Framework Synthesis Ontology Specification (FSO-1)

A House Specification for Structural Renderings of Framework-Objects

Abstract

This document specifies a reusable format for constructing a *framework synthesis ontology document*: a companion artifact whose task is to represent a framework-object explicitly and non-narratively. FSO-1 is designed for projects in which multiple theory-objects already exist, their order of dependence matters, and prose synthesis alone is too lossy a carrier of architectural identity. The goal of an FSO document is not readability, persuasion, or survey-style overview. The goal is architectural recoverability. FSO-1 therefore requires explicit treatment of framework identity, kernel, architectural spine, canonical objects, bridge objects, distinctions, mechanisms, dependencies, interface exports, burdens, failure modes, residues, propagation-sensitive revisions, and regeneration paths. It assumes that a framework-object is not identical with any one synthesis paper, diagram, kernel sheet, or sequence of component papers. A prose synthesis paper, a dependency map, a canonical lexicon, and an ontology document may all render the same framework-object under different presentation constraints.

1. Purpose

FSO-1 exists to solve a recurring problem in large theory projects: once a project grows into a framework or stack, its unity is often carried mainly by local memory, prose sequence, informal summaries, or the author's tacit sense of what depends on what. Individual theory papers may each remain intelligible in isolation while the framework as a whole remains only partially explicit. When this happens, the project risks drift, hidden dependency failure, silent interface change, redundancy, and loss of architectural clarity.

The purpose of a framework synthesis ontology document is therefore to state, in non-narrative form:

- what framework-object is being represented,
- what its kernel is,
- what its architectural spine is,
- what theory-objects are canonical within it,
- what roles those objects play,
- what mechanisms move the framework from one layer to another,
- what distinctions must not collapse,

- what depends on what,
- what the framework exports through stable interfaces,
- what failure modes recur across it,
- what remains unresolved or pressure-sensitive,
- and what kinds of change do or do not require propagation across the larger architecture.

2. What This Document Is For

A framework synthesis ontology document is *not* a replacement for a synthesis paper, a theory paper, or a canon index. It is a companion artifact with a different function.

The reader-facing synthesis paper is for:

- motivating the whole-framework problem,
- presenting the framework in readable prose,
- explaining how the pieces hang together,
- guiding entry into the project,
- and showing why the architecture matters.

The framework synthesis ontology document is for:

- architectural recoverability,
- explicit dependency control,
- distinction preservation across compression,
- propagation-sensitive revision management,
- canon or stack navigation,
- framework-level burden discipline,
- and regeneration of synthesis artifacts from a durable structural source.

3. Core Principle

The core principle of FSO-1 is:

> A framework should be representable as an explicit ontology of kernel, canonical objects, distinctions, mechanisms, dependencies, interfaces, burdens, failure modes, and residues rather than only as a sequence of separate papers or a reader-facing synthesis essay.

4. Upstream Assumption

FSO-1 assumes that a *framework-object* can be distinguished from its renderings, and that a framework-object is not reducible to any one component theory-object.

A framework-object is:

- a bounded but multi-object architecture,
- organized by a kernel, an inferential order, canonical objects, distinctions, mechanisms, and burdens,
- and capable of being rendered in more than one artifact.

A rendering may be:

- a synthesis paper,
- a framework ontology document,
- a dependency map,
- a kernel sheet,
- a canonical lexicon,
- a diagram,
- or another recognized canon artifact.

FSO-1 therefore treats the ontology document as one rendering of a framework-object, not as the framework-object itself.

5. Key Distinctions

Every document written under FSO-1 must explicitly distinguish the following.

5.1. Framework-Object vs Theory-Object

The document must distinguish:

- the framework-object as a structured whole,
- from the bounded theory-objects that compose it.

5.2. Framework-Object vs Rendering

The document must distinguish:

- the framework-object,
- from the synthesis paper, ontology document, diagram, kernel sheet, or map that renders it.

5.3. Kernel vs Architectural Spine

The document must distinguish:

- the framework's kernel: the shortest statement preserving its identity,
- from its architectural spine: the ordered sequence by which the framework unfolds.

5.4. Canonical Object vs Adjacent Object

The document must distinguish:

- canonical theory-objects required for the framework's identity,
- from adjacent, illustrative, or nearby objects that are useful but not structurally necessary.

5.5. Foundational vs Bridge vs Downstream

The document must distinguish:

- foundational objects that ground the framework,
- bridge objects that connect layers,
- downstream objects that elaborate consequences or applications,
- and auxiliary objects that support without anchoring the architecture.

5.6. Local Interface vs Framework Interface

The document must distinguish:

- what a specific theory-object exports locally,
- from what the framework as a whole exports through a stable interface.

5.7. Presentation Order vs Dependency Order

The document must distinguish:

- the order in which the framework is explained to readers,
- from the order in which its objects are structurally organized.

5.8. Internal Drift vs Architectural Drift

The document must distinguish:

- local revision that does not alter the framework's architectural identity,
- from architectural drift that changes kernel, spine, interface, or propagation-sensitive dependency.

5.9. Residue of an Object vs Residue of the Framework

The document must distinguish:

- what remains unresolved within a particular theory-object,
- from what remains unresolved at the level of the framework as a whole.

5.10. Upstream vs Internal vs Downstream

The document must distinguish:

- what the framework inherits from outside itself,
- what relations are internal to the framework,
- and what the framework enables downstream.

6. Framework-Object Types

Every framework synthesis ontology document must classify the target framework-object using one or more of the following types.

- ontological framework
- disclosure framework
- authority framework
- corrective framework
- normative framework
- design framework
- architectonic framework
- mixed framework

This requirement exists because many stacks contain several kinds of objects and it is easy to lose sight of what kind of whole is actually being synthesized.

7. Object Types to Be Represented

Every framework synthesis ontology document written under FSO-1 must make the following object types explicit where applicable.

7.1. Framework Kernel Objects

Compact statements preserving framework identity.

7.2. Architectural Spine Objects

Ordered inferential sequences through which the framework unfolds.

7.3. Canonical Theory Objects

Major theory-objects required for the framework's identity.

7.4. Bridge Objects

Objects whose main role is to connect layers or move the framework from one domain of concern to another.

7.5. Mechanism Objects

Named processes, pathways, or operator sequences that move the framework from one layer to another.

7.6. Distinction Objects

Load-bearing conceptual cuts that prevent architectural collapse.

7.7. Dependency Objects

Relations of grounding, inheritance, clarification, bridging, enabling, export, or constraint.

7.8. Interface Objects

Stable exports through which the framework relates to downstream work or adjacent projects.

7.9. Failure Mode Objects

Recurrent pathologies tracked across multiple theory-objects.

7.10. Burden Objects

Framework-level explanatory, argumentative, or architectural burdens.

7.11. Residue Objects

Open, deferred, unresolved, underdeveloped, or pressure-sensitive regions at framework level.

7.12. Regeneration Objects

Compressed structures from which synthesis artifacts can be reconstructed.

8. Status Classes

Every major object and claim should carry a status.

- `provisional`
- `working`
- `stabilized`
- `pressure-sensitive`
- `deferred`
- `frozen`
- `borrowed`
- `superseded`

9. Architectural Roles

Every canonical object should also carry an architectural role.

- `foundational`
- `bridge`
- `downstream`
- `auxiliary`
- `interface-only`
- `deprecated`

This requirement exists because framework-level clarity depends not only on what an object says, but on what structural role it plays.

10. Freeze Levels

Every framework synthesis ontology document should state a freeze level for the framework-object as a whole, and may optionally assign one to major objects or interfaces.

- `none`
- `soft`
- `interface-frozen`
- `hard`

11. Propagation Sensitivity

Every major framework object, interface, or dependency may optionally carry a propagation sensitivity marker.

- `local`
- `adjacent`
- `architectural`
- `canon-wide`

This marker states how widely a revision should propagate if the object changes.

12. Required Document Structure

A framework synthesis ontology document written under FSO-1 must contain the following sections.

0. Metadata

Include:

- target framework-object name,
- framework type,
- synthesis rendering title if applicable,
- framework version,
- ontology document version,
- status,
- freeze level,
- propagation sensitivity,
- domain,
- and whether the ontology is `minimal`, `expanded`, or `full`.

1. Ontology Purpose Statement

State what this ontology document is doing and what it is not doing.

Minimum requirement. The reader can tell that this is a structural rendering of a framework-object rather than a second synthesis draft.

2. Framework Identity

State:

- the framework kernel in compact form,
- the central problem the framework addresses,
- the framework-object type,
- the minimal theoretical ambition,
- the maximal intended scope,
- and the framework's current status.

3. Kernel and Architectural Spine

This section is required.

State:

- the framework kernel,
- the shortest master compression,
- the main inferential sequence,
- and what would count as loss of framework identity rather than internal revision.

4. Canonical Object Registry

List the framework's canonical theory-objects.

For each object, include:

- object ID,
- name,
- object type,
- architectural role,
- compact function,
- status,
- what it grounds,
- what depends on it,
- and propagation sensitivity.

5. Distinction Registry

List the framework's non-collapsible distinctions.

For each distinction, include:

- distinction ID,
- poles of the distinction,
- function in the framework,
- what collapse it blocks,
- status,
- and which objects or mechanisms rely on it.

6. Mechanism Registry

List the framework's core mechanisms.

For each mechanism, include:

- mechanism ID,
- name,
- input conditions,
- process schema,
- output,
- framework layers connected,
- status,
- and what objects depend on the mechanism.

7. Dependency Graph

This section is required.

State how canonical objects, distinctions, mechanisms, and interfaces depend on one another.

Each dependency entry should specify:

- source object,
- target object,
- dependency type,
- whether the dependency is necessary or supporting,

- whether it is upstream, internal, or downstream,
- and propagation sensitivity.

Dependency types may include:

- grounds
- presupposes
- clarifies
- bridges
- enables
- constrains
- exports
- inherits
- intensifies
- diagnoses

8. Framework Order

This section is required.

State:

- what is most primitive,
- what is foundational,
- what is bridge work,
- what is downstream,
- what can be removed without collapse,
- and what cannot be removed without altering framework identity.

9. Interface Map

This section is required.

State the framework's stable interface exports.

For each export, include:

- interface ID,
- statement,

- freeze level,
- who or what inherits it,
- and what would count as an interface change requiring propagation.

10. Failure Mode Map

This section is required.
List recurrent pathologies tracked across the framework.
For each failure mode, include:

- failure mode ID,
- name,
- compact definition,
- which mechanisms or distinction failures generate it,
- where it appears,
- and which theory-objects address it.

11. Burden Map

For each kernel claim, mechanism, or core architectural dependency, state:

- what burden it incurs,
- whether the burden is discharged,
- whether it is only partially discharged,
- or whether it is deferred,
- and where the burden is addressed.

12. Residue Statement

State what remains:

- unresolved,
- weakly specified,
- under-integrated,
- deferred,
- or pressure-sensitive

at framework level.

13. Regeneration Map

This section is required.

State how to regenerate from the ontology:

- the kernel sheet,
- the dependency map,
- the reader-facing synthesis paper,
- the framework lexicon,
- and the individual paper outlines or abstract skeletons where relevant.

14. Stack and Canon Relations

This section is required where the framework belongs to a larger project.

State:

- what the framework inherits from upstream projects,
- what it exports to downstream work,
- what neighboring frameworks are adjacent but not strongly dependent,
- and what changes would require propagation.

15. Drift and Revision Statement

State:

- what parts of the framework are expected to drift,
- what parts are stabilized,
- what would count as local revision only,
- what would count as architectural drift,
- and what would force reclassification or reopening.

13. Recommended Local Record Format

Framework Kernel Record

```
\paragraph{Framework Kernel}
ID: FK-01
Name: framework kernel
Object type: kernel object
Status: stabilized
```

```
Freeze level: interface-frozen
Definition: finite disclosure is selective but answerable because ...
Function: compresses framework identity
Used in: synthesis paper, kernel sheet, canon intro
Notes: changing this requires architectural review
```

Canonical Object Record

```
\paragraph{Canonical Object}
ID: CO-04
Name: Rendering to Authority Under Constraint
Object type: canonical theory-object
Architectural role: bridge
Status: stabilized
Function: explains how renderings acquire standing and become
          action-guiding authorities
Depends on: Structure in Reality, The Cut
Enables: Mediated Judgment Under Constraint
Propagation sensitivity: architectural
```

Mechanism Record

```
\paragraph{Mechanism}
ID: ME-02
Name: authority conversion
Object type: mechanism object
Status: stabilized
Input: rendering under stabilization and portability conditions
Process: embedding -> operational uptake -> decisional standing ->
         authority reinforcement
Output: operative authority over cases
Connects: disclosure layer to authority layer
```

Distinction Record

```
\paragraph{Distinction}
ID: DI-07
Poles: rendering / authority
Function: blocks collapse from selective representation to legitimate governance
Status: stabilized
Used by: RAUC, MJUC, CEUC
Blocks confusion: selective aid is already action-guiding authority
```

Failure Mode Record

```
\paragraph{Failure Mode}
ID: FM-04
Name: over-authorized abstraction
Definition: decisional standing granted beyond warranted scope
Generated by: standing + scope drift
Addressed in: RAUC, MJUC, CEUC
Status: stabilized
```

Interface Record

```
\paragraph{Interface Export}
ID: IF-03
Name: corrective necessity
Type: framework interface object
Status: interface-frozen
Definition: once authoritative renderings govern, correction becomes
            structurally necessary
Exported to: CEUC, CGUC
Propagation rule: changes require downstream review
```

14. Minimal Acceptance Test

A framework synthesis ontology document satisfies FSO-1 only if a later reader can answer all of the following without reconstructing the framework from separate papers alone:

1. What kind of framework-object is this?
2. What is its kernel?
3. What is its main inferential sequence?
4. What are its canonical objects?
5. Which objects are foundational, bridge, downstream, or auxiliary?
6. What distinctions must not collapse?
7. What mechanisms connect one layer to another?
8. What failure modes recur across the framework?
9. What does the framework export through a stable interface?
10. What burdens remain open?
11. What kinds of revision count as local drift and what kinds count as architectural change?
12. How would one regenerate the synthesis paper from the ontology?

15. Recommended Use

FSO-1 is best used:

1. once a project contains multiple theory-objects,
2. whenever the order of dependence matters,
3. whenever a reader could mistake the project for a thematic cluster rather than an architecture,
4. whenever interface changes in one object may propagate to others,
5. and whenever the framework needs a durable compression layer stronger than memory or prose summary.

A recommended workflow is:

1. stabilize or revise individual theory-objects,
2. update their AOS documents,
3. update the canonical object registry at framework level,
4. update kernel, spine, distinction, mechanism, and interface maps,
5. identify any architectural drift or propagation-sensitive change,
6. and regenerate or revise the reader-facing synthesis paper accordingly.

16. Final Statement

A synthesis paper may still be the best vehicle for readability and entry into a framework. But if the framework cannot also be rendered as an explicit ontology of kernel, canonical objects, distinctions, mechanisms, dependencies, interfaces, burdens, failure modes, residues, and regeneration paths, then too much of its identity remains trapped in prose sequence or tacit memory. FSO-1 is designed to prevent that. Its purpose is to make the framework-object itself structurally visible while still leaving room for finitude, selective stabilization, architectural revision, and disciplined incompleteness.