

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#**
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML



C# static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C# code

- All rules 409
- Vulnerability 34
- Bug 76
- Security Hotspot 28
- Code Smell 271
- Quick Fix 52

Tags ▾

Search by name... 🔍

| |
|---|
| Sections of code should not be commented out |
| Code Smell |
| Unused method parameters should be removed |
| Code Smell |
| Empty arrays and collections should be returned instead of null |
| Code Smell |
| Unused private types or members should be removed |
| Code Smell |
| Track uses of "FIXME" tags |
| Code Smell |
| "Obsolete" attributes should include explanations |
| Code Smell |
| Assignments should not be made from within sub-expressions |
| Code Smell |
| General exceptions should never be thrown |
| Code Smell |
| Utility classes should not have public constructors |
| Code Smell |
| Local variables should not shadow class fields |
| Code Smell |
| Redundant pairs of parentheses should be removed |
| Code Smell |
| Inheritance tree of classes should not be too deep |

Deserialization methods should be provided for "OptionalField" members

Analyze your code

Bug

Major

?

serialization

Fields marked with `System.Runtime.Serialization.OptionalFieldAttribute` are serialized just like any other field. But such fields are ignored on deserialization, and retain the default values associated with their types. Therefore, deserialization event handlers should be declared to set such fields during the deserialization process.

This rule raises when at least one field with the `System.Runtime.Serialization.OptionalFieldAttribute` attribute is declared but one (or both) of the following event handlers `System.Runtime.Serialization.OnDeserializingAttribute` or `System.Runtime.Serialization.OnDeserializedAttribute` are not present.

Noncompliant Code Example

```
[Serializable]
public class Foo
{
    [OptionalField(VersionAdded = 2)]
    int optionalField = 5;
}
```

Compliant Solution

```
[Serializable]
public class Foo
{
    [OptionalField(VersionAdded = 2)]
    int optionalField = 5;

    [OnDeserializing]
    void OnDeserializing(StreamingContext context)
    {
        optionalField = 5;
    }

    [OnDeserialized]
    void OnDeserialized(StreamingContext context)
    {
        // Set optionalField if dependent on other deseriali
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

 Code Smell

Nested blocks of code should not be left empty

 Code Smell

Methods should not have too many parameters

 Code Smell

Collapsible "if" statements should be merged

 Code Smell

OS commands should not be vulnerable to argument injection

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)