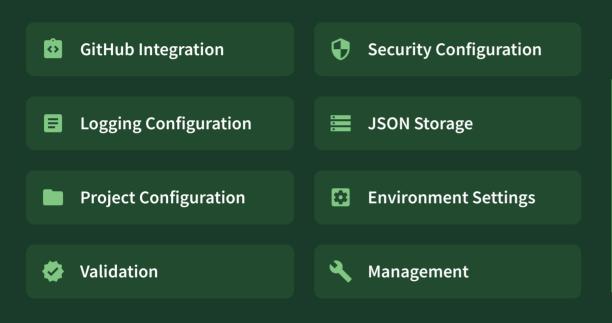


Configuration Overview

The AutoProjectManagement Configuration Guide provides comprehensive instructions for setting up and customizing all aspects of your project management environment.



Purpose of Proper Configuration

- Ensure system security and data protection
- Optimize performance for your specific environment
- Enable seamless integration with external services
- Provide reliable logging for troubleshooting
- Support different deployment environments

GitHub Integration

Environment Variables

A Key Settings

auto_sync Enable automatic synchronization with GitHub

sync_interval Time in seconds between sync operations

default_branch Default branch for repository operations

- Best Practices
- ✓ Use fine-grained tokens with minimal required permissions
- Store tokens securely, never commit to version control
- Set reasonable sync intervals to avoid API rate limits

{} Configuration Example

```
"username": "yourusername", "default_repo":
"username/repository", "default_branch": "main",
"auto_sync": true, "sync_interval": 300 } }
```

Security Configuration

• Security Architecture



Encryption
Protect data in transit



Rate Limiting
Prevent abuse

Configuration Options

Setting	Туре	Description
jwt_secret_key	string	JWT signing secret key
jwt_algorithm	string	JWT algorithm (HS256/RS256)
api_key_required	boolean	Require API key
cors_allowed_origins	list	CORS allowed origins

P Best Practices

Strong JWT Secrets

Use long, randomly generated secrets for JWT signing

Token Expiration

Set reasonable expiration times for access tokens

Security Components

○ JWT Tokens

JSON Web Tokens for secure authentication

SSL/TLS

Encrypted communication channels

○ API Kevs

Unique identifiers for API access control

© CORS Policies

Cross-origin resource sharing controls

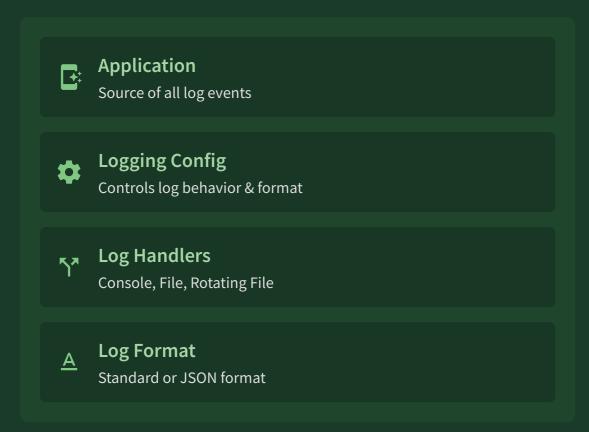
Configuration Examples

```
{ "security": { "jwt_secret_key": "your-super-secret-
key-change-this", "jwt_algorithm": "HS256",
"jwt_expiration_minutes": 60, "api_key_required":
true, "api_key_header": "X-API-Key",
"cors_allowed_origins": ["*"] } }
```

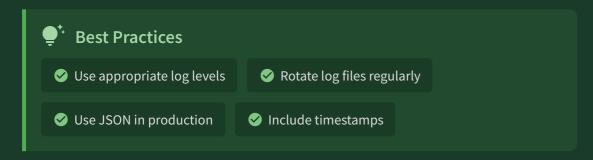
{ "security": { "ssl_cert_path": "/certs/server.crt",
 "ssl_key_path": "/certs/server.key" } }

Logging Configuration

A Logging Architecture



Setting	Туре	Description
level	string	Logging level (DEBUG, INFO, WARNING, ERROR)
format	string	Log format string
file_path	string	Log file path
max_file_size	string	Maximum log file size
json_format	boolean	Use JSON format



Configuration Examples

```
Simple configuration for development environment

{ "logging": { "level": "INFO", "format": "%
  (asctime)s - %(name)s - %(levelname)s - %
  (message)s" } }
```

```
File Logging with Rotation

Production-ready file logging with rotation

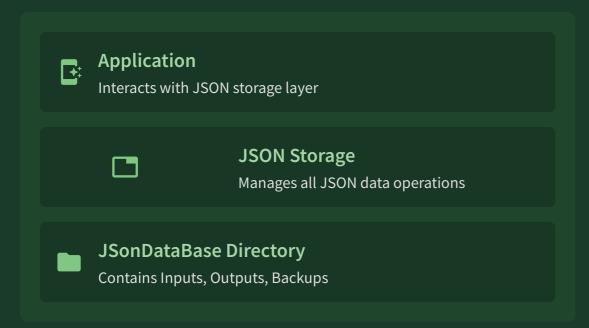
{ "logging": { "level": "DEBUG", "file_path":
   "logs/autoproject.log", "max_file_size": "10MB",
   "backup_count": 5 } }
```

```
{} JSON Structured Logging
Machine-readable logs for monitoring systems

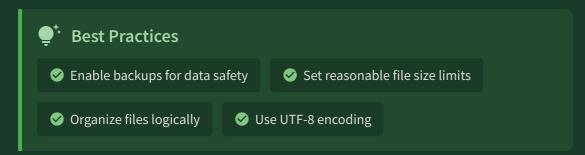
{ "logging": { "level": "INFO", "json_format":
  true, "include_extra": true } }
```

JSON Storage Configuration

Storage Architecture



Setting	Туре	Description
type	string	Storage type (always 'json')
json_path	string	Main JSON configuration file
data_directory	string	Directory for JSON data files
backup_enabled	boolean	Enable automatic backups
max_file_size	integer	Maximum JSON file size



Configuration Examples

```
Basic JSON Storage
Simple configuration with default paths

{ "database": { "type": "json", "json_path":
    "autoproject.json", "data_directory":
    "JSonDataBase", "backup_enabled": true } }
```

Custom Directory Structure

Custom paths for organized storage

```
{ "database": { "type": "json", "json_path":
"config/project_data.json", "data_directory":
"project_data", "inputs_path":
"project_data/inputs", "outputs_path":
"project_data/outputs", "backup_count": 10 } }
```

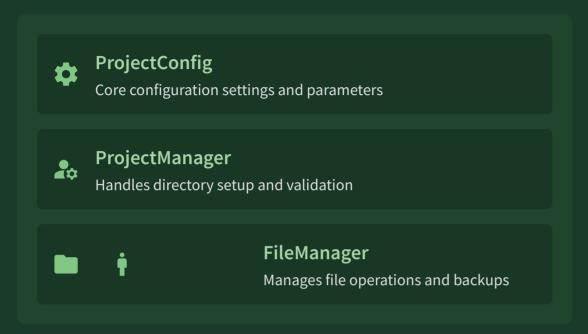
Production JSON Storage

Optimized for production environment

```
{ "database": { "type": "json", "json_path":
  "/opt/autoproject/data/main.json",
  "data_directory": "/opt/autoproject/data",
  "backup_enabled": true, "backup_count": 20,
  "max_file_size": 52428800 } }
```

Project Configuration

A Configuration Structure



Configuration Example

```
{ "project": { "base_path": "/path/to/project",
  "data_path": "JSonDataBase", "output_path":
  "JSonDataBase/OutPuts", "backup_path":
  "project_management/PM_Backups", "max_file_size":
  10485760, "allowed_extensions": [ ".py", ".js",
  ".json", ".md" ], "max_workers": 8, "timeout_seconds":
  300, "enable_auto_commit": true,
  "enable_risk_analysis": true,
  "enable_progress_tracking": true,
  "enable_github_integration": true } }
```


Set reasonable file size limits

Enable auto-commit for safety

Adjust workers based on system resources

{} data_path **b**ase_path Project base directory Data directory path max_file_size allowed_extensions Maximum file size (10MB) Permitted file types max workers timeout seconds Maximum worker threads **Operation timeout** commit enable_auto_commit • enable_risk_analysis **Automatic commits** Risk analysis features **Best Practices**

Restrict allowed file extensions

Auto Commit
Automatically save changes to repository

Risk Analysis
Evaluate potential project risks

Progress Tracking
Monitor project advancement

GitHub Integration
Connect with GitHub repositories

Environment-Specific Configurations

Configuration Matrix

Feature	Development	Production	Testing
API Debug	true	false	true
Storage Type	JSON files	JSON files	JSON files
Logging Level	DEBUG	INFO	DEBUG
GitHub Integration	true	true	false
SSL Required	false	true	false
Rate Limiting	relaxed	strict	disabled

(1-1) Environment Detection



Environment Examples

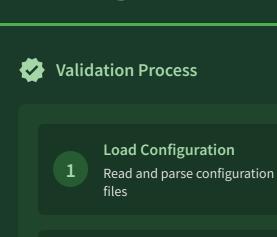
```
Development
 ≢ Debug: true
               ■ Logging: DEBUG
                                 GitHub: enabled
 { "api": { "host": "127.0.0.1", "port": 8000, "debug":
 true, "reload": true }, "logging": { "level": "DEBUG" } }
Production
 SSL: enabled
                ■ Logging: INFO
                                 Rate limiting: strict
 { "api": { "host": "0.0.0.0", "port": 8080, "debug": false
 }, "security": { "ssl_cert_path": "/certs/server.crt" },
 "logging": { "level": "INFO" } }
▲ Testing
               Storage: in-memory
 ≢ Debug: true

    Ø GitHub: disabled

 { "api": { "host": "127.0.0.1", "port": 8001, "debug": true
 }, "database": { "json_path": ":memory:" }, "project": {
```

"enable_github_integration": false } }

Configuration Validation & Management



- Validate Required FieldsCheck for presence of mandatory settings
- Validate Data Types

 Ensure correct data types for all values
- Validate Security Settings
 Check security parameters for risks
- Validate Dependencies
 Verify external service connections



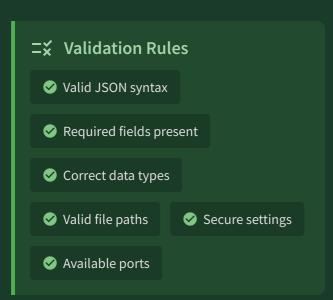


Commandline interface utilities

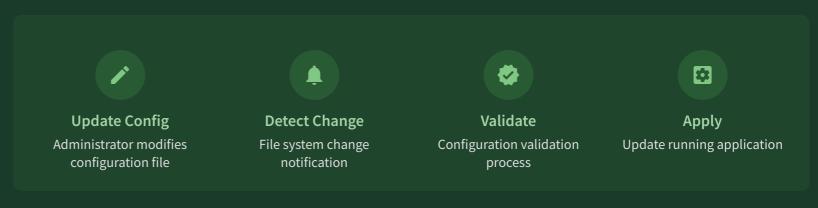


Live configuration updates Backup
System

Automatic configuration backups



Hot Reload Process



<> Management Commands

Python API config.validate() # Validate current configuration print_config()
Display current configuration reload_config() # Reload configuration from
files # CLI Arguments --config config.json # Specify configuration file -validate # Validate configuration --generate # Generate sample configuration -env development # Specify environment

♠ Backup & Restore



Restore Commands

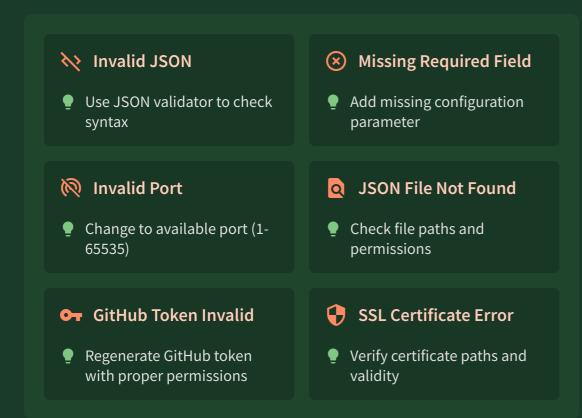
cp config.backup.json config.json

python -m
autoprojectmanagement.configuration
validate

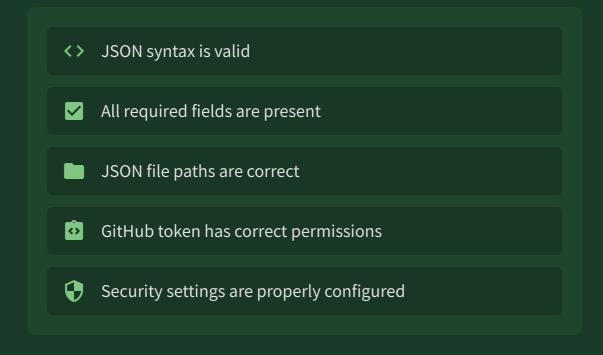
python -m
autoprojectmanagement.configuration
restore

Troubleshooting & Quick Reference

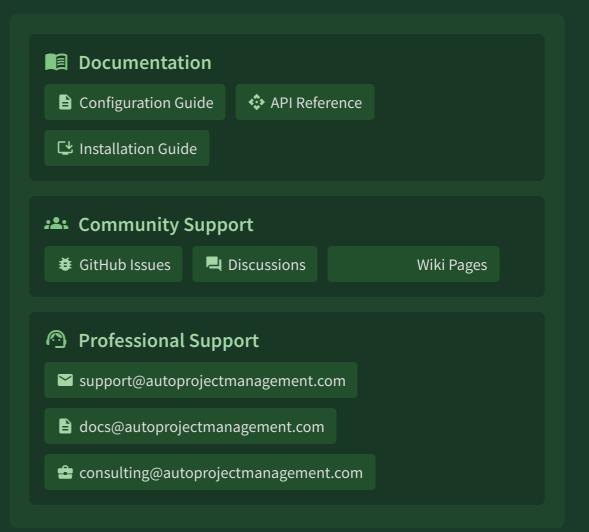
(!) Common Issues & Solutions



≠= Configuration Checklist



Getting Help



Diagnostic Commands

