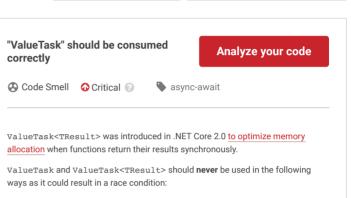




"=+" should not be used instead of

"+="

₩ Bug



- Calling await multiple times on a ValueTask / ValueTask<TResult>*. The wrapped object may have been reused by another operation. This differs from Task / Task<TResult>, on which you can await multiple times and always get the same result.
- Calling await concurrently on a ValueTask / ValueTask<TResult>*. The underlying object is not thread safe. What's more, it has the same effect as awaiting multiple times a ValueTask / ValueTask < TResult >. This again differs from Task / Task<TResult>, which support concurrent await.
- Using .Result or .GetAwaiter().GetResult() without checking if the operation completed*. IValueTaskSource / IValueTaskSource<TResult> implementations are not required to block until the operation completes. On the other hand, Task / Task<TResult> blocks the call until the task completes

It is recommended to use ValueTask / ValueTask<TResult> either by calling "await" on the function returning it, optionally calling ${\tt ConfigureAwait(false)}$ on it, or by calling .AsTask() on it.

This rule raises an issue when the following operations are performed on a ValueTask / ValueTask<TResult>instance:

- · Awaiting the instance multiple times
- · Calling AsTask multiple times.
- Using .Result or .GetAwaiter().GetResult() multiple times
- Using .Result or .GetAwaiter().GetResult() when the operation has not yet completed
- . Using more than one of these ways to consume the instance.

Noncompliant Code Example

```
ValueTask<int> vt = SomeValueTaskReturningMethodAsync();
int result = await vt:
int result2 = await vt; // Noncompliant, variable is awaited
int value = SomeValueTaskReturningMethodAsync().GetAwaiter()
```

Compliant Solution

```
int result = await SomeValueTaskReturningMethodAsync();
int result = await SomeValueTaskReturningMethodAsync().Confi
Task<int> t = SomeValueTaskReturningMethodAsync().AsTask();
```

"NaN" should not be used in comparisons

Bug

Conditionally executed code should be reachable

📆 Bug

Null pointers should not be dereferenced

📆 Bug

For-loop conditions should be true at least once

🖟 Bug

Exceptions

This rule does not raise any issue when a ValueTask / ValueTask<TResult> is awaited multiple time in a loop.

- ValueTask<TResult> official documentation
- Understanding the Whys, Whats, and Whens of ValueTask

Available In:

sonarlint ⊕ | sonarcloud & | sonarqube

© 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