API Specification Document

Sovereign AI Infrastructure: Bicameral Validator Ladder

Document Version:	1.0
Date:	February 11, 2026
Status:	Draft for Review
Owner:	API Lead / Backend Architect
API Version:	v1.0
Base URL:	http://localhost:8000/api/v1

Table of Contents

1. Executive Summary

This document defines the complete API specification for the Sovereign AI Infrastructure, a local, multi-model AI orchestration system featuring a Bicameral Validator Ladder architecture. The API provides programmatic access to:

- **Inference**: Submit tasks for AI processing with intelligent routing
- **Routing**: Query routing decisions and domain classification
- **Validation**: Request validation of generated content
- **Memory Ledger**: Access project state, scratchpad, and audit trails
- **System Status**: Monitor system health, model status, and resource utilization
- **Build Pipeline**: Trigger and monitor Constitutional Build Pipeline executions

> **Design Philosophy:** The API follows RESTful principles with JSON payloads. All endpoints are designed for local deployment with zero external dependencies, ensuring complete data sovereignty.

2. API Overview

2.1 Base URL

```
http://localhost:8000/api/v1
```

2.2 Content Type

All requests and responses use `application/json` unless otherwise specified.

2.3 HTTP Methods

Method	Usage
GET	Retrieve resources (status, history, ledger)
POST	Create resources (submit inference, trigger build)
PUT	Update resources (modify configuration)
DELETE	Remove resources (clear scratchpad)

2.4 Common Response Structure

```
{
  "success": true,
  "data": { ... },
  "meta": {
    "request_id": "req_abc123",
    "timestamp": "2026-02-11T10:30:00Z",
    "duration_ms": 1250
  }
}
```

---

3. Authentication

3.1 Authentication Method

The API supports token-based authentication via API keys passed in the request header.

```
Authorization: Bearer <api_key>
```

3.2 API Key Header (Single-User Mode)

For single-user deployments, authentication is optional. When enabled:

```
X-API-Key: <your_api_key>
```

3.3 Rate Limiting Headers

Header	Description
X-RateLimit-Limit	Maximum requests allowed per window
X-RateLimit-Remaining	Remaining requests in current window
X-RateLimit-Reset	Unix timestamp when limit resets

---

4. Core Endpoints

4.1 Inference API

> **POST** `/infer`

Description: Submit a task for AI processing. The system will route to the appropriate specialist model based on domain classification and stakes assessment.

Request Body

```
{
  "request": "string (required) - The user task description",
  "context": {
    "session_id": "string (optional) - Session identifier for continuity",
    "domain_hint": "string (optional) - Suggested domain override",
    "stakes_override": "string (optional) - low|medium|high",
    "model_override": "string (optional) - Force specific model",
    "validation_policy": "string (optional) - none|end_stage|block_by_block"
  },
  "attachments": [
    {
      "type": "string (required) - document|image",
      "content": "base64-encoded content or file path",
      "filename": "string (required)"
    }
  ],
  "options": {
    "stream": false,
    "max_tokens": 4096,
    "temperature": 0.7
  }
}
```

Response (200 OK)

```
{
  "success": true,
  "data": {
    "task_id": "task_abc123",
    "status": "completed",
    "routing": {
      "domain": "coding_architecture",
      "stakes": "high",
      "recommended_model": "qwen_coder_32b",
      "confidence": 0.92,
      "reasoning": "Keywords indicate architectural refactoring; high complexity
suggests high stakes"
    },
    "output": {
      "content": "Generated response text...",
```

```
      "format": "markdown"
    },
    "validation": {
      "policy": "block_by_block",
      "results": [
        {
          "block": 1,
          "verdict": "PASS",
          "checks": ["syntax", "logic", "security"]
        }
      ]
    },
    "tools_used": ["embeddings"],
    "grounding": {
      "sources": [
        {
          "type": "ocr",
          "page": 1,
          "excerpt": "..."
        }
      ]
    }
  },
  "meta": {
    "request_id": "req_xyz789",
    "timestamp": "2026-02-11T10:30:00Z",
    "duration_ms": 45230,
    "tokens_generated": 1250
  }
}
```

Error Responses

Status	Code	Description
400	INVALID_REQUEST	Missing or invalid request parameters
422	ROUTING_FAILED	Unable to classify request domain
503	MODEL_UNAVAILABLE	Requested model not loaded or OOM
504	TIMEOUT	Task exceeded maximum execution time

> **GET** `/infer/{task_id}`

Description: Retrieve the status and result of a previously submitted inference task.

Path Parameters

Parameter	Type	Description
task_id	string	Unique task identifier

Response (200 OK)

```
{
  "success": true,
  "data": {
    "task_id": "task_abc123",
    "status": "completed|processing|failed|queued",
    "progress": {
      "current_phase": "validation",
      "percent_complete": 75
    },
    "result": { ... },
    "error": null
  }
}
```

4.2 Routing API

> **POST** `/route/analyze`

Description: Analyze a request to determine routing decision without executing inference. Useful for debugging and pre-flight checks.

Request Body

```
{
  "request": "string (required) - The user task description"
}
```

Response (200 OK)

```
{
  "success": true,
  "data": {
    "domain": "coding_architecture",
    "stakes": "high",
    "recommended_model": "qwen_coder_32b",
    "validation_policy": "block_by_block",
    "tools_required": ["embeddings"],
    "confidence": 0.92,
```

```
    "reasoning": "Keywords indicate architectural refactoring; high complexity
suggests high stakes",
    "alternatives": [
      {
        "model": "gpt_oss_20b",
        "confidence": 0.67
      }
    ],
    "uncertainty": false
  }
}
```

GET `/route/domains`

Description: List all available domain classifications.

Response (200 OK)


```
{
  "success": true,
  "data": {
    "domains": [
      {
        "id": "coding_architecture",
        "name": "Coding (Architecture)",
        "description": "System design, refactoring, patterns, modularity",
        "keywords": ["refactor", "architecture", "design pattern",
"microservice"]
      },
      {
        "id": "coding_implementation",
        "name": "Coding (Implementation)",
        "description": "Writing functions, classes, debugging, optimization",
        "keywords": ["implement", "write function", "debug", "optimize"]
      },
      {
        "id": "reasoning",
        "name": "Reasoning & Planning",
        "description": "Logical analysis, strategy, comparison, problem-
solving",
        "keywords": ["analyze", "evaluate", "plan", "strategy", "compare"]
      },
      {
        "id": "creative",
        "name": "Creative Writing",
        "description": "Narrative, stylistic, open-ended content generation",
        "keywords": ["story", "blog", "narrative", "poem", "style"]
      },
      {
        "id": "documentation",
        "name": "Documentation",
```

```
        "description": "Explain code, write manuals, summaries, specifications",
        "keywords": ["document", "readme", "explain", "guide", "summary"]
      }
    ]
  }
}
```

4.3 Validation API

> **POST** `/validate`

Description: Validate content against specified criteria using the Validator model.

Request Body

```
{
  "content": "string (required) - Content to validate",
  "content_type": "string (required) - code|text|markdown",
  "validation_type": "string (required) - syntax|logic|security|policy|
comprehensive",
  "context": {
    "domain": "coding_architecture",
    "stakes": "high",
    "constraints": ["max_latency_ms: 100", "no_external_calls"]
  }
}
```

Response (200 OK)

```
{
  "success": true,
  "data": {
    "validation_id": "val_abc123",
    "verdict": "PASS|FAIL|PARTIAL",
    "checks": [
      {
        "check": "syntax",
        "status": "PASS",
        "details": "No syntax errors detected"
      },
      {
        "check": "logic",
        "status": "FAIL",
        "details": "Missing edge case handling for negative input",
        "location": "line 45"
      }
    ],
    "confidence": 0.88,
    "reasoning": "Code structure is sound but missing input validation"
```

```
  }
}
```

4.4 Memory Ledger API

> **GET** `/ledger/project-state`

Description: Retrieve the current project state from the memory ledger.

Response (200 OK)

```
{
  "success": true,
  "data": {
    "project_name": "Sovereign AI Infrastructure",
    "current_build": {
      "build_id": "build_20260206_001",
      "status": "Phase 7 - Implementation",
      "current_service": "payment_processor",
      "progress": "3/5 services complete"
    },
    "committed_components": [
      {
        "name": "payment_validator",
        "committed_at": "2026-02-06T10:30:00Z",
        "spec": "payment_validator_spec.md",
        "tests_status": "PASS",
        "validation": "PASS"
      }
    ],
    "global_constraints": [
      "Max VRAM: 16GB",
      "Single GPU constraint",
      "CPU validation required"
    ]
  }
}
```

> **GET** `/ledger/scratchpad`

Description: Retrieve the active scratchpad content (current reasoning and pending steps).

Query Parameters

Parameter	Type	Description
session_id	string	Filter by session
limit	integer	Number of entries (default: 50)

> **GET** `/ledger/audit-trail`

Description: Retrieve the complete audit trail for compliance and debugging.

Query Parameters

Parameter	Type	Description
start_date	string	ISO 8601 date (e.g., 2026-02-01)
end_date	string	ISO 8601 date
task_id	string	Filter by specific task
format	string	json|markdown|pdf

> **DELETE** `/ledger/scratchpad`

Description: Clear the scratchpad (requires confirmation).

Request Body

```
{
  "confirm": true,
  "reason": "string (optional) - Reason for clearing"
}
```

4.5 System Status API

> **GET** `/status`

Description: Check overall system health and status.

Response (200 OK)


```
{
  "success": true,
  "data": {
    "system_status": "healthy",
    "version": "1.0.0",
    "uptime_seconds": 86400,
    "components": {
      "router": {
        "status": "loaded",
        "model": "granite_micro_3b",
        "memory_mb": 6144
      },
      "validator": {
        "status": "loaded",
        "model": "granite_h_small",
        "memory_mb": 20480
      },
      "worker": {
        "status": "loaded",
        "model": "qwen_coder_32b",
        "vram_mb": 16384,
        "gpu_utilization": 45
      }
    },
    "resources": {
      "cpu_percent": 25,
      "ram_used_gb": 32,
      "ram_total_gb": 128,
      "vram_used_gb": 14,
      "vram_total_gb": 16
    }
  }
}
```

GET `/status/models`

Description: List all available models and their status.

Response (200 OK)


```
{
  "success": true,
  "data": {
    "models": [
      {
        "id": "qwen_coder_32b",
```

```
        "name": "Qwen Coder 32B",
        "type": "worker",
        "status": "loaded",
        "location": "gpu",
        "vram_mb": 16384,
        "capabilities": ["coding_architecture", "coding_implementation"],
        "proficiency": {
          "coding_architecture": 0.95,
          "coding_implementation": 0.85
        }
      },
      {
        "id": "nemotron_30b",
        "name": "Nemotron-3 Nano 30B",
        "type": "worker",
        "status": "warm_pool",
        "location": "ram",
        "vram_mb": 15360,
        "capabilities": ["coding_implementation"],
        "proficiency": {
          "coding_implementation": 0.95
        }
      },
      {
        "id": "gpt_oss_20b",
        "name": "GPT-OSS 20B",
        "type": "worker",
        "status": "unloaded",
        "location": "disk",
        "capabilities": ["reasoning", "coding_architecture"]
      },
      {
        "id": "granite_micro_3b",
        "name": "Granite-Micro 3B",
        "type": "router",
        "status": "loaded",
        "location": "cpu",
        "ram_mb": 6144
      },
      {
        "id": "granite_h_small",
        "name": "Granite-H-Small",
        "type": "validator",
        "status": "loaded",
        "location": "cpu",
        "ram_mb": 20480
      }
    ],
    "warm_pool": ["nemotron_30b"],
    "model_vault_path": "/opt/sovereign-ai/models/"
  }
}
```

POST `/status/models/{model_id}/load`

Description: Load a specific model into GPU VRAM.

Path Parameters

Parameter	Description
model_id	Model identifier (e.g., qwen_coder_32b)

Response (202 Accepted)

```
{
  "success": true,
  "data": {
    "operation_id": "op_load_abc123",
    "status": "loading",
    "estimated_seconds": 5
  }
}
```

GET `/history`

Description: Retrieve task history.

Query Parameters

Parameter	Type	Description
limit	integer	Number of tasks (default: 20, max: 100)
offset	integer	Pagination offset
domain	string	Filter by domain
status	string	Filter by status

4.6 Build Pipeline API

POST `/build`

Description: Trigger a Constitutional Build Pipeline execution.

Request Body

```
{
  "package_name": "string (required) - Name of the package to build",
  "requirements": "string (required) - Package requirements description",
  "options": {
    "start_phase": "ideation (optional) - Phase to start from",
    "end_phase": "ratification (optional) - Phase to end at",
    "skip_validation": false,
    "max_retries_per_block": 3
  },
  "context": {
    "prd_reference": "string (optional) - Reference to PRD document",
    "existing_codebase": "string (optional) - Path to existing code"
  }
}
```

Response (202 Accepted)

```
{
  "success": true,
  "data": {
    "build_id": "build_20260211_001",
    "status": "started",
    "current_phase": "ideation",
    "phases": [
      { "name": "ideation", "status": "in_progress" },
      { "name": "architecture", "status": "pending" },
      { "name": "flowchart", "status": "pending" },
      { "name": "module_definition", "status": "pending" },
      { "name": "constraints_checkpoint", "status": "pending" },
      { "name": "test_definition", "status": "pending" },
      { "name": "api_definition", "status": "pending" },
      { "name": "implementation", "status": "pending" },
      { "name": "assembly", "status": "pending" },
      { "name": "ratification", "status": "pending" }
    ],
    "estimated_duration_minutes": 30
  }
}
```

GET `/build/{build_id}`

Description: Get build status and progress.

Response (200 OK)

```
{
  "success": true,
  "data": {
    "build_id": "build_20260211_001",
    "package_name": "payment_processor",
    "status": "in_progress|completed|failed",
    "current_phase": "implementation",
    "progress": {
      "phase": 7,
      "total_phases": 10,
      "percent_complete": 65
    },
    "phases": [
      {
        "name": "ideation",
        "status": "completed",
        "completed_at": "2026-02-11T10:00:00Z",
        "artifacts": ["ideation_notes.md", "scope_draft.md"]
      },
      {
        "name": "implementation",
        "status": "in_progress",
        "current_service": "payment_validator",
        "services_completed": 2,
        "services_total": 5
      }
    ],
    "artifacts": {
      "sad_lite": "package_sad.md",
      "component_list": "component_list.md",
      "execution_graph": "execution_graph.json"
    }
  }
}
```

POST `/build/{build_id}/cancel`

Description: Cancel an in-progress build.

GET `/build/{build_id}/artifacts`

Description: List all artifacts generated by a build.

> **GET** `/build/{build_id}/artifacts/{artifact_path}`

Description: Download a specific build artifact.

Response

Returns the artifact content with appropriate Content-Type header.

---

5. Data Models

5.1 Domain Classification

Field	Type	Description
domain	string	Domain identifier (coding_architecture, coding_implementation, reasoning, creative, documentation, unknown)
confidence	float	Classification confidence (0.0 - 1.0)
reasoning	string	Human-readable explanation
keywords_matched	array	List of matched keywords

5.2 Stakes Assessment

Field	Type	Description
stakes	string	low, medium, or high
complexity_score	integer	Calculated complexity (0-10+)
risk_score	integer	Calculated risk (0-10+)
indicators	array	Keywords that triggered assessment

5.3 Routing Decision

Field	Type	Description
domain	string	Classified domain
stakes	string	Assessed stakes level
recommended_model	string	Selected model identifier

validation_policy	string	none, end_stage, or block_by_block
tools_required	array	List of required tools (ocr, vision, embeddings)
confidence	float	Overall routing confidence
reasoning	string	Explanation of decision
alternatives	array	Alternative model suggestions with confidence
uncertainty	boolean	True if confidence below threshold

5.4 Validation Result

Field	Type	Description
verdict	string	PASS, FAIL, or PARTIAL
checks	array	List of individual check results
confidence	float	Validator confidence
reasoning	string	Explanation of verdict
corrections	array	Suggested corrections (if FAIL)

5.5 Task Status

Status	Description
queued	Task waiting to be processed
processing	Task actively being processed
validating	Output being validated
completed	Task finished successfully
failed	Task failed (see error details)
cancelled	Task was cancelled

---

6. Error Handling

6.1 Error Response Format

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error description",
    "details": { ... },
    "request_id": "req_abc123"
  },
  "meta": {
    "timestamp": "2026-02-11T10:30:00Z"
  }
}
```

6.2 Error Codes

Code	HTTP Status	Description
INVALID_REQUEST	400	Missing or malformed request parameters
UNAUTHORIZED	401	Invalid or missing API key
FORBIDDEN	403	Insufficient permissions
NOT_FOUND	404	Requested resource not found
ROUTING_FAILED	422	Unable to classify request domain
VALIDATION_FAILED	422	Content failed validation checks
MODEL_UNAVAILABLE	503	Requested model not available
GPU_OOM	503	GPU out of memory
TIMEOUT	504	Request exceeded time limit
INTERNAL_ERROR	500	Unexpected server error

7. Rate Limiting

7.1 Default Limits

Endpoint Category	Limit	Window
Inference (/infer)	10	per minute
Routing (/route/*)	60	per minute
Validation (/validate)	30	per minute
Status (/status)	120	per minute
Build Pipeline (/build)	5	per hour
Ledger Read	120	per minute
Ledger Write	60	per minute

7.2 Rate Limit Response

When rate limit is exceeded:

```
HTTP 429 Too Many Requests
{
  "success": false,
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Rate limit exceeded. Try again in 45 seconds.",
    "retry_after": 45
  }
}
```

8. Versioning Strategy

8.1 URL Versioning

The API uses URL path versioning: `/api/v1/`, `/api/v2/`, etc.

8.2 Breaking Changes Policy

- Minor changes (new fields, new endpoints): Same version
- Breaking changes (removed fields, changed behavior): New version

• Deprecated features: Supported for 6 months with warning headers

8.3 Version Header

Clients can specify preferred version:

```
Accept-Version: v1
```

---

9. WebSocket Endpoints (Future)

> **Note:** WebSocket support is planned for v1.1 to enable streaming outputs and real-time build progress updates.

9.1 Planned Endpoints

Endpoint	Description
ws://localhost:8000/ws/infer	Streaming inference output
ws://localhost:8000/ws/build/{build_id}	Real-time build progress
ws://localhost:8000/ws/status	Live system metrics

---

10. Appendices

Appendix A: Prolog Router Integration

The API integrates with the Prolog-based router via the `pyswip` Python library. The `/route/analyze` endpoint directly queries the Prolog knowledge base.

Prolog Query Example

```
% Prolog predicate called by API
route("Refactor payment module for SOLID principles", Decision).

% Returns:
% Decision = {
%   domain: coding_architecture,
%   stakes: high,
%   recommended_model: qwen_coder_32b,
%   validation_policy: block_by_block,
```

```
%   tools_required: [embeddings],
%   confidence: 0.92,
%   reasoning: "Keywords indicate architectural refactoring..."
% }
```

Appendix B: Model Capability Matrix

Model	coding_architecture	coding_implementation	reasoning	creative	documentatio
qwen_coder_32b	0.95	0.85	0.70	-	0.75
nemotron_30b	0.75	0.95	0.65	-	-
gpt_oss_20b	0.75	0.70	0.90	-	0.80
mythomax_13b	-	-	0.60	0.95	0.70

Appendix C: Validation Policies

Policy	Stakes	Granularity	Use Case
none	low	No validation	Prototypes, creative writing
end_stage	medium	Per output	Internal tools, documentation
block_by_block	high	Per line/function	Production code, safety-critical

Appendix D: Tool Types

Tool	Trigger Keywords	Description
ocr	scan, document, pdf, extract	Extract text from images/documents
vision	image, picture, diagram, chart	Analyze visual content
embeddings	similar, search, retrieve, context	Semantic search from memory

Appendix E: OpenAPI Specification

A complete OpenAPI 3.0 specification is available at:

```
GET /openapi.json
GET /docs (Swagger UI)
```

Document Control

Dependencies:	PRD v1.0, System Architecture Document v2.0, Routing Logic Specification v1.0, Data Architecture & Memory Design Document v1.0, Security & Governance Design v1.0
Estimated Effort:	1 week
Reviewers:	Technical Lead, Backend Lead, API Lead
Next Review:	End of Phase 2

End of API Specification Document

Sovereign AI Infrastructure - Phase 1 Design & Specification