Solo Git - Complete API Documentation

Comprehensive API Reference for Solo Git v0.4.0

This document provides complete documentation for Solo Git's Python API and CLI commands.

Table of Contents

CLI Command Reference

Global Options

All commands support these global options:

```
evogitctl [GLOBAL OPTIONS] COMMAND [COMMAND OPTIONS]
```

Option	Description	Default
`--version`	Show version and exit	-
`--help`	Show help message	-
`-v, --verbose`	Enable verbose logging	`False`
`--config PATH`	Custom config file path	`~/.sologit/config.yaml`

---

config - Configuration Management

`config setup`

Interactive configuration wizard.

Usage:

```
evogitctl config setup [OPTIONS]
```

Options:

Option	Description	Default
`--force`	Overwrite existing config	`False`

Example:

```
evogitctl config setup
evogitctl config setup --force  # Reconfigure
```

Prompts:
1. API endpoint URL
2. API key (hidden input)
3. Model selection (planning, coding, fast)
4. Budget controls
5. Workflow settings

Output:
- Creates `~/.sologit/config.yaml`
- Tests API connection
- Validates model access

`config show`

Display current configuration.

Usage:

```
evogitctl config show [OPTIONS]
```

Options:

Option	Description	Default
`--secrets`	Show API keys (masked by default)	`False`
`--format FORMAT`	Output format: yaml, json, table	`yaml`

Example:

```
evogitctl config show
evogitctl config show --secrets     # Show full API key
evogitctl config show --format json # JSON output
```

Output:

```
Configuration: /home/ubuntu/.sologit/config.yaml

Abacus.ai API:
  endpoint: https://api.abacus.ai/v1
  api_key: sk-************************************

Models:
  Planning: gpt-4o
  Coding: deepseek-coder-33b
  Fast: llama-3.1-8b-instruct

Budget:
  daily_usd_cap: 10.0
  alert_threshold: 0.8

Workflow:
  promote_on_green: true
  rollback_on_ci_red: true
```

`config test`

Test configuration and API connectivity.

Usage:

```
evogitctl config test [OPTIONS]
```

Options:

Option	Description	Default
`--quick`	Skip model access checks	`False`

Example:

```
evogitctl config test
evogitctl config test --quick  # Faster, skip model tests
```

Tests:
1. ✅ Configuration file exists and is valid
2. ✅ API endpoint is reachable
3. ✅ Authentication succeeds
4. ✅ Models are accessible
5. ✅ Data directory is writable

Exit Codes:
- `0`: All tests passed
- `2`: Configuration error
- `3`: API error

`config init`

Create default configuration file.

Usage:

```
evogitctl config init [OPTIONS]
```

Options:

Option	Description	Default
`--force`	Overwrite existing config	`False`
`--minimal`	Create minimal config	`False`

Example:

```
evogitctl config init
evogitctl config init --force --minimal
```

`config path`

Show configuration file path.

Usage:

```
evogitctl config path
```

Example:

```
evogitctl config path
# Output: /home/ubuntu/.sologit/config.yaml
```

`config env-template`

Generate environment variable template.

Usage:

```
evogitctl config env-template [OPTIONS]
```

Options:

Option	Description	Default
`--output PATH`	Save to file	`stdout`

Example:

```
evogitctl config env-template > .env
```

Output:

```
# Abacus.ai API credentials
export ABACUS_API_ENDPOINT=https://api.abacus.ai/v1
export ABACUS_API_KEY=your-api-key-here

# Solo Git settings
export SOLOGIT_CONFIG_PATH=~/.sologit/config.yaml
export SOLOGIT_DATA_DIR=~/.sologit/data
export SOLOGIT_LOG_LEVEL=INFO
```

---

repo - Repository Management

`repo init`

Initialize repository from ZIP or Git URL.

Usage:

```
evogitctl repo init [OPTIONS]
```

Options:

Option	Description	Required
`--zip PATH`	Path to ZIP file	One of –zip or –git
`--git URL`	Git repository URL	One of –zip or –git
`--name TEXT`	Repository name	Optional

Example:

```
# From ZIP
evogitctl repo init --zip myproject.zip --name "My Project"

# From Git
evogitctl repo init --git https://github.com/user/repo.git

# Name inferred from source if not provided
evogitctl repo init --zip app.zip
```

Output:

```
📦 Initializing repository from ZIP...

✅ Repository initialized
   Repo ID: repo_a1b2c3d4
   Name: My Project
   Path: /home/ubuntu/.sologit/data/repos/repo_a1b2c3d4
   Files: 42 files extracted
   Trunk: main
   Initial commit: abc123def456

📊 Repository structure:
   src/ (12 files)
   tests/ (8 files)
   docs/ (3 files)
   [config files] (5 files)
```

Exit Codes:
- `0`: Success
- `1`: Invalid arguments
- `4`: Git operation failed

`repo list`

List all repositories.

Usage:

```
evogitctl repo list [OPTIONS]
```

Options:

Option	Description	Default
`--verbose`	Show detailed info	`False`
`--format FORMAT`	Output format: table, json, simple	`table`

Example:

```
evogitctl repo list
evogitctl repo list --verbose
evogitctl repo list --format json
```

Output:

```
Repositories:

ID            Name              Trunk  Workpads  Created
━━━━━━━━━━━━  ━━━━━━━━━━━━━━━━  ━━━━━  ━━━━━━━━  ━━━━━━━━━━━━━━━━━━━━
repo_a1b2c3d4 My Project        main   2         2025-10-17 14:30:45
repo_e5f6g7h8 Example App       main   0         2025-10-17 15:12:30

Total: 2 repositories
```

`repo info`

Show detailed repository information.

Usage:

```
evogitctl repo info REPO_ID [OPTIONS]
```

Arguments:

Argument	Description	Required
`REPO_ID`	Repository ID	Yes

Options:

Option	Description	Default
`--show-files`	List all files	`False`
`--show-history`	Show commit history	`False`

Example:

```
evogitctl repo info repo_a1b2c3d4
evogitctl repo info repo_a1b2c3d4 --show-files
evogitctl repo info repo_a1b2c3d4 --show-history
```

Output:

```
Repository Information

ID:       repo_a1b2c3d4
Name:     My Project
Path:     /home/ubuntu/.sologit/data/repos/repo_a1b2c3d4
Trunk:    main
Created:  2025-10-17 14:30:45

Statistics:
  Files:     42
  Size:      2.3 MB
  Commits:   15
  Workpads:  2 active

Active Workpads:
  • pad_x9y8z7w6 - add-auth-feature (3 checkpoints)
  • pad_v5u4t3s2 - fix-bug-123 (1 checkpoint)

Latest Commits:
  abc123 - Checkpoint 3 (2 hours ago)
  def456 - Checkpoint 2 (5 hours ago)
  ghi789 - Checkpoint 1 (1 day ago)
```

---

pad - Workpad Management

`pad create`

Create a new ephemeral workpad.

Usage:

```
evogitctl pad create TITLE [OPTIONS]
```

Arguments:

Argument	Description	Required
`TITLE`	Workpad title (human-readable)	Yes

Options:

Option	Description	Default
`--repo REPO_ID`	Repository ID	Required if multiple repos

Example:

```
evogitctl pad create "add-login-feature"
evogitctl pad create "fix-bug-123" --repo repo_a1b2c3d4
```

Output:

```
🎨 Creating workpad...

✅ Workpad created
   Pad ID: pad_x9y8z7w6
   Title: add-login-feature
   Repo: repo_a1b2c3d4 (My Project)
   Branch: pads/add-login-feature-20251017-1430
   Base: main @ abc123

📝 Workpad details:
   Status: Active
   Checkpoints: 0
   Changes: None yet

🎯 Next steps:
   1. Run tests: evogitctl test run --pad pad_x9y8z7w6
   2. Promote: evogitctl pad promote pad_x9y8z7w6
```

`pad list`

List all active workpads.

Usage:

```
evogitctl pad list [OPTIONS]
```

Options:

Option	Description	Default
`--repo REPO_ID`	Filter by repository	All repos

`--all`	Show inactive workpads too	`False`
`--format FORMAT`	Output format: table, json	`table`

Example:

```
evogitctl pad list
evogitctl pad list --repo repo_a1b2c3d4
evogitctl pad list --all
```

Output:

```
Active Workpads:

Pad ID        Title              Repo              Checkpoints  Age
------------  -----------------  ----------------  -----------  ------
pad_x9y8z7w6  add-login-feature  My Project        3            2h ago
pad_v5u4t3s2  fix-bug-123        My Project        1            5h ago

Total: 2 active workpads
```

`pad info`

Show detailed workpad information.

Usage:

```
evogitctl pad info PAD_ID [OPTIONS]
```

Arguments:

Argument	Description	Required
`PAD_ID`	Workpad ID	Yes

Options:

Option	Description	Default
`--show-diff`	Show diff from trunk	`False`

Example:

```
evogitctl pad info pad_x9y8z7w6
evogitctl pad info pad_x9y8z7w6 --show-diff
```

Output:

```
Workpad Information

Pad ID:   pad_x9y8z7w6
Title:    add-login-feature
Repo:     repo_a1b2c3d4 (My Project)
Branch:   pads/add-login-feature-20251017-1430
Base:     main @ abc123
Created:  2025-10-17 14:30:45
Age:      2 hours

Status:   Active
Tests:    Last run: 30 minutes ago (GREEN ✅)

Checkpoints: 3
  1. t1: Initial scaffold (def456) - 2h ago
  2. t2: Add auth logic (ghi789) - 1h ago
  3. t3: Add tests (jkl012) - 30m ago

Changes from trunk:
  Files changed: 5
  Insertions: +127 lines
  Deletions: -12 lines

Files modified:
  • src/auth/login.py (+45, -3)
  • src/auth/session.py (+32, -0)
  • tests/test_auth.py (+50, -0)
  • requirements.txt (+5, -2)
  • README.md (+0, -7)
```

`pad promote`

Promote workpad to trunk (fast-forward merge).

Usage:

```
evogitctl pad promote PAD_ID [OPTIONS]
```

Arguments:

Argument	Description	Required
`PAD_ID`	Workpad ID	Yes

Options:

Option	Description	Default
`--force`	Skip pre-promotion checks	`False`
`--no-delete`	Keep workpad branch after merge	`False`

Example:

```
evogitctl pad promote pad_x9y8z7w6
evogitctl pad promote pad_x9y8z7w6 --force
```

Prerequisites:
- Tests must be green (unless –force)
- Must be fast-forward mergeable
- No conflicts with trunk

Output:

```
🚀 Promoting workpad to trunk...

✅ Pre-promotion checks passed
   Tests: GREEN ✅
   Fast-forward: Possible ✅
   Conflicts: None ✅

✅ Workpad promoted
   Pad: pad_x9y8z7w6
   From: pads/add-login-feature-20251017-1430
   To: main
   Commit: mno345pqr678
   Merge: Fast-forward ⚡

📊 Promotion details:
   Files changed: 5
   Insertions: +127 lines
   Deletions: -12 lines
   Duration: 1.2 seconds

🧹 Cleanup:
   Branch deleted: pads/add-login-feature-20251017-1430
   Workpad removed from active list

🎉 Success! Your changes are now in trunk.
```

`pad diff`

Show diff between workpad and trunk.

Usage:

```
evogitctl pad diff PAD_ID [OPTIONS]
```

Arguments:

Argument	Description	Required
`PAD_ID`	Workpad ID	Yes

Options:

Option	Description	Default
`--unified N`	Context lines	`3`
`--stat`	Show only statistics	`False`
`--output PATH`	Save to file	`stdout`

Example:

```
evogitctl pad diff pad_x9y8z7w6
evogitctl pad diff pad_x9y8z7w6 --stat
evogitctl pad diff pad_x9y8z7w6 --output changes.patch
```

`pad delete`

Delete a workpad permanently.

Usage:

```
evogitctl pad delete PAD_ID [OPTIONS]
```

Arguments:

Argument	Description	Required
`PAD_ID`	Workpad ID	Yes

Options:

Option	Description	Default
`--force`	Skip confirmation	`False`

Example:

```
evogitctl pad delete pad_x9y8z7w6
evogitctl pad delete pad_x9y8z7w6 --force
```

Warning: This permanently deletes the workpad branch and all checkpoints.

---

test - Test Execution

`test run`

Run tests in sandboxed environment.

Usage:

```
evogitctl test run [OPTIONS]
```

Options:

Option	Description	Required
`--pad PAD_ID`	Workpad ID	Yes
`--target TARGET`	Test target: fast, full, smoke	No (default: fast)
`--parallel / --sequential`	Execution mode	`--parallel`
`--fail-fast`	Stop on first failure	`False`

Example:

```
evogitctl test run --pad pad_x9y8z7w6
evogitctl test run --pad pad_x9y8z7w6 --target full
evogitctl test run --pad pad_x9y8z7w6 --sequential --fail-fast
```

Output:

```
🧪 Running tests: fast
   Pad: pad_x9y8z7w6
   Repo: repo_a1b2c3d4
   Mode: Parallel
   Tests: 3

Running tests...

✅ unit-tests (2.3s)
   Passed: 12 tests

✅ integration-tests (5.1s)
   Passed: 8 tests

✅ linting (0.8s)
   No issues found

📊 Test Summary:
   Total: 3 test suites
   Passed: 3 ✅
   Failed: 0
   Duration: 5.2 seconds (parallel)
   Status: GREEN 🟢

🎯 All tests passed! Ready to promote.
   Run: evogitctl pad promote pad_x9y8z7w6
```

`test config`

Show test configuration.

Usage:

```
evogitctl test config [OPTIONS]
```

Options:

Option	Description	Default
`--repo REPO_ID`	Repository ID	Current repo
`--format FORMAT`	Output format: yaml, json	`yaml`

Example:

```
evogitctl test config
evogitctl test config --repo repo_a1b2c3d4
evogitctl test config --format json
```

`test analyze`

Analyze test failures with AI.

Usage:

```
evogitctl test analyze [OPTIONS]
```

Options:
Option	Description	Required
`--pad PAD_ID`	Workpad ID	Yes

Example:

```
evogitctl test analyze --pad pad_x9y8z7w6
```

Output:

```
🔍 Analyzing test failures...

Test Analysis Report

Status: RED 🔴
Total: 3 test suites
Passed: 2 ✅
Failed: 1 ❌

Failure Details:

Test: integration-tests
Category: ASSERTION_ERROR
Pattern: Expected 200, got 404

Root Cause:
  • Missing route configuration in app.py
  • Endpoint /api/login not registered

Affected Code:
  File: src/app.py
  Line: 45
  Function: create_app()

AI Suggestions:
  1. Add route registration: app.register_blueprint(auth_bp, url_prefix='/api')
  2. Verify blueprint is imported: from src.auth import auth_bp
  3. Check blueprint definition in src/auth/__init__.py

Recommended Actions:
  • Fix code based on suggestions
  • Re-run tests: evogitctl test run --pad pad_x9y8z7w6
```

Workflows (Phase 3)

`auto-merge run`

Execute complete auto-merge workflow.

Usage:

```
evogitctl auto-merge run [OPTIONS]
```

Options:

Option	Description	Required
`--pad PAD_ID`	Workpad ID	Yes
`--target TARGET`	Test target	No (default: fast)
`--no-auto-promote`	Skip auto-promotion	`False`

Example:

```
evogitctl auto-merge run --pad pad_x9y8z7w6
evogitctl auto-merge run --pad pad_x9y8z7w6 --target full
evogitctl auto-merge run --pad pad_x9y8z7w6 --no-auto-promote
```

Workflow:
1. Run tests
2. Analyze results
3. Evaluate promotion gate
4. Promote if approved (unless –no-auto-promote)
5. Trigger CI smoke tests (if configured)

Output:

```
🤖 Starting Auto-Merge Workflow

Step 1: Running tests (target: fast)
✅ All tests passed (5.2s)

Step 2: Analyzing test results
✅ No failures detected

Step 3: Evaluating promotion gate
✅ Promotion approved
   Tests: GREEN ✅
   Fast-forward: Possible ✅
   No conflicts ✅

Step 4: Promoting to trunk
✅ Promoted: main @ mno345pqr678

Step 5: Triggering CI smoke tests
⏳ CI job started: https://jenkins.example.com/job/123

🎉 Auto-merge complete!
   Duration: 8.5 seconds
   Status: SUCCESS
```

`auto-merge status`

Check auto-merge workflow status.

Usage:

```
evogitctl auto-merge status [OPTIONS]
```

Options:

Option	Description	Default
`--pad PAD_ID`	Workpad ID	Most recent

Example:

```
evogitctl auto-merge status
evogitctl auto-merge status --pad pad_x9y8z7w6
```

`promote`

Evaluate and promote with promotion gate.

Usage:

```
evogitctl promote PAD_ID [OPTIONS]
```

Arguments:

Argument	Description	Required
`PAD_ID`	Workpad ID	Yes

Example:

```
evogitctl promote pad_x9y8z7w6
```

Output:

```
📋 Evaluating Promotion Gate

Checks:
  ✅ Tests required: Passed
  ✅ Fast-forward possible: Yes
  ✅ Test status: GREEN
  ✅ Min coverage (80%): 87% ✅
  ✅ Max complexity (15): 12 ✅
  ✅ Max files changed (50): 5 ✅

Decision: APPROVE ✅

Promoting workpad...
✅ Promoted to trunk
```

`rollback`

Rollback last commit or specific commit.

Usage:

```
evogitctl rollback [OPTIONS]
```

Options:

Option	Description	Default
`--repo REPO_ID`	Repository ID	Current repo
`--commit HASH`	Specific commit to rollback	Last commit
`--recreate-pad`	Recreate workpad with changes	`False`

Example:

```
evogitctl rollback --last
evogitctl rollback --repo repo_a1b2c3d4 --commit abc123
evogitctl rollback --last --recreate-pad
```

Output:

```
⚠️  Rolling back commit...

Commit to rollback:
  Hash: mno345pqr678
  Message: Checkpoint 3
  Author: Solo Git
  Date: 2025-10-17 14:30:45

✅ Rollback complete
   New HEAD: ghi789jkl012
   Branch: main

✨ Workpad recreated
   Pad ID: pad_abc123def456
   Title: rollback-mno345-recreation
   Changes: Preserved from rolled-back commit
```

---

ci - CI/CD Integration

`ci smoke`

Run CI smoke tests.

Usage:

```
evogitctl ci smoke [OPTIONS]
```

Options:

Option	Description	Required
`--repo REPO_ID`	Repository ID	Yes
`--commit HASH`	Specific commit	No (default: HEAD)

Example:

```
evogitctl ci smoke --repo repo_a1b2c3d4
evogitctl ci smoke --repo repo_a1b2c3d4 --commit abc123
```

Output:

```
🔥 Running CI Smoke Tests

Commit: abc123def456
Branch: main
Repo: repo_a1b2c3d4

Smoke Tests:
  ✅ health-check (2.1s)
  ✅ basic-flow (3.5s)

📊 Summary:
   Total: 2 tests
   Passed: 2 ✅
   Failed: 0
   Duration: 5.6 seconds
   Status: GREEN 🟢

🎉 Smoke tests passed!
```

`ci status`

Check CI job status.

Usage:

```
evogitctl ci status [OPTIONS]
```

Options:
Option	Description	Default
`--repo REPO_ID`	Repository ID	Current repo

Example:

```
evogitctl ci status --repo repo_a1b2c3d4
```

Utilities

`version`

Show Solo Git version and API status.

Usage:

```
evogitctl version
```

Output:

```
Solo Git v0.4.0

Phases:
  Phase 0: Foundation & Setup       ✅ Complete
  Phase 1: Core Git Engine          ✅ Complete
  Phase 2: AI Integration           ✅ Complete
  Phase 3: Testing & Auto-Merge     ✅ Complete
  Phase 4: Polish & Beta            🚧 In Progress

API Status:
  Endpoint: https://api.abacus.ai/v1
  Status: Connected ✅

Python: 3.11.6
Git: 2.39.2
Platform: Linux-5.15.0-x86_64
```

`hello`

Verify installation and show welcome message.

Usage:

```
evogitctl hello
```

Python API Reference

Core Modules

sologit.engines.git_engine

GitEngine - Core Git operations

```python
from sologit.engines.git_engine import GitEngine, GitEngineError

# Initialize engine
engine = GitEngine(repos_path="/data/repos")

# Initialize repository from ZIP
repo_id = await engine.init_from_zip(
    zip_data=zip_bytes,
    name="My Project"
)

# Initialize from Git URL
repo_id = await engine.init_from_git(
    git_url="https://github.com/user/repo.git",
    name="Cloned Project"
)

# Create workpad
pad_id = await engine.create_workpad(
    repo_id=repo_id,
    title="add-feature"
)

# Apply patch
checkpoint = await engine.apply_patch(
    pad_id=pad_id,
    patch=unified_diff_string
)

# Promote workpad
commit_hash = await engine.promote_workpad(
    pad_id=pad_id
)

# Rollback
await engine.revert_last_commit(repo_id=repo_id)

# Get diff
diff = await engine.get_diff(
    pad_id=pad_id,
    base="trunk"
)

# Get repository map
file_tree = await engine.get_repo_map(repo_id=repo_id)
```

Classes:
- `GitEngine` - Main Git operations engine
- `Repository` - Repository data model
- `Workpad` - Workpad data model
- `GitEngineError` - Base exception class

Methods:

`GitEngine.__init__(repos_path: str = "/data/repos")`

Initialize Git engine.

Parameters:
- `repos_path` (str): Path to store repositories

`GitEngine.init_from_zip(zip_data: bytes, name: str) -> str`

Initialize repository from ZIP file.

Parameters:
- `zip_data` (bytes): ZIP file contents
- `name` (str): Repository name

Returns:
- `str`: Repository ID (e.g., "repo_a1b2c3d4")

Raises:
- `GitEngineError`: If initialization fails

Example:

```python
with open("project.zip", "rb") as f:
    zip_data = f.read()

repo_id = engine.init_from_zip(zip_data, "My Project")
print(f"Created repository: {repo_id}")
```

`GitEngine.init_from_git(git_url: str, name: str) -> str`

Initialize repository from Git URL.

Parameters:
- `git_url` (str): Git repository URL
- `name` (str): Repository name

Returns:
- `str`: Repository ID

Raises:
- `GitEngineError`: If clone fails

`GitEngine.create_workpad(repo_id: str, title: str) -> str`

Create ephemeral workpad.

Parameters:
- `repo_id` (str): Repository ID
- `title` (str): Human-readable title

Returns:
- `str`: Workpad ID (e.g., "pad_x9y8z7w6")

Raises:
- `GitEngineError`: If repository not found

Example:

```python
pad_id = engine.create_workpad(
    repo_id="repo_a1b2c3d4",
    title="add-authentication"
)
print(f"Created workpad: {pad_id}")
```

`GitEngine.apply_patch(pad_id: str, patch: str) -> str`

Apply unified diff patch to workpad.

Parameters:
- `pad_id` (str): Workpad ID
- `patch` (str): Unified diff format patch

Returns:
- `str`: Checkpoint commit hash

Raises:
- `GitEngineError`: If patch application fails
- `PatchConflictError`: If patch has conflicts

Example:

```python
patch = """
diff --git a/src/auth.py b/src/auth.py
index abc123..def456 100644
--- a/src/auth.py
+++ b/src/auth.py
@@ -10,0 +11,5 @@ def login(username, password):
+def logout(session_id):
+    # Clear user session
+    sessions.pop(session_id, None)
+    return True
"""

checkpoint = engine.apply_patch(pad_id, patch)
print(f"Created checkpoint: {checkpoint}")
```

`GitEngine.promote_workpad(pad_id: str) -> str`

Promote workpad to trunk (fast-forward merge).

Parameters:
- `pad_id` (str): Workpad ID

Returns:
- `str`: New trunk commit hash

Raises:
- `GitEngineError`: If promotion fails
- `FastForwardError`: If fast-forward not possible

Example:

```python
try:
    commit_hash = engine.promote_workpad("pad_x9y8z7w6")
    print(f"Promoted to trunk: {commit_hash}")
except FastForwardError:
    print("Cannot fast-forward, trunk has diverged")
```

`GitEngine.revert_last_commit(repo_id: str) -> None`

Rollback last commit on trunk.

Parameters:
- `repo_id` (str): Repository ID

Raises:
- `GitEngineError`: If rollback fails

`GitEngine.get_diff(pad_id: str, base: str = "trunk") -> str`

Get diff between workpad and base.

Parameters:
- `pad_id` (str): Workpad ID
- `base` (str): Base reference (default: “trunk”)

Returns:
- `str`: Unified diff

`GitEngine.get_repo_map(repo_id: str) -> Dict`

Get repository file tree.

Parameters:
- `repo_id` (str): Repository ID

Returns:
- `Dict`: Hierarchical file tree

Example:

```python
tree = engine.get_repo_map("repo_a1b2c3d4")
print(json.dumps(tree, indent=2))
```

---

sologit.engines.patch_engine

PatchEngine - Patch generation and application

```python
from sologit.engines.patch_engine import PatchEngine

# Initialize
patch_engine = PatchEngine(git_engine)

# Generate patch
patch = patch_engine.generate_patch(
    file_path="src/app.py",
    old_content="...",
    new_content="..."
)

# Apply with conflict detection
result = patch_engine.apply_with_conflict_detection(
    pad_id="pad_x9y8z7w6",
    patch=patch
)

if result.conflicts:
    print("Conflicts detected!")
    for conflict in result.conflicts:
        print(f"  {conflict.file}: {conflict.reason}")
```

---

sologit.engines.test_orchestrator

TestOrchestrator - Test execution and orchestration

```python
from sologit.engines.test_orchestrator import TestOrchestrator, TestConfig

# Initialize
test_orch = TestOrchestrator(git_engine)

# Define tests
tests = [
    TestConfig(
        name="unit-tests",
        cmd="pytest tests/unit --quiet",
        timeout=30
    ),
    TestConfig(
        name="integration-tests",
        cmd="pytest tests/integration",
        timeout=60,
        depends_on=["unit-tests"]
    )
]

# Run tests
results = await test_orch.run_tests(
    repo_path="/data/repos/repo_a1b2c3d4",
    tests=tests,
    parallel=True
)

# Check results
all_passed = all(r.status == "passed" for r in results)
print(f"All tests passed: {all_passed}")

for result in results:
    status_icon = "✅" if result.status == "passed" else "❌"
    print(f"{status_icon} {result.name}: {result.duration_ms}ms")
```

AI Orchestration

sologit.orchestration.ai_orchestrator

AIOrchestrator - Main AI coordination

```python
from sologit.orchestration.ai_orchestrator import AIOrchestrator

# Initialize
ai_orch = AIOrchestrator(config)

# Execute pair loop
result = await ai_orch.execute_pair_loop(
    prompt="Add Redis caching to search endpoint",
    repo_id="repo_a1b2c3d4",
    pad_id="pad_x9y8z7w6"
)

print(f"Plan: {result.plan}")
print(f"Patches: {len(result.patches)}")
print(f"Tests: {result.test_results.status}")
```

sologit.orchestration.model_router

ModelRouter - Intelligent model selection

```python
from sologit.orchestration.model_router import ModelRouter

# Initialize
router = ModelRouter(config)

# Select model
model = router.select_model(
    prompt="Implement OAuth2 authentication",
    context=repo_context
)

print(f"Selected: {model.tier} - {model.name}")
# Output: Selected: planning - gpt-4o
```

---

sologit.orchestration.cost_guard

CostGuard - Budget tracking and enforcement

```python
from sologit.orchestration.cost_guard import CostGuard

# Initialize
cost_guard = CostGuard(config)

# Check budget before operation
can_proceed, remaining = cost_guard.check_budget()
if not can_proceed:
    print(f"Daily budget exceeded!")
else:
    print(f"Remaining: ${remaining:.2f}")

# Track operation cost
cost_guard.track_operation(
    model="gpt-4o",
    tokens_used=1500,
    operation_type="planning"
)

# Get usage report
report = cost_guard.get_daily_report()
print(f"Today's spend: ${report.total_usd:.2f}")
print(f"Operations: {report.operation_count}")
```

Analysis & Workflows

sologit.analysis.test_analyzer

TestAnalyzer - Intelligent test failure analysis

```python
from sologit.analysis.test_analyzer import TestAnalyzer

# Initialize
analyzer = TestAnalyzer(ai_orchestrator)

# Analyze failures
analysis = await analyzer.analyze_failures(
    test_results=test_results,
    repo_context=repo_context
)

print(f"Category: {analysis.category}")
print(f"Root cause: {analysis.root_cause}")
print(f"Suggestions: {len(analysis.suggestions)}")

for suggestion in analysis.suggestions:
    print(f"  • {suggestion}")
```

---

sologit.workflows.promotion_gate

PromotionGate - Configurable merge gate

```python
from sologit.workflows.promotion_gate import PromotionGate, PromotionRules

# Configure rules
rules = PromotionRules(
    require_tests=True,
    require_fast_forward=True,
    min_coverage=0.80,
    max_complexity=15
)

# Initialize gate
gate = PromotionGate(git_engine, test_orchestrator, rules)

# Evaluate
decision = await gate.evaluate(
    pad_id="pad_x9y8z7w6",
    test_results=test_results
)

if decision.approved:
    print("✅ Promotion approved")
    # Promote...
else:
    print(f"❌ Promotion rejected: {decision.reason}")
```

---

sologit.workflows.auto_merge

AutoMergeWorkflow - Complete auto-merge automation

```python
from sologit.workflows.auto_merge import AutoMergeWorkflow

# Initialize
workflow = AutoMergeWorkflow(
    git_engine,
    test_orchestrator,
    promotion_gate,
    ai_orchestrator
)

# Execute workflow
result = await workflow.execute(
    pad_id="pad_x9y8z7w6",
    test_target="fast",
    auto_promote=True
)

print(f"Workflow status: {result.status}")
print(f"Duration: {result.duration_seconds}s")

if result.promoted:
    print(f"✅ Promoted to trunk: {result.commit_hash}")
else:
    print(f"❌ Not promoted: {result.reason}")
```

Configuration API

Config Manager

```python
from sologit.config.manager import ConfigManager

# Load config
config = ConfigManager.load()

# Access settings
api_key = config.abacus.api_key
planning_model = config.models.planning_model
daily_cap = config.budget.daily_usd_cap

# Validate config
is_valid, errors = config.validate()
if not is_valid:
    for error in errors:
        print(f"Error: {error}")

# Save config
config.budget.daily_usd_cap = 15.0
config.save()

# Test API connection
success = await config.test_connection()
if success:
    print("✅ API connection successful")
```

Data Models

Repository

```
@dataclass
class Repository:
    id: str                      # e.g., "repo_a1b2c3d4"
    name: str                    # Human-readable name
    path: str                    # Filesystem path
    trunk_branch: str            # Usually "main"
    created_at: datetime

    # Methods
    def get_workpads(self) -> List[Workpad]: ...
    def get_history(self, limit: int = 10) -> List[Commit]: ...
```

Workpad

```
@dataclass
class Workpad:
    id: str                      # e.g., "pad_x9y8z7w6"
    repo_id: str
    title: str                   # Human-readable
    branch_name: str             # Git branch
    base_commit: str             # Trunk commit it branched from
    created_at: datetime
    checkpoints: List[str]       # Checkpoint commit hashes

    # Properties
    @property
    def age(self) -> timedelta: ...

    @property
    def checkpoint_count(self) -> int: ...

    # Methods
    def get_diff(self, base: str = "trunk") -> str: ...
    def get_latest_checkpoint(self) -> str: ...
```

TestResult

```
@dataclass
class TestResult:
    name: str
    status: Literal["passed", "failed", "timeout", "error"]
    duration_ms: int
    exit_code: int
    stdout: str
    stderr: str
    error: Optional[str] = None
```

TestAnalysis

```
@dataclass
class TestAnalysis:
    status: Literal["green", "yellow", "red"]
    passed: int
    failed: int
    timeout: int
    error: int
    category: Optional[str]           # Failure category
    root_cause: Optional[str]          # Root cause description
    suggestions: List[str]             # AI suggestions
    actionable: bool                   # Can be auto-fixed?
```

PromotionDecision

```
@dataclass
class PromotionDecision:
    decision: Literal["APPROVE", "REJECT", "MANUAL_REVIEW"]
    approved: bool
    reason: str
    checks: Dict[str, bool]            # Individual check results
    timestamp: datetime
```

Error Handling

Exception Hierarchy

```
SoloGitError                      # Base exception
├── ConfigurationError            # Config issues
│   ├── InvalidConfigError
│   └── MissingCredentialsError
├── GitEngineError                # Git operations
│   ├── RepositoryNotFoundError
│   ├── WorkpadNotFoundError
│   ├── PatchConflictError
│   └── FastForwardError
├── TestExecutionError            # Test failures
│   ├── TestTimeoutError
│   └── TestSetupError
├── AIOrchestrationError          # AI issues
│   ├── ModelNotAvailableError
│   ├── APIConnectionError
│   └── BudgetExceededError
└── WorkflowError                 # Workflow issues
    ├── PromotionRejectedError
    └── RollbackFailedError
```

Error Handling Examples

```python
from sologit.engines.git_engine import GitEngine, GitEngineError, FastForwardError

try:
    engine.promote_workpad("pad_x9y8z7w6")
except FastForwardError as e:
    print(f"Cannot fast-forward: {e}")
    # Handle rebase or manual merge
except GitEngineError as e:
    print(f"Git error: {e}")
    # Log and notify
```

```python
from sologit.orchestration.ai_orchestrator import AIOrchestrator, BudgetExceededError

try:
    result = await ai_orch.plan_changes(prompt)
except BudgetExceededError as e:
    print(f"Budget exceeded: ${e.current_spend:.2f} / ${e.daily_cap:.2f}")
    # Fallback to manual or wait
```

Examples

Example 1: Complete Workflow

```python
from sologit.engines.git_engine import GitEngine
from sologit.engines.test_orchestrator import TestOrchestrator, TestConfig
from sologit.workflows.auto_merge import AutoMergeWorkflow

# Initialize
git_engine = GitEngine()
test_orch = TestOrchestrator(git_engine)

# Initialize repo
repo_id = git_engine.init_from_zip(zip_data, "My Project")

# Create workpad
pad_id = git_engine.create_workpad(repo_id, "add-feature")

# Apply changes (manually or via AI)
patch = generate_patch()  # Your patch generation
git_engine.apply_patch(pad_id, patch)

# Run tests
tests = [
    TestConfig(name="unit-tests", cmd="pytest tests/", timeout=30)
]
results = await test_orch.run_tests(repo_path, tests)

# Auto-merge if green
if all(r.status == "passed" for r in results):
    commit = git_engine.promote_workpad(pad_id)
    print(f"✅ Promoted: {commit}")
else:
    print("❌ Tests failed, not promoting")
```

Example 2: AI Pair Programming

```python
from sologit.orchestration.ai_orchestrator import AIOrchestrator

# Initialize
ai_orch = AIOrchestrator(config)

# Execute pair loop
result = await ai_orch.execute_pair_loop(
    prompt="Add Redis caching to search endpoint with 5-minute TTL",
    repo_id="repo_a1b2c3d4",
    pad_id="pad_x9y8z7w6"
)

# Review results
print(f"📋 Plan:\n{result.plan}")
print(f"\n📝 Generated {len(result.patches)} patches")
print(f"\n🧪 Tests: {result.test_results.status}")

if result.test_results.status == "green":
    print("\n✅ Ready to promote!")
else:
    print(f"\n❌ Tests failed: {result.test_results.analysis.root_cause}")
```

Example 3: Custom Promotion Gate

```python
from sologit.workflows.promotion_gate import PromotionGate, PromotionRules

# Custom rules
rules = PromotionRules(
    require_tests=True,
    require_fast_forward=True,
    min_coverage=0.85,           # 85% coverage required
    max_complexity=12,            # Max cyclomatic complexity
    max_files_changed=30,         # Limit scope
    no_todos=True,                # No TODO comments
)

gate = PromotionGate(git_engine, test_orch, rules)

# Evaluate
decision = await gate.evaluate(pad_id, test_results)

print(f"Decision: {decision.decision}")
print(f"Approved: {decision.approved}")

if not decision.approved:
    print(f"Reason: {decision.reason}")
    print("\nFailed checks:")
    for check, passed in decision.checks.items():
        if not passed:
            print(f"  ❌ {check}")
```

Environment Variables Reference

Variable	Description	Default
`ABACUS_API_ENDPOINT`	Abacus.ai API endpoint	-
`ABACUS_API_KEY`	API key	-
`SOLOGIT_CONFIG_PATH`	Config file path	`~/.sologit/config.yaml`
`SOLOGIT_DATA_DIR`	Data directory	`~/.sologit/data`
`SOLOGIT_LOG_LEVEL`	Logging level	`INFO`
`SOLOGIT_LOG_FILE`	Log file path	`~/.sologit/logs/sologit.log`
`SOLOGIT_DAILY_CAP`	Daily budget cap (USD)	From config
`SOLOGIT_PARALLEL_TESTS`	Enable parallel tests	`true`

Exit Codes

Code	Meaning	Example
0	Success	Command completed successfully
1	General error	Unexpected error occurred
2	Configuration error	Missing or invalid config
3	API error	API connection or auth failed
4	Git operation failed	Repository, workpad, or merge error
5	Tests failed	Test execution returned failures
6	Promotion rejected	Gate rejected promotion
7	Budget exceeded	Daily budget cap reached

API Versioning

Solo Git follows semantic versioning (SemVer):

- **v0.x.x**: Beta versions (current)
- **v1.0.0**: First stable release (future)
- **v1.x.x**: Stable with backwards compatibility
- **v2.0.0**: Breaking changes

Current Version: v0.4.0 (Phase 4 - Beta Preparation)

Rate Limits

Abacus.ai API:
- Rate limits depend on your API plan
- Monitor usage via Cost Guard
- Daily budget caps prevent overspending

Solo Git:
- No built-in rate limits
- Configurable via `budget.daily_usd_cap`
- Escalation policies can reduce costs

Support & Contributing

- **Documentation**: docs/wiki/Home.md (wiki/Home.md)
- **Issues**: GitHub Issues (https://github.com/yourusername/solo-git/issues)
- **API Questions**: support@sologit.dev
- **Contributing**: See CONTRIBUTING.md

Last Updated: October 17, 2025
Solo Git v0.4.0 - Phase 4 (Beta Preparation)