AutoProjectManagement Configuration Guide

Complete Setup and Configuration Instructions

Streamline Your Project Management

Configuration Guide Overview

AutoProjectManagement is a comprehensive project management system with flexible configuration options. This guide covers all essential configuration areas to help you set up and optimize your environment.

GitHub Integration
Connect repositories, manage tokens

Security Configuration
JWT, API keys, SSL/TLS settings

Logging Configuration
Log levels, formats, file rotation

JSON Storage
Data paths, backup settings

Project Configuration
Paths, extensions, workers

Environment Config
Dev, production, testing settings

Configuration Validation
Rules, error handling, methods

Configuration Management
Commands, tools, hot-reload

Troubleshooting
Common issues, debugging, diagnostics

GitHub Integration

Environment Variables

```bash
export GITHUB_TOKEN=ghp_xxxxxxxxxxxxxxxxxxxx
export GITHUB_USERNAME=yourusername
export GITHUB_DEFAULT_REPO=username/repository
```

Set these variables in your shell environment or .env file

Configuration Example

```json
{
  "github": {
    "token": "ghp_xxxxxxxxxxxxxxxxxxxx",
    "username": "yourusername",
    "default_repo": "username/repository",
    "default_branch": "main",
    "auto_sync": true,
    "sync_interval": 300
  }
}
```

Add this to your configuration file for GitHub integration

Security Configuration

Security Architecture

Authentication	Authorization	Encryption	Rate Limiting
JWT Tokens	API Keys	SSL/TLS	CORS Policies

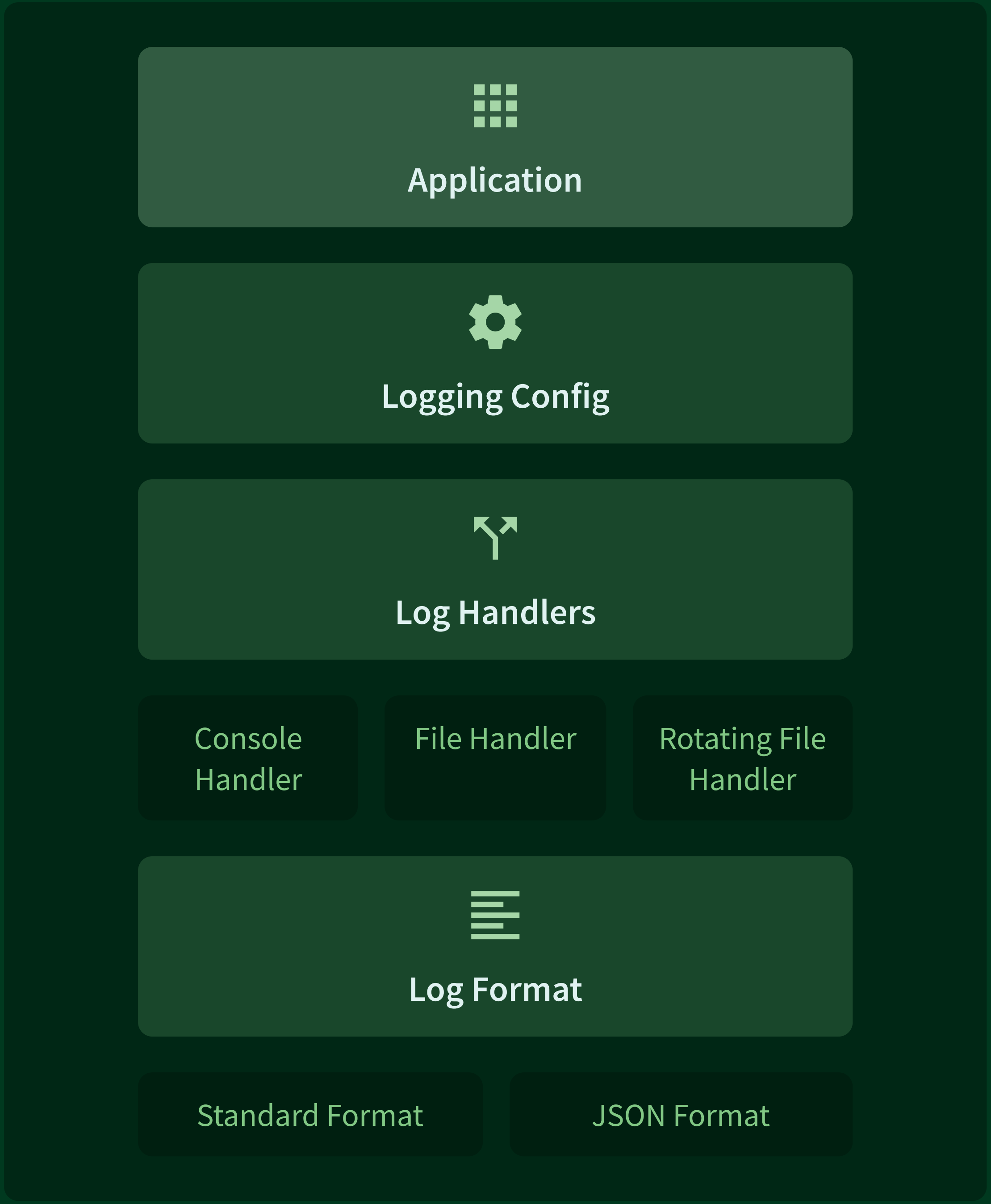
Configuration Options

Setting	Type	Default
jwt_secret_key	string	change-this
jwt_algorithm	string	HS256
jwt_expiration_minutes	integer	60
api_key_header	string	X-API-Key
api_key_required	boolean	false
cors_allowed_origins	list	["*"]
ssl_cert_path	string	-
ssl_key_path	string	-

Multi-layered security approach protects your project data

Logging Configuration

Logging Architecture



Flexible logging system with multiple output options

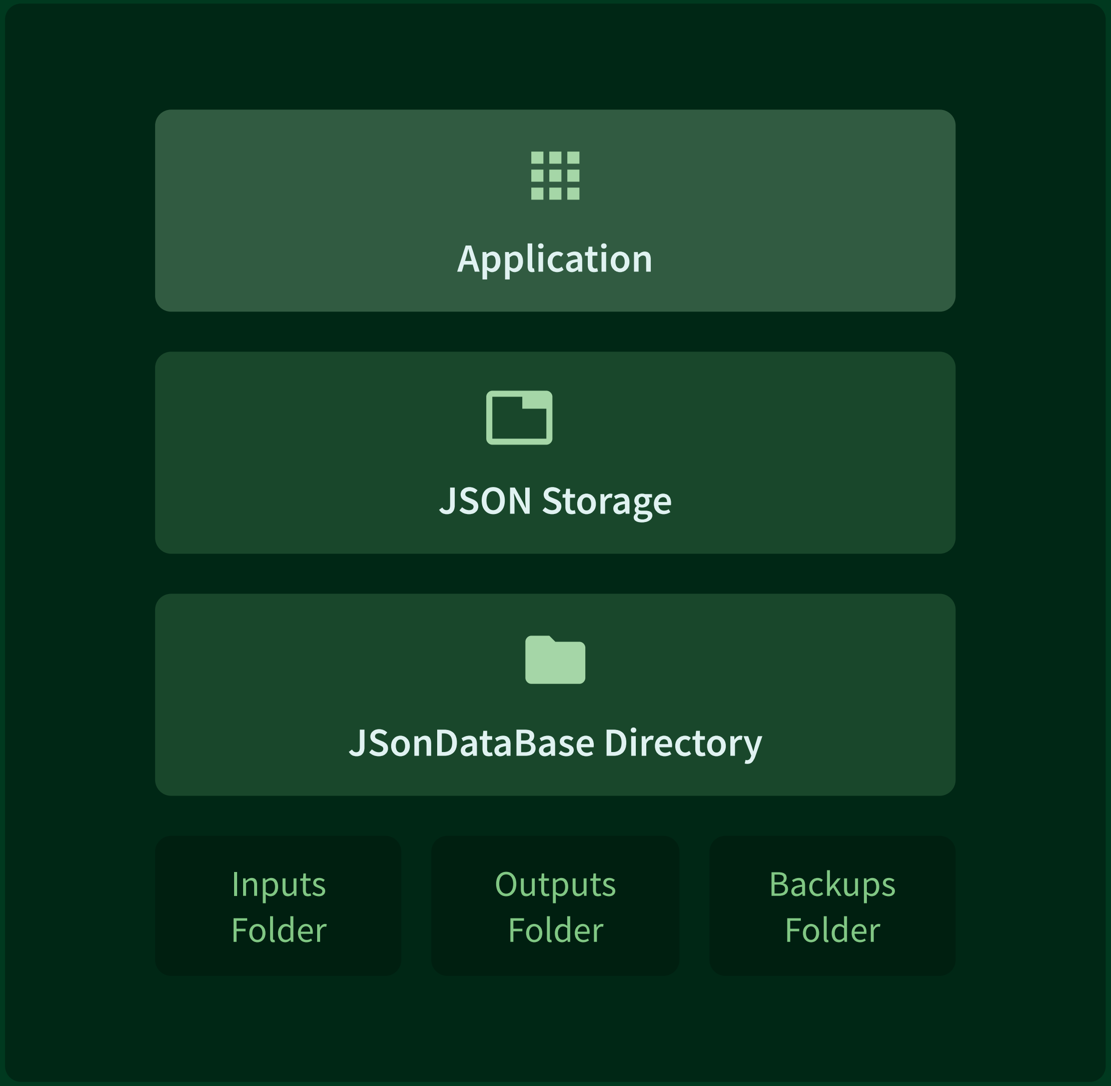
Configuration Options

Setting	Type	Default
level	string	INFO
format	string	standard
file_path	string	-
max_file_size	string	10MB
backup_count	integer	5
json_format	boolean	false
include_extra	boolean	true

Customize logging behavior for different environments

JSON Storage Configuration

Storage Architecture



Hierarchical JSON file storage system with automatic backups

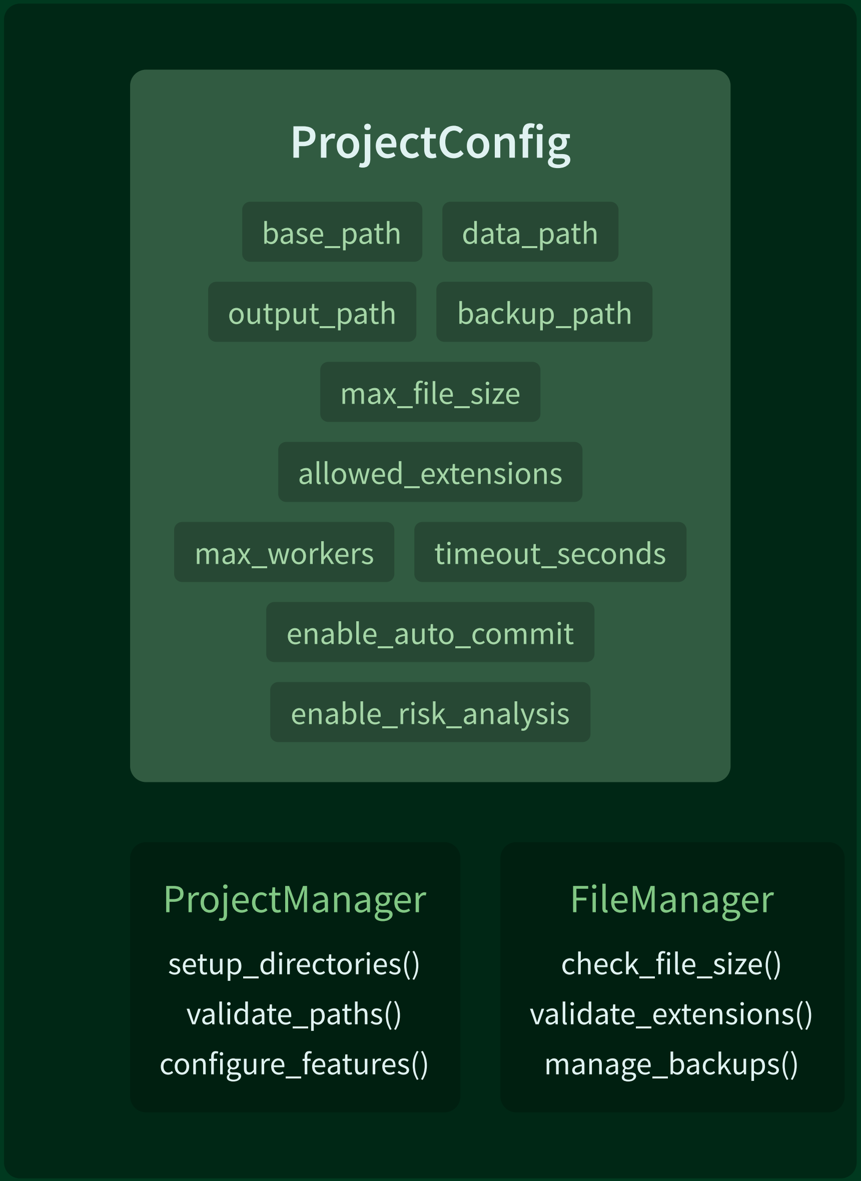
Configuration Options

Setting	Type	Default
type	string	json
json_path	string	autoproject.json
data_directory	string	JSonDataBase
inputs_path	string	JSonDataBase/Inputs
outputs_path	string	JSonDataBase/OutPuts
backup_enabled	boolean	true
backup_count	integer	5
max_file_size	integer	10485760
encoding	string	utf-8

Customize storage paths and backup settings for your project

Project Configuration

Configuration Structure



Object-oriented design for flexible project management

Configuration Options

Setting	Type	Default
base_path	string	current directory
data_path	string	JSonDataBase
output_path	string	JSonDataBase/OutPuts
backup_path	string	project_management/PM_Backups
max_file_size	integer	10485760
max_workers	integer	4
timeout_seconds	integer	300
enable_auto_commit	boolean	true
enable_risk_analysis	boolean	true
enable_github_integration	boolean	true

Customize project settings to match your workflow

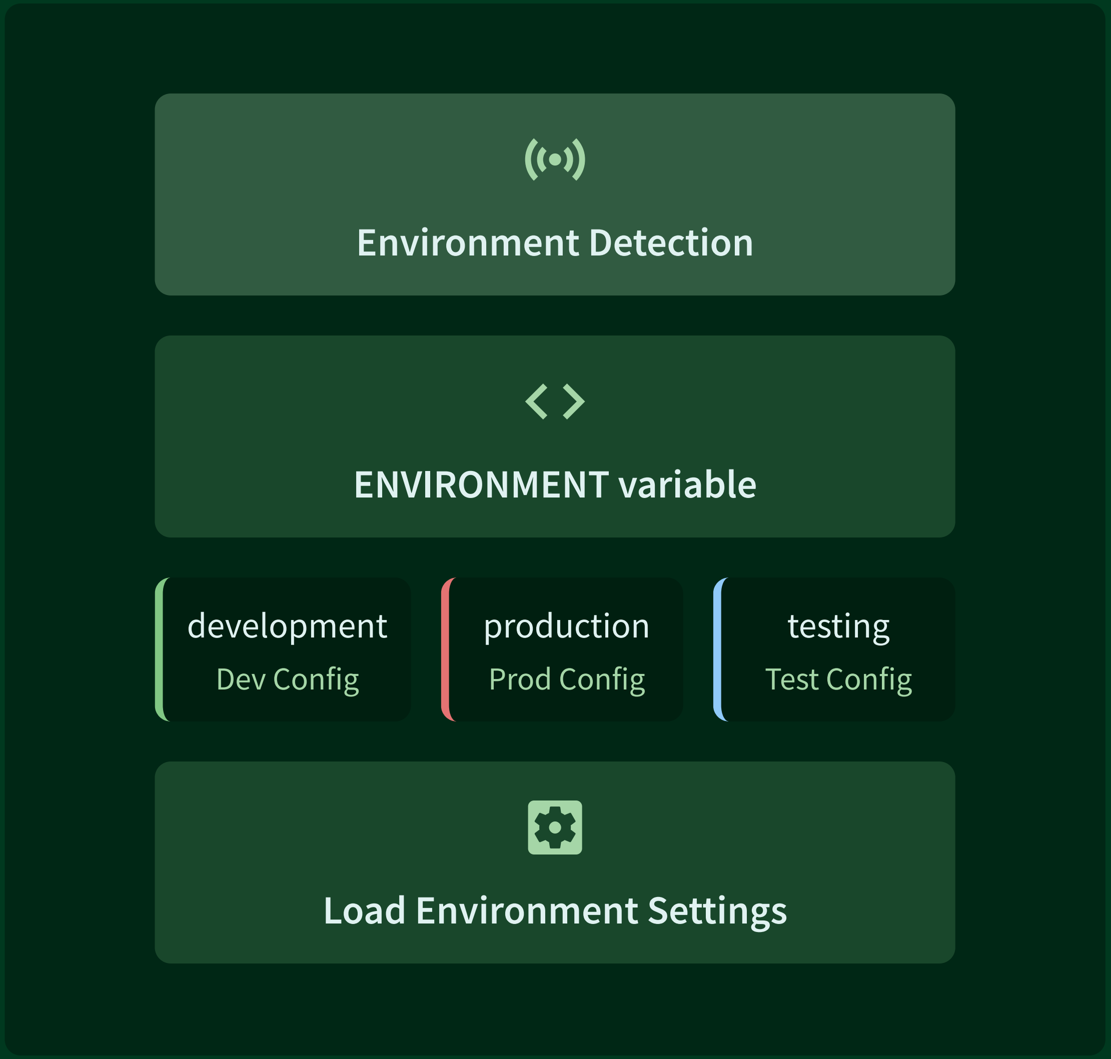
Environment-Specific Configurations

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

Optimize settings for each environment to balance security and functionality

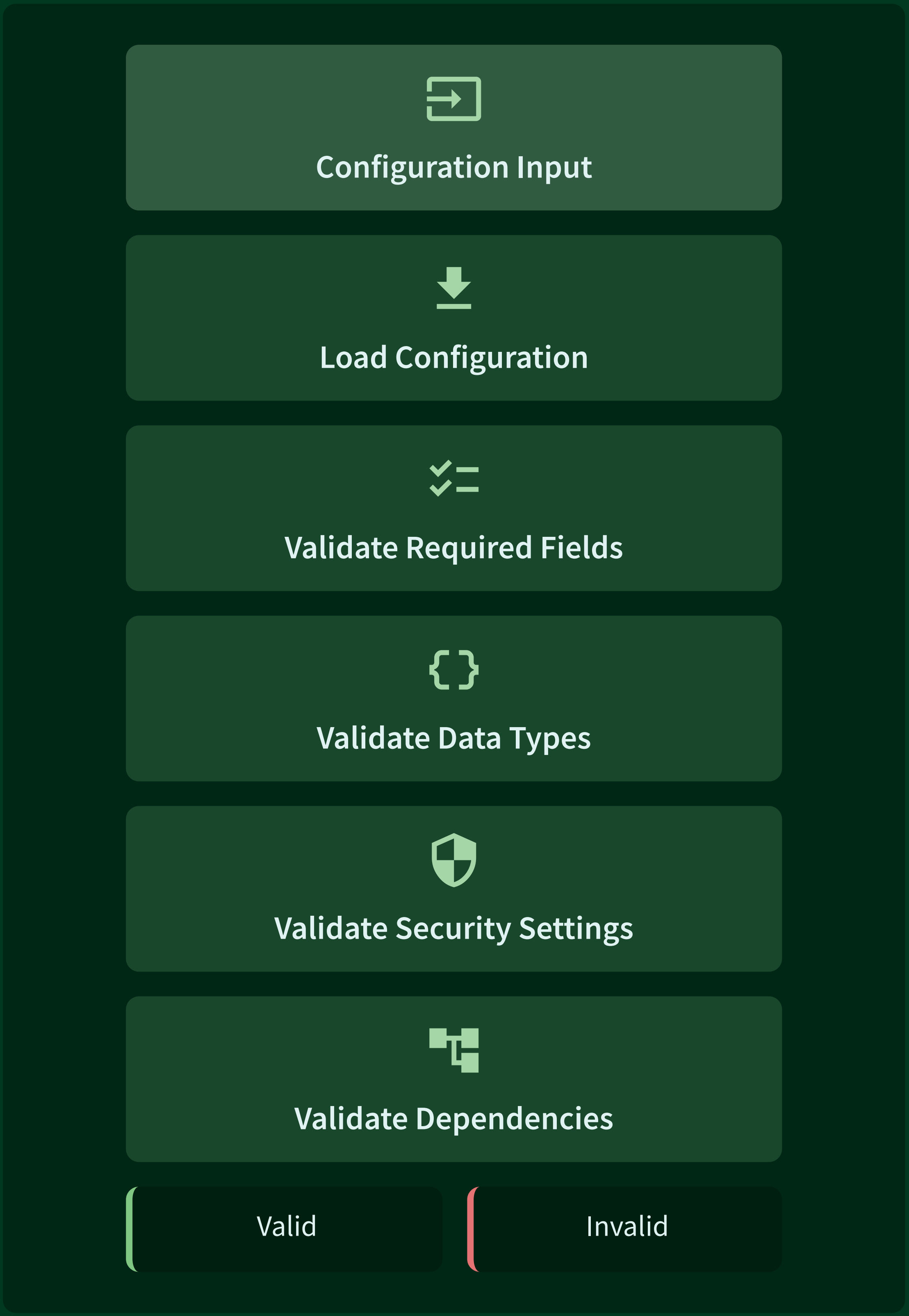
Environment Detection



Automatic environment detection simplifies configuration management across deployment stages

Configuration Validation

Validation Process



Systematic validation ensures configuration integrity and security

Validation Rules

Section	Required Fields	Validation Rules
API	host, port	host must be valid IP/domain, port 1-65535
Database	type	must be 'json'
GitHub	token (if enabled)	must be valid GitHub token format
Security	jwt_secret_key	must not be default value
Logging	level	must be valid log level
Project	base_path	must be valid directory path

Comprehensive validation rules prevent configuration errors and security issues

Configuration Management

Management Commands

config.validate()
Validate current configuration

print_config()
Display current configuration

reload_config()
Reload configuration from files

--config
Specify configuration file

--validate
Validate configuration

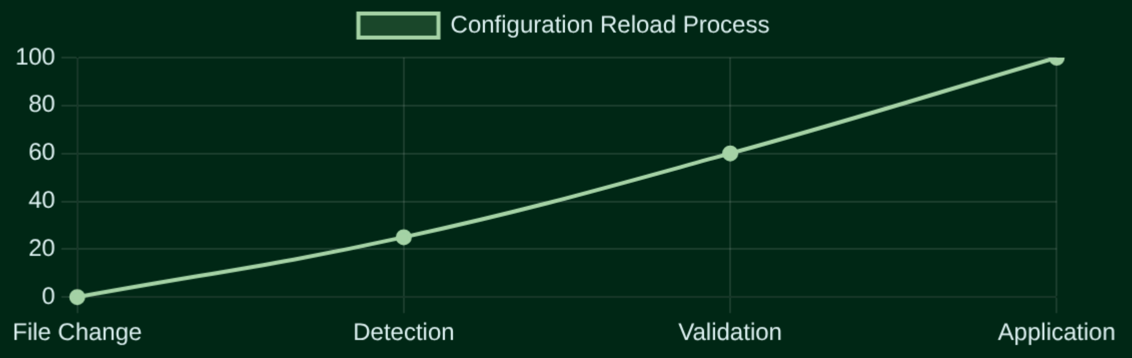
Hot-Reload Process

Administrator
Updates config.json file

File System
Detects file change event

Config Manager
Validates new configuration

Application
Applies new configuration



Management Tools

Configuration CLI

```
# Validate configuration
python -m autoprojectmanagement.configuration
validate --config config.json
# Generate sample configuration
python -m autoprojectmanagement.configuration
generate --env development
```

Configuration Script

```
from autoprojectmanagement.configuration.manager
import ConfigManager

manager = ConfigManager()
manager.load_config('config.json')
manager.validate_config()
manager.apply_config()
```

Backup & Restore

Backup Configuration

```
# Backup current
configuration
cp config.json
config.backup.json

# Backup with
timestamp
cp config.json
config.backup.$(date
+%Y%m%d_%H%M%S).json
```

Restore Configuration

```
# Restore from backup
cp config.backup.json config.json

# Validate after restore
python -m
autoprojectmanagement.configuration
validate
```

Troubleshooting

Common Configuration Issues

Issue	Symptom	Solution
Invalid JSON	JSON parsing error	Use JSON validator
Missing required field	Validation error	Add missing configuration
Invalid port	Port already in use	Change port number
JSON file not found	File not found error	Check file paths
GitHub token invalid	401 Unauthorized	Regenerate GitHub token
SSL certificate error	SSL handshake failed	Check certificate paths

Debugging & Diagnostics

Enable Debug Logging

Set logging level to DEBUG for detailed information

```
{
  "logging": {
    "level": "DEBUG"
  }
}
```

Configuration Test Script

Validate configuration loading and settings

```
import json
from autoproject_configuration import Config

config = Config()
config.validate()
print(f"Storage: {config.database.type}")
```

Diagnostic Commands

Check configuration syntax
```
python -m json.tool config.json
```

Validate configuration
```
python -c "from autoproject_configuration import config; config.validate()"
```

Test JSON storage
```
python -c "from autoproject_configuration import config; print(f'Storage: {config.database.type}')"
```

Systematic debugging approach helps identify configuration issues quickly

Quick Reference

Configuration Template

```
{ "api": { "host": "127.0.0.1", "port": 8000,
"debug": false, "cors_origins":
["http://localhost:3000"] }, "database": {
"type": "json", "json_path": "autoproject.json",
"backup_enabled": true }, "github": { "token":
"your-github-token", "default_repo":
"username/repository" }, "security": {
"jwt_secret_key": "your-secret-key",
"jwt_expiration_minutes": 60 }, "logging": {
"level": "INFO", "file_path":
"logs/autoproject.log" }, "project": {
"base_path": "/path/to/project",
"enable_auto_commit": true } }
```

Development

```
export
API_HOST=127.0.0.1
export API_PORT=8000
export
API_DEBUG=true
export DB_TYPE=json
export
GITHUB_TOKEN=your-
token
```

Production

```
export
API_HOST=0.0.0.0
export API_PORT=8080
export
API_DEBUG=false
export DB_TYPE=json
export
SECURITY_SSL=true
```

Validation Checklist

- ✓ JSON syntax is valid
- ✓ All required fields are present
- ✓ JSON file paths are correct
- ✓ GitHub token has correct permissions
- ✓ Security settings are properly configured
- ✓ File paths exist and are accessible
- ✓ Port numbers are available
- ✓ Environment variables are set

Quick Validation Commands

```
# Validate configuration
python -m autoprojectmanagement.configuration
validate

# Check configuration syntax
python -m json.tool config.json

# Test GitHub integration
python -c "from
autoprojectmanagement.services.github_integration
import GitHubIntegration; print('GitHub ready')"
```

Always validate your configuration before deploying to production