RuleKeeper

Policy Quick Reference

Complete Policy Lookup Guide

This document provides a quick reference lookup for all RuleKeeper policies. Use this guide when modifying the rulekeeper.yaml configuration file.

Severity Levels:

CRITICAL	Block deployment - Security/compliance risk
HIGH	Fix before deployment
MEDIUM	Address in current sprint
LOW	Best practice recommendation

Naming Conventions

ID	Name	Description	Severity
CS-NAME-001	Class/Interface Naming	Classes and Interfaces must use PascalCase	HIGH
CS-NAME-002	Method Naming	Methods must use PascalCase	HIGH
CS-NAME-003	Variable/Field/Parameter ...	Variables, fields, and parameters must use camelCase	HIGH
CS-NAME-004	Constant Naming	Constants must use UPPER_SNAKE_CASE	HIGH
CS-NAME-005	Private Field Naming	Private fields must use _camelCase (underscore prefix)	HIGH
CS-NAME-006	Async Method Naming	Async methods must end with 'Async' suffix	HIGH
CS-NAME-007	Interface Naming	Interfaces must be prefixed with 'I'	HIGH
CS-NAME-008	Request/Response DTO Naming	Request and Response DTOs must end with Request or Response ...	HIGH
CS-NAME-009	Boolean Variable Naming	Boolean variables should use is/has/can/should prefixes	MEDI
CS-NAME-010	Event Handler Naming	Event handlers should follow 'On' + EventName pattern	MEDI

File & Project Organization

ID	Name	Description	Severity
CS-FILE-001	One Class Per File	Each file should contain only one class	MEDI
CS-FILE-002	File Name Matches Class	File name must match the class name it contains	HIGH
CS-FILE-003	Feature-Based Organization	Group files logically by feature, not layer (vertical slicing)	MEDI
CS-FILE-004	Namespace Matches Folder ...	Namespace should reflect the folder structure	MEDI

Method Design & Readability

ID	Name	Description	Severity
CS-METHOD-001	Single Responsibility	Methods should be small and do one thing	HIGH
CS-METHOD-002	Method Length	Keep method length at or below 30 lines	MEDI
CS-METHOD-003	Parameter Count	Avoid long parameter lists - use DTOs instead	MEDI
CS-METHOD-004	Cyclomatic Complexity	Methods should have low cyclomatic complexity	MEDI
CS-METHOD-005	No Nested Ternary	Avoid nested ternary operators	MEDI

Secure Coding Practices

ID	Name	Description	Severity
CS-SEC-001	Parameterized Queries	Never concatenate SQL or user inputs - use parameterized que...	CRIT
CS-SEC-002	Input Validation	Always validate user input	CRIT
CS-SEC-003	Log Sanitization	Sanitize logs - no sensitive data (PIN, password, token)	CRIT
CS-SEC-004	Secret Protection	Use SecureString or data masking for secrets	CRIT
CS-SEC-005	Configuration Security	Protect configuration via Azure Key Vault or AWS Secrets Man...	CRIT
CS-SEC-006	No Hardcoded Credentials	Never hardcode credentials in source code	CRIT
CS-SEC-007	XSS Prevention	Sanitize output to prevent Cross-Site Scripting	CRIT
CS-SEC-008	Path Traversal Prevention	Validate file paths to prevent directory traversal	CRIT

Exception Handling & Logging

ID	Name	Description	Severity
CS-EXC-001	Meaningful Exception Hand...	Use try-catch only where you can handle errors meaningfully	HIGH
CS-EXC-002	Contextual Logging	Log exceptions with context, but not sensitive data	HIGH
CS-EXC-003	No Empty Catch Blocks	Avoid empty catch blocks	CRIT
CS-EXC-004	Domain Exceptions	Throw domain-specific exceptions when needed	MEDI
CS-EXC-005	No Catch-All Without Rethrow	Catching all exceptions should rethrow or terminate	HIGH

Asynchronous Programming

ID	Name	Description	Severity
CS-ASYNC-001	Always Await	Always await async calls	HIGH
CS-ASYNC-002	No Blocking Async	Don't block async with .Result or .Wait()	CRIT
CS-ASYNC-003	ConfigureAwait in Libraries	Use ConfigureAwait(false) in library code	MEDI
CS-ASYNC-004	Async Void Avoidance	Avoid async void except for event handlers	HIGH
CS-ASYNC-005	Proper Cancellation Token...	Async methods should accept and use CancellationToken	MEDI

Dependency Injection & SOLID

ID	Name	Description	Severity
CS-DI-001	Depend on Abstractions	Depend on interfaces, not concrete types	HIGH
CS-DI-002	Use IoC Container	Use built-in IServiceCollection or IoC containers	HIGH
CS-DI-003	Avoid New in Business Logic	Avoid 'new' keyword for dependencies inside business logic	HIGH
CS-DI-004	Constructor Injection Only	Use constructor injection, not property or method injection	MEDI
CS-DI-005	Service Lifetime Consistency	Ensure consistent service lifetimes in DI registration	HIGH

Constants & Magic Numbers

ID	Name	Description	Severity
CS-CONST-001	No Magic Numbers	Avoid magic numbers or strings in code	MEDI
CS-CONST-002	Use Named Constants	Use named constants or enums instead of literals	MEDI
CS-CONST-003	No Magic Strings	Avoid magic strings in code	MEDI

Data Validation

ID	Name	Description	Severity
CS-VAL-001	DTO Validation	Always validate input DTOs using attributes or FluentValidation	HIGH
CS-VAL-002	Client and Server Validation	Validate both client and server side	HIGH
CS-VAL-003	Null Checks	Check for null before using objects	HIGH
CS-VAL-004	Guard Clauses	Use guard clauses for parameter validation	MEDI

Logging Standards

ID	Name	Description	Severity
CS-LOG-001	Structured Logging	Use structured logging	HIGH
CS-LOG-002	No Sensitive Data in Logs	Never log sensitive data (PIN, password, token)	CRIT
CS-LOG-003	Appropriate Log Levels	Log at appropriate levels (Info, Warning, Error, Critical)	MEDI
CS-LOG-004	Include Correlation ID	Include correlation/trace ID in logs for distributed tracing	MEDI

Code Comments & Documentation

ID	Name	Description	Severity
CS-DOC-001	XML Comments for Public APIs	Use XML comments for public APIs	MEDI
CS-DOC-002	Comment Why Not What	Comment why, not what - avoid redundant comments	LOW
CS-DOC-003	TODO Comments	TODO comments should include ticket/issue reference	LOW

Immutability & Defensive Coding

ID	Name	Description	Severity
CS-IMM-001	Use Readonly	Use readonly for fields that don't change after construction	MEDI
CS-IMM-002	No Mutable Collections	Avoid exposing mutable collections	MEDI
CS-IMM-003	Clone External Data	Clone or copy external data inputs	MEDI
CS-IMM-004	Use Records for DTOs	Consider using records for immutable DTOs	LOW

Secure Configuration

ID	Name	Description	Severity
CS-CFG-001	No Secrets in Source	No secrets in source code or appsettings.json	CRIT
CS-CFG-002	Use Secret Managers	Use environment variables or secret managers	CRIT
CS-CFG-003	Secure Connection Strings	Connection strings should use integrated security or managed...	HIGH

Secure String Handling

ID	Name	Description	Severity
CS-STR-001	No Plain Text Secrets	Avoid keeping secrets as plain strings in memory	HIGH
CS-STR-002	Use SecureString	Use SecureString or encrypt secrets in memory	HIGH

Unit Testing Standards

ID	Name	Description	Severity
CS-TEST-001	Test Naming Convention	Use clear test names: MethodName_StateUnderTest_ExpectedBeha...	MEDI
CS-TEST-002	Single Assertion	Prefer one logical assertion per test	LOW
CS-TEST-003	No External Dependencies	No dependency on external systems in unit tests	HIGH
CS-TEST-004	Arrange-Act-Assert Pattern	Tests should follow Arrange-Act-Assert pattern	LOW
CS-TEST-005	Test Class Naming	Test classes should be named {ClassName}Tests	LOW

CORS Configuration

ID	Name	Description	Severity
API-CORS-001	Specific CORS Origins	Configure CORS with specific allowed origins, not AllowAnyOr...	CRIT
API-CORS-002	No Credentials with Any O...	AllowCredentials cannot be used with AllowAnyOrigin	CRIT

API Design

ID	Name	Description	Severity
API-REST-001	RESTful Endpoints	Use proper HTTP methods for CRUD operations	HIGH
API-HTTP-001	Appropriate Status Codes	Return appropriate HTTP status codes	HIGH
API-VER-001	API Versioning	Implement API versioning in routes	HIGH
API-RESP-001	Consistent Response Format	Use a consistent API response wrapper	MEDI
API-DOC-001	Endpoint Documentation	Document API endpoints with XML comments and response types	MEDI

Encryption

ID	Name	Description	Severity
API-ENC-001	Proper RSA Encryption	Use proper RSA encryption with OAEP padding	CRIT
API-ENC-002	Strong Hashing Algorithms	Use SHA-256 or stronger for hashing	CRIT

Idempotency

ID	Name	Description	Severity
API-IDEMP-001	Idempotency Keys	Use idempotency keys for financial operations	CRIT

Authentication & Authorization

ID	Name	Description	Severity
API-AUTH-001	Endpoint Authorization	Protect endpoints with proper authorization	CRIT
API-AUTH-002	Resource-Level Authorization	Verify user has access to specific resources	CRIT

Error Handling

ID	Name	Description	Severity
API-ERR-001	Domain Exception Handling	Handle domain-specific exceptions with appropriate responses	HIGH
API-SAN-001	Input Sanitization	Sanitize all user inputs before processing	CRIT

Rate Limiting

ID	Name	Description	Severity
API-RATE-001	Rate Limiting	Implement rate limiting on API endpoints	HIGH

YAML Configuration Options

Option	Type	Description
id	string	Unique identifier for the rule
name	string	Human-readable rule name
description	string	Detailed description of what the rule enforces
severity	critical|high|medium|low	Severity level of violations
enabled	true|false	Whether the rule is active
skip	true|false	Skip this rule during scanning
pattern	regex	Regex pattern for matching valid code
anti_pattern	regex	Regex pattern that indicates violations
applies_to	list	Code elements this rule applies to
file_pattern	glob	Glob pattern for files to scan
custom_validator	string	Name of custom validation function
prebuilt	string	Reference to prebuilt policy template
message	string	Custom message on violation
fix_hint	string	Suggestion for fixing the violation
examples.good	string	Example of correct code
examples.bad	string	Example of incorrect code
tags	list	Tags for categorization
parameters	object	Additional parameters for validators

Prebuilt Policy Templates

Template	Description	Includes
dotnet_naming	Standard .NET naming conventions	CS-NAME-001 through CS-NAME-007
security_essentials	Essential security rules for financial apps	CS-SEC-001-005, CS-CFG-001-002, API-AUTH-001-002, API-SAN-001
async_best_practices	Async/await best practices	CS-ASYNC-001 through CS-ASYNC-003
api_security	API security standards	API-CORS-001, API-VAL-001, API-ENC-001, API-AUTH-001-002, API-RATE-0

Custom Validators

Validator	Description
ValidatePasswordComplexity	Validates password meets complexity requirements
ValidateAccountNumber	Validates account number format
ScanForSecrets	Scans for potential secrets in code
DetectSqlInjection	Detects potential SQL injection vulnerabilities
DetectSensitiveLogging	Detects sensitive data in log statements
ValidateSingleClassPerFile	Validates one class per file
ValidateMethodLength	Validates method length constraints
ValidateCyclomaticComplexity	Validates cyclomatic complexity

Complete Policy ID List

CS-NAME-001: Class/Interface Nami
CS-NAME-002: Method Naming
CS-NAME-003: Variable/Field/Param
CS-NAME-004: Constant Naming
CS-NAME-005: Private Field Naming
CS-NAME-006: Async Method Naming
CS-NAME-007: Interface Naming
CS-NAME-008: Request/Response DTO
CS-NAME-009: Boolean Variable Nam
CS-NAME-010: Event Handler Naming
CS-FILE-001: One Class Per File
CS-FILE-002: File Name Matches Cl
CS-FILE-003: Feature-Based Organi
CS-FILE-004: Namespace Matches Fo
CS-METHOD-001: Single Responsibilit
CS-METHOD-002: Method Length
CS-METHOD-003: Parameter Count
CS-METHOD-004: Cyclomatic Complexit
CS-METHOD-005: No Nested Ternary
CS-SEC-001: Parameterized Querie
CS-SEC-002: Input Validation
CS-SEC-003: Log Sanitization
CS-SEC-004: Secret Protection
CS-SEC-005: Configuration Securi
CS-SEC-006: No Hardcoded Credent
CS-SEC-007: XSS Prevention
CS-SEC-008: Path Traversal Preve
CS-EXC-001: Meaningful Exception
CS-EXC-002: Contextual Logging
CS-EXC-003: No Empty Catch Block
CS-EXC-004: Domain Exceptions
CS-EXC-005: No Catch-All Without
CS-ASYNC-001: Always Await
CS-ASYNC-002: No Blocking Async
CS-ASYNC-003: ConfigureAwait in Li
CS-ASYNC-004: Async Void Avoidance
CS-ASYNC-005: Proper Cancellation
CS-DI-001: Depend on Abstractio
CS-DI-002: Use IoC Container
CS-DI-003: Avoid New in Busines
CS-DI-004: Constructor Injectio
CS-DI-005: Service Lifetime Con
CS-CONST-001: No Magic Numbers
CS-CONST-002: Use Named Constants
CS-CONST-003: No Magic Strings
CS-VAL-001: DTO Validation
CS-VAL-002: Client and Server Va
CS-VAL-003: Null Checks
CS-VAL-004: Guard Clauses
CS-LOG-001: Structured Logging
CS-LOG-002: No Sensitive Data in
CS-LOG-003: Appropriate Log Leve
CS-LOG-004: Include Correlation
CS-DOC-001: XML Comments for Pub
CS-DOC-002: Comment Why Not What
CS-DOC-003: TODO Comments
CS-IMM-001: Use Readonly
CS-IMM-002: No Mutable Collectio
CS-IMM-003: Clone External Data
CS-IMM-004: Use Records for DTOs
CS-CFG-001: No Secrets in Source
CS-CFG-002: Use Secret Managers
CS-CFG-003: Secure Connection St
CS-STR-001: No Plain Text Secret
CS-STR-002: Use SecureString
CS-TEST-001: Test Naming Conventi
CS-TEST-002: Single Assertion
CS-TEST-003: No External Dependen
CS-TEST-004: Arrange-Act-Assert P
CS-TEST-005: Test Class Naming
API-CORS-001: Specific CORS Origin
API-CORS-002: No Credentials with
API-REST-001: RESTful Endpoints
API-HTTP-001: Appropriate Status C
API-VER-001: API Versioning
API-RESP-001: Consistent Response
API-DOC-001: Endpoint Documentati
API-ENC-001: Proper RSA Encryptio
API-ENC-002: Strong Hashing Algor
API-IDEMP-001: Idempotency Keys
API-AUTH-001: Endpoint Authorizati
API-AUTH-002: Resource-Level Autho
API-ERR-001: Domain Exception Han
API-SAN-001: Input Sanitization
API-RATE-001: Rate Limiting