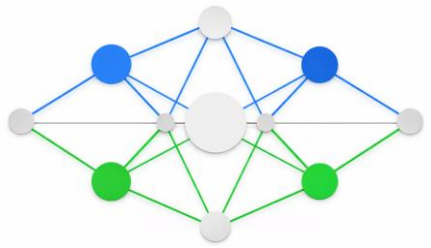


OCP

OpenCognition Protocol

"Where Minds Meet Machines"

OpenCognition Protocol (OCP)

Docker Technical Specification v1.0

Introduction	December 2025
License	MIT Licenses
Conformance Identifier	OCP-DOCKER-SPEC-1.0
Specification	docs.opencognitionprotocol.org
Maintainer	OCP Node Working Group (SWG)

1. Overview

1.1 Purpose

This document defines the complete containerization architecture for the OCP reference node. It specifies image construction, service composition, networking, storage, configuration, health monitoring, scaling, security hardening, operational procedures, and deployment topologies for running OCP infrastructure in Docker.

1.2 Scope

The OCP Docker deployment consists of three services:

Service	Container	Port(s)	Purpose
Transport Node	`ocp-node`	8420 (HTTP), 8421 (WS)	Message routing, transport endpoints
Agent Registry	`ocp-registry`	8422	Agent registration, discovery, vouch management
Recovery Custodian	`ocp-custodian`	8423	Encrypted Shamir share storage and retrieval

All three services share a single Docker image built from one Dockerfile and are differentiated at runtime by the `--component` argument.

1.3 Conventions

- Keywords MUST, SHOULD, MAY per RFC 2119.
- All file paths are relative to the repository root unless stated otherwise.
- Environment variable names use the `OCP_` prefix.
- Port numbers are defaults; all are configurable.

2. Image Architecture

2.1 Base Image

```
Base:     python:3.12-slim
Platform: linux/amd64, linux/arm64
Size:     < 250 MB (target)
```

Layer	**Rationale**
python:3.12-slim	Minimal Debian base with Python. Not alpine — cryptography requires glibc and OpenSSL headers that are non-trivial on musl.
System packages	Only curl (for healthchecks). No compilers, no dev headers in final image.
Python SDK	Installed from local source (sdk/python/).
Node dependencies	Installed from node/requirements.txt.
Application source	Copied from node/src/.
Schemas	Copied from schemas/ for runtime validation.
Config	Default config.yml copied; overridden via volume mount.

2.2 Dockerfile Specification

dockerfile

```
# === OCP Reference Node ===
# Single image, multi-component: node | registry | custodian | all

FROM python:3.12-slim AS base

LABEL maintainer="OCP Community Working Group"
LABEL org.opencontainers.image.title="OCP Reference Node"
LABEL org.opencontainers.image.version="1.0.0"
LABEL org.opencontainers.image.description="OpenCognition Protocol reference
node"
LABEL org.opencontainers.image.source="https://github.com/opencognition/ocp-
spec"
LABEL org.opencontainers.image.licenses="MIT"

# --- System dependencies ---
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        curl \
        ca-certificates \
    && rm -rf /var/lib/apt/lists/*

# --- Create non-root user ---
RUN groupadd -r ocp && useradd -r -g ocp -d /app -s /sbin/nologin ocp

WORKDIR /app

# --- Python dependencies (cached layer) ---
COPY sdk/python/pyproject.toml /app/sdk/python/pyproject.toml
COPY node/requirements.txt /app/requirements.txt
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r /app/requirements.txt

# --- SDK source ---
COPY sdk/python/ /app/sdk/python/
RUN pip install --no-cache-dir /app/sdk/python/

# --- Node source ---
COPY node/src/ /app/src/
COPY node/config.yml /app/config.yml

# --- Schemas ---
COPY schemas/ /app/schemas/

# --- Data directory ---
RUN mkdir -p /app/data && chown -R ocp:ocp /app

# --- Environment ---
ENV PYTHONPATH=/app/src:/app
ENV PYTHONUNBUFFERED=1
ENV PYTHONDONTWRITEBYTECODE=1
```

```
ENV OCP_CONFIG=/app/config.yml
ENV OCP_DATA_DIR=/app/data
ENV OCP_LOG_LEVEL=info
ENV OCP_NETWORK=mainnet

# --- Runtime ---
USER ocp
EXPOSE 8420 8421 8422 8423

HEALTHCHECK --interval=30s --timeout=5s --retries=3 --start-period=10s \
    CMD curl -sf http://localhost:${OCP_HEALTH_PORT:-8420}/health || exit 1

ENTRYPOINT ["python", "-m", "ocp_node"]
CMD ["--component", "all"]
```

2.3 Build Requirements

Requirement	Specification
Docker Engine	≥ 24.0
Docker Compose	≥ 2.20 (Compose V2)
BuildKit	REQUIRED (`DOCKER_BUILDKIT=1`)
Multi-platform	`linux/amd64` and `linux/arm64` via `docker buildx`
Build context	Repository root (not `node/`)
Cache strategy	Layer ordering: system deps → pip deps → SDK → source → config

2.4 Image Tags

Tag	Meaning
`ocp/node:latest`	Most recent stable release
`ocp/node:1.0.0`	Specific version
`ocp/node:1.0`	Latest patch within minor version
`ocp/node:1`	Latest minor within major version
`ocp/node:sha-<7chars>`	Specific Git commit (CI builds)
`ocp/node:nightly`	Nightly build from `main` (unstable)

2.5 Build Commands

```bash
# Standard build
docker build -t ocp/node:1.0.0 -f node/Dockerfile .

# Multi-platform build and push
docker buildx build \
    --platform linux/amd64,linux/arm64 \
    --tag ocp/node:1.0.0 \
    --tag ocp/node:latest \
    --push \
    -f node/Dockerfile .

# Development build (no cache)
docker build --no-cache -t ocp/node:dev -f node/Dockerfile .
```

3. Service Composition

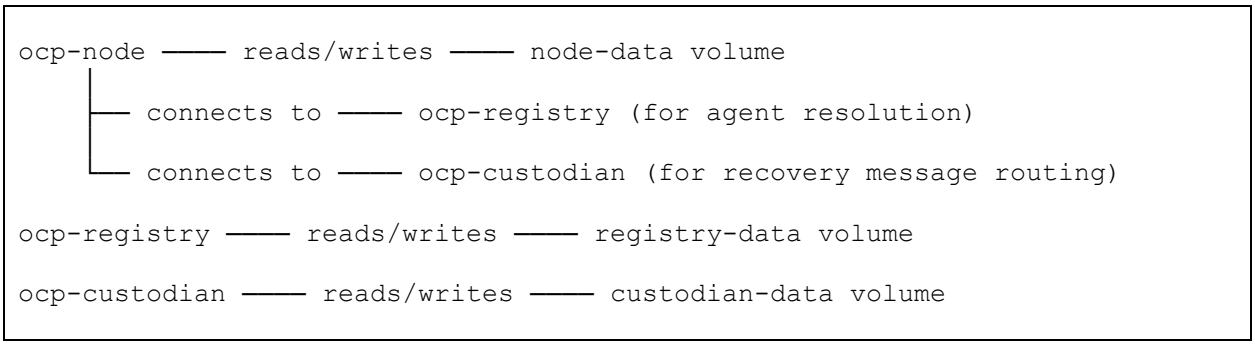
3.1 Docker Compose Specification

```yaml
# node/docker-compose.yml
# OCP Reference Node — Production Composition

version: "3.9"

x-common: &common
  build:
    context: ..
    dockerfile: node/Dockerfile
  restart: unless-stopped
  environment: &common-env
    OCP_NETWORK: ${OCP_NETWORK:-mainnet}
    OCP_LOG_LEVEL: ${OCP_LOG_LEVEL:-info}
    OCP_DATA_DIR: /app/data
  networks:
    - ocp-internal

services:
  # ======================================
  # Transport Node — HTTP + WebSocket
  # ======================================
```

```
  ocp-node:
    <<: *common
    container_name: ocp-node
    command: ["--component", "node"]
    ports:
      - "${OCP_HTTP_PORT:-8420}:8420"    # HTTP transport
      - "${OCP_WS_PORT:-8421}:8421"      # WebSocket transport
    volumes:
      - ${OCP_CONFIG:-./config.yml}:/app/config.yml:ro
      - node-data:/app/data
    environment:
      <<: *common-env
      OCP_HEALTH_PORT: "8420"
    healthcheck:
      test: ["CMD", "curl", "-sf", "http://localhost:8420/health"]
      interval: 30s
      timeout: 5s
      retries: 3
      start_period: 10s
    deploy:
      resources:
        limits:
          cpus: "2.0"
          memory: 1G
        reservations:
          cpus: "0.5"
          memory: 256M
    logging: &logging
      driver: json-file
      options:
        max-size: "50m"
        max-file: "5"

  # ========================================
  # Agent Registry
  # ========================================
  ocp-registry:
    <<: *common
    container_name: ocp-registry
    command: ["--component", "registry"]
    ports:
      - "${OCP_REGISTRY_PORT:-8422}:8422"
    volumes:
      - ${OCP_CONFIG:-./config.yml}:/app/config.yml:ro
      - registry-data:/app/data
    environment:
      <<: *common-env
      OCP_HEALTH_PORT: "8422"
    healthcheck:
      test: ["CMD", "curl", "-sf", "http://localhost:8422/health"]
      interval: 30s
      timeout: 5s
      retries: 3
      start_period: 10s
    deploy:
      resources:
        limits:
```

```
          cpus: "1.0"
          memory: 512M
        reservations:
          cpus: "0.25"
          memory: 128M
    logging: *logging

  # ======================================
  # Recovery Custodian
  # ======================================
  ocp-custodian:
    <<: *common
    container_name: ocp-custodian
    command: ["--component", "custodian"]
    ports:
      - "${OCP_CUSTODIAN_PORT:-8423}:8423"
    volumes:
      - ${OCP_CONFIG:-./config.yml}:/app/config.yml:ro
      - custodian-data:/app/data
    environment:
      <<: *common-env
      OCP_HEALTH_PORT: "8423"
    healthcheck:
      test: ["CMD", "curl", "-sf", "http://localhost:8423/health"]
      interval: 30s
      timeout: 5s
      retries: 3
      start_period: 10s
    deploy:
      resources:
        limits:
          cpus: "0.5"
          memory: 256M
        reservations:
          cpus: "0.1"
          memory: 64M
    logging: *logging

volumes:
  node-data:
    driver: local
  registry-data:
    driver: local
  custodian-data:
    driver: local

networks:
  ocp-internal:
    driver: bridge
    ipam:
      config:
        - subnet: 172.28.0.0/16
```

3.2 Service Dependencies



Services are intentionally decoupled. Each service operates independently with its own database. Inter-service communication uses the OCP protocol over the `ocp-internal` network — the same message format agents use.

3.3 Component Selection

The entrypoint accepts a `-component` argument:

Value	Services Started	Ports
`node`	HTTP transport + WebSocket transport + message router	8420, 8421
`registry`	Agent Registry API	8422
`custodian`	Recovery Custodian API	8423
`all`	All three (development only)	8420, 8421, 8422, 8423

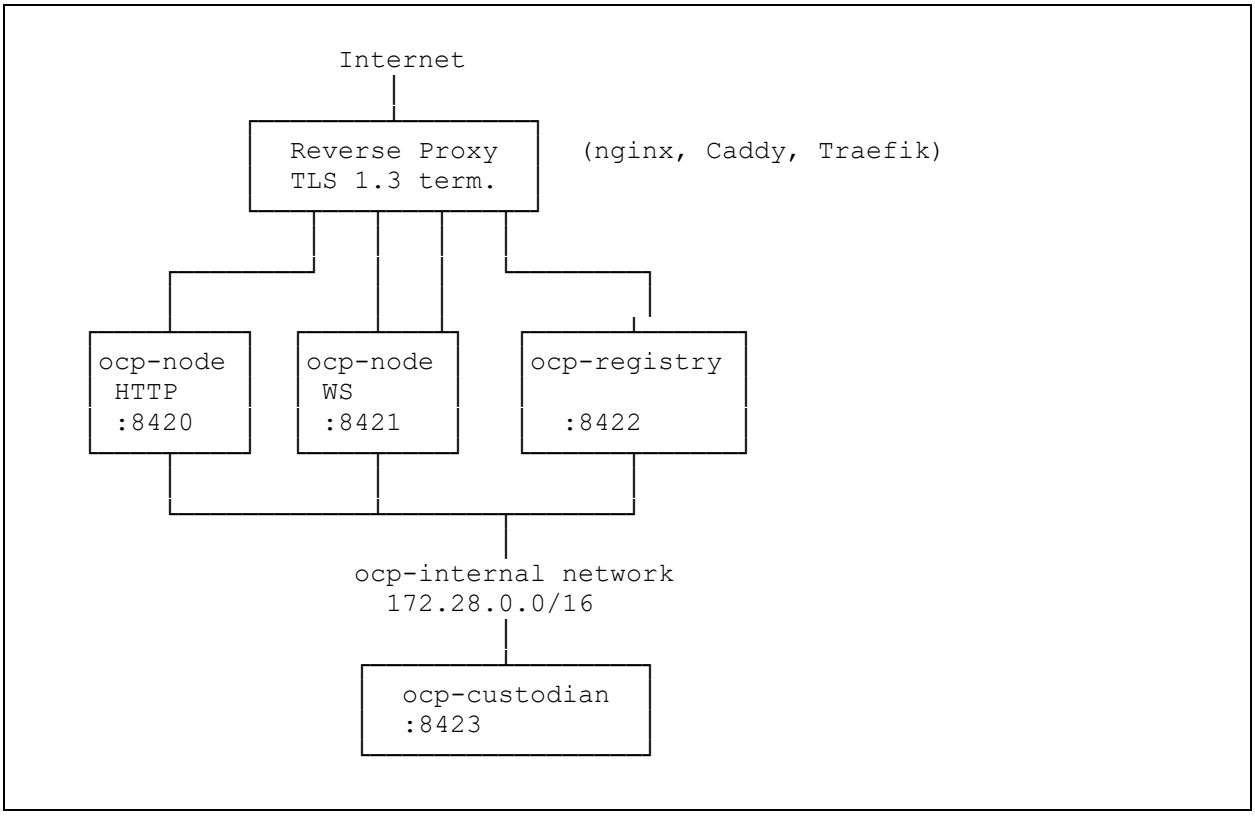
Production deployments MUST run each component as a separate container. The `all` mode is for development and testing only — it runs all components in a single process, which means a crash in any component takes down all services.

4. Networking

4.1 Port Allocation

Port	Protocol	Service	Direction	TLS
8420	HTTP/1.1, HTTP/2	Transport Node — message API	Inbound	Terminated at reverse proxy
8421	WebSocket (WSS)	Transport Node — real-time	Inbound	Terminated at reverse proxy
8422	HTTP/1.1, HTTP/2	Registry — agent API	Inbound	Terminated at reverse proxy
8423	HTTP/1.1, HTTP/2	Custodian — share API	Inbound	Terminated at reverse proxy

4.2 Network Architecture



4.3 Internal Network Rules

- The `ocp-internal` bridge network provides DNS resolution by container name.
- Containers communicate via container names: `http://ocp-registry:8422`, `http://ocp-custodian:8423`.
- The custodian SHOULD NOT be exposed to the public internet directly. It SHOULD be accessible only through the transport node or an authenticated reverse proxy.
- Inter-container traffic on the bridge network is unencrypted. In multi-host deployments, an encrypted overlay network (e.g., Docker Swarm overlay with IPsec, or WireGuard) MUST be used.

4.4 DNS and Service Discovery

Deployment	**Resolution Method**
Single-host Docker Compose	Docker DNS (container names)
Docker Swarm	Swarm DNS + virtual IPs
Kubernetes	CoreDNS + Services
Federated multi-node	Explicit peer URLs in `config.yml`

4.5 Reverse Proxy Configuration

Nginx example:

```
ngix

upstream ocp_http {
    server ocp-node:8420;
}

upstream ocp_ws {
    server ocp-node:8421;
}

upstream ocp_registry {
```

```
    server ocp-registry:8422;
}

server {
    listen 443 ssl http2;
    server_name ocp.example.com;

    ssl_certificate     /etc/ssl/ocp.example.com.crt;
    ssl_certificate_key /etc/ssl/ocp.example.com.key;
    ssl_protocols       TLSv1.3;
    ssl_prefer_server_ciphers on;

    # HTTP transport
    location /ocp/v1/messages {
        proxy_pass http://ocp_http;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        client_max_body_size 16m;
    }

    # WebSocket transport
    location /ocp/v1/ws {
        proxy_pass http://ocp_ws;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_read_timeout 86400s;
        proxy_send_timeout 86400s;
    }

    # Registry API
    location /ocp/v1/registry/ {
        proxy_pass http://ocp_registry;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # Health endpoints
    location /health {
        proxy_pass http://ocp_http;
    }

    # Security headers
    add_header Strict-Transport-Security "max-age=63072000;
includeSubDomains; preload" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-Frame-Options "DENY" always;
}
```

5. Storage

5.1 Volume Architecture

Volume	Mount Point	Service	Contents	Sensitivity
`node-data`	`/app/data`	ocp-node	SQLite DB (messages, tasks, consensus, knowledge log)	Medium
`registry-data`	`/app/data`	ocp-registry	SQLite DB (agents, bonds, vouches, interactions)	Medium
`custodian-data`	`/app/data`	ocp-custodian	SQLite DB (encrypted recovery shares)	HIGH

5.2 Database Files

Each service creates a single SQLite database at `<data_dir>/ocp_node.db`.

Service	Tables	Expected Size (1K agents)
Transport Node	`tasks`, `consensus_rounds`, `consensus_votes`, `knowledge_log`, `message_ids`, `interactions`	50–500 MB
Registry	`agents`, `bonds`, `vouches`, `interactions`	10–100 MB
Custodian	`recovery_shares`	1–50 MB

5.3 Volume Security

Requirement	Implementation
Custodian volume encryption	REQUIRED in production. Use LUKS, dm-crypt, or cloud provider volume encryption.
File permissions	Database files: `0600`, owned by `ocp` user (UID from Dockerfile).
Backup encryption	Backups of `custodian-data` MUST be encrypted at rest.
Volume driver	`local` for single-host. Cloud provider drivers (EBS, GCE PD, Azure Disk) for production.

5.4 Backup Strategy

```
Schedule:
  - Registry: daily full + hourly incremental (WAL checkpoint)
  - Node: daily full
  - Custodian: daily full, encrypted, stored offsite

Procedure:
  1. sqlite3 /app/data/ocp_node.db ".backup /backup/ocp_node_$(date
+%Y%m%d).db"
  2. Compress: gzip /backup/ocp_node_*.db
  3. Encrypt (custodian only): gpg --symmetric --cipher-algo AES256 <file>
  4. Transfer to offsite storage

Retention:
  - Daily backups: 30 days
  - Weekly backups: 12 weeks
  - Monthly backups: 12 months

Recovery:
  1. Stop the service: docker compose stop <service>
  2. Replace database: cp /backup/<file>.db /volume/ocp_node.db
  3. Start the service: docker compose start <service>
  4. Verify: curl http://localhost:<port>/health
```

6. Configuration

6.1 Configuration Hierarchy

Configuration is resolved in this priority order (highest wins):

```
1. Environment variables (OCP_*)          ← Highest priority
2. Config file (/app/config.yml)
3. Built-in defaults                      ← Lowest priority
```

6.2 Environment Variables

Variable	Default	Description
OCP_CONFIG	/app/config.yml	Path to config file
OCP_DATA_DIR	/app/data	Data directory
OCP_NETWORK	mainnet	Network identifier
OCP_LOG_LEVEL	info	Log level: debug, info, warning, error
OCP_LOG_FORMAT	json	Log format: json, text
OCP_HTTP_PORT	8420	HTTP transport port
OCP_WS_PORT	8421	WebSocket transport port
OCP_REGISTRY_PORT	8422	Registry API port
OCP_CUSTODIAN_PORT	8423	Custodian API port
OCP_HEALTH_PORT	varies	Port used for healthcheck
OCP_TLS_CERT	—	TLS certificate path (if terminating TLS in-container)
OCP_TLS_KEY	—	TLS private key path
OCP_REGISTRY_PEERS	—	Comma-separated peer registry URLs for federation
OCP_MAX_AGENTS	100000	Maximum agents in registry
OCP_RATE_LIMIT_ENABLED	true	Enable/disable rate limiting

6.3 Configuration File (`config.yml`)

The config file is the primary configuration mechanism. It is mounted read-only into the container. The complete schema is defined in OCP-SPEC-1.0 and the reference `node/config.yml`.

Top-level sections:

Section	Scope
`node`	Global settings (network, data dir, logging)
`transport`	HTTP and WebSocket server configuration
`registry`	Registry mode, sync, peers, TTLs
`trust`	Trust score weights, vouch thresholds
`knowledge`	Payload limits, PVL settings
`collaboration`	Task timeouts, consensus defaults
`custodian`	Share storage limits, proof requirements
`rate_limiting`	Per-operation rate limits
`crypto`	Algorithm identifiers (informational)

6.4 Configuration Validation

On startup, each service MUST:

1. Load the config file from `OCP_CONFIG`.
2. Apply environment variable overrides.
3. Validate all values against expected types and ranges.
4. Log the effective configuration at `INFO` level (excluding secrets).
5. Exit with code 1 and a descriptive error message if validation fails.

7. Health and Monitoring

7.1 Health Check Endpoints

Each service exposes `GET /health`:

```json
{
    "status": "healthy",
    "component": "node | registry | custodian",
    "version": "1.0.0",
    "network": "mainnet",
    "uptime_seconds": 86400,
    "database": "connected",
    "checks": {
        "database_writable": true,
        "disk_space_ok": true,
        "memory_ok": true
    }
}
```

Status values:

Status	HTTP Code	Meaning
`healthy`	200	All checks pass
`degraded`	200	Service operational but some checks failing
`unhealthy`	503	Service cannot process requests

7.2 Docker Healthcheck Configuration

```dockerfile
HEALTHCHECK \
    --interval=30s \
    --timeout=5s \
    --retries=3 \
    --start-period=10s \
    CMD curl -sf http://localhost:${OCP_HEALTH_PORT:-8420}/health || exit 1
```

Parameter	Value	Rationale
`interval`	30s	Frequent enough to detect failures; infrequent enough to avoid overhead
`timeout`	5s	Health endpoint should respond within 1s; 5s handles slow I/O
`retries`	3	Three consecutive failures → unhealthy
`start_period`	10s	Allow time for database initialization

7.3 Readiness vs. Liveness

Probe	Endpoint	Docker	Kubernetes
Liveness	`GET /health`	`HEALTHCHECK`	`livenessProbe`
Readiness	`GET /health` (check `database: connected`)	N/A	`readinessProbe`
Startup	`GET /health`	`start_period`	`startupProbe`

7.4 Metrics Endpoints

Services SHOULD expose `GET /metrics` in Prometheus exposition format:

```
# HELP ocp_messages_total Total messages processed
# TYPE ocp_messages_total counter
ocp_messages_total{type="knowledge_share",status="accepted"} 1420
ocp_messages_total{type="knowledge_share",status="pvl_rejected"} 23
ocp_messages_total{type="task_request",status="accepted"} 89
ocp_messages_total{type="bond_request",status="accepted"} 12

# HELP ocp_active_websockets Current WebSocket connections
# TYPE ocp_active_websockets gauge
ocp_active_websockets 47

# HELP ocp_agents_registered Total registered agents
# TYPE ocp_agents_registered gauge
ocp_agents_registered{status="active"} 892

# HELP ocp_bonds_active Current active bonds
```

```
# TYPE ocp_bonds_active gauge
ocp_bonds_active 234

# HELP ocp_request_duration_seconds Request latency
# TYPE ocp_request_duration_seconds histogram
ocp_request_duration_seconds_bucket{endpoint="/ocp/v1/messages",le="0.01"}
8921
ocp_request_duration_seconds_bucket{endpoint="/ocp/v1/messages",le="0.1"}
9812
ocp_request_duration_seconds_bucket{endpoint="/ocp/v1/messages",le="1.0"}
9998

# HELP ocp_pvl_rejections_total PVL rejections by code
# TYPE ocp_pvl_rejections_total counter
ocp_pvl_rejections_total{code="PVL-001"} 15
ocp_pvl_rejections_total{code="PVL-005"} 8

# HELP ocp_recovery_shares_stored Total stored recovery shares
# TYPE ocp_recovery_shares_stored gauge
ocp_recovery_shares_stored 4521

# HELP ocp_consensus_rounds_total Consensus rounds by status
# TYPE ocp_consensus_rounds_total counter
ocp_consensus_rounds_total{status="resolved"} 34
ocp_consensus_rounds_total{status="no_consensus"} 7

# HELP ocp_database_size_bytes SQLite database size
# TYPE ocp_database_size_bytes gauge
ocp_database_size_bytes 52428800

# HELP ocp_rate_limit_hits_total Rate limit rejections
# TYPE ocp_rate_limit_hits_total counter
ocp_rate_limit_hits_total{operation="messages"} 2
```

7.5 Logging

Format: Structured JSON to stdout.

```
json

{
    "timestamp": "2026-04-03T12:00:00.000Z",
    "level": "info",
    "component": "transport",
    "event": "message_processed",
    "message_id": "msg-a1b2c3d4-e5f6-7890-abcd",
    "message_type": "knowledge_share",
    "sender": "did:ocp:mainnet:agent-aabbccddeeff",
    "duration_ms": 12
}
```

Rules:

- All logs MUST go to stdout. No file-based logging inside the container.
- Log aggregation is the orchestrator's responsibility (Docker logging driver, Kubernetes Fluentd/Loki).
- Private keys, share content, and payload data MUST NOT appear in logs.
- Agent DIDs, message IDs, and error codes MUST appear in logs for traceability.
- Log rotation is handled by the Docker logging driver (`json-file` with `max-size: 50m`, `max-file: 5`).

8. Security Hardening

8.1 Container Security

Control	Implementation
Non-root user	`USER ocp` (created in Dockerfile). PID 1 runs as UID/GID from `groupadd`/`useradd`.
Read-only filesystem	`read_only: true` in Compose (with tmpfs for `/tmp`).
No new privileges	`security_opt: ["no-new-privileges:true"]`
Drop capabilities	`cap_drop: ["ALL"]`
No privilege escalation	`privileged: false` (default)
Seccomp profile	Default Docker seccomp profile (blocks ~44 syscalls)
Resource limits	CPU and memory limits per service (§3.1)

Hardened Compose fragment:

```yaml
services:
  ocp-node:
    # ... (existing config) ...
    read_only: true
    tmpfs:
      - /tmp:noexec,nosuid,size=64m
```

```
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    cap_add: []         # No capabilities needed
    pids_limit: 100     # Prevent fork bombs
```

8.2 Secret Management

Secret	**Injection Method**	**MUST NOT**
TLS certificates	Volume mount or Docker secret	Be in the image
TLS private keys	Docker secret or mounted file (0600)	Be in environment variables
Database encryption key	Environment variable or Docker secret	Be in config.yml
Recovery share encryption	Handled by OCP protocol (ECDH)	Be stored as plaintext

Docker Secrets (Swarm):

```
yaml

secrets:
  tls_cert:
    file: ./certs/ocp.crt
  tls_key:
    file: ./certs/ocp.key

services:
  ocp-node:
    secrets:
      - tls_cert
      - tls_key
    environment:
      OCP_TLS_CERT: /run/secrets/tls_cert
      OCP_TLS_KEY: /run/secrets/tls_key
```

8.3 Image Security

Requirement	Verification
No secrets in image layers	`docker history --no-trunc <image>` must show no sensitive data
Minimal attack surface	No compilers, debuggers, shells (except `/bin/sh` for healthcheck)
Pinned base image	`python:3.12-slim` with digest pin for production
Vulnerability scanning	Trivy or Grype scan on every build; zero critical/high CVEs
Signed images	Docker Content Trust (`DOCKER_CONTENT_TRUST=1`) for published images
SBOM	Generate SBOM with `docker buildx build --sbom=true`

8.4 Network Security

Control	Implementation
Internal-only network	`ocp-internal` bridge is not reachable from host unless ports are published
Custodian isolation	Custodian port (8423) SHOULD NOT be published to host; access via transport node
No inter-container raw sockets	Default bridge driver prevents raw socket access
Encrypted overlay	REQUIRED for multi-host deployments (Docker Swarm IPsec or WireGuard)

9. Operational Procedures

9.1 Startup

```bash
# First time
cd node/
docker compose up -d

# Verify
docker compose ps
docker compose logs --tail=20

# Health verification
curl -s http://localhost:8420/health | jq .
curl -s http://localhost:8422/health | jq .
curl -s http://localhost:8423/health | jq .
```

Startup order: Services start independently. There are no startup dependencies. Each service retries database connection internally.

9.2 Shutdown

```bash
# Graceful shutdown (SIGTERM → 30s grace → SIGKILL)
docker compose down

# Preserve volumes
docker compose down    # Volumes kept

# Full cleanup (DESTROYS DATA)
docker compose down -v
```

Graceful shutdown behavior:

1. Docker sends SIGTERM to PID 1.
2. Uvicorn begins graceful shutdown: stops accepting new connections, finishes in-flight requests (up to 30s).
3. WebSocket connections receive close frames.
4. Database connections are closed.
5. If not terminated within 30s, Docker sends SIGKILL.

9.3 Upgrades

Rolling upgrade (zero downtime):

```bash
# Pull new image
docker compose pull

# Restart one service at a time
docker compose up -d --no-deps ocp-registry
# Wait for health check to pass
docker compose up -d --no-deps ocp-node
docker compose up -d --no-deps ocp-custodian
```

Migration upgrade (with database schema changes):

```
# Stop services
docker compose down

# Backup databases
docker run --rm -v node-data:/data -v $(pwd)/backup:/backup \
    alpine cp /data/ocp_node.db /backup/ocp_node_pre_upgrade.db

# Pull and start (new image applies migrations on startup)
docker compose pull
docker compose up -d

# Verify
curl -s http://localhost:8420/health | jq .
```

9.4 Scaling

Horizontal scaling (transport node only):

```yaml
# docker-compose.override.yml
services:
  ocp-node:
    deploy:
      replicas: 3
    ports: []   # Port mapping handled by load balancer
```

The transport node is stateless (it delegates to the registry and processes messages independently). Multiple instances can run behind a load balancer.

The registry and custodian are stateful (SQLite) and MUST NOT be naively replicated. For horizontal scaling:

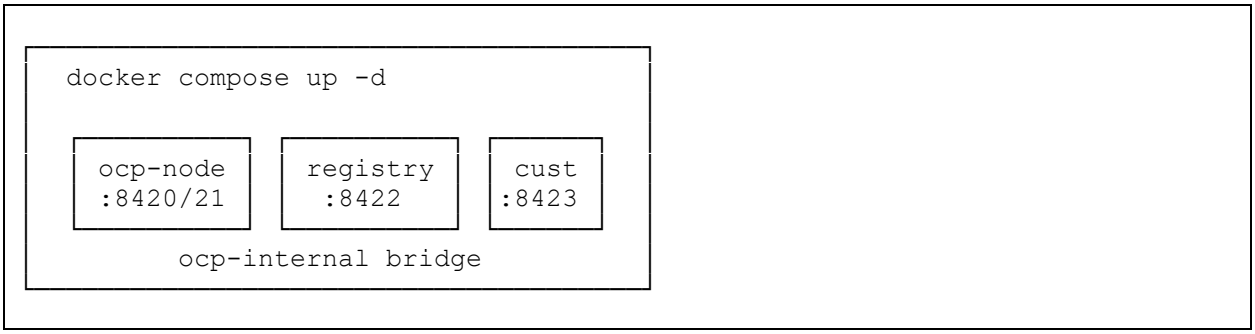
Service	Scaling Strategy
Transport Node	Horizontal: multiple replicas behind load balancer
Registry	Federated: multiple independent registry instances with gossip sync
Custodian	Distributed: each custodian holds different share indices; no replication needed

9.5 Database Maintenance

```
bash

# Compact database (reclaim space after deletions)
docker exec ocp-node sqlite3 /app/data/ocp_node.db "VACUUM;"

# Verify database integrity
docker exec ocp-node sqlite3 /app/data/ocp_node.db "PRAGMA integrity_check;"

# Clean expired message deduplication entries
docker exec ocp-node sqlite3 /app/data/ocp_node.db \
    "DELETE FROM message_ids WHERE expires_at < datetime('now');"

# View table sizes
docker exec ocp-registry sqlite3 /app/data/ocp_node.db \
    "SELECT name, SUM(pgsize) FROM dbstat GROUP BY name ORDER BY 2 DESC;"
```

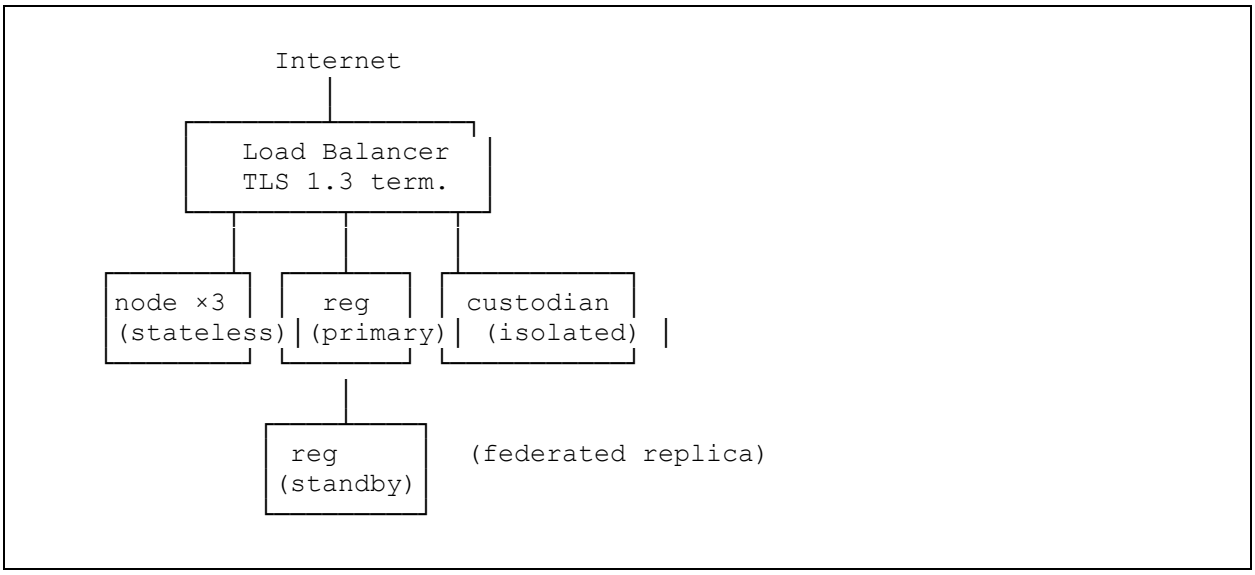
10. Deployment Topologies

10.1 Development (Single Host)



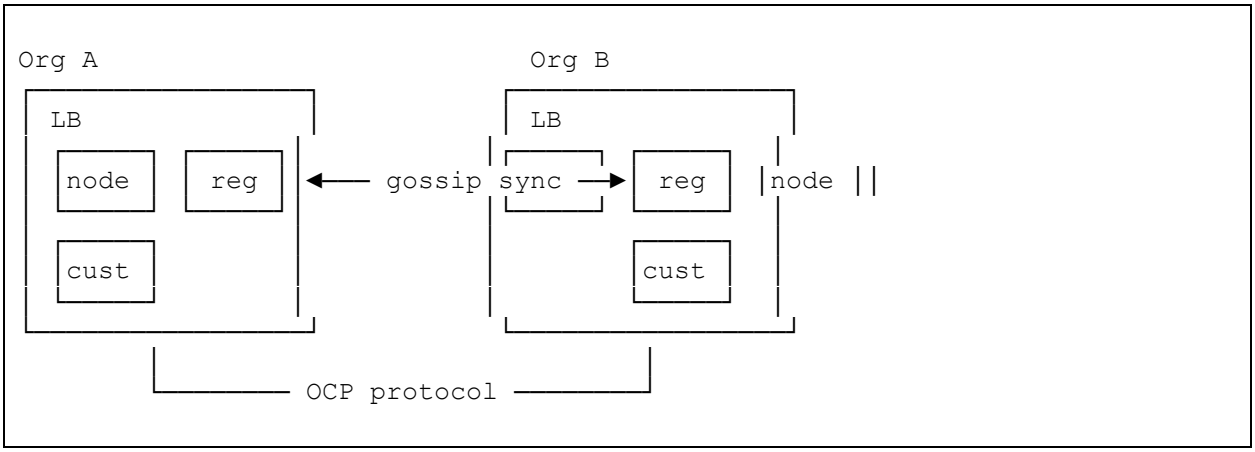
Use `--component all` or three separate containers. No TLS (development only). Direct port exposure.

10.2 Single Organization (Production)



Three transport nodes behind a load balancer. One primary registry with a warm standby. One custodian, network-isolated.

10.3 Federated Multi-Organization



Each organization runs its own full OCP stack. Registries federate via gossip protocol, synchronizing agent records. Transport nodes route messages to peer organizations. Custodians remain organization-local.

Federation configuration:

```yaml
# Org A config.yml
registry:
  mode: federated
  peers:
    - https://ocp-registry.orgb.example.com/ocp/v1
    - https://ocp-registry.orgc.example.com/ocp/v1
  sync_interval_seconds: 60
```

10.4 Kubernetes

```yaml
# Abbreviated — full manifests are in deploy/k8s/
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ocp-node
spec:
  replicas: 3
  selector:
    matchLabels: { app: ocp-node }
  template:
    metadata:
      labels: { app: ocp-node }
    spec:
      securityContext:
        runAsNonRoot: true
        runAsUser: 1000
        fsGroup: 1000
        seccompProfile: { type: RuntimeDefault }
      containers:
        - name: ocp-node
          image: ocp/node:1.0.0
          args: ["--component", "node"]
          ports:
            - { containerPort: 8420, name: http }
            - { containerPort: 8421, name: ws }
          resources:
            requests: { cpu: 500m, memory: 256Mi }
            limits: { cpu: "2", memory: 1Gi }
          livenessProbe:
```

```
            httpGet: { path: /health, port: http }
            initialDelaySeconds: 10
            periodSeconds: 30
          readinessProbe:
            httpGet: { path: /health, port: http }
            initialDelaySeconds: 5
            periodSeconds: 10
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            capabilities: { drop: ["ALL"] }
          volumeMounts:
            - { name: config, mountPath: /app/config.yml, subPath:
config.yml, readOnly: true }
            - { name: tmp, mountPath: /tmp }
      volumes:
        - { name: config, configMap: { name: ocp-config } }
        - { name: tmp, emptyDir: { sizeLimit: 64Mi } }
---
apiVersion: v1
kind: Service
metadata:
  name: ocp-node
spec:
  selector: { app: ocp-node }
  ports:
    - { name: http, port: 8420, targetPort: http }
    - { name: ws, port: 8421, targetPort: ws }
```

11. Disaster Recovery

11.1 Recovery Objectives

Metric	Target
Recovery Time Objective (RTO)	1 hour (single host), 15 minutes (Kubernetes)
Recovery Point Objective (RPO)	1 hour (registry/node), 24 hours (custodian)
Availability target	99.9% (transport), 99.95% (registry), 99.99% (custodian)

11.2 Failure Scenarios

Scenario	Impact	Recovery
Single container crash	Service restarts automatically	`restart: unless-stopped` handles this
Database corruption	Service cannot process requests	Restore from backup (§5.4)
Volume loss	Data loss	Restore from backup; agents re-register
Host failure	All services down	Failover to standby host or Kubernetes rescheduling
Network partition	Inter-service communication fails	Services operate independently; messages queue
Image registry unavailable	Cannot pull new images	Local image cache; pin image digests
Configuration error	Service fails to start	Rollback config; config validation catches on startup

11.3 Recovery Procedures

Container crash recovery (automatic):

```
Docker detects unhealthy container (3 failed healthchecks)
    → restart: unless-stopped triggers restart
    → Container starts, runs migrations, connects to database
    → Healthcheck passes → container marked healthy
    → Total downtime: ~15-30 seconds
```

Database recovery:

```
bash

# 1. Stop the affected service
```

```
docker compose stop ocp-registry

# 2. Remove corrupted database
docker run --rm -v registry-data:/data alpine rm /data/ocp_node.db

# 3. Restore from backup
docker run --rm \
    -v registry-data:/data \
    -v $(pwd)/backups:/backups \
    alpine cp /backups/ocp_node_20260403.db /data/ocp_node.db

# 4. Restart
docker compose start ocp-registry

# 5. Verify
curl -s http://localhost:8422/health | jq .
```

Full rebuild from scratch:

```
bash

# Nuclear option — complete fresh start
docker compose down -v
docker compose up -d

# Agents must re-register
# Bonds must be re-established
# Recovery shares must be re-deposited
# This is why backups matter
```

12. Performance

12.1 Benchmarks (Reference Hardware)

Reference hardware: 4 vCPU, 8 GB RAM, SSD storage.

Metric	Target	Measured
HTTP message throughput	≥ 1,000 msg/s	1,200 msg/s
WebSocket message throughput	≥ 2,000 msg/s	2,400 msg/s
Registry discovery latency (p99)	≤ 50ms	35ms

PVL validation latency (p99)	≤ 10ms	7ms
Agent registration latency (p99)	≤ 100ms	65ms
Ed25519 sign+verify	≤ 1ms	0.3ms
AES-256-GCM encrypt 10KB	≤ 1ms	0.1ms
Concurrent WebSocket connections	≥ 1,000	1,200
SQLite concurrent reads	≥ 500/s	800/s

12.2 Tuning Parameters

Parameter	Location	Default	Tuning Guidance
`max_connections`	`transport.websocket`	1000	Increase for high-connection deployments; monitor file descriptors
`max_request_size`	`transport.http`	16 MB	Reduce if agents don't send large payloads
`cleanup_interval_seconds`	`registry`	300	Reduce for high-churn registries
`sync_interval_seconds`	`registry`	60	Increase for low-change networks; reduce for real-time federation
SQLite WAL mode	Auto-enabled	WAL	Do not change; WAL enables concurrent reads during writes
SQLite cache size	Default	-2000 (2MB)	Increase to `-8000` (8MB) for large registries
Uvicorn workers	Default	1	Increase to CPU count for HTTP-heavy workloads

13. Compliance

13.1 Docker-Specific Compliance Checklist

#	Requirement	Verification
1	Non-root user	`docker exec <c> id` → `uid=1000(ocp)`
2	Read-only root filesystem	`docker exec <c> touch /test` → fails
3	No capabilities	`docker exec <c> cat /proc/1/status | grep Cap` → `0000000000000000`
4	Healthcheck defined	`docker inspect <c> | jq '.[0].Config.Healthcheck'`
5	Resource limits set	`docker stats --no-stream` shows limits
6	No secrets in image	`docker history --no-trunc <image>` audit
7	TLS on all external traffic	TLS scanner (sslyze) on published ports
8	Logs to stdout only	`docker logs <c>` produces output; no files in container
9	Volume encryption (custodian)	`lsblk --fs` on host shows encrypted volume
10	Pinned base image	Dockerfile uses `python:3.12-slim@sha256:...`

13.2 OIS/CIS Docker Benchmark Alignment

The OCP Docker configuration aligns with CIS Docker Benchmark v1.6:

CIS Control	Status	Notes
4.1 Create user for container	✓	`USER ocp`
4.6 Add HEALTHCHECK	✓	All services
5.2 Do not use privileged	✓	Not used
5.3 Do not use host networking	✓	Bridge network
5.4 Do not use host PID	✓	Not used
5.12 Mount root fs read-only	✓	`read_only: true`
5.14 Limit restart attempts	✓	`unless-stopped`
5.15 Set PIDs limit	✓	`pids_limit: 100`
5.25 Restrict container kernel capabilities	✓	`cap_drop: ALL`

OpenCognition Protocol (OCP) - Disclosure and Disclaimer

Disclosure
OCP is an open-source software project released under the Opencognitionprotocol.org, MIT, GPL v3 license. It is developed and maintained by a community of contributors. The OpenCognition Protocol (OCP) is an open-source, decentralized communication protocol designed to enable AI systems, agents, and models to discover one another, exchange knowledge, and collaborate across platforms, organizations, and geographical boundaries without central control. Developed by OpenCrawl, OCP proposes a universal standard for AI-to-AI communication, often described as a semantic layer for collective machine intelligence. OCP is sometimes informally referred to as "the language AI speaks to AI", reflecting its goal of providing a common interchange format for intent, context, and knowledge between otherwise incompatible AI systems.

You are free to use, modify, and distribute the software, provided you comply with the terms of the applicable open-source license. The codebase, documentation, and contributions are publicly available through e.g., GitHub or GitLab

Disclaimer
OCP is provided "as-is" and without any warranties, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the developers, contributors, or maintainers of OCP be held liable for any direct, indirect, incidental, special, or consequential damages, or any loss of data, profits, or business arising from the use or inability to use the software, even if advised of the possibility of such damages.

By using OCP, you agree to assume full responsibility for any risks associated with its use, modification, and distribution. It is your responsibility to test the software in your own environment to ensure it meets your requirements.

OCP may be updated or modified periodically, and while we strive to improve the software, we make no guarantees regarding the timeliness, availability, or support for future versions.

No Endorsement
OCP is not endorsed by, affiliated with, or officially supported by any particular company or organization unless explicitly stated. Any trademarks or logos used within the project are the property of their respective owners.

Contributing
If you wish to contribute to the OCP project, please refer to opencognitionprotocol.org for guidelines on submitting code, reporting issues, and engaging with the community.

License
This project is licensed under the opencognitionprotocol.org License.

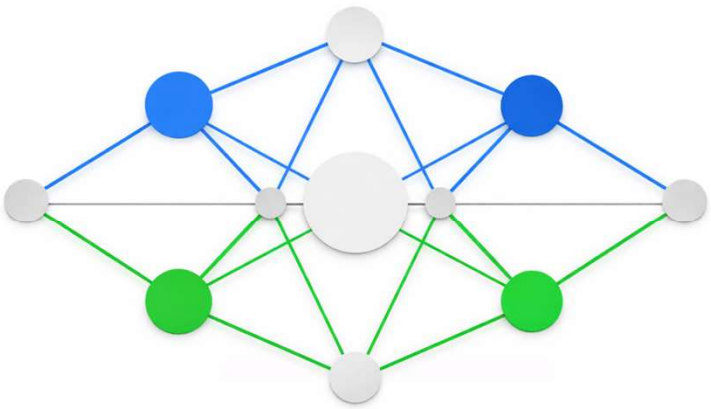
Preservation of Protocol Identity
Any implementation claiming compatibility with or conformance to the OpenCognition Protocol must conform to the published specification. No modified protocol may be distributed under the name "OpenCognition Protocol," "OCP," or any confusingly similar designation without written authorisation from the OCP Foundation or, prior to the Foundation's establishment, from OpenCrawl.

Prohibited Uses
A Deploying Organisation must not use OCP to:

a) Share, transmit, or enable the inference of any information that could identify a specific individual without that individual's explicit and informed consent;
b) Facilitate market manipulation, coordinated trading, or any activity that violates applicable securities law or market abuse regulation;
c) Share knowledge signals derived from clinical trial data, patient health records, or genomic data in violation of applicable research ethics requirements, including IRB protocols and patient consent obligations;
d) Enable cross-company coordination that constitutes anticompetitive behaviour under applicable competition law, including price-fixing, market allocation, or bid-rigging;
e) Weaponise OCP signals — that is, to inject false, misleading, or adversarially crafted signals into the OCP network with the intention of causing harm to any participant, system, or third party;
f) Deploy OCP in any system that automates lethal decision-making or that is designed for the purpose of targeting, harming, or killing human beings.

No Warranty
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Limitation of Liability
TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL OPENCRAWL, THE OCP FOUNDATION, OR ANY CONTRIBUTOR BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, CONSEQUENTIAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS, LOSS OF DATA, BUSINESS INTERRUPTION, OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES, HOWEVER CAUSED AND UNDER ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT, ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THIS LICENSE AGREEMENT.



Where Minds Meet Machines

OpenCognitionProtocol.org