

- 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... 🔍

"protected" members

Code Smell

Underscores should be used to make large numbers readable

Code Smell

"ToString()" calls should not be redundant

Code Smell

"==" should not be used when "Equals" is overridden

Code Smell

An abstract class should have both abstract and concrete methods

Code Smell

Multiple variables should not be declared on the same line

Code Smell

Culture should be specified for "string" operations

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell

break statements should not be used except for switch cases

Code Smell

String literals should not be duplicated

Code Smell

Files should contain an empty newline at the end

Code Smell

Unused "using" should be removed

Code Smell

"interface" instances should not be cast to concrete types

Analyze your code

Code Smell Critical ? design

Needing to cast from an interface to a concrete type indicates that something is wrong with the abstractions in use, likely that something is missing from the interface. Instead of casting to a discrete type, the missing functionality should be added to the interface. Otherwise there is a risk of runtime exceptions.

Noncompliant Code Example

```
public interface IMyInterface
{
    void DoStuff();
}

public class MyClass1 : IMyInterface
{
    public int Data { get { return new Random().Next(); } }

    public void DoStuff()
    {
        // TODO...
    }
}

public static class DowncastExampleProgram
{
    static void EntryPoint(IMyInterface interfaceRef)
    {
        MyClass1 class1 = (MyClass1)interfaceRef; // Noncompliant
        int privateData = class1.Data;





        class1 = interfaceRef as MyClass1; // Noncompliant
        if (class1 != null)
        {
            // ...
        }
    }
}
```

Exceptions

Casting to object doesn't raise an issue, because it can never fail.

```
static void EntryPoint(IMyInterface interfaceRef)
{
    var o = (object)interfaceRef;
    ...
}
```

Available In:

A close curly brace should be located at the beginning of a line
 Code Smell
Tabulation characters should not be used
 Code Smell
Methods and properties should be named in PascalCase
 Code Smell
Track uses of in-source issue suppressions
 Code Smell