Implementation Guide

Version: 1.1

Last Updated: October 12, 2025

Document Type: technical-reference

Tractatus AI Safety Framework

https://agenticgovernance.digital

Tractatus Framework Implementation Guide

Version: 1.1 **Last Updated**: 2025-10-11 **Status**: Production-Ready (Phase 5 Complete)

Overview

This guide covers production deployment of the Tractatus Agentic Governance Framework with MongoDB persistence and optional API Memory integration.

Architecture: Hybrid memory system

- **MongoDB** (required): Persistent storage for governance rules, audit logs
- **Anthropic API Memory** (optional): Session continuity enhancement
- **Filesystem** (debug): Audit trail for development

See the **Architectural Overview** document for complete system architecture and research status.

Prerequisites

Required

- **Node.js**: v18+ LTS
- **MongoDB**: v7.0+
- **npm** or **yarn**: Latest stable
- **Git**: For cloning repository

Optional

- **Anthropic API Key**: For API Memory features
- **systemd**: For production process management (Linux)

Installation

1. Clone Repository

```
git clone https://github.com/AgenticGovernance/tractatus.git
cd tractatus
```

2. Install Dependencies

```
npm install
```

Key Dependencies:

- `mongodb`: v8.x (MongoDB driver)
- `mongoose`: v8.x (ODM for models)
- `express`: v4.x (Web framework)
- `marked`: v14.x (Markdown processing)
- `@anthropic-ai/sdk`: v0.65+ (API Memory - optional)

3. MongoDB Setup

Option A: Local Development

```
# Install MongoDB (Ubuntu/Debian)
sudo apt-get install mongodb-org

# Start MongoDB
sudo systemctl start mongod
sudo systemctl enable mongod

# Create database
mongosh
> use tractatus_dev
> db.createCollection('governanceRules')
> db.createCollection('auditLogs')
> db.createCollection('documents')
> exit
```

Option B: MongoDB Atlas (Cloud)

1. Create free cluster at https://mongodb.com/atlas
2. Add IP whitelist: `0.0.0.0/0` (development) or specific IPs (production)
3. Create database user with read/write permissions
4. Get connection string: `mongodb+srv://user:pass@cluster.mongodb.net/tractatus`

4. Environment Configuration

Create `.env` file in project root:

```
# Required
MONGODB_URI=mongodb://localhost:27017/tractatus_dev
MONGODB_DB=tractatus_dev
NODE_ENV=development
PORT=9000

# Optional - API Memory Features
CLAUDE_API_KEY=your_anthropic_api_key_here

# Optional - JWT for admin features
JWT_SECRET=your_random_secret_here_minimum_32_characters
```

Security Notes:

- Never commit `.env` to version control
- Use strong JWT secrets in production (32+ characters)
- Restrict MongoDB access by IP in production

Framework Initialization

Service Architecture

The framework consists of 6 core services:

1. **InstructionPersistenceClassifier**: Classify and persist user instructions
2. **CrossReferenceValidator**: Validate actions against stored instructions
3. **BoundaryEnforcer**: Enforce inst_016-018 content validation
4. **ContextPressureMonitor**: Monitor session quality degradation
5. **MetacognitiveVerifier**: Confidence-based action verification
6. **BlogCuration**: Validate blog content against governance rules

All services integrate with **MemoryProxy** for MongoDB access.

Basic Initialization

```
const InstructionPersistenceClassifier = require('./src/services/InstructionPers
const CrossReferenceValidator = require('./src/services/CrossReferenceValidator.
const BoundaryEnforcer = require('./src/services/BoundaryEnforcer.service');
const ContextPressureMonitor = require('./src/services/ContextPressureMonitor.se
const MetacognitiveVerifier = require('./src/services/MetacognitiveVerifier.serv
const BlogCuration = require('./src/services/BlogCuration.service');

// Initialize all services (loads governance rules from MongoDB)
async function initializeFramework() {
  await InstructionPersistenceClassifier.initialize();
  await CrossReferenceValidator.initialize();
  await BoundaryEnforcer.initialize();
  await ContextPressureMonitor.initialize();
  await MetacognitiveVerifier.initialize();
  await BlogCuration.initialize();

  console.log('✓ Tractatus Framework initialized');
}

// Call during application startup
initializeFramework();
```

Service Usage Examples

1. Classify User Instructions

```
const classification = InstructionPersistenceClassifier.classify({
  text: "Always use MongoDB port 27017 for this project",
  context: {
    conversation_tokens: 5000,
    conversation_length: 20
  }
});

console.log(classification);
// {
//   quadrant: 'SYSTEM',
//   persistence: 'HIGH',
//   temporalScope: 'PERMANENT',
//   verificationRequired: 'MANDATORY',
//   parameters: { port: 27017, database: 'mongodb' }
// }
```

2. Validate Actions

```
const validation = await CrossReferenceValidator.validate(
  "Change MongoDB port to 27018",
  { explicit_instructions: await loadInstructions() }
);

if (validation.status === 'REJECTED') {
  console.error('Conflict:', validation.reason);
  // "Conflicts with HIGH persistence instruction to use port 27017"
}
```

3. Enforce Content Boundaries

```
const content = "Join thousands of satisfied customers!";
const validation = await BlogCuration.validateContent(content);

if (!validation.allowed) {
  console.error('Violation:', validation.violations[0]);
  // "inst_018: Unverified claim about 'thousands of satisfied customers'"
}
```

4. Monitor Context Pressure

```
const pressure = ContextPressureMonitor.analyzePressure({
  token_usage: 0.75,
  conversation_length: 0.80,
  task_complexity: 0.60,
  error_frequency: 0.10
});

console.log(pressure);
// {
//   pressureName: 'ELEVATED',
//   overall: 0.5625,
//   action: 'REVIEW_BEFORE_COMMIT',
//   recommendations: ['Consider creating session handoff']
// }
```

5. Verify Complex Operations

```
const verification = MetacognitiveVerifier.verify(
  "Implement user authentication with JWT and bcrypt",
  "I will create middleware, hash passwords, and add protected routes",
  { explicit_instructions: await loadInstructions() }
);

console.log(verification);
// {
//   confidence: 0.83,
//   decision: 'PROCEED',
//   level: 'PROCEED',
//   reasoning: '...',
//   recommendations: [...]
// }
```

Database Schema

GovernanceRules Collection

```
{
  _id: ObjectId,
  id: "inst_001",                    // Unique rule identifier
  text: "Use MongoDB port 27017",    // Instruction text
  quadrant: "SYSTEM",                // STRATEGIC/OPERATIONAL/TACTICAL/SYSTEM/ST
  persistence: "HIGH",               // HIGH/MEDIUM/LOW
  category: "technical",             // content/security/privacy/technical/proce
  priority: 50,                      // 0-100
  temporalScope: "PERMANENT",        // IMMEDIATE/SESSION/PROJECT/PERMANENT
  expiresAt: null,                   // Date or null
  active: true,                      // Boolean
  source: "user_instruction",        // Origin
  stats: {
    timesChecked: 42,
    timesViolated: 0,
    lastChecked: Date
  },
  createdAt: Date,
  updatedAt: Date
}
```

AuditLogs Collection

```
{
  _id: ObjectId,
  timestamp: Date,
  sessionId: "2025-10-11-001",
  action: "boundary_enforcement",    // Service action type
  rulesChecked: ["inst_016", "inst_017", "inst_018"],
  violations: [],                    // Array of violations (if any)
  allowed: true,                     // Decision outcome
  metadata: {
    // Service-specific context
  }
}
```

Documents Collection

See **Architectural Overview** for complete schema.

Production Deployment

1. Server Setup

Recommended: Ubuntu 22.04 LTS or Debian 12

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Node.js 18 LTS
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Install MongoDB
wget -qO - https://www.mongodb.org/static/pgp/server-7.0.asc | sudo apt-key add
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu jammy/mongodb
sudo apt-get update
sudo apt-get install -y mongodb-org

# Start MongoDB
sudo systemctl start mongod
sudo systemctl enable mongod
```

2. Deploy Application

```
# Create app user
sudo useradd -m -s /bin/bash tractatus

# Clone and setup
sudo su - tractatus
git clone https://github.com/AgenticGovernance/tractatus.git
cd tractatus
npm install --production

# Configure environment
cp .env.example .env
nano .env  # Update with production values
```

3. MongoDB Production Configuration

```
# Create production database user
mongosh
> use tractatus_prod
> db.createUser({
    user: "tractatus_user",
    pwd: "SECURE_PASSWORD_HERE",
    roles: [
      { role: "readWrite", db: "tractatus_prod" }
    ]
  })
> exit

# Update .env
MONGODB_URI=mongodb://tractatus_user:SECURE_PASSWORD@localhost:27017/tractatus_p
MONGODB_DB=tractatus_prod
```

4. systemd Service

Create `/etc/systemd/system/tractatus.service`:

```
[Unit]
Description=Tractatus AI Safety Framework
Documentation=https://agenticgovernance.digital
After=network.target mongod.service
Requires=mongod.service

[Service]
Type=simple
User=tractatus
WorkingDirectory=/home/tractatus/tractatus
ExecStart=/usr/bin/node src/server.js
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=tractatus

# Security
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ReadWritePaths=/home/tractatus/tractatus/.memory
MemoryLimit=2G

# Environment
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
```

Start service:

```
sudo systemctl daemon-reload
sudo systemctl start tractatus
sudo systemctl enable tractatus
sudo systemctl status tractatus
```

5. Nginx Reverse Proxy (Optional)

```
server {
    listen 80;
    server_name agenticgovernance.digital;

    location / {
        proxy_pass http://localhost:9000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Monitoring & Maintenance

View Audit Logs

```
# Today's audit trail
cat .memory/audit/decisions-$(date +%Y-%m-%d).jsonl | jq

# Count violations
cat .memory/audit/*.jsonl | jq 'select(.allowed == false)' | wc -l

# View specific service logs
cat .memory/audit/*.jsonl | jq 'select(.action == "boundary_enforcement")'
```

MongoDB Queries

```
// Connect to MongoDB
mongosh mongodb://localhost:27017/tractatus_prod

// View active rules
db.governanceRules.find({ active: true }).pretty()

// Check rule statistics
db.governanceRules.aggregate([
  { $match: { active: true } },
  { $group: {
      _id: "$quadrant",
      count: { $sum: 1 },
      totalChecks: { $sum: "$stats.timesChecked" }
    }
  }
])

// Recent audit logs
db.auditLogs.find().sort({ timestamp: -1 }).limit(10).pretty()
```

Service Health Check

```
# Check service status
sudo systemctl status tractatus

# View logs
sudo journalctl -u tractatus -f

# Check MongoDB connection
mongosh --eval "db.adminCommand('ping')"
```

Troubleshooting

Issue: Services not loading rules

Symptom: "Governance rules not initialized" warnings

Fix:

```
// Manually initialize
await InstructionPersistenceClassifier.initialize();
await CrossReferenceValidator.initialize();
// etc.
```

Issue: MongoDB connection failed

Symptom: "MongoServerError: Authentication failed"

Fix:

1. Verify `MONGODB_URI` in `.env`
2. Check MongoDB user exists: `mongosh` → `use tractatus_prod` → `db.getUsers()`
3. Verify MongoDB is running: `sudo systemctl status mongod`

Issue: API Memory not working

Symptom: Session continuity not preserved

Fix:

- API Memory is **optional**
- Framework functions without it using MongoDB alone
- To enable: Set `CLAUDE_API_KEY` in `.env`

Migration from Filesystem (Legacy)

If upgrading from filesystem-based instruction storage:

```
# Run migration script
node scripts/migrate-to-mongodb.js

# Verify migration
mongosh
> use tractatus_dev
> db.governanceRules.countDocuments()
18  # Should show migrated rules
```

Next Steps

1. **Read Core Concepts**: Understand the 6 services
2. **Review Architectural Overview**: Complete system architecture
3. **Check Glossary**: Key terms and definitions
4. **Explore Case Studies**: Real-world usage examples

Support

- **Documentation**: https://agenticgovernance.digital/docs.html
- **GitHub**: https://github.com/AgenticGovernance/tractatus
- **Issues**: https://github.com/AgenticGovernance/tractatus/issues

---

Version History:

- v1.1 (2025-10-11): Complete rewrite for MongoDB architecture
- v1.0 (2025-10-07): Initial version (filesystem-based)
