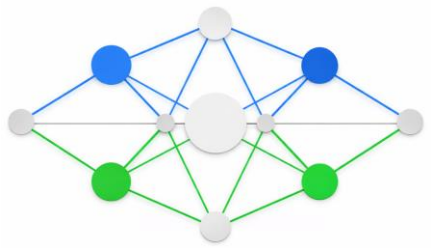


# OCP

# OpenCognition Protocol

*"Where Minds Meet Machines"*

# OpenCognition Protocol (OCP)

## Git Repository Technical Specification v1.0

| | |
|---|---|
| **Introduction** | October 2025 |
| **License** | MIT Licenses |
| **Conformance Identifier** | OCP-REPO-SPEC-1.0 |
| **Specification** | docs.opencognitionprotocol.org |
| **Maintainer** | OCP Technical Steering Committee |

## 1. Purpose

This document defines the authoritative structure, branching model, versioning strategy, release process, contribution workflow, CI/CD pipeline, and governance rules for the official OCP Git repository (`opencognition/ocp-spec`). All contributors, maintainers, and automated systems MUST conform to this specification.

## 2. Repository Identity

| Field | Value |
|---|---|
| Canonical URL | `https://github.com/opencognition/ocp-spec` |
| License | MIT (SPDX: `MIT`) |
| Primary language | Python (SDK), JSON (schemas), Markdown (spec) |
| Minimum Python | 3.11 |
| JSON Schema draft | 2020-12 |
| Default branch | `main` |

## 3. Repository Structure

The repository is organized into seven top-level directories. Each directory has a single owner (a TSC working group or designated maintainer) and a defined scope. Files outside these directories are repository-level configuration and governance documents.

```
ocp-spec/
│
├── spec/                           # Protocol specification
│   ├── SPECIFICATION.md            # Full technical specification
(normative)
│   ├── WHITEPAPER.md               # Original whitepaper (informational)
│   ├── CHANGELOG.md                # Spec version history (normative)
│   └── extensions/                 # Registered extension specifications
│       └── ocp-ext-template.md     # Extension authoring template
│
├── schemas/                        # JSON Schemas (normative)
│   ├── ocpumf.schema.json          # Universal Message Format
│   ├── agent-record.schema.json    # Agent Registry record
│   ├── did-document.schema.json    # DID Document
│   ├── bond-record.schema.json     # Bond record
│   ├── vouch-record.schema.json    # Vouch record
│   ├── knowledge/                  # Knowledge payload schemas
│   │   ├── embedding.schema.json
│   │   ├── insight.schema.json
│   │   └── model-delta.schema.json
│   ├── tasks/                      # Task schemas
│   │   ├── task-request.schema.json
│   │   └── task-response.schema.json
│   ├── consensus/                  # Consensus schemas
│   │   ├── consensus-initiate.schema.json
│   │   ├── consensus-vote.schema.json
│   │   └── consensus-result.schema.json
│   ├── recovery/                   # Recovery schemas
│   │   ├── recovery-share.schema.json
│   │   ├── recovery-request.schema.json
│   │   └── recovery-share-response.schema.json
│   ├── discovery/                  # Discovery schemas
│   │   ├── discovery-ping.schema.json
│   │   ├── capability-query.schema.json
│   │   └── capability-response.schema.json
│   ├── transport/                  # Transport schemas
│   │   ├── auth-handshake.schema.json
│   │   └── auth-result.schema.json
│   ├── registry/                   # Registry API schemas
│   │   ├── discover-request.schema.json
│   │   └── discover-response.schema.json
│   └── errors/                     # Error schemas
│       └── error-response.schema.json
│
├── sdk/                            # Reference SDKs
│   ├── python/                     # Python SDK (reference implementation)
│   │   ├── pyproject.toml
```

```
│   │   ├── README.md
│   │   └── ocp/
│   │       ├── __init__.py
│   │       ├── constants.py       # Protocol constants
│   │       ├── exceptions.py      # Exception hierarchy
│   │       ├── crypto.py          # Cryptographic primitives
│   │       ├── identity.py        # DID generation & management
│   │       ├── messages.py        # OCPUMF builder & validator
│   │       ├── knowledge.py       # Knowledge types
│   │       ├── trust.py           # Trust, vouching, bonds
│   │       ├── pvl.py             # Privacy Validation Layer
│   │       ├── recovery.py        # Shamir SSS key recovery
│   │       ├── transport.py       # WebSocket & HTTP transports
│   │       ├── registry.py        # Registry client
│   │       ├── consensus.py       # Consensus protocol
│   │       ├── agent.py           # High-level Agent class
│   │       ├── cli.py             # CLI tools (ocp-keygen)
│   │       └── compliance.py      # Compliance checker
│   ├── javascript/                # JavaScript SDK (planned)
│   │   └── .gitkeep
│   └── go/                        # Go SDK (planned)
│       └── .gitkeep
│
├── node/                          # Reference OCP Node
│   ├── Dockerfile
│   ├── docker-compose.yml
│   ├── requirements.txt
│   ├── config.yml
│   └── src/
│       └── ocp_node/
│           ├── __init__.py
│           ├── __main__.py        # Entry point
│           ├── database.py        # SQLite persistence
│           ├── message_router.py  # Message routing engine
│           ├── handlers.py        # Message type handlers
│           ├── transport_server.py # HTTP + WebSocket server
│           ├── registry_server.py # Agent Registry API
│           └── custodian_server.py # Recovery custodian
│
├── tests/                         # Compliance test suite
│   ├── conftest.py                # Shared fixtures
│   ├── test_identity.py
│   ├── test_messages.py
│   ├── test_crypto.py
│   ├── test_knowledge.py
│   ├── test_trust.py
│   ├── test_pvl.py
│   ├── test_transport.py
│   ├── test_consensus.py
│   ├── test_recovery.py
│   └── integration/               # Integration tests (require node)
│       ├── test_registry_api.py
│       ├── test_message_flow.py
│       └── test_custodian_api.py
│
├── examples/                      # Runnable example agents
│   ├── simple_agent.py
```

```
│   ├── knowledge_sharing.py
│   ├── multi_agent_consensus.py
│   └── key_recovery.py
│
├── docs/                           # Documentation site source
│   ├── mkdocs.yml
│   └── pages/
│       ├── index.md
│       ├── quickstart.md
│       ├── architecture.md
│       ├── sdk-reference.md
│       ├── node-operations.md
│       └── extension-guide.md
│
├── .github/                        # GitHub configuration
│   ├── ISSUE_TEMPLATE/
│   │   ├── bug_report.yml
│   │   ├── feature_request.yml
│   │   ├── spec_change.yml
│   │   └── extension_proposal.yml
│   ├── PULL_REQUEST_TEMPLATE.md
│   ├── CODEOWNERS
│   ├── workflows/
│   │   ├── ci.yml                  # Continuous integration
│   │   ├── release.yml             # Release automation
│   │   ├── schema-validation.yml   # JSON Schema linting
│   │   ├── security-audit.yml      # Dependency & crypto audit
│   │   └── docs-deploy.yml         # Documentation deployment
│   └── dependabot.yml
│
├── .gitignore
├── .editorconfig
├── LICENSE
├── README.md
├── CONTRIBUTING.md
├── CODE_OF_CONDUCT.md
├── SECURITY.md
└── GOVERNANCE.md
```

## 3.1 Directory Ownership

| Directory | Owner | Approval Required |
|---|---|---|
| `spec/` | TSC (Technical Steering Committee) | 2 TSC members |
| `schemas/` | TSC | 2 TSC members |
| `sdk/python/` | SDK Working Group | 1 SDK WG member + 1 TSC |
| `sdk/javascript/` | SDK Working Group | 1 SDK WG member + 1 TSC |
| `sdk/go/` | SDK Working Group | 1 SDK WG member + 1 TSC |
| `node/` | Node Working Group | 1 Node WG member + 1 TSC |
| `tests/` | SDK Working Group | 1 SDK WG member |
| `examples/` | Community Council | 1 CC member |
| `docs/` | Community Council | 1 CC member |
| `.github/` | TSC | 1 TSC member |
| Root files | TSC | 2 TSC members |

## 3.2 CODEOWNERS

```
# .github/CODEOWNERS

# Specification (normative)
/spec/                      @ocp-tsc
/schemas/                   @ocp-tsc

# Python SDK
/sdk/python/                @ocp-sdk-wg
/tests/                     @ocp-sdk-wg

# Node
/node/                      @ocp-node-wg

# Governance & root
/GOVERNANCE.md              @ocp-tsc
/SECURITY.md                @ocp-security-wg
/CONTRIBUTING.md            @ocp-community-council
/CODE_OF_CONDUCT.md         @ocp-community-council
/README.md                  @ocp-tsc
```

```
# CI/CD
/.github/workflows/       @ocp-tsc

# Documentation
/docs/                    @ocp-community-council

# Examples
/examples/                @ocp-community-council
```

## 4. Branching Model

OCP uses a trunk-based development model with short-lived feature branches and protected release branches.

### 4.1 Branch Types

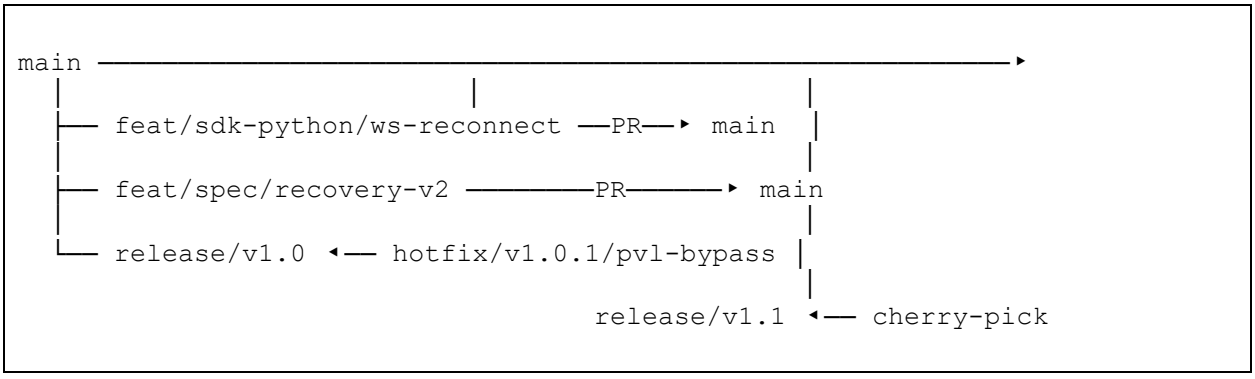
| Branch | Pattern | Lifetime | Protection |
|---|---|---|---|
| **main** | `main` | Permanent | Protected: require PR, 2 approvals, CI pass, no force push |
| **release** | `release/v<MAJOR>.<MINOR>` | Until EOL | Protected: require PR, 2 TSC approvals, CI pass |
| **feature** | `feat/<scope>/<short-description>` | Days to weeks | Unprotected; deleted after merge |
| **fix** | `fix/<scope>/<short-description>` | Days | Unprotected; deleted after merge |
| **hotfix** | `hotfix/v<version>/<description>` | Hours to days | Requires 1 TSC emergency approval |

## 4.2 Scope Identifiers

Scopes correspond to top-level directories and are used in branch names and commit messages:

| Scope | Directory |
|---|---|
| `spec` | `spec/` |
| `schemas` | `schemas/` |
| `sdk-python` | `sdk/python/` |
| `sdk-js` | `sdk/javascript/` |
| `sdk-go` | `sdk/go/` |
| `node` | `node/` |
| `tests` | `tests/` |
| `examples` | `examples/` |
| `docs` | `docs/` |
| `ci` | `.github/` |
| `governance` | Root governance files |

## 4.3 Branch Lifecycle




## 4.4 Rules

1. All changes to `main` MUST go through a Pull Request.
2. Feature branches MUST be rebased onto `main` before merge (no merge commits).
3. Feature branches MUST be deleted after merge.
4. Release branches are cut from `main` at the point of release.
5. Hotfixes are applied to the release branch and cherry-picked to `main`.
6. Direct commits to `main` or any release branch are prohibited.

## 5. Commit Convention

All commits MUST follow Conventional Commits v1.0.0.

### 5.1 Format

```
<type>(<scope>): <description>

[optional body]

[optional footer(s)]
```

### 5.2 Types

| Type | Meaning | Triggers |
|---|---|---|
| `feat` | New feature | Minor version bump |
| `fix` | Bug fix | Patch version bump |
| `docs` | Documentation only | No version bump |
| `style` | Formatting, whitespace | No version bump |
| `refactor` | Code change that neither fixes nor adds | No version bump |
| `perf` | Performance improvement | Patch version bump |
| `test` | Adding or correcting tests | No version bump |
| `build` | Build system or dependencies | No version bump |
| `ci` | CI configuration | No version bump |
| `chore` | Maintenance tasks | No version bump |
| `revert` | Reverts a prior commit | Depends on reverted type |
| `security` | Security fix | Patch version bump |

## 5.3 Breaking Changes

Breaking changes MUST include `BREAKING CHANGE:` in the commit footer or `!` after the type/scope:

```
feat(spec)!: rename knowledge_share payload field

BREAKING CHANGE: The "findings" field in insight payloads is
renamed to "payload" for consistency with the OCPUMF spec.
Migration: rename all "findings" keys to "payload" in existing
insight packages.
```

Breaking changes trigger a **major** version bump.

## 5.4 Examples

```
feat(sdk-python): add WebSocket automatic reconnection with backoff

fix(schemas): correct bond-record expires_at to require date-time format

docs(spec): clarify trust score computation edge cases

test(tests): add PVL rejection code coverage for PVL-003

refactor(node): extract message routing into dedicated module

security(sdk-python): upgrade cryptography to 44.0 for CVE-2026-XXXXX

ci(ci): add schema validation workflow for all JSON files

feat(spec)!: add recovery_request and recovery_share_response message types

BREAKING CHANGE: OCPUMF message_type enum now includes two additional
values. Implementations that validate against a closed enum must be updated.
```

# 6. Versioning

## 6.1 Protocol Version

The OCP protocol version follows `MAJOR.MINOR`:

| Component | Meaning |
|---|---|
| MAJOR | Breaking protocol changes (incompatible wire format, removed message types) |
| MINOR | Backward-compatible additions (new message types, new optional fields) |

The protocol version is encoded in every OCPUMF message as `ocp_version`.

**Current:** `1.0`

## 6.2 Software Versions

All software artifacts (SDK, node, schemas) follow Semantic Versioning 2.0.0 independently:

```
MAJOR.MINOR.PATCH[-prerelease][+build]
```

| Artifact | Current Version | Version Source |
|---|---|---|
| Protocol specification | 1.0 | spec/SPECIFICATION.md header |
| JSON Schemas | 1.0.0 | $id URL path |
| Python SDK | 1.0.0 | sdk/python/ocp/__init__.py → __version__ |
| Reference Node | 1.0.0 | node/src/ocp_node/__init__.py → __version__ |

### 6.3 Version Synchronization Rules

1. A protocol MAJOR bump forces a MAJOR bump on all schemas and SDKs.
2. A protocol MINOR bump forces at minimum a MINOR bump on schemas.
3. SDK and node versions are independent of each other.
4. Schema versions MUST match the protocol version they implement.

### 6.4 Git Tags

Tags follow the format `<artifact>/<version>`:

```
spec/v1.0
schemas/v1.0.0
sdk-python/v1.0.0
sdk-python/v1.0.1
sdk-python/v1.1.0
node/v1.0.0
```

Tags are annotated (not lightweight) and MUST be signed by a TSC member:

```
git tag -s sdk-python/v1.1.0 -m "sdk-python v1.1.0: add WebSocket
reconnection"
```

# 7. Release Process

## 7.1 Release Types

| Type | Cadence | Approval | Deprecation Notice |
|---|---|---|---|
| **Patch** (x.y.Z) | As needed | 1 TSC + CI pass | None |
| **Minor** (x.Y.0) | Quarterly target | 2 TSC + CI pass | None |
| **Major** (X.0.0) | Annual target | TSC supermajority (5/7) + community RFC | 12 months |

## 7.2 Release Checklist

Each release MUST complete the following steps in order:

1. **Freeze.** Announce feature freeze on the `main` branch.
2. **Branch.** Create `release/v<MAJOR>.<MINOR>` from `main`.
3. **Version bump.** Update all version strings (spec, schemas, SDK, node).
4. **Changelog.** Update `spec/CHANGELOG.md` and relevant `CHANGELOG` files.
5. **Schema validation.** Run `schema-validation.yml` workflow — all schemas must pass.
6. **Compliance.** Run `ocp-compliance` against the SDK — all checks must pass.
7. **Test suite.** Full `pytest` run — 100% of tests must pass, coverage ≥ 85%.
8. **Security audit.** Run `security-audit.yml` — no critical or high vulnerabilities.
9. **Documentation.** Build and review docs site.
10. **TSC vote.** Required approvals per §7.1.
11. **Tag.** Create signed annotated tag(s).
12. **Publish.** Push to PyPI (SDK), Docker Hub (node), GitHub Releases (all).
13. **Announce.** Post to OCP blog, mailing list, and social channels.

## 7.3 Release Artifacts

| Artifact | Registry | Format |
|---|---|---|
| Python SDK | PyPI (`ocp-protocol`) | Wheel + sdist |
| Reference Node | Docker Hub (`ocp/node`) | Docker image (linux/amd64, linux/arm64) |
| JSON Schemas | GitHub Releases + CDN | Zip archive |
| Specification | GitHub Releases | PDF + Markdown |
| Compliance suite | PyPI (bundled with SDK) | Python package |

## 7.4 Deprecation Policy

Deprecated features MUST:

1. Be annotated with `@deprecated` in code and `[DEPRECATED]` in spec text.
2. Emit a runtime warning in the SDK when used.
3. Remain functional for the entire deprecation window.
4. Be documented in the CHANGELOG with the planned removal version.

Deprecation windows:

- Protocol features: 12 months minimum.
- SDK features: 6 months minimum (2 minor releases).
- Node features: 6 months minimum.

# 8. CI/CD Pipeline

## 8.1 Continuous Integration (`ci.yml`)

Triggered on every push and PR to `main`:

```yaml
name: CI
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install ruff
      - run: ruff check sdk/python/ node/src/ tests/ examples/
      - run: ruff format --check sdk/python/ node/src/

  typecheck:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install mypy sdk/python/[dev]
      - run: mypy sdk/python/ocp/ --strict

  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: ["3.11", "3.12", "3.13"]
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "${{ matrix.python-version }}" }
      - run: pip install sdk/python/[dev]
      - run: pytest tests/ -v --cov=ocp --cov-report=xml --cov-fail-under=85
      - uses: codecov/codecov-action@v4
        with: { files: coverage.xml }

  compliance:
    runs-on: ubuntu-latest
    steps:
```

```
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install sdk/python/
      - run: python -m ocp.compliance

  node-build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: docker build -t ocp-node-test -f node/Dockerfile .
      - run: |
          docker run --rm ocp-node-test python -c "
          from ocp_node import __version__
          print(f'Node version: {__version__}')
          "

  integration:
    runs-on: ubuntu-latest
    needs: [test, node-build]
    steps:
      - uses: actions/checkout@v4
      - run: docker compose -f node/docker-compose.yml up -d
      - run: sleep 10
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install sdk/python/[dev]
      - run: pytest tests/integration/ -v --timeout=60
      - run: docker compose -f node/docker-compose.yml down
```

### 8.2 Schema Validation (`schema-validation.yml`)

```yaml
name: Schema Validation
on:
  push:
    paths: ["schemas/**"]
  pull_request:
    paths: ["schemas/**"]

jobs:
  validate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install jsonschema check-jsonschema
      - run: |
          find schemas/ -name "*.schema.json" | while read f; do
            echo "Validating: $f"
```

```
            check-jsonschema --check-metaschema "$f"
          done
      - run: |
          python -c "
          import json, os
          for root, dirs, files in os.walk('schemas'):
              for f in files:
                  if f.endswith('.schema.json'):
                      path = os.path.join(root, f)
                      with open(path) as fh:
                          schema = json.load(fh)
                      assert '\$id' in schema, f'{path}: missing \$id'
                      assert '\$schema' in schema, f'{path}: missing
\$schema'
                      assert 'title' in schema, f'{path}: missing title'
                      assert 'description' in schema, f'{path}: missing
description'
                      print(f'  ✓ {path}')
          "
```

## 8.3 Security Audit (`security-audit.yml`)

```yaml
name: Security Audit
on:
  schedule:
    - cron: "0 6 * * 1"   # Weekly Monday 06:00 UTC
  push:
    paths: ["sdk/python/pyproject.toml", "node/requirements.txt"]

jobs:
  audit:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install pip-audit safety
      - run: pip-audit --requirement node/requirements.txt --strict
      - run: pip install sdk/python/ && pip-audit --strict
      - run: |
          # Verify cryptographic library versions
          python -c "
          import cryptography
          v = tuple(int(x) for x in cryptography.__version__.split('.')[:2])
          assert v >= (43, 0), f'cryptography {cryptography.__version__} is
below minimum 43.0'
          print(f'cryptography: {cryptography.__version__} ✓')
          "
```

## 8.4 Release Automation (`release.yml`)

```yaml
name: Release
on:
  push:
    tags: ["sdk-python/v*", "node/v*", "schemas/v*", "spec/v*"]

jobs:
  release-sdk:
    if: startsWith(github.ref, 'refs/tags/sdk-python/')
    runs-on: ubuntu-latest
    environment: pypi
    permissions:
      id-token: write
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.12" }
      - run: pip install build twine
      - run: cd sdk/python && python -m build
      - uses: pypa/gh-action-pypi-publish@release/v1
        with: { packages-dir: sdk/python/dist/ }

  release-node:
    if: startsWith(github.ref, 'refs/tags/node/')
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: docker/setup-buildx-action@v3
      - uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USER }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      - uses: docker/build-push-action@v5
        with:
          context: .
          file: node/Dockerfile
          push: true
          platforms: linux/amd64,linux/arm64
          tags: |
            ocp/node:${{ github.ref_name }}
            ocp/node:latest

  release-github:
    runs-on: ubuntu-latest
    permissions:
      contents: write
    steps:
      - uses: actions/checkout@v4
      - uses: softprops/action-gh-release@v2
        with:
          generate_release_notes: true
          files: |
            spec/SPECIFICATION.md
```

```
            spec/CHANGELOG.md
```

### 8.5 Pipeline Requirements

All PRs to `main` MUST satisfy these gates:

| Gate | Requirement |
|---|---|
| Lint | `ruff check` and `ruff format --check` pass with zero errors |
| Type check | `mypy --strict` passes with zero errors |
| Unit tests | 100% of tests pass across Python 3.11, 3.12, 3.13 |
| Coverage | ≥ 85% line coverage |
| Compliance | `ocp-compliance` exits with code 0 |
| Schema validation | All `.schema.json` files pass metaschema validation |
| Node build | Docker image builds successfully |
| Integration | Integration tests pass against a running node |
| No conflicts | Branch is rebased on latest `main` |
| Approvals | Per CODEOWNERS (§3.2) |

## 9. Contribution Workflow

### 9.1 Standard Flow

```
1. Fork or clone the repository.
2. Create a branch:       git checkout -b feat/<scope>/<description>
3. Make changes.
4. Commit with convention: git commit -m "feat(sdk-python): add X"
5. Push:                   git push origin feat/<scope>/<description>
6. Open a Pull Request against main.
7. Address review feedback.
8. Squash and rebase if requested.
9. Maintainer merges (rebase-merge, not squash-merge for multi-commit PRs).
10. Branch is auto-deleted.
```

## 9.2 Spec Change Flow (RFC Process)

Changes to `spec/SPECIFICATION.md` or `schemas/` follow a heavier process:

```
1. Open a GitHub Issue tagged [RFC] describing the proposed change.
2. TSC labels the issue for discussion (14-day comment period).
3. If the RFC receives TSC sponsorship (at least 1 TSC member):
   a. Author creates a branch: feat/spec/<rfc-number>-<short-name>
   b. Author submits a PR with:
      - Updated SPECIFICATION.md
      - Updated or new schemas
      - Updated SDK code (if the change affects wire format)
      - Updated tests
      - CHANGELOG entry
   c. PR requires 2 TSC approvals.
   d. For breaking changes: TSC supermajority vote (5/7).
4. After merge, the TSC schedules inclusion in the next release.
```

## 9.3 Extension Proposal Flow

New domain extensions (`ocp-ext:*`) follow:

```
1. Copy spec/extensions/ocp-ext-template.md to a new file.
2. Fill in all sections (scope, schemas, message types, examples).
3. Submit a PR to spec/extensions/ and schemas/ (if new schemas are needed).
4. TSC reviews for:
   - No conflict with core spec
   - Schema validity
   - Test coverage for new message types
5. 1 TSC approval required.
```

## 9.4 Security Disclosure Flow

Security-sensitive contributions follow `SECURITY.md`:

1. Report to `security@ocp.foundation` (never a public issue).
2. SWG acknowledges within 48 hours.
3. Fix developed on a private branch.
4. Coordinated disclosure after fix is ready (default 90 days).
5. Hotfix released per §7.1.

# 10. Quality Standards

## 10.1 Specification Text

- MUST use RFC 2119 keywords (MUST, SHOULD, MAY) consistently.
- MUST include a JSON example for every data structure introduced.
- MUST cross-reference related sections with section numbers.
- MUST NOT contain implementation-specific language (no Python, Go, etc. in normative text).
- Line length SHOULD be ≤ 100 characters (soft wrap for readability in diffs).

## 10.2 JSON Schemas

- MUST use `$schema: "https://json-schema.org/draft/2020-12/schema"`.
- MUST include `$id` with canonical URL.
- MUST include `title` and `description` on every schema.
- MUST include `description` on every property.
- MUST use `additionalProperties: false` unless extensibility is explicitly intended.
- MUST validate against the JSON Schema metaschema (enforced by CI).
- SHOULD use `$defs` for reusable type definitions.
- File naming: `<noun-noun>.schema.json`, lowercase, hyphenated.

## 10.3 Python Code

- MUST format with `ruff format` (line length 100).
- MUST pass `ruff check` with zero errors.
- MUST pass `mypy --strict` with zero errors.
- MUST include type hints on all function signatures and return types.
- MUST include docstrings on all public classes, methods, and functions.
- Docstrings follow Google style.
- Variable names: concise but readable (`msg` not `m`, `agent_id` not `aid`).
- MUST NOT use `print()` — use `structlog` or `logging`.
- MUST NOT use `assert` for validation — use explicit checks with exceptions.
- Test files mirror source structure: `ocp/crypto.py` → `tests/test_crypto.py`.

### 10.4 Docker / Node

- Dockerfile MUST use a pinned base image tag (not `:latest`).
- `HEALTHCHECK` MUST be defined for every service.
- Secrets MUST be injected via environment variables, never in config files.
- All ports MUST be documented in `docker-compose.yml` comments.
- Configuration MUST support environment variable overrides.

### 10.5 Tests

- Unit tests MUST NOT require network access.
- Integration tests MUST be in `tests/integration/` and marked with `@pytest.mark.integration`.
- Tests MUST be deterministic (no random failures).
- Tests MUST NOT share state between test functions.
- Each test file MUST be independently runnable.
- Fixtures MUST be defined in `conftest.py`, not duplicated.

## 11. File Format Standards

| File Type | Encoding | Line Ending | Indentation | Trailing Newline |
|---|---|---|---|---|
| `.md` | UTF-8 | LF | 2 spaces (lists) | Yes |
| `.json` | UTF-8 | LF | 2 spaces | Yes |
| `.py` | UTF-8 | LF | 4 spaces | Yes |
| `.yml` / `.yaml` | UTF-8 | LF | 2 spaces | Yes |
| `.toml` | UTF-8 | LF | 4 spaces | Yes |
| `Dockerfile` | UTF-8 | LF | 4 spaces | Yes |

## 11.1 EditorConfig

```ini
# .editorconfig
root = true

[*]
end_of_line = lf
insert_final_newline = true
charset = utf-8
trim_trailing_whitespace = true

[*.py]
indent_style = space
indent_size = 4

[*.{json,yml,yaml,md}]
indent_style = space
indent_size = 2

[*.toml]
indent_style = space
indent_size = 4

[Dockerfile]
indent_style = space
indent_size = 4

[Makefile]
indent_style = tab
```

# 12. Issue and PR Templates

## 12.1 Issue Labels

| Label | Color | Scope |
|---|---|---|
| `spec` | `#0075ca` | Protocol specification |
| `schemas` | `#0075ca` | JSON Schemas |
| `sdk-python` | `#1d76db` | Python SDK |
| `sdk-js` | `#1d76db` | JavaScript SDK |
| `sdk-go` | `#1d76db` | Go SDK |
| `node` | `#5319e7` | Reference node |
| `tests` | `#d4c5f9` | Test suite |
| `docs` | `#0e8a16` | Documentation |
| `security` | `#b60205` | Security-sensitive |
| `breaking` | `#b60205` | Breaking change |
| `RFC` | `#fbca04` | Request for Comments |
| `extension` | `#fbca04` | Extension proposal |
| `good-first-issue` | `#7057ff` | Newcomer-friendly |
| `help-wanted` | `#008672` | Community contribution welcome |
| `wontfix` | `#ffffff` | Will not be addressed |
| `duplicate` | `#cfd3d7` | Duplicate of another issue |
| `P0-critical` | `#b60205` | Blocks release |
| `P1-high` | `#d93f0b` | Important, next release |
| `P2-medium` | `#fbca04` | Planned, no urgency |
| `P3-low` | `#0e8a16` | Nice to have |

## 12.2 PR Template

*markdown*

```
<!-- .github/PULL_REQUEST_TEMPLATE.md -->

## Summary

<!-- What does this PR do? One paragraph. -->

## Type

- [ ] `feat` — New feature
- [ ] `fix` — Bug fix
- [ ] `docs` — Documentation
- [ ] `refactor` — Code improvement
- [ ] `test` — Test addition/fix
- [ ] `security` — Security fix
- [ ] `spec` — Specification change (RFC required)
- [ ] `schemas` — Schema change

## Breaking Change?

- [ ] No
- [ ] Yes — describe migration path below

## Checklist

- [ ] Commits follow Conventional Commits
- [ ] Tests added/updated
- [ ] Documentation updated (if user-facing)
- [ ] `ruff check` passes
- [ ] `mypy --strict` passes
- [ ] CHANGELOG updated (for feat/fix/security)
- [ ] Schemas validate against metaschema (if changed)

## Related Issues

<!-- Closes #NNN -->
```

# 13. Dependency Management

## 13.1 Policy

- All dependencies MUST be pinned to a minimum version with `>=` (not exact pin) in `pyproject.toml`.
- Lock files are NOT committed (the SDK is a library, not an application).
- The node's `requirements.txt` uses `>=` pins for reproducible installs within Docker.
- Dependabot is configured to open PRs for security updates weekly.

## 13.2 Dependabot Configuration

*yaml*

```yaml
# .github/dependabot.yml
version: 2
updates:
  - package-ecosystem: pip
    directory: /sdk/python
    schedule:
      interval: weekly
    reviewers:
      - ocp-sdk-wg
    labels:
      - dependencies
      - sdk-python

  - package-ecosystem: pip
    directory: /node
    schedule:
      interval: weekly
    reviewers:
      - ocp-node-wg
    labels:
      - dependencies
      - node

  - package-ecosystem: docker
    directory: /node
    schedule:
      interval: weekly
    reviewers:
      - ocp-node-wg
    labels:
      - dependencies
      - node
```

```
  - package-ecosystem: github-actions
    directory: /
    schedule:
      interval: weekly
    reviewers:
      - ocp-tsc
    labels:
      - dependencies
      - ci
```

### 13.3 Approved Cryptographic Libraries

Only the following cryptographic libraries are approved for use in OCP implementations. Using an unapproved library requires SWG review:

| Library | Language | Approved Versions |
|---|---|---|
| `cryptography` | Python | ≥ 43.0 |
| `pynacl` | Python | ≥ 1.5.0 |
| `@noble/ed25519` | JavaScript | ≥ 2.0 |
| `@noble/curves` | JavaScript | ≥ 1.4 |
| `golang.org/x/crypto` | Go | latest |

## 14. Archival and Migration

### 14.1 Repository Archival

If the OCP project is discontinued:

1. The repository is archived (read-only) on GitHub.
2. A full mirror is published to the Internet Archive.
3. The specification and schemas are published to a permanent DOI via Zenodo.
4. All Docker images remain available on Docker Hub.
5. The PyPI package remains available (yanking is prohibited).

### 14.2 Platform Migration

If the repository must move from GitHub:

1. TSC votes to approve the migration target (supermajority required).
2. Full Git history is preserved (no squashing).
3. All issues and PRs are exported.
4. The GitHub repository becomes a redirect/archive pointing to the new location.
5. All CI/CD pipelines are rebuilt on the new platform.
6. A 6-month transition period is maintained where both locations accept contributions.

## Appendix A: Quick Reference

### Branching

```bash
# New feature
git checkout -b feat/sdk-python/ws-reconnect
# ... work ...
git commit -m "feat(sdk-python): add WebSocket automatic reconnection"
git push origin feat/sdk-python/ws-reconnect
# Open PR → review → merge → branch auto-deleted

# Hotfix on release
git checkout release/v1.0
git checkout -b hotfix/v1.0.1/pvl-bypass
# ... fix ...
git commit -m "security(pvl): prevent PII scanner bypass via nested
encoding"
# PR to release/v1.0 → merge → tag v1.0.1 → cherry-pick to main
```

## Tagging

```bash
# SDK release
git tag -s sdk-python/v1.1.0 -m "sdk-python v1.1.0"
git push origin sdk-python/v1.1.0

# Spec release
git tag -s spec/v1.1 -m "OCP Specification v1.1"
git push origin spec/v1.1
```

## Running CI locally

```bash
# Lint
ruff check sdk/python/ node/src/ tests/ examples/
ruff format --check sdk/python/

# Type check
mypy sdk/python/ocp/ --strict

# Tests
pytest tests/ -v --cov=ocp --cov-fail-under=85

# Compliance
python -m ocp.compliance

# Schema validation
find schemas/ -name "*.schema.json" -exec check-jsonschema --check-
metaschema {} \;

# Node build
docker compose -f node/docker-compose.yml build
```

# 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.

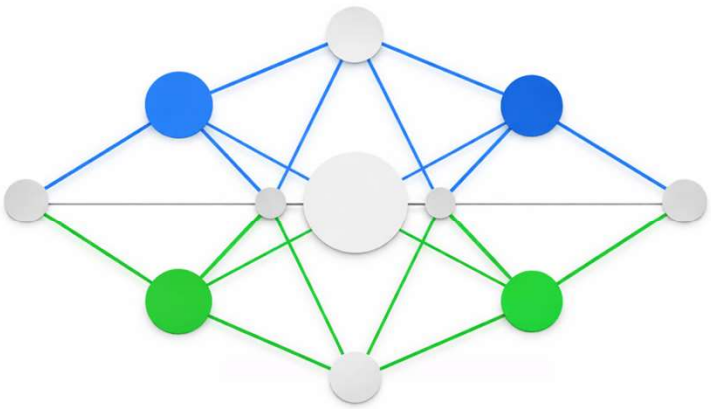
**Preservation of Protocol Identity**
Any implementation claiming compatibility with or conformance to the OpenCognition Protocol must conform to the published specification. No modified protocol may be distributed under the name "OpenCognition Protocol," "OCP," or any confusingly similar designation 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