Coverity Connect MCP Server - 関数仕様書

Version: 1.0.0
作成日: 2025年7月21日
更新日: 2025年7月21日

📋 概要

本仕様書では、`main.py`と`coverity_client.py`の全関数について詳細な仕様を記載します。

🔧 main.py 関数仕様

1. initialize_client()

概要

Coverity Clientインスタンスを環境変数から初期化する関数

仕様

```
python

def initialize_client() -> CoverityClient
```

詳細仕様

項目	内容
戻り値	`CoverityClient` - 初期化されたクライアントインスタンス
例外	`ValueError` - 必須環境変数が不足している場合
副作用	グローバル変数 `coverity_client` を設定

環境変数要件

```
python
```

```
required_env_vars = [
    'COVERITY_HOST',     # 必須: Coverityサーバーホスト
    'COVAUTHUSER',       # 必須: 認証ユーザー名
    'COVAUTHKEY'         # 必須: 認証キー
]

optional_env_vars = [
    'COVERITY_PORT',     # オプション: ポート番号 (デフォルト: 8080)
    'COVERITY_SSL'       # オプション: SSL使用 (デフォルト: True)
]
```

実装詳細

python

```
def initialize_client() -> CoverityClient:
    """Initialize Coverity client with environment variables"""
    global coverity_client

    # シングルトンパターン実装
    if coverity_client is not None:
        return coverity_client

    # 環境変数取得
    host = os.getenv('COVERITY_HOST')
    port = int(os.getenv('COVERITY_PORT', '8080'))
    use_ssl = os.getenv('COVERITY_SSL', 'True').lower() == 'true'
    username = os.getenv('COVAUTHUSER')
    password = os.getenv('COVAUTHKEY')

    # バリデーション
    if not host or not username or not password:
        raise ValueError("Missing required environment variables")

    # クライアント作成
    coverity_client = CoverityClient(
        host=host, port=port, use_ssl=use_ssl,
        username=username, password=password
    )

    logger.info(f"Initialized Coverity client for {host}:{port}")
    return coverity_client
```

2. create_server()

概要

FastMCPサーバーを作成し、全てのツールとリソースを登録する関数

仕様

```
python

def create_server() -> FastMCP
```

詳細仕様

項目	内容
戻り値	`FastMCP` - 設定済みMCPサーバーインスタンス
例外	クライアント初期化失敗時に例外を再発生
副作用	- MCPツール登録（8個） - MCPリソース登録（2個） - ログ出力

登録されるMCPツール

```
python

mcp_tools = [
    'search_defects',          # 欠陥検索
    'get_defect_details',      # 欠陥詳細取得
    'list_projects',           # プロジェクト一覧
    'list_streams',            # ストリーム一覧
    'get_project_summary',     # プロジェクトサマリー
    'list_users',              # ユーザー一覧
    'get_user_details',        # ユーザー詳細
    'get_user_roles'           # ユーザー権限情報
]
```

登録されるMCPリソース

```
python

mcp_resources = [
    'coverity://projects/{project_id}/config',    # プロジェクト設定
    'coverity://streams/{stream_id}/defects'      # ストリーム欠陥
]
```

3. MCPツール関数群

3.1 search_defects()

```
python
```

```python
@mcp.tool()
async def search_defects(
    query: str = "",
    stream_id: str = "",
    checker: str = "",
    severity: str = "",
    status: str = "",
    limit: int = 50
) -> List[Dict[str, Any]]
```

パラメータ	型	必須	デフォルト	説明
`query`	str	No	`""`	汎用検索クエリ
`stream_id`	str	No	`""`	ストリームIDフィルタ
`checker`	str	No	`""`	チェッカー名フィルタ
`severity`	str	No	`""`	重要度フィルタ
`status`	str	No	`""`	ステータスフィルタ
`limit`	int	No	`50`	最大結果数

戻り値: `List[Dict[str, Any]]` - 欠陥情報のリスト **例外処理**: 全ての例外をキャッチしてエラー辞書を返却

3.2 get_defect_details()

```python
@mcp.tool()
async def get_defect_details(cid: str) -> Dict[str, Any]
```

パラメータ	型	必須	説明
`cid`	str	Yes	Coverity Issue Identifier

戻り値: `Dict[str, Any]` - 詳細欠陥情報 **例外処理**: 欠陥が見つからない場合はエラー辞書を返却

3.3 list_projects()

```python
@mcp.tool()
async def list_projects() -> List[Dict[str, Any]]
```

パラメータ: なし **戻り値**: `List[Dict[str, Any]]` - プロジェクト情報のリスト **例外処理**: エラー時は空リストまたはエラー辞書を返却

3.4 list_streams()

```python
@mcp.tool()
async def list_streams(project_id: str = "") -> List[Dict[str, Any]]
```

パラメータ	型	必須	デフォルト	説明
`project_id`	str	No	`""`	プロジェクトIDフィルタ

戻り値: `List[Dict[str, Any]]` - ストリーム情報のリスト

3.5 get_project_summary()

```python
@mcp.tool()
async def get_project_summary(project_id: str) -> Dict[str, Any]
```

パラメータ	型	必須	説明
`project_id`	str	Yes	プロジェクト識別子

戻り値: `Dict[str, Any]` - プロジェクトサマリー情報 **処理フロー**:

1. プロジェクト詳細取得
2. プロジェクトのストリーム一覧取得
3. 各ストリームの欠陥情報取得（最大1000件）
4. 重要度・ステータス別集計
5. サマリー情報構築

3.6 list_users() ✨ 新機能

```python
@mcp.tool()
async def list_users(
    include_disabled: bool = False,
    limit: int = 200
) -> List[Dict[str, Any]]
```

パラメータ	型	必須	デフォルト	説明
`include_disabled`	bool	No	`False`	無効化ユーザーを含める
`limit`	int	No	`200`	最大取得ユーザー数

戻り値: `List[Dict[str, Any]]` - ユーザー情報のリスト

3.7 get_user_details() ✨ 新機能

```python
@mcp.tool()
async def get_user_details(username: str) -> Dict[str, Any]
```

パラメータ	型	必須	説明
`username`	str	Yes	ユーザー名

戻り値: `Dict[str, Any]` - ユーザー詳細情報

3.8 get_user_roles() ✨ 新機能

```python
@mcp.tool()
async def get_user_roles(username: str) -> Dict[str, Any]
```

パラメータ	型	必須	説明
`username`	str	Yes	権限を調べるユーザー名

戻り値: `Dict[str, Any]` - ロール・権限詳細情報 **特殊処理**:

- 日本語ロール説明の追加
- ロール情報の構造化
- ユーザーステータス情報の整理

```python
role_descriptions = {
    "administrator": "システム全体の管理権限",
    "projectOwner": "プロジェクトの所有者権限",
    "developer": "開発者権限",
    "analyst": "分析者権限",
    "viewer": "閲覧権限"
}
```

4. MCPリソース関数群

4.1 get_project_config()

```python
```

```
@mcp.resource("coverity://projects/{project_id}/config")
async def get_project_config(project_id: str) -> str
```

パラメータ	型	必須	説明
`project_id`	str	Yes	プロジェクト識別子

戻り値: `str` - フォーマット済みプロジェクト設定情報 **処理内容**:

1. プロジェクト詳細取得
2. 設定情報の抽出・整理
3. 人間可読形式でのフォーマット

4.2 get_stream_defects()

```
python

@mcp.resource("coverity://streams/{stream_id}/defects")
async def get_stream_defects(stream_id: str) -> str
```

パラメータ	型	必須	説明
`stream_id`	str	Yes	ストリーム識別子

戻り値: `str` - フォーマット済みストリーム欠陥情報 **制限**: 表示は最初の10件まで（可読性のため）

5. ユーティリティ関数

5.1 run_server()

```
python

def run_server() -> None
```

概要: MCPサーバーを起動する関数 **処理フロー**:

1. `create_server()`でサーバー作成
2. `mcp.run()`でサーバー起動
3. 例外時は`sys.exit(1)`で終了

5.2 cli()

```
python
```

```
@click.command()
@click.option('--host', default='localhost', help='Coverity Connect host')
@click.option('--port', default=8080, help='Coverity Connect port')
@click.option('--ssl/--no-ssl', default=True, help='Use SSL connection')
@click.option('--username', help='Coverity username')
@click.option('--password', help='Coverity password')
def cli(host, port, ssl, username, password) -> None
```

概要: CLIインターフェース関数 **処理内容**:

- CLI引数を環境変数に設定
- `run_server()`呼び出し

🔌 coverity_client.py 関数仕様

1. CoverityClient.init()

概要

CoverityClientインスタンスを初期化するコンストラクタ

仕様

```
python

def __init__(
    self,
    host: str,
    port: int = 8080,
    use_ssl: bool = True,
    username: str = "",
    password: str = ""
) -> None
```

詳細仕様

パラメータ	型	必須	デフォルト	説明
`host`	str	Yes	-	Coverity Connectサーバーホスト
`port`	int	No	`8080`	サーバーポート番号
`use_ssl`	bool	No	`True`	SSL/HTTPS使用フラグ
`username`	str	No	`""`	認証ユーザー名
`password`	str	No	`""`	認証パスワード/キー

初期化処理

```python
# インスタンス変数設定
self.host = host
self.port = port
self.use_ssl = use_ssl
self.username = username
self.password = password

# base_URL構築
protocol = "https" if use_ssl else "http"
self.base_url = f"{protocol}://{host}:{port}"

# セッション初期化（遅延）
self._session: Optional[aiohttp.ClientSession] = None
```

2. _get_session()

概要

HTTPセッションを取得または作成する内部関数

仕様

```python
async def _get_session(self) -> aiohttp.ClientSession
```

詳細仕様

項目	内容
戻り値	`aiohttp.ClientSession` - 設定済みHTTPセッション
副作用	`self._session`の設定
キャッシュ	セッション再利用によるパフォーマンス最適化

SSL設定詳細

```python
ssl_context = ssl.create_default_context()
# 開発・テスト環境用設定
ssl_context.check_hostname = False
ssl_context.verify_mode = ssl.CERT_NONE
```

セッション設定

```python
# 認証設定
auth = aiohttp.BasicAuth(self.username, self.password)

# タイムアウト設定
timeout = aiohttp.ClientTimeout(total=30)

# セッション作成
self._session = aiohttp.ClientSession(
    auth=auth,
    timeout=timeout,
    connector=aiohttp.TCPConnector(ssl=ssl_context),
    headers={
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    }
)
```

3. _make_request()

概要

HTTP リクエストを実行する内部関数

仕様

```python
async def _make_request(
    self,
    method: str,
    endpoint: str,
    params: Dict[str, Any] = None,
    data: Dict[str, Any] = None
) -> Dict[str, Any]
```

詳細仕様

パラメータ	型	必須	説明
method	str	Yes	HTTPメソッド (GET, POST等)
endpoint	str	Yes	APIエンドポイントパス
params	Dict	No	クエリパラメータ
data	Dict	No	リクエストボディデータ

戻り値: `Dict[str, Any]` - JSON レスポンスデータ

エラーハンドリング

```python
# HTTP ステータスコード別処理
if response.status == 200:
    return await response.json()
elif response.status == 401:
    raise Exception("Authentication failed - check credentials")
elif response.status == 404:
    logger.warning(f"Resource not found: {url}")
    return {}
else:
    text = await response.text()
    raise Exception(f"HTTP {response.status}: {text}")
```

接続エラー処理

```python
except aiohttp.ClientError as e:
    logger.error(f"Request failed: {e}")
    raise Exception(f"Connection error: {e}")
```

4. get_projects()

概要

Coverity Connect から全プロジェクト情報を取得

仕様

```python
async def get_projects(self) -> List[Dict[str, Any]]
```

API エンドポイント

```
GET /api/viewContents/projects/v1
```

戻り値形式

```json
```

```
[
  {
    "projectKey": "PROJ001",
    "projectName": "WebApplication",
    "description": "メインWebアプリケーション",
    "createdDate": "2024-01-15T10:30:00Z",
    "lastModified": "2024-07-20T15:45:00Z"
  }
]
```


フォールバック処理

API失敗時は開発用ダミーデータを返却

5. get_project()

概要

指定プロジェクトの詳細情報を取得

仕様

```python
async def get_project(self, project_id: str) -> Optional[Dict[str, Any]]
```

処理アルゴリズム

1. `get_projects()`で全プロジェクト取得
2. `project_id`でフィルタリング
3. マッチング条件: `projectKey` または `projectName`

6. get_streams()

概要

ストリーム情報を取得（プロジェクトフィルタ可能）

仕様

```python
async def get_streams(self, project_id: str = "") -> List[Dict[str, Any]]
```

API エンドポイント

```
GET /api/viewContents/streams/v1?projectId={project_id}
```

戻り値形式

```json
[
  {
    "name": "main-stream",
    "description": "メイン開発ストリーム",
    "projectId": "WebApplication",
    "language": "MIXED"
  }
]
```

7. get_defects()

概要

欠陥情報を検索・取得（高度なフィルタリング対応）

仕様

```python
async def get_defects(
    self,
    stream_id: str = "",
    query: str = "",
    filters: Dict[str, str] = None,
    limit: int = 100
) -> List[Dict[str, Any]]
```

API エンドポイント

```
GET /api/viewContents/issues/v1?rowCount={limit}&streamId={stream_id}&query={query}
```

フィルタリング機能

フィルタ	説明	例
streamId	ストリーム限定	main-stream
checker	チェッカー限定	NULL_RETURNS
severity	重要度限定	High, Medium, Low
status	ステータス限定	New, Triaged, Fixed

戻り値形式

```json
[
  {
    "cid": "12345",
    "checkerName": "NULL_RETURNS",
    "displayType": "Null pointer dereference",
    "displayImpact": "High",
    "displayStatus": "New",
    "displayFile": "src/main.c",
    "displayFunction": "main",
    "firstDetected": "2024-01-15T10:00:00Z",
    "streamId": "main-stream"
  }
]
```

8. get_defect_details()

概要

CIDによる詳細欠陥情報取得

仕様

```python
async def get_defect_details(self, cid: str) -> Optional[Dict[str, Any]]
```

API エンドポイント

```
GET /api/viewContents/issues/v1/{cid}
```

拡張情報

```json
```

```
{
  "cid": "12345",
  "occurrenceCount": 1,
  "events": [
    {
      "eventNumber": 1,
      "eventTag": "assignment",
      "eventDescription": "Null assignment detected",
      "fileName": "src/main.c",
      "lineNumber": 42
    }
  ]
}
```


9. get_users() ✨ 新機能

概要

Coverity Connect ユーザー一覧を取得

仕様

```python
async def get_users(
    self,
    disabled: bool = False,
    include_details: bool = True,
    locked: bool = False,
    limit: int = 200
) -> List[Dict[str, Any]]
```

API エンドポイント

```
GET /api/v2/users?disabled={disabled}&includeDetails={include_details}&locked={locked}&rowCount={limit}
```

クエリパラメータ

パラメータ	型	説明
`disabled`	bool	無効化ユーザーを含める
`includeDetails`	bool	詳細情報を含める
`locked`	bool	ロックユーザーを含める
`rowCount`	int	最大取得数
`sortColumn`	str	ソート列 (`name`)
`sortOrder`	str	ソート順 (`asc`)

戻り値形式

```json
[
  {
    "name": "developer1",
    "email": "dev1@company.com",
    "familyName": "開発",
    "givenName": "太郎",
    "disabled": false,
    "locked": false,
    "superUser": false,
    "groupNames": ["Users"],
    "roleAssignments": [
      {
        "roleName": "developer",
        "scope": "global",
        "username": "developer1"
      }
    ],
    "lastLogin": "2024-07-20T15:30:00Z",
    "dateCreated": "2024-02-01T00:00:00Z",
    "local": true
  }
]
```

10. get_user_details()

概要

指定ユーザーの詳細情報を取得

仕様

```python
```

```
async def get_user_details(self, username: str) -> Optional[Dict[str, Any]]
```

API エンドポイント

```
GET /api/v2/users/{username}
```

フォールバック処理

1. 直接API呼び出し
2. 失敗時は`get_users()`でフィルタリング検索

11. close()

概要

HTTPセッションをクリーンアップ

仕様

```
python

async def close(self) -> None
```

処理内容

```
python

if self._session and not self._session.closed:
    await self._session.close()
```

12. Context Manager メソッド

aenter()

```
python

async def __aenter__(self):
    return self
```

aexit()

```
python

async def __aexit__(self, exc_type, exc_val, exc_tb):
    await self.close()
```

🔍 関数間依存関係

main.py 依存関係

```
cli()
└── run_server()
    └── create_server()
        ├── initialize_client()
        │   └── CoverityClient.__init__()
        └── MCP Tools Registration
            ├── search_defects() → CoverityClient.get_defects()
            ├── get_defect_details() → CoverityClient.get_defect_details()
            ├── list_projects() → CoverityClient.get_projects()
            ├── list_streams() → CoverityClient.get_streams()
            ├── get_project_summary() → multiple CoverityClient methods
            ├── list_users() → CoverityClient.get_users()
            ├── get_user_details() → CoverityClient.get_user_details()
            └── get_user_roles() → CoverityClient.get_user_details()
```

coverity_client.py 依存関係

```
CoverityClient.__init__()
├── _get_session()
│   ├── ssl.create_default_context()
│   ├── aiohttp.BasicAuth()
│   └── aiohttp.ClientSession()
└── Public API Methods
    ├── get_projects() → _make_request()
    ├── get_project() → get_projects()
    ├── get_streams() → _make_request()
    ├── get_defects() → _make_request()
    ├── get_defect_details() → _make_request()
    ├── get_users() → _make_request()
    └── get_user_details() → _make_request() + get_users()
```

🧪 テスト関数仕様

test_client() (coverity_client.py)

概要

開発用テスト関数

仕様

```python
async def test_client() -> None
```

テスト内容

1. ダミーサーバーとの接続テスト
2. 基本API呼び出しテスト
3. パフォーマンス検証

この関数仕様書は、Coverity Connect MCP Serverの全ての関数について実装レベルの詳細を提供しています。開発者が各関数の動作、パラメータ、戻り値、エラーハンドリングを完全に理解できるよう設計されています。