POLYXAN–RSVP STARSPACE
A Formal Specification for a Xanadu–RSVP Social Hyperstructure
with Multi-Galaxy Semantic Dynamics and Generative Field Algorithms

Flyxion System Specification Draft

2025

Abstract

This document specifies the Polyxan–RSVP Starspace System: a hybrid Xanadu-style hypermedia architecture, an RSVP generative substrate (Scalar–Vector–Entropy fields) driving semantic and dynamical evolution, and a starspace MMO interface in which each user occupies a partially isolated galaxy region. We define content atoms, spans, typed bidirectional links, media quines generated by a Polycompiler, semantic force embeddings, user-galaxy sheaves, RSVP field evolution, and the global g-reset operator. We give a Lagrangian formulation of RSVP fields on semantic space and their coupling to the hypergraph, a category-theoretic architecture, an operational semantics for resets, a database schema and API sketch, simulation pseudocode, UML diagrams, and verification-oriented invariants suitable for mechanized proof assistants.

Contents

1 System Overview

The Polyxan–RSVP Starspace system integrates:

- A **Xanadu-style hypergraph** of Content Atoms with fine-grain spans.
- The **RSVP generative field system**: scalar Φ, vector $\mathbf{v}$, entropy S, producing semantic gradients, cluster morphologies, and viewpoint curvature over a latent semantic manifold.
- A **semantic latent space** embedded in $\mathbb{R}^3$ as a star map, with N-body relaxation guided by RSVP fields.
- A **galaxy-shard universe**: each user u sees a localized galaxy generated as a sheaf section over the global semantic space X.
- A **ship-projection MMO layer**: users appear as anonymous triangular ships; projections are holographic and non-destructive.
- A **global reset operator** triggered by holding key **g** for five seconds, recomputing the embedding and galaxy layouts under RSVP constraints.
- **Autoblink** stability constraints to keep certain users' local patches approximately invariant through resets.

2 Core Data Types

2.1 Content Atoms

A Content Atom is the basic unit of meaning, regardless of media type. Formally:

$$\text{Atom} := \{\text{id} : \mathbb{N},\ \text{media} : M,\ \text{payload} : B,\ \text{tags} : \mathcal{P}(T),\ \text{version} \in \mathbb{N},\ \text{polyGroup} \in \mathbb{N} \cup \{\varnothing\}\}.$$

Here M is the set of media types (text, audio, video, image, code, composite), B is a blob reference (to object storage or stream), and T is a tag alphabet (topics, languages, etc.).

2.2 Spans

A Span provides fine-grained addressability.

$$\text{Span} = (\text{spanId}, \text{atomId}, s, e)$$

with $s < e$ and s, e representing byte or time offsets within the payload.

2.3 Typed Links

A Typed Link is an edge:

$$L = (\text{linkId}, \text{from}, \text{to}, \tau, \text{creator}, t)$$

where:

- from, to are Spans or Atoms.
- $\tau \in \Lambda$ is a link type (reply, critique, support, transclusion, translation, summary, remix, etc.).
- creator is the persona that authored the link.
- t is a timestamp.

We maintain bidirectionality by ensuring that for each $L : X \to Y$, adjacency structures store both outgoing and incoming references.

3 Polycompiler and Media Quines

The Polycompiler is a system service:

$$\text{Polycompile} : \text{Atom} \times \Sigma \to \text{Atom}$$

where Σ is a specification of the target modality or transformation (e.g. (*summary*, 1-minute video), (*translation*, es_MX), etc.).

All media variants $\{A_\sigma\}$ of a seed Atom A share a common polyGroup identifier g.

[Media Quine] A *media quine* is a polyGroup G of Atoms such that for every $A_i \in G$ there exists σ with $\text{Polycompile}(A_i, \sigma) \approx A_j$ for some $A_j \in G$, up to a specified semantic equivalence relation $\approx$.

4 Semantic Latent Space and Star Map

Each semantic entity x (Atom, PolyGroup, Persona, Topic cluster) has an embedding:

$$\mathbf{z}_x \in \mathbb{R}^d.$$

This embedding is constructed from:

- multimodal content encoders (f_{mm}),
- graph-structural features (typed links, centrality),
- RSVP field values $(\Phi, \mathbf{v}, S)$ at the node.

We then define a projection:

$$\pi : \mathbb{R}^d \to \mathbb{R}^3$$

which is a parametric, time-stable projection onto coordinates interpreted as star positions:

$$\mathbf{x}_x := \pi(\mathbf{z}_x).$$

An N-body relaxation step adjusts $\mathbf{x}_x$ to satisfy aesthetic and semantic constraints (e.g. cluster compactness, repulsion between blocked regions), while preserving the relative ordering implied by $\mathbf{z}_x$ and the RSVP fields.

5 Galaxy-Shard Architecture

Let X denote the global semantic manifold (the embedding space). For each user u, define a center $\mathbf{z}_u$ corresponding to the persona embedding, and an open neighborhood:

$$U_u := B_R(\mathbf{z}_u) \subset X$$

for some radius R in the latent metric. The *galaxy map* for user u is:

$$\mathcal{G}_u := \{(\mathbf{x}_x, x) \mid x \in X,\ \mathbf{z}_x \in U_u\}.$$

Semantic isolation arises because traveling from U_u to another user v's neighborhood U_v requires multiple in-game steps: ship movement at finite speed across $\mathbf{x}$-space.

6 User Ships, Projection, and Anonymity

Each persona p is displayed in its own galaxy as a triangular ship at position:

$$\mathbf{x}_p = \pi(\mathbf{z}_p).$$

When p projects into another user's galaxy, a *ghost ship* representation appears (triangle with no explicit username), subject to privacy rules.

[Holographic Projection] A holographic projection from user u to galaxy v is a morphism:

$$\mathsf{Proj}_{u \to v} : \mathcal{G}_v \to \mathcal{G}_v$$

that augments $\mathcal{G}_v$ with an anonymous ship entity s_u, allowing u to *observe* $\mathcal{G}_v$ but not mutate its contents.

All content created during projection is anchored in u's own galaxy (contexts, groups), but may link to spans originating from $\mathcal{G}_v$.

7 The g-Key Reset Operator and Autoblink

Holding key g for 5 seconds triggers a *Reset Event*:

$$\mathsf{Reset} : \Sigma \to \Sigma'$$

where Σ is the full system state: embeddings $\mathbf{z}_x$, field values $(\Phi, \mathbf{v}, S)$, layout positions $\mathbf{x}_x$, and galaxy views $\mathcal{G}_u$.

7.1 Global Reset Transformation

We can formalize reset as:

$$\mathbf{z}'_x = \mathcal{R}_z(\mathbf{z}_x, \Phi, \mathbf{v}, S), \quad \mathbf{x}'_x = \mathcal{R}_x(\mathbf{z}'_x),$$

with the following constraints:

- $\mathcal{R}_z$ respects RSVP dynamics (Section 8), so embeddings adjust according to updated fields.
- $\mathcal{R}_x$ is a new N-body relaxation seeded by $\mathbf{z}'_x$.

7.2 Autoblink Constraint

Users with `autoblink` enabled impose a local stability constraint:

$$\|\mathbf{x}'_u - \mathbf{x}_u\| \leq \epsilon$$

for some small $\epsilon > 0$. In the relaxation solver, these points become soft constraints or pinned nodes; other stars flow around them.

8 RSVP–Polyxan Lagrangian

We now define a Lagrangian for the RSVP fields over the semantic manifold X and couple it to the Polyxan content graph.

8.1 Fields and Densities

Let X be a Riemannian manifold representing semantic space with metric g. Over X we define:

$$\Phi : X \times \mathbb{R} \to \mathbb{R}, \quad \mathbf{v} : X \times \mathbb{R} \to TX, \quad S : X \times \mathbb{R} \to \mathbb{R}.$$

Define a node density $\rho : X \to \mathbb{R}_{\geq 0}$ induced by the content graph, e.g. via kernel smoothing over embeddings. Define a link curvature scalar $\kappa : X \to \mathbb{R}$ that measures non-local connectivity complexity (e.g. triangle density, motif structure).

8.2 Action Functional

We propose an action:

$$\mathcal{A}[\Phi, \mathbf{v}, S] = \int_{\mathbb{R}} dt \int_X d\mu_g \; \mathcal{L}(\Phi, \partial_t \Phi, \nabla\Phi, \mathbf{v}, \nabla\mathbf{v}, S, \nabla S; \rho, \kappa)$$

with Lagrangian density:

$$
\begin{aligned}
\mathcal{L} = &\underbrace{\frac{1}{2}(\partial_t \Phi)^2 - \frac{c_\Phi^2}{2}\|\nabla \Phi\|^2}_{\text{scalar kinetic/elastic}} \\
&+ \underbrace{\frac{1}{2}\|\partial_t \mathbf{v}\|^2 - \frac{c_v^2}{2}\|\nabla \mathbf{v}\|^2}_{\text{vector field kinetic/elastic}} \\
&+ \underbrace{\frac{1}{2}(\partial_t S)^2 - \frac{c_S^2}{2}\|\nabla S\|^2}_{\text{entropy field smoothing}} \\
&- V(\Phi, \mathbf{v}, S; \rho, \kappa),
\end{aligned}
$$

where V is a potential encoding:

- attraction of Φ to high-density regions (cluster formation),
- negentropic flows (alignment of $\mathbf{v}$ with $\nabla\Phi$),
- entropy minimization in well-structured semantic neighborhoods,
- penalties for excessive curvature κ (graph over-complexity).

Variation of $\mathcal{A}$ yields Euler–Lagrange equations for the fields, which can be discretized on the embedding graph.

8.3 Coupling to Graph Nodes

Each Atom x sits at an embedding $\mathbf{z}_x \in X$. We define the field values at node x by restriction: $\Phi_x(t) = \Phi(\mathbf{z}_x, t)$, etc. A simple discrete evolution for embeddings is:

$$
\frac{d\mathbf{z}_x}{dt} = -\alpha \nabla \Phi(\mathbf{z}_x, t) + \beta\, \mathbf{v}(\mathbf{z}_x, t) - \gamma \nabla S(\mathbf{z}_x, t),
$$

where α, β, γ are hyperparameters governing attraction to semantic wells, vector-flow drift, and entropy smoothing.

9 Category-Theoretic Architecture

We now sketch a categorical view.

9.1 Content Category

Define a category $\mathbf{C}$:

- Objects: Content Atoms and Spans.
- Morphisms: Typed Links $L : X \to Y$.

Composition is given by path concatenation when link types are composable; identity morphisms are trivial self-links.

9.2 Polycompiler as Endofunctor

The Polycompiler induces an endofunctor:

$$\mathsf{Poly} : \mathbf{C} \to \mathbf{C}$$

that:

- on objects: sends an Atom A to a PolyGroup object, or to a specific media variant A_σ.
- on morphisms: lifts links along media quine equivalences, preserving semantic type when possible.

9.3 RSVP Functor

Define a functor:

$$\mathsf{RSVP} : \mathbf{C} \to \mathbf{F}$$

where $\mathbf{F}$ is a category of field configurations, e.g.:

- objects: triples $(\Phi, \mathbf{v}, S)$ defined on finite subsets of X,
- morphisms: restriction maps and field reparameterizations.

RSVP maps content/link structure into field source terms (e.g. node densities, curvature contributions), and in turn field evolution feeds back into embedding updates.

9.4 Galaxy Sheaf

Over X we define a presheaf $\mathscr{G}$:

$$\mathscr{G}(U) = \{\text{all galaxy renderings over } U\}$$

with restriction maps $\rho_{UV} : \mathscr{G}(U) \to \mathscr{G}(V)$ for $V \subset U$.

[Sheaf Condition (Sketch)] If:

- the projection π is deterministic and smooth,
- RSVP fields are continuous on X,
- content IDs and links are globally unique,

then $\mathscr{G}$ is a sheaf: compatible local views glue uniquely to a global galaxy rendering.

Idea. Galaxy renderings are determined by $(\mathbf{z}_x, \pi)$ and field values. Compatibility on overlaps corresponds to agreeement on shared nodes and their local layout under the same π and field configuration. Uniqueness follows from determinism of the layout algorithm. $\square$

10 Operational Semantics of g-Reset

We describe small-step semantics for the g key in a simplified form.

10.1 State

Let a system state be:

$$\Sigma = (\mathcal{C}, \mathbf{Z}, \mathbf{X}, F, \mathcal{U})$$

where:

- $\mathcal{C}$ = content graph (Atoms, Spans, Links),
- $\mathbf{Z} = \{\mathbf{z}_x\}$ embeddings,
- $\mathbf{X} = \{\mathbf{x}_x\}$ layout positions,
- $F = (\Phi, \mathbf{v}, S)$ RSVP fields,
- $\mathcal{U}$ = user metadata (autoblink flags, ship positions).

10.2 Events

We introduce an event $\mathsf{GPress}(u, t)$ for a user u holding g from time t to $t + \Delta$.

We define two rules:

Broadcast Rule (short press). If $0 < \Delta < 5$s:

$$\frac{}{\Sigma \xrightarrow{\mathsf{GPress}(u,\Delta)} \Sigma'}$$

where Σ' has $\mathcal{U}'$ updated to broadcast u's current ship position in nearby galaxies for some time window, but no change to $\mathbf{Z}, \mathbf{X}, F$.

Reset Rule (long press). If $\Delta \geq 5$s:

$$\frac{}{\Sigma \xrightarrow{\mathsf{GPress}(u,\Delta)} \Sigma''}$$

where:

$$\begin{aligned}
\mathbf{Z}'' &= \mathcal{R}_z(\mathbf{Z}, F, \mathcal{C}) \\
F'' &= \mathcal{R}_F(F, \mathcal{C}) \\
\mathbf{X}'' &= \mathcal{R}_x(\mathbf{Z}'', F'', \mathcal{U}) \\
\mathcal{U}'' &= \mathcal{U} \text{ (up to transient fields)}
\end{aligned}$$

and $\mathcal{R}_x$ respects autoblink constraints by pinning or softly constraining selected user positions.

11 Database Schema and API Sketch

11.1 Relational/Core Schema (Sketch)

Tables (conceptual):

- `atoms`(`id`, `media_type`, `payload_ref`, `version`, `poly_group_id`, `created_at`, `author_id`)
- `spans`(`id`, `atom_id`, `start`, `end`)
- `links`(`id`, `from_span_id`, `to_span_id`, `link_type`, `creator_id`, `created_at`)
- `poly_groups`(`id`, `root_atom_id`)
- `personas`(`id`, `user_id`, `name`, `avatar_atom_id`)

- `embeddings`(`entity_id`, `entity_type`, `vector`)
- `galaxy_views`(`user_id`, `center_embedding`, `params`, `last_updated`)
- `rsvp_fields`(`patch_id`, `phi_params`, `v_params`, `s_params`)
- `reset_events`(`id`, `trigger_user_id`, `at_time`)

Embeddings can be stored in a vector-capable store or separate service.

11.2 API Sketch

Representative endpoints (REST or gRPC-ish):

- `GET /atoms/{id}` – fetch Atom metadata and (optionally) payload.
- `POST /atoms` – create Atom.
- `POST /links` – create Typed Link.
- `POST /polycompile` – request Polycompiler to generate variants.
- `GET /galaxy/{userId}` – fetch GalaxyView for user.
- `POST /galaxy/{userId}/project` – project to another user's galaxy.
- `POST /events/gpress` – notify backend of `g` press; backend decides whether to broadcast or reset.
- `GET /embeddings/{entityId}` – fetch embedding(s).
- `POST /rsvp/step` – advance RSVP fields and embeddings by one timestep.

12 Simulation Appendix: RSVP Dynamics on the Graph

We sketch a discrete-time simulation to evolve RSVP fields and embeddings.

12.1 Discrete State

Let $G = (V, E)$ be the content graph (nodes V = entities, edges E = links). At each node $i \in V$ we maintain:

$$\Phi_i^t, \quad \mathbf{v}_i^t, \quad S_i^t, \quad \mathbf{z}_i^t.$$

12.2 Update Equations (Example)

For time step Δt:

$$
\begin{aligned}
\Phi_i^{t+\Delta t} &= \Phi_i^t + \Delta t \left(D_\Phi \sum_{j\sim i} (\Phi_j^t - \Phi_i^t) - \lambda_\Phi \Phi_i^t + f_\Phi(\rho_i, \kappa_i) \right), \\
S_i^{t+\Delta t} &= S_i^t + \Delta t \left(D_S \sum_{j\sim i} (S_j^t - S_i^t) + f_S(\rho_i, \kappa_i) \right), \\
\mathbf{v}_i^{t+\Delta t} &= \mathbf{v}_i^t + \Delta t \left(D_v \sum_{j\sim i} (\mathbf{v}_j^t - \mathbf{v}_i^t) - \nabla \Phi_i^t - \eta \mathbf{v}_i^t \right), \\
\mathbf{z}_i^{t+\Delta t} &= \mathbf{z}_i^t + \Delta t \left(-\alpha \nabla \Phi_i^t + \beta \mathbf{v}_i^t - \gamma \nabla S_i^t \right),
\end{aligned}
$$

where $j \sim i$ denotes neighbors in the graph, and ρ_i, κ_i are local density/curvature estimates.

12.3 Pseudocode

```
for t in range(T):
    # compute local graph Laplacians, densities, curvatures
    for i in V:
        lap_Phi[i] = sum(Phi[j] - Phi[i] for j in neighbors[i])
        lap_S[i]   = sum(S[j]   - S[i]   for j in neighbors[i])
        lap_v[i]   = sum(v[j]   - v[i]   for j in neighbors[i])
        rho[i]     = density_estimate(i)
        kappa[i]   = curvature_estimate(i)

    # update fields
    for i in V:
        Phi[i] += dt * (D_Phi * lap_Phi[i] - lambda_Phi * Phi[i] + f_Phi(rho[i],kappa[
        S[i]   += dt * (D_S   * lap_S[i]   + f_S(rho[i],kappa[i]))
        v[i]   += dt * (D_v   * lap_v[i]   - grad_Phi[i] - eta * v[i])

    # update embeddings
    for i in V:
        z[i] += dt * (-alpha * grad_Phi[i] + beta * v[i] - gamma * grad_S[i])
```

The projection $\mathbf{x}_i = \pi(\mathbf{z}_i)$ and N-body relaxation are performed periodically (e.g. every K timesteps or after a reset).

13 Additional UML Sketches

13.1 Sequence: Polycompile and Reset

The following is a textual UML-style sequence sketch (you can convert to `tikz-uml` as desired):

14 Verification-Oriented Invariants and Proof Sketches

We outline properties suitable for formal verification (e.g. Coq/Lean).

14.1 Invariant: Link Bidirectionality

Property. For every stored TypedLink L with $(\text{from} = X, \text{to} = Y)$, the adjacency indices satisfy:

$$Y \in \text{succ}(X) \iff X \in \text{pred}(Y).$$

Sketch. By construction: insertion of links is atomic and updates both `succ` and `pred` indices. Deletion is symmetric. Inductive reasoning over link operations proves preservation. □

14.2 Invariant: Sheaf Compatibility of Galaxy Views

Property. For any two users u, v with $U_u \cap U_v \neq \varnothing$:

$$\rho_{U_u \cap U_v}(G_u) = \rho_{U_u \cap U_v}(G_v),$$

where $G_u \in \mathscr{G}(U_u)$, $G_v \in \mathscr{G}(U_v)$ are galaxy views derived from the same global embedding/-field configuration.

14.3 Safety Property: Reset Preserves Connectivity

Property. A g-reset does not change the content graph:

$$\mathcal{C}' = \mathcal{C},$$

i.e. Atoms, Spans, and Links are unchanged.

Sketch. By definition of Reset, only $\mathbf{Z}, \mathbf{X}, F$ are recomputed. No content insertion/deletion occurs. Thus connectivity is preserved. □

14.4 Coq/Lean-Style Theorem Template

In a Coq-like pseudo-syntax:

```
Record State := {
  C : ContentGraph;
  Z : Embeddings;
  X : Layout;
  F : Fields;
  U : UserMeta;
}.
```

```
Inductive step : State -> State -> Prop :=
| StepGShort : forall s s',
    g_press_short s s' ->
    step s s'
| StepGLong : forall s s',
    g_press_long s s' ->
    step s s'.

Theorem reset_preserves_graph :
  forall s s',
    step s s' ->
    C s' = C s.
```

A similar structure can encode invariants about autoblink pinning:

```
Definition autoblink_invariant (s s' : State) : Prop :=
  forall u, autoblink u = true ->
    dist (ship_pos s u) (ship_pos s' u) <= eps.
```

15 Conclusion

We have specified a unified architecture for Polyxan–RSVP Starspace: a Xanadu-inspired hypermedia graph coupled to RSVP fields, projected into a semantic starspace with user-local galaxies, ships, holographic projections, and a global g-reset operator. We provided a Lagrangian for the RSVP fields, a category-theoretic framing, sheaf-theoretic consistency conditions for galaxy views, an operational semantics for reset events, a database and API sketch, simulation pseudocode, UML sketches, and verification-oriented invariants. This document is intended as a scaffold for both theoretical refinement and practical implementation.