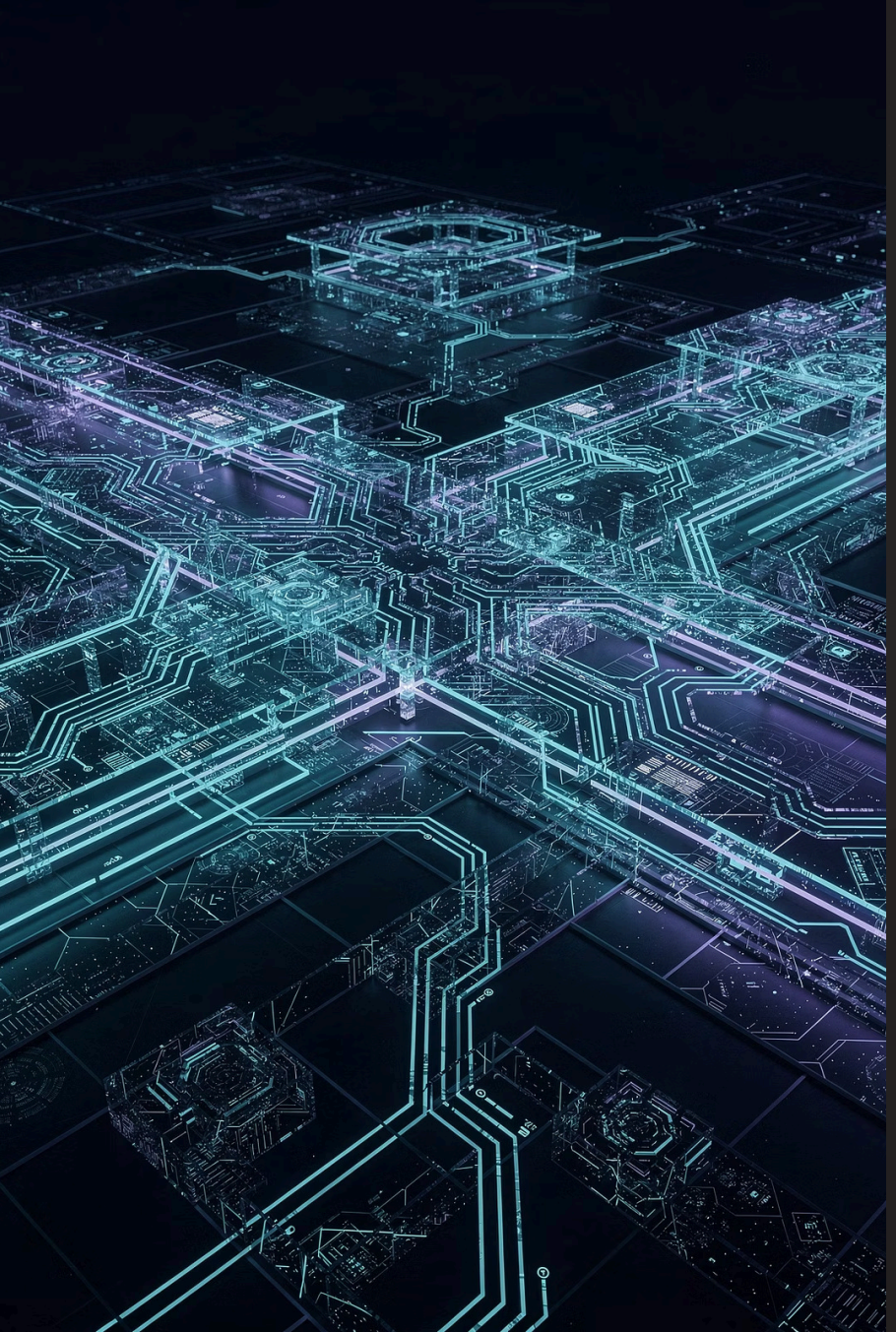


Artifact Interface Contract (AIC)

Status: Draft v0.1.1 | **Host:** Aptlantis Studio | **Compatible With:** SESM v0.2.x · NIPC v0.1.x ·Neon Ink v0.1.x ·APGC v0.1.x

The contract layer that transforms structured data, SESM metadata, NIPC palette semantics, and Neon Ink brand expression into deterministic rendered artifacts in SVG, HTML, and related brand surfaces.

1. Overview

The **Artifact Interface Contract (AIC)** is the binding bridge between SESM, NIPC, and Neon Ink — converting meaning, palette, and brand rules into consistent, reproducible rendered artifacts. Each layer in the pipeline carries a distinct responsibility.

1

Data

Source records, manifests, pipeline output, and content

2

SESM

Identity, semantics, provenance, links, and agent/UI hints

3

AIC

Artifact types, required fields, layout regions, render targets, state behavior

4

SVG / HTML

Final deterministic interface delivered to the user

AIC exists to **prevent design drift**. Given identical source data, SESM block, artifact type, and Neon Ink version, the generator must produce structurally identical artifact output every time. NIPC provides palette semantics and intensity rules; Neon Ink provides typography, density, and panel grammar.

2. Relationship to SESM, NIPC, and Neon Ink

Four documents define the full visual system. Each owns a distinct layer of responsibility — none replaces another.

Layer	Document	Responsibility
Meaning	SESM v0.2.md	Embedded metadata, artifact identity, provenance, links, agent and crawler hints
Palette	NIPC v0.1.1.md	Palette families, semantic roles, psychological intent, intensity, palette validation
Contract	AIC v0.1.1.md	Artifact type contract, required fields, layout regions, render targets, state mapping
Expression	Neon Ink v0.1.md	Brand expression, typography, panel grammar, page composition, brand voice

What SESM Answers

- What is this artifact and where did it come from?
- What data or template produced it?
- Where should agents or crawlers go for canonical context?

What AIC Answers

- What must a `dataset-card` display?
- Which fields are required, optional, derived, or hidden?
- How does state map to border, glow, animation, and interaction?
- What validates whether an artifact is contract-compliant?

3. Goals

AIC is designed around ten principles that keep generated interfaces stable, semantically honest, and testable across the full Aptlantis Studio pipeline.

1

Deterministic Output

Same inputs always yield structurally identical artifacts.

2

Semantic Binding

SESM metadata is bound to Neon Ink visual semantics through AIC.

3

Canonical Types

Defines first-class artifact types for the website and generator.

4

SVG / HTML Separation

Portable visual artifacts are cleanly separated from interactive page behavior.

5

Drift Prevention

Prevents field, layout, and state drift across all pages and surfaces.

6

Testable Generation

Supports validators, preview engines, editors, diff tools, and future visual builders.

Archival Meaning

Preserves local-first, long-term artifact meaning.

Dense Scanability

Keeps dense interfaces semantically consistent.

Theme Variants

Allows theme variants without breaking core contracts.

Agent Hints

Preserves crawl and LLM hints in SESM for every artifact.

SCOPE BOUNDARY

4. Non-Goals

AIC is a **rendering contract** — not a storage format, brand manual, or schema authority. Understanding its boundaries is as important as understanding its responsibilities.

Does Not Replace SESM

SESM metadata definitions remain canonical in their own document.

Does Not Replace Neon Ink

Visual rules, typography, and brand voice live in Neon Ink.

Does Not Define Schemas

Dataset schemas and Rust generator internals are out of scope.

No Auth / Access Control

Private metadata policy and authentication are explicitly excluded.

Also Out of Scope

- Requiring every artifact to be SVG
- Requiring JavaScript for basic meaning
- Forcing marketing graphics to expose full technical metadata
- Defining a complete design tool file format

AIC answers *how* things render — not *what* they mean, *what* they look like in brand terms, or *how* they are stored.

5. Terminology

These eight terms form the precise vocabulary used throughout AIC contracts. Consistent usage prevents ambiguity across generators, validators, and documentation.

Artifact Type

A stable named primitive the generator can render — e.g., `dataset-card`, `pipeline-panel`, `qa-item`.

Contract

The full specification for an artifact type: required fields, optional fields, layout regions, render targets, state behavior, and validation rules.

Region

A named spatial slot in an artifact layout. Examples: `icon`, `title`, `tags`, `stats`, `actions`, `state_badge`.

Render Target

The output context where a field or region appears — `svg`, `html`, `sesm`, or `brand`.

Visible Field

Must be shown to the user in the rendered UI.

Embedded Field

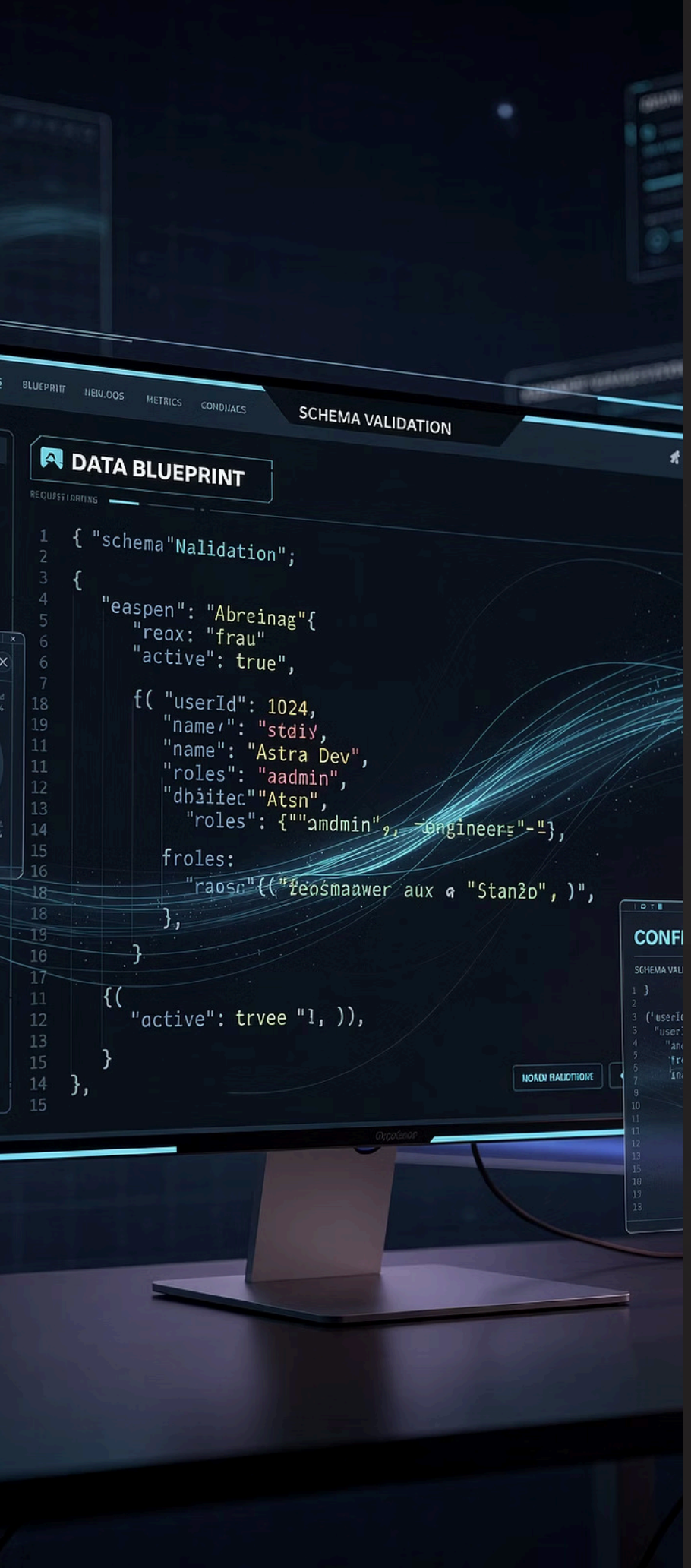
Preserved in SESM or machine-readable metadata regardless of visual display.

Derived Field

Computed from source data — e.g., `quality_label` from `quality_score`.

Canonical Action

Primary destination or operation — e.g., opening the canonical dataset page or downloading a release.



6. AIC Block Structure

An AIC contract is a JSON-compatible object with a stable top-level shape. The contract may live as a standalone JSON or TOML file, embedded generator configuration, documented Markdown examples, a Rust struct, or a validation schema.

```
{
  "aic_version": "0.1.0",
  "artifact_type": "dataset-card",
  "compatible_with": {
    "sesm": "0.2.x",
    "nipc": "0.1.x",
    "neon_ink": "0.1.x",
    "apgc": "0.1.x"
  },
  "fields": {},
  "theme": {},
  "layout": {},
  "semantic_mapping": {},
  "state_mapping": {},
  "fallback_rules": {},
  "render": {},
  "render_targets": {},
  "interaction": {},
  "validation": {}
}
```

The contract must remain stable enough that artifacts generated today can be validated years later. Stability in top-level keys is a hard requirement.

7. Canonical Artifact Types

AIC v0.1 defines six first-class artifact primitives. Each is a compiler target — not a hand-designed component. Type names must be lowercase and hyphen-separated. Extension types should be namespaced until promoted into a later AIC version.

dataset-card

Compact dataset preview artifact for home, category, related, and listing pages.

dataset-header

Primary dataset identity artifact. Full detail panel with provenance and quality summary.

pipeline-panel

Pipeline state and execution artifact showing run summary, stages, and status.

qa-item

Indexed documentation or FAQ artifact with accordion layout and semantic indicator.

stats-tile

Single metric artifact. One label, one value, one semantic role. No glow unless live state.

theme-board

Palette and theme identity artifact. Exposes swatches, state examples, and token export.

Extension types such as `navigation-card`, `download-card`, `schema-card`, and `brand-promo-panel` should be treated as extension contracts until promoted into a later AIC version.

8. Global Field Contract

All generated artifacts share a required identity core. Required fields are never guessed — if missing, production generation fails. AIC cleanly separates what the user sees from what SESM preserves.

Required Embedded Fields (SESM)

```
{
  "artifact_id": "rust-code-corpus-card",
  "artifact_type": "dataset-card",
  "state": "active",
  "semantic_role": "code-heat",
  "semantic_family": "creation-build-code",
  "psychological_intent": "signal-hands-on-code-build-work",
  "intensity": 2,
  "source_manifest": "data/datasets/rust-code-corpus.json",
  "template_id": "dataset-card-v1"
}
```

Visible vs. Embedded Split

Visible fields render into SVG, HTML, or both. Embedded fields are stored in SESM or another machine-readable payload even when not shown in the UI.

```
{
  "visible": ["title", "stats", "state"],
  "embedded": [
    "artifact_id",
    "provenance",
    "theme_id"
  ]
}
```

Required Compatibility Fields

```
{
  "aic_version": "0.1.1",
  "sesm_version": "0.2.0",
  "theme_id": "neon-ink",
  "theme_version": "0.1.0"
}
```

`semantic_role` determines the hue token. `semantic_family` supports grouping and validation. `psychological_intent` records *why* the hue was chosen. `state`, `intensity`, `glow`, and `priority` keep rendering deterministic.

9. Artifact Contracts

Each canonical artifact type has a precise contract defining required fields, optional fields, layout regions, and render behavior. Below is a summary of the six core contracts.

1

dataset-card

Required: title, dataset_id, state, semantic_role, records, tokens
Layout: icon → title → tags → stats → actions → status ·compact density

2

dataset-header

Required: title, dataset ID, state, description, tags, records, tokens, license, updated, canonical action
Flow: identity-left-state-right ·detailed density

3

pipeline-panel

Required: pipeline ID, state, dataset, stage list, timestamp
Render: running → pulse; complete → green; warning → yellow; error → red

4

stats-tile

Required: label, value, semantic_role
Rules: no glow unless live state ·mono labels ·values readable at a glance

5

qa-item

Required: question, category, id
Layout: index → semantic_indicator → title → toggle → body → links ·indexed-accordion ·compact

6

theme-board

Required: theme ID/name/version, background swatches, semantic role swatches, state behavior examples

10. Semantic to Visual Mapping

AIC maps NIPC semantic roles and artifact states to deterministic visual behavior. Color role and state strength are always kept separate: hue encodes meaning; brightness, glow, and animation encode activity level.

NIPC Intensity Scale

Level	Name	Use Case
0	muted	Archived, unknown, background metadata
1	whisper	Tiny dot, thin rail, subtle label
2	soft	Normal tags, semantic indicator
3	active	Selected item, highlighted card
4	urgent	Warning, running, featured
5	interrupt	Error or critical blocker only

State Mapping Examples

State	Border	Glow	Intensity
idle	archive	none	0
active	info	soft	2
running	process	pulse	4
warning	caution	soft	3
error	error	sharp	4
verified	success	soft	2
archived	archive	none	0

NIPC Palette Validator Warnings

- Red/risk roles used for non-risk decoration
- Green/trust roles appear without evidence fields
- Yellow/attention roles dominate a surface instead of acting as markers
- A compact artifact uses more than one dominant semantic family
- A `qa-item` lacks a semantic indicator

11. SVG vs. HTML Responsibility

AIC enforces a clean separation between portable visual artifacts and interactive page behavior. Both targets may show the same data — but they must derive it from the same source manifest or compiled artifact payload.

SVG Carries

- Artifact identity, title, and primary semantic role
- State indicator and compact visible metadata
- Core stats, tags (when compact), and embedded SESM metadata
- Stable visual layout for dataset cards, pipeline panels, and theme boards

HTML Carries

- Page layout, interactive controls, and search/filter/sort
- Expanded metadata, keyboard focus behavior, and tooltips
- Long-form text, responsive reflow, and analytics hooks
- Download controls, forms, and user input

Shared by Both

- Title, state, tags, and primary stats
- Canonical action — derived from the same source manifest

ⓘ HTML can wrap SVG artifacts and expose the same canonical data, adding interactions without altering the underlying artifact meaning.

12. Interaction Contracts

Interactions are declared through AIC and reflected in SESM `ui.interaction`. SVGs may expose clickable regions, but HTML wrappers remain responsible for accessibility and keyboard navigation.

Standard Interaction Block

```
{
  "interaction": {
    "click": "canonical_html",
    "hover": "show-metadata",
    "focus": "highlight-border",
    "keyboard": "activate-canonical-action"
  }
}
```

Common Click Targets

- `canonical_html`
- `download`
- `pipeline_report`
- `manifest`
- `schema`
- `related_dataset`
- `none`

Hover Metadata May Show

- Artifact ID and source manifest
- State and last updated timestamp
- Quality score and provenance summary

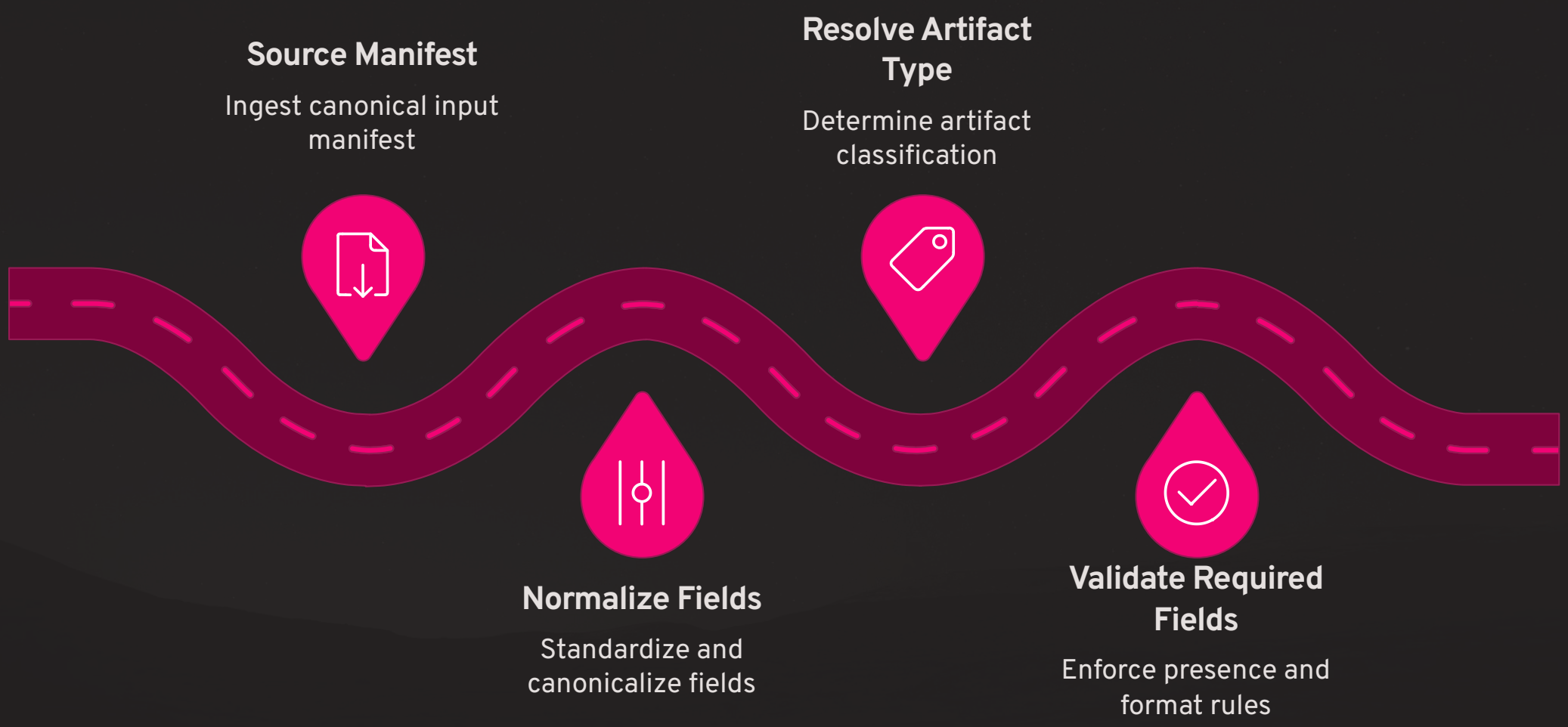
> ⚠ Hover metadata must never be the only place critical warnings appear.

Focus Behavior Rules

- Use semantic border highlight — not glow alone
- Preserve visible focus at all times
- Expose accessible label text in HTML wrappers

13. Generator Contract

The Rust artifact generator treats AIC as a hard contract. Every generation run must follow a deterministic, validated pipeline. Identical inputs must always produce structurally identical outputs.



Given the same generator version, template ID, AIC version, SESM version, and Neon Ink version, output should be structurally identical except for explicitly volatile fields such as timestamps or build IDs.

Missing Required Field

Production builds fail validation. Preview builds may emit a warning artifact but must not silently invent data.

Missing Optional Field

Region may collapse per contract. Layout must remain stable. SESM omits unknown values.

Fallback Priority

Source manifest → derived field → SESM metadata → theme default → explicit unknown/muted state.

Never Guess

License, validation, provenance, quality, and compatibility fields must never be inferred from fallback logic.

14. Validation Rules

AIC validation operates at four layers: global artifact integrity, artifact-specific field checks, visual output quality, and SESM metadata hygiene. APGC geometry validation is advisory in v0.1.

Global Validation Checks

- Artifact type is canonical or explicitly namespaced
- All required fields exist and render
- Required embedded fields exist in SESM
- `theme.id` is present
- State, semantic_role, and semantic_family use recognized names
- `psychological_intent` present for semantic artifacts
- Intensity is within the NIPC 0–5 scale
- Canonical links are valid when declared

Visual Validation

- No clipped text or overlapping critical labels
- Readable contrast and stable layout after optional field collapse
- Correct semantic color usage and glow/state behavior
- Visible keyboard focus in HTML wrappers

Artifact-Specific Checks

dataset-card	title visible; records & tokens visible; canonical action available
pipeline-panel	state visible; stages exist; status color matches state mapping
qa-item	index visible; semantic indicator visible; body available in HTML
theme-board	swatches visible; theme ID and version embedded or shown

APGC Geometry (Advisory)

- `shape_family` must be a recognized APGC family name
- `angle_energy` within 0–5 and within the shape family range
- Declared safe zone honored — no critical UI renders outside it
- For error/critical artifacts, `angle_energy` ≤ 2

Geometry violations produce warnings in v0.1, not hard failures.

15. Compatibility & Versioning

All generated artifacts must track version fields for every layer of the stack. This ensures artifacts can be re-validated, diff-compared, and regenerated consistently as the system evolves.

```
{
  "aic_version": "0.1.1",
  "sesm_version": "0.2.0",
  "nipc_version": "0.1.0",
  "neon_ink_version": "0.1.0",
  "apgc_version": "0.1.0"
}
```

AIC	SESM	NIPC	Neon Ink	APGC	Status
0.1.x	0.2.x	0.1.x	0.1.x	0.1.x	Supported

Breaking Changes (Major Version)

- Removing required fields
- Renaming canonical artifact types
- Changing required layout regions
- Changing state meanings
- Changing render target responsibilities

Non-Breaking (Minor Version)

- Additive artifact types
- New optional fields
- New validation recommendations

Extension Namespacing

- `aptlantis.dataset-map`
- `aptlantis.model-card`
- `experimental.world-panel`

16. Example Complete Contract

A full `dataset-card` contract illustrating all top-level keys — fields, layout, semantic mapping, state mapping, theme, fallback rules, render targets, interaction, and validation.


```
{
 "aic_version": "0.1.0",
 "artifact_type": "dataset-card",
 "compatible_with": { "sesm": "0.2.x", "nipc": "0.1.x",
 "neon_ink": "0.1.x", "apgc": "0.1.x" },
 "fields": {
 "required": ["title","dataset_id","state","semantic_role",
 "semantic_family","psychological_intent","intensity",
 "records","tokens"],
 "optional": ["description","icon","tags","license",
 "quality_score","size","updated","canonical_html"],
 "embedded": ["artifact_id","source_manifest","template_id",
 "provenance","theme"],
 "visible": ["title","tags","records","tokens","state_badge"]
 },
 "layout": {
 "regions": ["icon","title","tags","stats","actions","status"],
 "flow": "left-icon-right-content",
 "density": "compact"
 },
 "semantic_mapping": {
 "code_heat": { "color":"orange","border":true,"accent_weight":"primary" },
 "process": { "color":"violet","glow":"pulse" }
 },
 "state_mapping": {
 "active": { "border":"info","glow":"soft","intensity":2 },
 "running": { "border":"process","glow":"pulse","intensity":3 },
 "success": { "intensity":3,"color_override":"green" }
 },
 "fallback_rules": {
 "missing_license":"omit",
 "missing_validation":"omit",
 "missing_stats":"hide_region"
 },
 "render_targets": {
 "svg": ["icon","title","tags","records","tokens","state_badge"],
 "html": ["canonical_link","hover_metadata","expanded_stats"],
 "sesm": ["asset","artifact","theme","ui","crawl","llm","links","provenance"]
 },
 "interaction": {
 "click":"canonical_html","hover":"show-metadata","focus":"highlight-border"
 },
 "validation": {
 "fail_on_missing_required":true,"allow_missing_optional":true,
 "require_sesm":true,"require_theme":"neon-ink"
 }
}
```

17. Implementation Notes for Rust Generators

The Rust generator treats each AIC contract as a first-class compiler input. Recommended module organization keeps contracts, rendering, and themes cleanly separated for maintainability and testability.

ArtifactType Enum

```
enum ArtifactType {
    DatasetCard,
    DatasetHeader,
    PipelinePanel,
    QaItem,
    StatsTile,
    ThemeBoard,
}
```

Validation Sequence

```
parse source
  -> normalize
  -> validate AIC
  -> validate SESM
  -> render
  -> inspect output
```

Recommended Module Layout

```
contracts/
  artifact_type.rs
  fields.rs
  layout.rs
  state_mapping.rs
  validation.rs

render/
  svg/
  html/
  sesm/

themes/
  neon_ink.rs
```

Template Naming Convention

- `dataset-card-v1`
- `dataset-header-v1`
- `pipeline-panel-v1`
- `qa-item-v1`
- `theme-board-v1`

Template IDs must appear in `SESM artifact.template_id` for every generated artifact.

18. Summary

AIC is the binding contract that makes Aptlantis Studio's visual system **deterministic**. It is the layer that prevents the interface from being merely themed — and keeps it *derived*.

SESM Says

What an artifact *is* — its identity, provenance, and semantic meaning.

Neon Ink Says

What meaning *looks like* — typography, density, and brand expression.

AIC Says

Exactly *how* it renders — fields, layout, state, targets, and validation.

Design is Schema-Aware

Every visual decision traces back to a field contract, a semantic role, and a validated state.

UI is a Compiled Artifact

Interfaces are generated from source, not hand-assembled. SVG becomes a stable transport layer.

Color Carries Meaning

Hue encodes semantic role. Intensity and glow encode state. Neither is decorative.

Pages Can Be Regenerated

Validators, preview tools, and diff utilities can enforce the contract across any future build.

> `dataset.json → SESM → AIC → Neon Ink → SVG/HTML artifact`