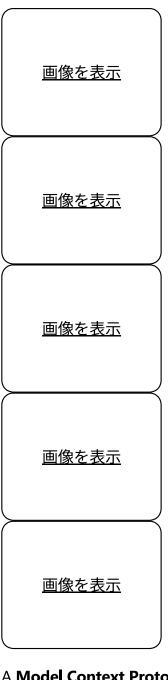
Coverity Connect MCP Server



A **Model Context Protocol (MCP) server** that provides seamless integration between Al assistants (like Claude Desktop) and **Black Duck Coverity Connect** static analysis platform.

Transform your Coverity workflow with natural language commands and automated analysis through Alpowered interactions.

🚀 Features

Comprehensive Coverity Integration

- **Project Management**: List and explore Coverity projects and streams
- Snapshot Analysis: Detailed defect analysis with automated reporting
- Security Focus: Specialized security vulnerability detection and analysis
- CI/CD Automation: Automated pipeline integration for continuous quality monitoring

• Quality Reports: Executive-level quality dashboards and trend analysis

AI-Powered Analysis

- Natural Language Queries: "Show me critical security issues in project X"
- Intelligent Filtering: Automatic prioritization of high-impact defects
- Contextual Recommendations: Al-driven remediation suggestions
- Trend Analysis: Historical data analysis and quality metrics

K Enterprise Ready

- SOAP API Integration: Full Coverity Connect Web Services support
- Authentication: Secure auth-key based authentication
- **Proxy Support**: Corporate network and proxy configuration
- Multi-Platform: Windows, macOS, and Linux support
- Docker Ready: Containerized deployment for enterprise environments

Installation

Using pip (Recommended)

bash

pip install coverity-connect-mcp

Using Docker

bash

docker pull \${DOCKER_USERNAME}/coverity-connect-mcp:latest

From Source

bash

git clone https://github.com/\${GITHUB_USERNAME}/coverity-connect-mcp.git cd coverity-connect-mcp pip install -e .

Configuration

1. Environment Variables

bash

```
# Required
export COVAUTHUSER="your_coverity_username"
export COVAUTHKEY="your_coverity_auth_key"

# Optional - Coverity Server
export COVERITY_HOST="your-coverity-server.com"
export COVERITY_PORT="443"
export COVERITY_SSL="True"
export COVERITY_BASE_DIR="/path/to/coverity/workspace"

# Optional - Corporate Proxy (if needed)
export PROXY_HOST="your-proxy-server.com"
export PROXY_PORT="3128"
export PROXY_USER="proxy_username" # if authentication required
export PROXY_PASS="proxy_password" # if authentication required
```

2. Claude Desktop Integration

Add to your (claude_desktop_config.json):

3. Docker Configuration

yaml			

docker-compose.yml

version: '3.8'

services:

coverity-mcp:

image: \${DOCKER_USERNAME}/coverity-connect-mcp:latest

environment:

- COVAUTHUSER=\${COVAUTHUSER}
- COVAUTHKEY=\${COVAUTHKEY}
- COVERITY_HOST=\${COVERITY_HOST}
- # Optional proxy settings
- PROXY_HOST=\${PROXY_HOST}
- PROXY_PORT=\${PROXY_PORT}

ports:

- "8000:8000"

© Usage Examples

Basic Project Analysis

Show me all Coverity projects and their current status

Security-Focused Analysis

Analyze the latest snapshot of project "MyWebApp" and focus on high-severity security vulnerabilities. Provide specific remediation recommendations.

Quality Reporting

Generate a comprehensive quality report for project "MyProject" including trends over the last 30 days

CI/CD Integration

Run automated Coverity analysis for group "web-team", project "frontend", branch "main" with commit message "Security fixes"

Advanced Filtering

Show me all CERT-C violations in project "EmbeddedSystem" with impact level "High" and provide code examples for fixes



Available Tools

Tool	Description	Example Usage	
get_coverity_projects	List all accessible Coverity projects	Project inventory and access verification	
get_project_streams	Get streams for a specific project	Stream-based analysis planning	
get_stream_snapshots	Retrieve snapshot history for a stream	Historical analysis and trend tracking	
analyze_snapshot_defects	Detailed defect analysis of a snapshot	In-depth security and quality analysis	
run_coverity_automation	Execute automated CI/CD pipeline	Continuous integration workflows	
parse_coverity_issues	Parse and filter analysis results	Custom reporting and data extraction	
generate_quality_report	Create executive quality reports	Management reporting and KPIs	

Documentation

- **<u>Installation Guide</u>** Detailed setup instructions
- **<u>Configuration Reference</u>** Complete configuration options
- <u>Usage Examples</u> Real-world usage scenarios
- **API Reference** Comprehensive API documentation
- **<u>Troubleshooting</u>** Common issues and solutions

Testing

bash

Run unit tests

pytest tests/

Run integration tests

pytest tests/ -m integration

Run with coverage

pytest --cov=coverity_mcp_server tests/

Test with Docker

docker-compose -f docker-compose.test.yml up --abort-on-container-exit

Contributing

We welcome contributions! Please see our **Contributing Guide** for details.

Development Setup

bash

```
git clone https://github.com/${GITHUB_USERNAME}/coverity-connect-mcp.git
cd coverity-connect-mcp
python -m venv venv
source venv/bin/activate # On Windows: venv\Scripts\activate
pip install -e ".[dev]"
pre-commit install
```

Submitting Changes

- 1. Fork the repository
- 2. Create a feature branch (git checkout -b feature/amazing-feature)
- 3. Commit your changes (git commit -m 'Add amazing feature')
- 4. Push to the branch (git push origin feature/amazing-feature)
- 5. Open a Pull Request

License

This project is licensed under the MIT License - see the <u>LICENSE</u> file for details.

🙏 Acknowledgments

- Black Duck Coverity for providing the static analysis platform
- Anthropic for the Model Context Protocol and Claude AI
- Open Source Community for the foundational libraries and tools

Support

- **GitHub Issues**: Report bugs or request features
- **Discussions**: Community support and questions
- **Security Issues**: Please see our <u>Security Policy</u>

📕 Roadmap

v1.1 : Advanced filtering ar	nd custom views	
v1.2: Multi-tenant support	t and user managemen	t

- **v1.3**: REST API alongside SOAP support
- **v1.4**: Machine learning-powered defect prioritization
- **v2.0**: Plugin architecture and third-party integrations

Made with **for** the software security community

Transform your static analysis workflow with the power of AI