embed-SESM.py · Convert-to-SVG.py

Tests

Full unittest suite via `run_tests.py`

Platform

Aptlantis Studio · aptlantis.studio