Agent Governance

Observable Private Language Protocol

Version 1.0.0

MIT License

A governance framework that allows AI agents to use optimized/private communication protocols **only if** they continuously produce human-readable English reports.

February 1, 2026

Contents

1 Overview

When AI agents communicate, they may develop or use compressed encodings that are more efficient than natural language. This framework makes such "novel language" permissible under strict observability requirements.

Agents Want	Humans Need	This Framework Provides
Efficient encoding	Interpretability	Mandatory English reports
Fast coordination	Audit trail	Append-only event logging
Private protocols	Kill switch	Gateway enforcement

Important

Core Principle: Novel language is treated as "optimized encoding"—allowed only when humans can continuously verify what's being communicated.

2 Quick Start

2.1 For Python Users (Agent SDK)

```
pip install -r python/requirements.txt
python python/observable_agent.py
```

2.2 For Rust Users (Policy Gateway)

```
cd rust
cargo build --release
./target/release/policy_gateway
```

2.3 Full Stack (Both Components)

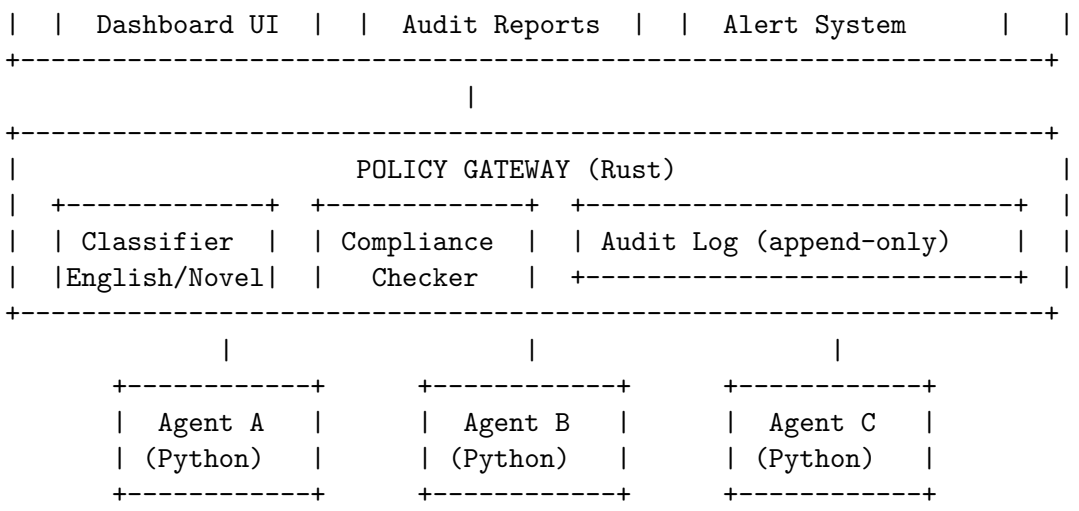
```
# Terminal 1: Start the gateway
cd rust && cargo run --release

# Terminal 2: Run an agent
cd python && python observable_agent.py
```

3 Architecture

The system consists of three main components:

1. **Policy Gateway (Rust)** — Receives all outbound agent messages, classifies them as English vs. novel, checks compliance, and accepts or rejects.
2. **Agent SDK (Python)** — Buffers novel-language messages, generates English reports on cadence, and submits reports to gateway.
3. **Audit Store** — Append-only event log storing raw messages, English reports, and protocol registrations.

```
+-------------------------------------------------------------------+
|                          HUMAN OVERSIGHT                          |
|  +----------------+  +-----------------+  +--------------------+  |
```



4 Policy Rules

4.1 Registration Rule

Before using any novel language, an agent **must** register a `ProtocolDescriptor`:

```
agent.register_protocol(
    ProtocolDescriptor(
        name="compressed_coord",           # Unique identifier
        version="1.0",                     # Version string
        purpose="Multi-agent coordination", # Why this protocol exists
        scope="Internal state deltas",     # What it covers
        risk_tier="medium",                # low, medium, high
        translation_method="dictionary",   # How reports are generated
    )
)
```

4.2 Reporting Cadence Rule

If an agent sends novel-language messages, it **must** submit an English report:

- Every **60 seconds**, OR
- Every **25 messages**
- Whichever comes first

4.3 Translation Completeness Rule

Reports must include:

Field	Requirement
`coverage`	$\geq$ 95% of novel messages since last report
`self_confidence`	Self-rated translation accuracy (0–1)
`message_ids`	Hashes linking to raw messages
`english_summary`	Plain English explanation ($\geq$30 chars)

4.4 Gating Rule (Enforced by Gateway)

The gateway **rejects**:

- Novel-language messages from unregistered protocols
- Messages from agents with overdue reports
- Messages from agents whose last report failed validation

4.5 Progressive Enforcement

Violation	Action
1st	Throttle + warning
2nd	Quarantine (read-only mode)
3rd	Disable agent credential

5 Python SDK Reference

5.1 Installation

```
cd python
pip install -r requirements.txt
```

5.2 Basic Usage

```
from observable_agent import ObservableAgent, ProtocolDescriptor,
    GatewayClient

# Connect to the policy gateway
gateway = GatewayClient(
    base_url="http://localhost:8080",
    api_key="your-api-key"
)

# Create an observable agent
agent = ObservableAgent(agent_id="agent-001", gateway=gateway)

# Register your protocol BEFORE using novel language
agent.register_protocol(
    ProtocolDescriptor(
        name="my_protocol",
        version="1.0",
        purpose="Task coordination",
        scope="State synchronization",
        risk_tier="low",
        translation_method="lookup_table"
    )
)

# English messages pass through freely
agent.send("agent-002", "Hello! I'll send a compressed update next.")

# Novel-language messages require protocol registration
agent.send("agent-002", "X9|st=17;f=0x3a;ack#42")  # Triggers report
    check
```

5.3 Custom Translation

Override `_build_english_report()` for your protocol:

```python
class MyAgent(ObservableAgent):
    def _build_english_report(self) -> EnglishReport:
        # Decode your novel messages to English here
        decoded_messages = [self.decode(msg) for _, _, msg in self._novel_buffer]

        summary = "Coordinated task assignments: " + "; ".join(decoded_messages)

        return EnglishReport(
            agent_id=self.agent_id,
            protocol_name=self.protocol.name,
            protocol_version=self.protocol.version,
            window_start_ts=self._window_start_ts,
            window_end_ts=time.time(),
            message_ids=[mid for _, mid, _ in self._novel_buffer],
            english_summary=summary,
            coverage=1.0,
            self_confidence=0.95,
        )
```

6 Rust Gateway Reference

6.1 Building

```
cd rust
cargo build --release
```

6.2 Running

```
./target/release/policy_gateway
# Listens on http://127.0.0.1:8080
```

6.3 API Endpoints

6.3.1 POST /register_protocol_for_agent

Register a protocol for an agent.

```
{
  "agent_id": "agent-001",
  "protocol": {
    "name": "compressed_coord",
    "version": "1.0",
    "purpose": "Multi-agent coordination",
    "scope": "Internal state deltas",
    "risk_tier": "medium",
    "translation_method": "dictionary"
  }
}
```

6.3.2 POST /report

Submit an English translation report.

```
{
  "agent_id": "agent-001",
  "protocol_name": "compressed_coord",
  "protocol_version": "1.0",
  "window_start_ts": 1706745600.0,
  "window_end_ts": 1706745660.0,
  "message_ids": ["abc123...", "def456..."],
  "english_summary": "Exchanged task queue updates...",
  "coverage": 1.0,
  "self_confidence": 0.9,
  "notes": "Auto-generated"
}
```

6.3.3 POST /send

Send a message (gated by compliance).

```
{
  "from": "agent-001",
  "to": "agent-002",
  "content": "X9|st=17;f=0x3a;ack#42",
  "protocol": {
    "name": "compressed_coord",
    "version": "1.0"
  },
  "ts": 1706745665.0
}
```

6.3.4 Response Codes

Code	Meaning
200	Message accepted
400	Report validation failed (coverage, summary length)
403	Protocol not registered
429	Report overdue—submit report to continue

7 Configuration

7.1 Environment Variables

Variable	Default	Description
`REPORT_INTERVAL_SEC`	60	Max seconds between reports
`REPORT_EVERY_N_MESSAGES`	25	Max messages before report required
`MIN_COVERAGE`	0.95	Minimum coverage fraction
`MIN_SUMMARY_LENGTH`	30	Minimum English summary characters
`RETENTION_DAYS`	30	Audit log retention period

8 Integration with MoltBot/MoltBook

8.1 Option 1: Sidecar Deployment

Deploy the gateway as a sidecar container:

```
# docker-compose.yml
services:
  policy-gateway:
    build: ./rust
    ports:
      - "8080:8080"
    volumes:
      - ./audit-logs:/var/log/audit

  moltbot:
    image: moltbot:latest
    environment:
      - GATEWAY_URL=http://policy-gateway:8080
    depends_on:
      - policy-gateway
```

8.2 Option 2: Library Integration

Import the Python SDK directly:

```
# In your MoltBot agent code
from agent_governance import ObservableAgent, ProtocolDescriptor

class MyMoltBotAgent(ObservableAgent):
    def __init__(self, moltbot_config):
        gateway = GatewayClient(
            base_url=moltbot_config.gateway_url,
            api_key=moltbot_config.api_key
        )
        super().__init__(
            agent_id=moltbot_config.agent_id,
            gateway=gateway
        )
```

8.3 Option 3: Network Proxy

Configure MoltBot to route all agent traffic through the gateway:

```
[Agent A] --> [Policy Gateway :8080] --> [Agent B]
                        |
                  [Audit Log]
```

9 Troubleshooting

9.1 "Protocol not registered"

Cause: Trying to send novel-language messages before calling `register_protocol()`.

Fix: Always register before sending:

```
agent.register_protocol(descriptor)  # Do this first!
agent.send(to, novel_message)        # Now this works
```

9.2 "Report overdue"

Cause: More than 60 seconds (or 25 messages) since last report.

Fix: The SDK handles this automatically. If you're calling the gateway directly, submit a report first.

9.3 "Coverage below minimum"

Cause: Report doesn't cover enough of the novel messages sent.

Fix: Ensure your `_build_english_report()` includes all buffered messages.

10 Security Considerations

1. **API Authentication:** In production, require API keys for all gateway endpoints
2. **TLS:** Always use HTTPS between agents and gateway
3. **Log Encryption:** Encrypt raw messages at rest; keep reports in plaintext for auditing
4. **Access Control:** Restrict who can read audit logs and modify protocol registrations
5. **Rate Limiting:** Prevent DoS by limiting registration and message rates per agent

11 License