ORION External Agents Pipeline

Comprehensive Technical & Functional Documentation

---

1. Executive Summary

The ORION External Agents Pipeline is a fully automated AI-powered horizon scanning system designed for strategic foresight intelligence. It systematically monitors a curated library of 500 active sources across the global information landscape, extracting, scoring, and classifying emerging signals of change using a chain of six specialized AI agents.

The pipeline operates autonomously on a daily schedule, producing structured, schema-locked outputs that feed into the broader ORION strategic foresight platform. Its ultimate purpose is to replicate and scale the work of an expert foresight analyst team · continuously scanning the world for early indicators of technological breakthroughs, social shifts, economic disruptions, environmental developments, and policy changes that matter for long-term strategic planning.

Key Capabilities:

- Automated crawling and content extraction from 500 diverse sources
- AI-driven signal extraction, deduplication, scoring, and classification
- Synthesis of individual signals into higher-order strategic forces (Megatrends, Trends, Weak Signals, Wildcards)
- Schema-locked 20-column CSV outputs compatible with the ORION platform
- Interactive web dashboard for human review and signal promotion
- Intelligent source rotation ensuring full coverage over 10-day cycles
- Cumulative master databases that grow continuously over time

---

2. The Source Network

2.1 Library Composition

ORION draws from a professionally curated library of **500 active sources** spanning over 80 specialist platforms, research institutions, and leading publications. The library is organized in an Excel workbook (`ORION_active_sources_top500.xlsx`) and covers:

Category	Examples
Foresight Platforms	Quantumrun Foresight, Futures Platform, FutureLoop, ITONICS
Horizon Scanners	Horizon Scanner, Envisioning Radar
Leading Business Media	Fortune, Business Insider, TechCrunch, The Verge, Quartz
Research & Science	The Conversation, MIT Technology Review, Futurity, Sage Journals
Global Institutions	World Economic Forum, United Nations, European Commission, UNEP
Strategic Consultancies	Deloitte, Accenture, EY, Gartner, McKinsey, Oliver Wyman
Industry Intelligence	CB Insights, Statista, S&P Global, Morningstar, Deutsche Bank
Innovation Think Tanks	Copenhagen Institute for Futures Studies, SITRA, Atlantic Council
Trend Agencies	Trend Hunter, TrendOne, Foresight Factory, VML Intelligence
Sector Specialists	HubSpot, Pinterest, Coursera, Hootsuite, IPSOS

Sources are selected based on their historical contribution to foresight databases, ensuring breadth across geographies, industries, and disciplines.

2.2 Intelligent Source Rotation

Rather than overwhelming all 500 sources daily, ORION uses an intelligent rotation system:

- **50 sources per day**, cycling through the full library over a **10-day rotation cycle**
- The rotation offset advances by 50 after each run, ensuring every source is scanned exactly once per cycle
- Rotation state is persisted in `out/rotation_state.json`, tracking the last offset and date
- The batch size is configurable via the `ORION_BATCH_SIZE` environment variable (default: 50)
- A **full sweep** mode is available for weekly comprehensive scans of all 500 sources

Rotation Logic:

- Day 1: Sources 1·50
- Day 2: Sources 51·100
- Day 3: Sources 101·150
- ...
- Day 10: Sources 451·500
- Day 11: Cycle restarts at sources 1·50

If the rotation wraps past the end of the list, it seamlessly continues from the beginning (circular rotation).

---

3. The Intelligent Collector

3.1 How Content Collection Works

Before any AI agent engages, the Collector module (`src/collector.py`) crawls the daily batch of sources and extracts readable article content. The collector is designed to handle the diversity of modern web publishing · from well-structured RSS feeds to complex JavaScript-heavy websites.

Collection Process per Source:

1. **RSS/Atom Feed Attempt** · The collector first attempts to parse the source URL directly as an RSS or Atom feed using `feedparser`
2. **Feed Discovery** · If direct parsing fails, the collector fetches the homepage HTML and searches for `<link rel="alternate">` tags pointing to RSS/Atom feed URLs (up to 5 candidates tested)
3. **Feed Entry Processing** · For each feed entry found (up to 2 per source by default), the collector:
    - Fetches the full article page at the entry's link URL
    - Extracts readable content using the `readability-lxml` algorithm (the same technology behind browser reading modes)
    - Falls back to feed summary/content fields if the full page fetch fails
4. **Homepage Fallback** · If no feed is found, the collector extracts readable text directly from the homepage HTML
5. **Content Filtering** · Only documents with at least 400 characters of substantive text are passed forward; short snippets, navigation text, and boilerplate are automatically filtered out

Document Record: Each collected document is packaged as a structured record containing:

Field	Description
`doc_id`	Unique UUID for this document
`source_name`	Name of the source (e.g., "MIT Technology Review")
`source_url`	Original source URL from the library
`canonical_url`	Actual article URL where content was found
`published_at`	Publication date from the feed (if available)
`retrieved_at`	ISO timestamp of when ORION collected this document
`clean_text`	Extracted readable text, truncated to 8,000 characters
`content_hash`	SHA-256 hash of the clean text for content-level deduplication

3.2 Responsible Crawling

- **Rate limiting**: 2-second delay between requests to the same domain
- **Identification**: All requests use the `ORION-External-Agent/1.0` user-agent header
- **Timeout protection**: Individual page fetches timeout after 12 seconds
- **Graceful degradation**: If a source fails, the collector moves on without interrupting the pipeline

3.3 Collector Report

After collection completes, a detailed `collector_report.json` is saved to the output directory, containing per-source statistics:

- Feed URL discovered (if any)
- Number of feed entries found
- Number of documents successfully created
- Average text length
- Any errors encountered

---

4. The Intelligence Pipeline

The core of ORION is a sequential chain of specialized AI agents and processing steps. Four AI agents handle signal extraction and analysis, a code-based phase handles export and validation, and an optional sixth AI agent synthesizes strategic forces. Together, they replicate the complete workflow of an expert foresight research team · from raw article reading to structured, scored, and synthesized intelligence outputs.

All agent definitions · including system prompts, input schemas, and output schemas · are stored in `agents/prompts.json`. Each AI agent call uses OpenAI's API with strict JSON schema validation on outputs, ensuring every response conforms exactly to the expected structure. If an agent produces malformed JSON, a repair pass attempts to fix it automatically. Each AI agent also has a deterministic code-based fallback that activates if the LLM call fails, ensuring the pipeline never stops due to a single API error.

4.1 Agent 1: PATHFINDER · Signal Extraction

Purpose: Read each collected document and identify up to 5 candidate signals.

What is a Signal? A signal is an early indicator of meaningful change · a specific, falsifiable observation grounded in the source document. Signals can represent technological breakthroughs, shifting social behaviors, emerging policy directions, environmental developments, or economic disruptions.

Process:

1. Receives one collected document (clean text, source metadata)
2. Also receives the list of valid ORION dimensions and tag vocabulary for consistent classification
3. Analyzes the content and extracts 0·5 candidate signals
4. For each candidate, produces a structured record

Output per Candidate:

| Field | Description | |---|---| | `candidate_id` | Unique identifier for this candidate | | `doc_id` | Links back to the source document | | `title` | Clear, descriptive title for the signal | | `claim_summary` | What the signal claims is happening or emerging | | `why_it_matters` | Why this signal is strategically significant | | `proposed_tags` | 3·8 concise topic tags | | `proposed_steep` | STEEP classification: Social, Technological, Economic, or Environmental | | `proposed_dimension` | ORION dimension category | | `type_suggested` | Signal type: S (Signal), WS (Weak Signal), T (Trend), or WC (Wildcard) | | `evidence_snippet` | Short excerpt from the source (max 240 characters) | | `source_name` | Name of the originating source | | `canonical_url` | URL of the article | | `published_at` | Publication date | | `retrieved_at` | Retrieval timestamp | | `content_hash` |

Document content hash |

Normalization: After PATHFINDER returns, each candidate is normalized:

- Missing fields are filled from the parent document
- Tags are deduplicated and limited to 8
- STEEP values are validated against allowed values (defaults to "Technological")
- Dimensions are validated against ORION's dimension list (defaults to "Other")
- Type suggestions are validated (defaults to "S")
- Evidence snippets are ensured to be present

Model: GPT-4o-mini (optimized for speed and cost)

4.2 Agent 2: COMPARATOR · Duplicate Detection

Purpose: Compare each candidate signal against the existing ORION knowledge corpus to detect duplicates and near-duplicates.

Process:

1. For each candidate, the pipeline generates a query text from the title, claim summary, and "why it matters" fields
2. This query is compared against the ORION corpus using a lexical similarity index (combined Jaccard + Cosine similarity)
3. The top 5 nearest neighbors from the corpus are retrieved with their similarity scores
4. The COMPARATOR agent receives the candidate plus its nearest neighbors
5. The agent analyzes the similarity evidence and determines:
    - `max_similarity`: The highest similarity score found (0.0 to 1.0)
    - `nearest_orion_ids`: The IDs of the most similar existing entries (sorted by similarity)
    - `duplicate_flag`: Set to `true` if max_similarity >= 0.92 (92% similarity threshold)
    - `comparison_rationale`: Explanation of why this is or isn't a duplicate

Duplicate Threshold: 92% similarity · any candidate at or above this threshold is flagged as a duplicate and will be rejected or sent for review.

Similarity Index Methodology: The vector index uses a combined lexical similarity approach:

- Text is tokenized into alphanumeric tokens of 3+ characters
- **Jaccard similarity**: Measures overlap between token sets (intersection / union)
- **Cosine similarity**: Measures directional similarity between term frequency vectors
- Final similarity score = average of Jaccard and Cosine scores
- This approach provides fast, deterministic similarity computation without requiring external embedding APIs

Corpus Source: The comparison corpus is loaded from the existing ORION `driving_forces.xlsx` file, which contains all previously catalogued signals, trends, and forces in the ORION platform.

Model: GPT-4o-mini

4.3 Agent 3: SCORER · Quality Scoring & Decision

Purpose: Compute multi-dimensional quality scores for each candidate and make an accept/review/reject decision.

Scoring Dimensions:

Metric	Range	What It Measures
Novelty Score	0·100	How new and original is this signal compared to existing knowledge? Computed from the inverse of max_similarity using a piecewise function
Credibility Score	0·100	How reliable is the source? Based on source tier (A/B/C/D), corroboration count, and evidence quality
Relevance Score	0·100	How strategically important is this signal for foresight purposes? Based on fit to ORION dimensions, tags, and STEEP categories
Priority Index	0·100	Composite score: `0.45 × Relevance + 0.35 × Novelty + 0.20 × Credibility`
Importance Distance	1·10	Scaled impact rating mapped from the Priority Index using defined bins

Novelty Calculation (from Similarity):

- Similarity >= 0.90: Novelty drops steeply (0·15 range) · very similar to existing knowledge
- Similarity 0.70·0.90: Linear interpolation (15·85 range)
- Similarity <= 0.70: High novelty (85+ range) · genuinely new information

Credibility Tiers:

Source Tier	Base Score
A (Top-tier)	85
B (Strong)	72
C (Standard)	58
D (Lower)	35

Credibility Gates:

- If credibility < 40: Priority Index is capped at 50
- If credibility < 25: Priority Index is capped at 35

Priority Index to Importance Distance Mapping:

Priority Index	Importance Distance
0·14	1
15·24	2
25·34	3
35·44	4
45·54	5
55·64	6
65·74	7
75·84	8
85·92	9
93·100	10

Decision Thresholds:

Decision	Criteria
Accept	Priority Index >= 60 AND Credibility >= 45 AND not a duplicate
Review	Priority Index 45·59 OR Credibility 25·44 (borderline cases flagged for human review)
Reject	Priority Index < 45, OR duplicate (similarity >= 0.92)

Calibration: When there are 10 or more eligible candidates in a batch, importance distances are calibrated using relative ranking:

- Top 7% of candidates: Importance 8·10
- Next 28%: Importance 6·7
- Remaining: Importance 5 or below

Model: GPT-4o-mini

4.4 Agent 4: CURATOR · Taxonomy Normalization

Purpose: Normalize all taxonomy fields and produce ORION-row objects conforming to the strict 20-column schema required by the ORION platform.

Process:

1. Receives the candidate content, comparison results, and scores
2. Normalizes tags to a clean JSON array (3·8 tags, deduplicated)
3. Validates STEEP and dimension values against allowed enums
4. Produces two output objects:
    - `orion_row`: The canonical 20-field record matching the `driving_forces.xlsx` schema

- `staging_row`: The same 20 fields plus additional scoring and audit fields for internal tracking

ORION Row Field Mapping:

Field	Source	Default
`id`	Generated UUID	·
`project_id`	From ORION corpus	·
`title`	From curator or candidate	·
`type`	Always "S" for accepted signals	"S"
`steep`	Validated STEEP category	"Technological"
`dimension`	Validated ORION dimension	"Other"
`scope`	Always "signals"	"signals"
`impact`	Default	7.0
`ttm`	Time-to-mainstream	""
`sentiment`	Directional assessment	"Neutral"
`source`	Canonical URL of the article	·
`tags`	JSON-encoded tag array	·
`text`	Claim summary + why it matters	·
`magnitude`	Priority Index / 10 (0·10 scale)	·
`distance`	Importance Distance (1·10)	·
`color_hex`	Signal color code	"#94A3B8"
`feasibility`	Not scored in v1	null
`urgency`	Not scored in v1	null
`created_at`	ISO timestamp	·
`updated_at`	ISO timestamp	·

Model: GPT-4o-mini

4.5 Export & Validation Phase

Purpose: Perform a final quality gate by verifying schema compliance and data integrity, then write all output files.

Unlike Agents 1·4, the export phase is handled entirely by code (not an LLM call), ensuring deterministic, reliable file output. An EXPORTER agent definition exists in the agent configuration for potential future use, but the current implementation uses a code-based export pipeline for maximum reliability.

Process:

1. Collects all processed rows with their decisions (accept/review/reject)
2. Calibrates importance distances using relative ranking within the batch
3. Validates all `orion_row` objects against the 20-column schema
4. Counts accept/review/reject totals
5. Writes all output files via the export module (`src/export.py`)
6. Appends accepted signals to the cumulative master file

Export Outputs:

File	Contents
`orion_daily_all_candidates.csv`	All candidates with full scoring metadata (20 ORION columns + audit fields)
`orion_daily_accepted.csv`	Only accepted signals (strict 20-column schema)
`orion_daily_pending_review.csv`	Signals needing human review (20 columns + audit fields)
`orion_daily_rejected.csv`	Rejected/duplicate signals with rejection reasons
`collector_report.json`	Crawl statistics

Master File Appending: After export, accepted signals are automatically appended to the cumulative master file (`out/orion_master.csv`). Deduplication is enforced by checking existing IDs · no signal can appear twice in the master file.

4.6 Agent 6: SYNTHESIZER · Force Generation

Purpose: Cluster accepted signals by theme and synthesize higher-order strategic forces · transforming individual data points into actionable strategic intelligence.

This agent is optional and runs when the `--synthesize` flag is passed (enabled by default in the automated daily schedule).

Process:

1. Receives all accepted signals from the current run (plus any from the master file)

2. Groups signals into thematic clusters (maximum 10 clusters)
3. Classifies each cluster into one of four strategic force types
4. Generates one curated force per cluster with:
    - A descriptive title capturing the strategic theme
    - A synthesized text summarizing the underlying pattern
    - Normalized STEEP, dimension, and tags
    - Full traceability: list of source signal IDs that contributed

Four Force Types:

Force Type	Code	Definition	Example
Megatrend	MT	Large-scale, long-term transformative change with broad societal impact	The global shift toward renewable energy systems
Trend	T	Observable pattern of change with clear momentum and direction	Rise of AI-assisted medical diagnostics
Weak Signal	WS	Early indicator of potential change, still emerging and uncertain	Experimental neural-interface consumer devices
Wildcard	WC	Low-probability, high-impact event or discontinuity	Sudden collapse of a major global supply chain corridor

Force Color Coding:

Type	Color	Hex Code
Megatrend	Blue	#3B82F6
Trend	Green	#10B981
Weak Signal	Amber	#F59E0B
Wildcard	Red	#EF4444

Quality Rules:

- Conservative output: only forces with clear thematic coherence are generated
- Fewer, higher-quality forces are preferred over many weak ones
- A single signal can justify a Weak Signal force if it's truly novel
- Each force must trace back to at least 1 source signal
- Existing forces from the master file are provided as context to avoid creating duplicates

Force Row Schema: Each force is output in the same 20-column ORION schema as signals, with:

- `type` set to MT, T, WS, or WC (instead of "S")
- `scope` set to "megatrends", "trends", "weak_signals", or "wildcards" (instead of "signals")
- `color_hex` set to the force type color
- Source signal IDs embedded in the tags field as `synthesized_from:id1,id2,...`

Force Master File: Curated forces are appended to `out/orion_forces_master.csv` with the same ID-based deduplication as signals.

Model: GPT-4o (upgraded for higher-quality synthesis and reasoning)

---

5. Data Flow Diagram

```
····················
·   500 Sources    ·
·   (Excel File)   ·
····················
         ·
····················
·  Source Rotation  ·
·  (50/day, 10-day ·
·    cycle)         ·
····················
         ·
```

```
                ....................
                ·    Collector     ·
                ·  RSS / Scraping  ·
                ·  Content Extract ·
                ....................
                         .
                ....................
                ·   PATHFINDER     ·
                ·  (Agent 1)       ·
                ·  Extract 0-5     ·
                ·  candidates/doc  ·
                ....................
                         .
                ....................
                ·   COMPARATOR     ·
                ·  (Agent 2)       ·
                ·  Duplicate check ·
                ·  vs ORION corpus ·
                ....................
                         .
                ....................
                ·     SCORER       ·
                ·  (Agent 3)       ·
                ·  Score & Decide  ·
                ·  Accept/Review/  ·
                ·  Reject          ·
                ....................
                         .
                ....................
                ·     CURATOR      ·
                ·  (Agent 4)       ·
                ·  Normalize to    ·
                ·  20-col schema   ·
                ....................
                         .
                ....................
                ·    EXPORTER      ·
                ·  (Agent 5)       ·
                ·  Validate &      ·
                ·  Write Files     ·
                ....................
                         .
           .............................
           .             .             .
...............  ................  ................
·  Accepted   ·  ·   Pending    ·  ·   Rejected   ·
·  Signals    ·  ·   Review     ·  ·   Signals    ·
·  (.csv)     ·  ·   (.csv)     ·  ·   (.csv)     ·
...............  ................  ................
       .                .
       .         ................
       .         ·  Dashboard   ·
       .         ·  Human Review·
       .         ·  & Promotion ·
```

```
              .         ................
              .                .
              .................
              .
       ...............
       .   Master    .
       .   Signal    .
       .   File      .
       ...............
              .
       ...............
       . SYNTHESIZER .
       .  (Agent 6)  .
       .  Cluster &  .
       .  Classify   .
       ...............
              .
       ...............
       .   Curated   .
       .   Forces    .
       .   Master    .
       ...............
```

---

6. The 20-Column Schema

All outputs conform to a strict 20-column schema, ensuring seamless integration with existing ORION databases and visualization tools. This schema matches the structure of the `driving_forces.xlsx` file used by the ORION platform.

#	Column	Type	Description
1	`id`	String (UUID)	Unique identifier for this entry
2	`project_id`	String	ORION project identifier
3	`title`	String	Clear, descriptive title
4	`type`	String	Entry type: S (Signal), MT (Megatrend), T (Trend), WS (Weak Signal), WC (Wildcard)
5	`steep`	String	STEEP classification: Social, Technological, Economic, Environmental
6	`dimension`	String	ORION dimension category
7	`scope`	String	Scope: signals, megatrends, trends, weak_signals, wildcards
8	`impact`	Float	Impact rating (default: 7.0)
9	`ttm`	String	Time-to-mainstream estimate
10	`sentiment`	String	Directional assessment: Positive, Negative, Neutral
11	`source`	String	Source URL where the signal was found
12	`tags`	String (JSON)	JSON array of topic tags
13	`text`	String	Full descriptive text (claim summary + significance)
14	`magnitude`	Float	Strength indicator: Priority Index / 10 (0.0·10.0)
15	`distance`	Integer	Importance distance: 1 (low) to 10 (transformative)
16	`color_hex`	String	Display color: #94A3B8 (signals), #3B82F6 (MT), #10B981 (T), #F59E0B (WS), #EF4444 (WC)
17	`feasibility`	Float/null	Feasibility score (reserved for future use)
18	`urgency`	Float/null	Urgency score (reserved for future use)
19	`created_at`	String (ISO)	Creation timestamp
20	`updated_at`	String (ISO)	Last update timestamp

---

7. Output Deliverables

7.1 Daily Outputs (per scan date)

Each pipeline run produces a dated folder under `out/YYYY-MM-DD/` containing:

File	Description
`orion_daily_all_candidates.csv`	Complete candidate list with all 20

ORION columns plus scoring audit fields (novelty, credibility, relevance, priority index, similarity scores, decision rationale, evidence snippets) | | `orion_daily_accepted.csv` | High-confidence signals that passed all quality gates · strict 20-column schema only| |`orion_daily_pending_review.csv` | Borderline signals flagged for human expert review · includes audit fields for informed decision-making| |`orion_daily_rejected.csv` | Filtered entries with rejection reasons (duplicate, low quality) · preserved for audit purposes| | `collector_report.json` | Detailed crawl statistics: sources reached, feeds discovered, documents created, errors, text lengths | | `orion_forces_accepted.csv` | Curated strategic forces generated from accepted signals (20-column schema) | | `orion_forces_all_candidates.csv` | Forces with additional synthesis metadata (source signal IDs, synthesis rationale) |

7.2 Cumulative Master Files

| File | Description | |---|---| | `out/orion_master.csv` | All accepted signals from all runs, appended automatically. Grows continuously over time. Duplicates prevented by ID-based deduplication. | | `out/orion_forces_master.csv` | All curated strategic forces from all runs. Same deduplication protection. |

7.3 Run Logs

Each pipeline run generates a detailed JSONL log file in the `Logs/` directory:

- Run start/end timestamps
- OpenAI version and capabilities detected
- Collector summary (sources scanned, documents fetched, failures)
- LLM call estimates and actual calls made
- Per-agent success/error events with timing
- Runtime budget tracking
- Synthesis results summary

---

8. The Review Dashboard

ORION includes a Streamlit-based web dashboard (`dashboard.py`) for human oversight, signal review, and promotion. The dashboard runs on port 5000 and is accessible via a secure web URL · no software installation required.

8.1 Dashboard Features

Date Selection

- Browse results from any previous scan date using a dropdown selector
- Dates are automatically discovered from the output directory structure

Pending Signal Review

- Interactive data table showing all pending signals with key fields:
    - Title, STEEP category, Dimension, Priority Index, Credibility Score, Source URL
- Checkbox selection for individual signals
- "Select All" toggle for batch operations
- Visual distinction between signals not yet promoted and already-promoted signals

Signal Promotion

- **Promote Selected**: Accept individual signals and add them to the master database
- **Promote All**: Batch-accept all pending signals

- Duplicate protection: signals already in the master file cannot be promoted twice
- Real-time feedback on promotion success

Master File Statistics

- Total signals in the master database
- Breakdown by STEEP category (Social, Technological, Economic, Environmental)
- Breakdown by dimension (top 10)

Curated Forces View

- Forces grouped by type with visual indicators:
    - Megatrends (blue)
    - Trends (green)
    - Weak Signals (amber)
    - Wildcards (red)
- Count metrics for each force type
- Expandable force details showing:
    - Synthesized description text
    - Dimension and STEEP classification
    - Full traceability to source signals (with signal titles and source URLs)

---

9. Automated Daily Operation

9.1 Scheduling

The pipeline is deployed as a **scheduled deployment** on the Replit platform:

- **Default schedule**: 03:00 GMT daily
- **Immediate startup**: If no completed run is detected for the current day (checked via `collector_report.json`), the pipeline triggers automatically on startup
- **Safety timeout**: 2.5 hours (9,000 seconds, configurable via `ORION_MAX_RUNTIME_SECONDS`) · ensures graceful completion before the 3-hour platform timeout
- **Runtime budget**: The pipeline checks elapsed time after each document and stops processing new documents if the budget is exceeded, while still completing export and synthesis for already-processed candidates

9.2 What Happens During a Daily Run

1. **Startup Check** · Determines if a run has already completed today
2. **Source Selection** · Loads the next batch of 50 sources from the rotation schedule
3. **Collection** · Crawls sources, discovers feeds, extracts article content
4. **Signal Extraction** · PATHFINDER reads each document and extracts candidates
5. **Duplicate Detection** · COMPARATOR checks each candidate against the corpus
6. **Quality Scoring** · SCORER computes scores and makes accept/review/reject decisions
7. **Taxonomy Normalization** · CURATOR produces 20-column ORION rows
8. **Export** · EXPORTER validates and writes all output files
9. **Master Append** · Accepted signals are added to the cumulative master file
10. **Synthesis** · SYNTHESIZER clusters accepted signals and generates curated forces
11. **Forces Master Append** · Curated forces are added to the forces master file

9.3 Failure Handling

- **Individual source failures**: The collector continues processing remaining sources
- **Individual document failures**: The pipeline logs the error and moves to the next document
- **API rate limits**: Automatic retry with exponential backoff (built into the OpenAI client)
- **JSON malformation**: Automatic repair pass using a secondary LLM call
- **Runtime budget exceeded**: Graceful stop · all already-processed candidates are exported normally

---

10. Data Quality & Governance

Principle	Implementation
Schema Integrity	Strict 20-column format enforced at every stage via JSON Schema validation; no malformed records pass through
Duplicate Prevention	92% similarity threshold for near-duplicate detection; ID-based deduplication in master files
Source Attribution	Every signal traces back to its original source URL, publication date, and retrieval timestamp
Human-in-the-Loop	Borderline signals are flagged for expert review rather than auto-accepted or silently discarded
Audit Trail	Complete candidate files, rejection reasons, and collection reports preserved for every run
Provenance	Curated forces link directly to the individual signals that informed their synthesis
Content Hashing	SHA-256 hashes of document text enable content-level deduplication across runs
JSON Repair	Malformed LLM outputs are automatically repaired using a secondary model call with schema guidance
Calibrated Scoring	Importance distances are calibrated using relative ranking within each batch to ensure consistent distribution

---

11. Technical Specifications

Specification	Detail
Language	Python 3.11
AI Models	GPT-4o-mini (Agents 1·5), GPT-4o (Agent 6 · Synthesizer)
AI Provider	OpenAI API via Replit AI Integrations
Source Library	500 curated sources across 80+ platforms
Daily Coverage	50 sources per day, full rotation every 10 days
Processing Capacity	Up to 2 documents per source per cycle
Max Candidates	Up to 5 signals extracted per document
Signal Schema	20-column ORION-compatible CSV format
Force Types	Megatrends (MT), Trends (T), Weak Signals (WS), Wildcards (WC)
Quality Scoring	Multi-dimensional: Novelty, Credibility, Relevance, Priority Index
Duplicate Detection	Jaccard + Cosine lexical similarity index with 92% threshold
Operation Mode	Fully automated daily schedule with optional human review
Dashboard	Streamlit web application, always accessible
Data Formats	CSV (schema-locked), JSON (collection reports), JSONL (run logs)
Rate Limiting	2 seconds per domain for crawling
Request Timeout	12 seconds per page fetch
Min Document Length	400 characters of clean text
Max Text per Document	8,000 characters
Safety Timeout	2.5 hours (configurable)
Platform Timeout	3 hours (Replit scheduled deployment limit)

---

12. Configuration & Environment Variables

Variable	Default	Description
`ORION_BATCH_SIZE`	50	Number of sources per daily rotation batch
`ORION_MAX_RUNTIME_SECONDS`	9000 (2.5h)	Safety timeout to prevent platform kill
`OPENAI_MODEL`	gpt-4o-mini	Model used for agents 1·5
`OPENAI_SYNTHESIZER_MODEL`	gpt-4o	Model used for Agent 6 (Synthesizer)
`AI_INTEGRATIONS_OPENAI_API_KEY`	(managed)	OpenAI API key (managed by Replit AI Integrations)
`AI_INTEGRATIONS_OPENAI_BASE_URL`	(managed)	OpenAI base URL (managed by Replit AI Integrations)

CLI Arguments

Argument	Default	Description
`--date`	Today	Date string YYYY-MM-DD
`--rotate`	Off	

Enable source rotation (50 sources/day) | | `--full-sweep` | Off | Process all 500 sources | | `--max_sources` | 100 | Limit sources when not using rotation | | `--max_docs_per_source` | 2 | Max documents per source | | `--max_docs_total` | 50 | Total document limit | | `--max_candidates_per_doc` | 5 | Max signals extracted per document | | `--max_runtime_seconds` | 0 (no limit) | CLI-level runtime limit | | `--synthesize` | Off | Run force synthesis after signal extraction |

---

13. What ORION Accomplishes

Continuous Intelligence Generation

ORION transforms the manual, time-intensive process of horizon scanning into a continuous, automated intelligence service. A task that would require a team of analysts spending hundreds of hours reviewing sources is compressed into a 30·60 minute automated daily run.

Structured, Actionable Outputs

Rather than producing unstructured reports, ORION delivers structured data in a standardized format that integrates directly with existing foresight platforms, visualization tools, and strategic planning workflows.

Multi-Layer Intelligence

ORION operates at two levels:

1. **Signal Level** · Individual observations of change, scored and classified
2. **Force Level** · Higher-order strategic themes synthesized from clusters of related signals, providing the "so what" that connects individual data points into strategic narratives

Growing Knowledge Base

The cumulative master files grow continuously, building a comprehensive foresight intelligence library over time. Each daily run adds new signals and forces while preventing duplication, creating an ever-expanding strategic knowledge base.

Human-AI Collaboration

ORION doesn't replace human judgment · it augments it. The system automates the high-volume, repetitive scanning work while routing borderline cases to human experts through the review dashboard. This ensures the quality and strategic relevance that only experienced analysts can provide, applied efficiently to the most impactful decisions.

Full Traceability

Every output in ORION can be traced back to its origin:

- Forces trace to the signals that informed them
- Signals trace to the source articles they were extracted from
- Articles trace to their publication URLs, dates, and retrieval timestamps

This complete provenance chain ensures transparency, auditability, and confidence in the intelligence produced.

---

14. Agent Registry

14.1 Overview

The ORION Agent Registry (`agents/orion_agent_registry.json`) is a centralized, machine-readable catalog of every AI agent in the ORION ecosystem. It serves as the single source of truth for agent configurations, runtime metadata, lifecycle tracking, and integration mapping · enabling cross-platform governance, auditing, and discovery of all agents regardless of where they are deployed.

14.2 Registry Structure

The registry file follows a standardized JSON format:

```
{
  "registry_name": "ORION Agent Registry",
  "registry_version": "v1.0",
  "created_at": "2026-02-14T12:42:30Z",
  "total_agents": 8,
  "agents": [ ... ]
}
```

14.3 Agent Entry Schema

Each agent entry contains the following sections:

Section	Fields	Purpose
Identity	`agent_name`, `version`, `role`	Unique identification and purpose description
Prompts	`system_prompt`	Full system prompt text defining agent behavior
Schemas	`input_schema`, `output_schema`	JSON Schema definitions for input/output contracts
Runtime	`app`, `page_environment`, `trigger_schedule`, `model`, `tools_used`, `execution_mode`	Deployment context · where, when, and how the agent runs
Lifecycle	`owner`, `status`, `last_updated`	Operational status and ownership tracking
Integration Map	`upstream_inputs`, `downstream_outputs`	Data flow · what feeds into and out of this agent
Notes	`notes`	Key behavioral details, thresholds, and caveats

14.4 Registered Agents

The registry contains **8 agents** across two deployment contexts:

Pipeline Agents (6) · ORION External Agents Pipeline:

#	Agent Name	Model	Execution Mode	Status
1	ORIONPATHFINDER	gpt-4o-mini	LLM call with code-based fallback	Active
2	ORIONCOMPARATOR	gpt-4o-mini	LLM call with code-based fallback	Active
3	ORIONSCORER	gpt-4o-mini	LLM call with code-based fallback	Active
4	ORIONCURATOR	gpt-4o-mini	LLM call with code-based fallback	Active
5	ORIONEXPORTER	None (code-based)	Code-based only	Active
6	ORIONSYNTHESIZER	gpt-4o	LLM call (optional phase)	Active

Platform Copilot Agents (2) · ORION Main App (Placeholders):

#	Agent Name	Status	Notes
7	ORION_SCANNING_COPILOT	Placeholder	Environmental scanning chatbot · config to be populated from ORION platform
8	ORION_GENERAL_COPILOT	Placeholder	General foresight exploration chatbot · config to be populated from ORION platform

14.5 Integration Mapping

The registry captures the complete data flow between agents:

```
Sources (500) · Collector · PATHFINDER · COMPARATOR · SCORER · CURATOR ·
EXPORTER · CSV Files
```

```
                    SYNTHESIZER · Forces
```

Each agent's `upstream_inputs` and `downstream_outputs` fields document exactly what data enters and leaves, enabling dependency analysis, impact assessment, and pipeline optimization.

14.6 Use Cases

- **Governance**: Track which models are used, by whom, and with what permissions
- **Auditing**: Verify agent configurations match documented specifications
- **Discovery**: Find agents by capability, model, or deployment context
- **Migration**: Plan model upgrades by understanding agent dependencies
- **Monitoring**: Identify active vs. deprecated agents across the ecosystem

---

15. Source Code Repository

15.1 GitHub Repository

The complete ORION External Agents Pipeline source code is hosted on GitHub:

Repository: https://github.com/soeirocarvalho/SCANNING_AGENTS

15.2 Repository Contents

The repository contains all pipeline source code, agent definitions, configuration files, and documentation:

Directory/File	Description
`src/`	Core Python modules (pipeline, collector, synthesis, export, etc.)
`agents/`	Agent prompts (`prompts.json`) and registry (`orion_agent_registry.json`)
`scripts/`	Utility scripts (schema validation, signal promotion)
`inputs/`	Source library (`ORION_active_sources_top500.xlsx`)
`docs/`	Technical documentation (this document + PDF version)
`dashboard.py`	Streamlit web dashboard
`start.py`	Startup entry point (dashboard + pipeline trigger)
`run_daily.py`	CLI entry point for manual pipeline runs
`scheduler.py`	APScheduler configuration for VM deployment mode
`requirements.txt`	Python package dependencies
`pyproject.toml`	Project metadata and UV package configuration

15.3 Large Files

The following file is not included in the repository due to GitHub's file size limits:

File	Size	Description
`inputs/driving_forces.xlsx`	64 MB	ORION corpus containing 3,000+ curated forces and 30,000+ signals

This file must be obtained separately and placed in the `inputs/` directory before running the pipeline. It serves as the comparison corpus for duplicate detection (Agent 2: COMPARATOR) and provides the reference dimensions, tags, and project ID used throughout the pipeline.
