

- 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... 🔍

Inappropriate casts should not be made

Code Smell

Constructors should only call non-overrideable methods

Code Smell

"GC.Collect" should not be called

Code Smell

Methods should not be empty

Code Smell

Exceptions should not be thrown in finally blocks

Code Smell

Method overrides should not change parameter defaults

Code Smell

Types allowed to be deserialized should be restricted

Vulnerability

Server-side requests should not be vulnerable to forging attacks

Vulnerability

Members should not have conflicting transparency annotations

Vulnerability

"PartCreationPolicyAttribute" should be used with "ExportAttribute"

Bug

"ConstructorArgument" parameters should exist in constructors

Bug

Windows Forms entry points should be marked with STAThread

Calls to delegate's method "BeginInvoke" should be paired with calls to "EndInvoke"

Analyze your code

Bug Critical ?

Calling the `BeginInvoke` method of a delegate will allocate some resources that are only freed-up when `EndInvoke` is called. This is why you should always pair `BeginInvoke` with an `EndInvoke` to complete your asynchronous call.

This rule raises an issue when:

- the `BeginInvoke` method is called without any callback and it is not paired with a call to `EndInvoke` in the same block.
- a callback with a single parameter of type `IAsyncResult` doesn't contain a call to `EndInvoke`.

Noncompliant Code Example

BeginInvoke without callback

```
public delegate string AsyncMethodCaller();

public static void Main()
{
    AsyncExample asyncExample = new AsyncExample();
    AsyncMethodCaller caller = new AsyncMethodCaller(asyncEx

    // Initiate the asynchronous call.
    IAsyncResult result = caller.BeginInvoke(null, null); //
}
```

BeginInvoke with callback

```
public delegate string AsyncMethodCaller();

public static void Main()
{
    AsyncExample asyncExample = new AsyncExample();
    AsyncMethodCaller caller = new AsyncMethodCaller(asyncEx





    IAsyncResult result = caller.BeginInvoke(
        new AsyncCallback((IAsyncResult ar) => {}),
        null); // Noncompliant - not paired with EndInvoke
}
```

Compliant Solution

BeginInvoke without callback

```
public delegate string AsyncMethodCaller();

public static void Main()
{
    AsyncExample asyncExample = new AsyncExample();
```

 Bug
Collection elements should not be replaced unconditionally
 Bug
Exceptions should not be created without being thrown
 Bug
Collection sizes and array length comparisons should make sense
 Bug
Serialization event handlers should be implemented correctly

```
AsyncMethodCaller caller = new AsyncMethodCaller(asyncEx

IAsyncResult result = caller.BeginInvoke(null, null);

string returnValue = caller.EndInvoke(result);

}
```

BeginInvoke with callback

```
public delegate string AsyncMethodCaller();

public static void Main()
{
    AsyncExample asyncExample = new AsyncExample();
    AsyncMethodCaller caller = new AsyncMethodCaller(asyncEx

    IAsyncResult result = caller.BeginInvoke(
        new AsyncCallback((IAsyncResult ar) =>
        {
            // Call EndInvoke to retrieve the results.
            string returnValue = caller.EndInvoke(ar);
        })), null);
}
```

See

[Calling Synchronous Methods Asynchronously](#)

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 