Log Task

The task of creating a small log component has been given to you. The consultant that has been doing the task is not in the company any more.

You are given the code as it is. Your task is to finish the component, fix any bugs and give it a good architecture.

Log Component

The component is a small library that can do asynchronous writing of strings to a file.

Demands

1: A call to Write on the interface must be as fast as possible so the calling application can get on with its work without waiting for the log to be written to a file.

2: If we cross midnight a new file with new timestamp must be made

3: It must be possible to stop the component in two ways:

- Ask it to stop right away and if any outstanding logs they are not written
- Ask it to stop by wait for it to finish writing outstanding logs if any

4: If an error occur the calling application must not be put down. It is more important that the application continues to run than lines not being written to log

Tests

The following must be proven by unit/integration tests:

1: A call to Ilog will end up in writing something

2: New files are created if midnight is crossed

3: The stop behavior is working as described above

Projects

LogComponent.csproj

This is the project containing all the logic around logging things.

The ILog is the public interface

Application.csproj

This project is the application using the LogComponent.

Your task

Your task is to make the log component do what is specified in the Demands section and create the mentioned unit/integration tests to prove it works.

Things to do

- Make the design more SOLID
- Make the code more readable/maintainable
- Find and fix bugs
- Create unit/integration tests

There are no limits in what you can do with the code. You can create new classes, new interfaces, use other technologies within .Net as long as the LogComponent will do as specified.

Regarding unit/integration test: You are welcome to use any test framework of your own choice, or just do tests without a backing framework.

If you lack technical skills within any part of the task, you can write a textual description of what you would have done, and why.

Work outcome

- Rearranged code and unit/integration tests. If task is too big to do within the available time please do as much as possible and describe the missing steps in text, pseudo code etc.
- The log files, written as they are now. Two log-files, one with numbers going from 50 down to something – when the component is stopped without flush. One file having logs with numbers going from 0 to 14
- A few words on what is fixed, refactored and why the code ended up looking as it does in the end