ChatGPT

에이전트 설정

- `.agents/config.json` **구조:** 이 파일에는 활성 에이전트 이름(`"active"`)과 훅 설정(`"hooks"`) 등이 포함됩니다. 기본값은 `"gemini"`이며, `invoke agent.set --name <agent>` 명령으로 변경 가능합니다. 실제 활성 에이전트는 우선적으로 `ACTIVE_AGENT` 환경변수로 지정하며, 없으면 config.json의 `"active"` 값을 사용합니다 [1] [2].
- **비상 래퍼 토글:** `.agents/emergency.json` 파일의 `"enabled": true`로 설정하면 Codex CLI 호출이 `tools/codex_emergency.ps1` 래퍼를 통해 실행되도록 설정됩니다. 이 상태에서는 환경변수(`CODEX_…`)를 통해 `rps`, `max_tokens`, `retry` 등의 값이 주입되며, 비상모드 해제 시 `"enabled": false`로 변경 후 세션을 재시작합니다 [3].
- **에이전트 전환 정책:** 현재 활성 에이전트는 `invoke agent.status`로 확인 가능하며, `invoke agent.set --name codex` 처럼 명령어로 전환합니다. config.json의 `"active"` 값을 업데이트하는 스크립트(`scripts/agent_manager.py`)가 있어, `ACTIVE_AGENT`를 일시적으로 설정하여 프로세스 단위로 라벨링할 수도 있습니다 [1] [2]. Codex와 Gemini 에이전트 간에는 “파일 수정 프로토콜”이 정의되어 있어 시스템 파일 수정은 Codex가 직접 수행하고, Gemini는 사용자의 승인 후 Codex에게 작업 메시지를 보내는 방식으로 협업합니다 [1].

Git 훅/보안 정책

- **Pre-commit 훅 동작:** `.agents/config.json`의 `"hooks.enabled"` 값으로 전역 훅 활성화/비활성화를 제어합니다 [4]. 커밋 시 기본적으로 `invoke commit_safe`를 사용하며, 인터랙티브한 변경 사항 확인을 건너뛰려면 `--skip-diff-confirm` 플래그를 추가하거나 환경변수 `SKIP_DIFF_CONFIRM=1`을 설정합니다 [4]. 외부 GUI 툴(Sourcetree 등) 사용 시에는 훅을 끄도록 권장합니다.
- **Secrets Guard:** 커밋 시 `.gemini/` 디렉터리의 민감파일(`oauth, creds, token, secret`, 확장자 .json/.db/.sqlite/.pem/.p12/.key 등)과 `projects/` 경로는 스테이징 자체를 차단합니다 [5]. `.githooks/pre-commit`과 `scripts/hooks/precommit_secrets_guard.py`가 이러한 패턴을 검사하여 위반 시 커밋을 중단합니다. 예를 들어 `.gemini/` 내 토큰이나 인증 정보 파일은 로컬 전용으로 간주되어 커밋이 금지됩니다 [5].
- `.gemini/` **보안 규칙:** `.gemini/` 폴더는 로컬 설정·비밀 정보 저장용입니다. 이에 따라 `*.creds*.json`, `*oauth*.*` 등 인증 정보 파일은 절대 커밋하지 않습니다 [5]. 유일하게 버전 관리가 허용되는 파일은 `context_policy.yaml`뿐으로, 이 파일은 컨텍스트 생성 정책(whitelist 스키마)을 정의합니다 [6]. 컨텍스트 정책은 `sources`, `tokens`, `context_limits` 키만 유효하며, 다른 키는 무시되거나 향후 제거됩니다 [6]. 비밀·토큰 노출 시에는 즉시 **키 회전** 및 **Git 이력 정리** 절차를 따라야 합니다 [5] [7].

품질 게이트

- **테스트 전략:** `pytest -q` 명령으로 전체 테스트를 통과해야 합니다. 프로젝트에 정의된 테스트들이 pytest.ini 설정(예: xfail·mark 등) 아래에서 정상 실행되어야 합니다. 예를 들어 `invoke test`로 테스트를 실행하여 모든 테스트가 성공해야 PR 머지가 가능합니다 [8].
- **주요 의존성:** `requirements.txt`에는 주요 라이브러리가 명시되어 있습니다. 예를 들어 Invoke(태스크 관리), PyYAML(구성 파일 파싱), rich(콘솔 출력), LLM SDK(예: Codex/Gemini 클라이언트) 등과 테스트/분석 도구(pytest, mypy, ruff 등)가 포함됩니다. 개발 환경은 venv 기반이 권장되며, Windows 환경에서는 `%REPO%/venv/Scripts/python.exe`를, 그렇지 않으면 시스템 파이썬을 사용하도록 권장합니다 [9] [4].

- **정적 분석 및 CI:** 코드 품질 강화를 위해 ruff(스타일·버그 탐지), mypy(타입 검증) 등의 정적분석 도구와 비밀 검사(Secret Scan)를 CI 파이프라인에 도입합니다 [8]. 모든 PR은 Windows 기반 CI(예: GitHub Actions)에서 이들 도구와 테스트를 통과해야 병합할 수 있습니다 [8].

운영 표준 연계

- **표준과 설정의 일치:** GEMINI.md에서 명시한 운영 원칙(Windows-first, UTF-8, 레포 내부 작업, 비밀 커밋 금지 등)은 실제 설정에 반영되어 있습니다. 예를 들어 Windows 우선 정책(`Python 직접 호출`, `UTF-8 고정`)이 문서화되어 있으며 [9], 실제 셸 프로필 스크립트에도 PowerShell 7 프로필로 UTF-8 인코딩 설정 등이 포함되어 있습니다.
- **Git 훅 및 시크릿 관리:** GEMINI.md는 `.gemini/*` 내 비밀 금지와 프로젝트 로컬 작업 등을 규정하고 있으며 [5], 실질적으로는 pre-commit 가드 스크립트가 이를 구현합니다. 훅 온/오프 토글 기능(`invoke git.set-hooks`)과 `SKIP_DIFF_CONFIRM` 플래그도 문서에 언급된 대로 적용되어 있습니다 [4] [5].
- **품질 게이트 연계:** 문서에는 `pytest` 통과와 정적 분석 도입, Windows CI 필수 통과 등이 명시되어 있으며 [8], 실제 CI 설정에서도 해당 조건이 병합 규칙으로 적용됩니다. 즉, 로컬 `invoke test`와 원격 빌드 모두에서 동일한 품질 기준이 유지됩니다. 이처럼 운영 표준이 각 설정 파일 및 CI 워크플로우에 충실히 반영되어, 보안·품질이 통일성 있게 관리되고 있습니다 [5] [8].

장점과 한계

- **장점:** 엄격한 시크릿 관리와 훅 정책 덕분에 비밀 유출 위험을 줄이고, 모든 변경사항에 대해 통일된 품질 검사(테스트·정적 분석)를 보장합니다. 운영 환경·절차가 문서화되어 있어 재현성과 인수인계가 용이합니다. 예를 들어 비밀 키 노출 시 회전·로그 정리 절차가 명확히 정의되어 있습니다 [5] [7].
- **한계:** 그러나 **Windows-first** 정책(예: PowerShell/인코딩 고정)은 다른 OS에서의 사용을 제약할 수 있습니다 [9]. `SKIP_DIFF_CONFIRM` 환경변수나 훅 비활성 옵션은 사용자가 손쉽게 보안 검사를 우회할 가능성을 열어두며, 훅이 완전히 신뢰될 수는 없습니다. 또한 `.gemini/`와 같은 로컬 비밀 관리 구조는 초깃값 설정과 키 갱신 관리가 다소 복잡하여 사용자 부담이 될 수 있습니다.

사용 맥락 요약

- **커밋 제한:** `.gemini/` 디렉터리 및 `projects/` 아래의 비밀/토큰 파일은 **절대 커밋해서는 안 됩니다** [5]. 이를 위반하면 pre-commit 스크립트가 커밋을 차단합니다.
- **안전한 커밋:** 변경사항을 커밋할 때는 `invoke git.review`나 `invoke commit_safe`를 사용하고, Diff 확인이 필요 없으면 `SKIP_DIFF_CONFIRM=1`을 설정합니다 [4].
- **테스트 및 CI 준수:** 모든 코드 변경은 `pytest -q` 테스트를 통과해야 하며, 정적분석(rsss/ruff/mypy)도 주기적으로 수행해야 합니다. PR을 병합하려면 Windows 기반 CI가 이들 체크를 모두 성공시켜야 합니다 [8].
- **컨텍스트 정책:** LLM 프롬프트 생성 시 `.gemini/context_policy.yaml`에 정의된 스키마(`sources`, `tokens`, `context_limits`)만 사용해야 하며, 이 파일만 버전 관리가 허용됩니다 [5] [6].
- **기타:** 작업 허브(`docs/HUB.md`) 절차, 파일 수정 프로토콜(파일 직접 수정은 Codex 담당) 등은 AGENTS.md/GEMINI.md에 명시된 정책을 따릅니다. Claude 기반 작업 시에도 이 모든 정책을 준수해야 합니다. 이는 곧 **비밀 커밋 금지, 테스트 통과, 품질 검사 수행** 등이 반드시 지켜져야 함을 의미합니다 [5] [8].

출처: 위 내용은 해당 레포지토리의 구성 파일 및 정책 문서에 근거하였으며 [3] [4] [5] [8], 실제 스크립트 구현을 참고해 종합하였습니다.

---

[1] [4] [5] [6] [7] [8] GEMINI.md
https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md

2 agent_manager.py

https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/scripts/agent_manager.py

3 9 AGENTS.md

https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md