Inhaltsverzeichnis

NetLogWin – Compact Logging with User Interaction

Introduction

NetLogWin is a modular logging framework for **.NET Framework 4.8** (and optionally **.NET 8+**) that focuses on the essentials: structured logging to files and simple user interaction in both console and Windows applications.

Unlike many bloated logging frameworks, NetLogWin allows you to:

- **Write log files quickly, targeted, and with multiple levels** (Info, Warning, Error, Success, etc.).
- **Ask the user questions** (e.g., Yes/No/Cancel) – directly within the logging call.
- **Easily swap output targets** (Console, GUI) using clear interfaces.

NetLogWin Architecture Overview

NetLogWin is a lean, powerful logging framework for .NET 4.8 and .NET 8 Windows applications. It combines file logging with direct user interaction – setting it apart from larger frameworks.

Packages and Responsibilities

Packagename	Typ	Description
HostApp_exe	Executable	The application's startup class (MainApplication) and initializer (LogInitializer).
NetLogger48_dll	Core-DLL	Contains the main logging classes: LogWriter, LogCore, LogFile-Sinks, and the Display-Policy.
LogEnum48_dll	Enum-DLL	Common Enums (LogLevel, UserResponse, ResponseButtons, LogFileRotation) for maximum decoupling.
IODisplay48Win_dll	Display-DLL	Windows GUI-based implementation of the display plugin (dialogs, MessageBox, etc.).
IODisplay48Con_dll	Display-DLL	Console-based implementation of the display plugin (color output, user input in the console).

Architectural Principles

- **Modularity:** Clear separation of core logging logic (`NetLogger48_dll`) from display implementations (`IODisplay48Win_dll`/`IODisplay48Con_dll`).
- **Decoupling:** Common data types in `LogEnum48_dll` minimize dependencies between DLLs.
- **Flexibility:** Interchangeable display plugins enable use in various application types.
- **Combination of Logging & User Interaction:** The integration of user queries into the logging process in a single call is unique.
- **Lightweight:** Compact and clear, without unnecessary ballast – ideal for small to medium-sized applications.

Log Process Flow

1. The application calls **`LogWriter`**, e.g., `LogWriter.Information("...")`.

- `LogWriter` passes the message to **`LogCore`**.
- `LogCore` writes to log files (FileLogSink, FileDebugLogSink) depending on the **`LogLevel`**.

- A display callback method (e.g., `IODisplayServiceWin.ShowMessage`) is called via the configured **`ILogDisplayPolicy`**.
- User interaction (e.g., a Yes/No question) is processed directly and returns values.
- The result is returned to the application.

This documentation provides developers with clear guidance on how to use, extend, and maintain NetLogWin.

Installation & Integration

Dependencies

- **.NET Framework 4.8** (Optional adaptations for .NET 6/8 are available)

Integrate DLLs

- Reference the following DLLs in your project:

- **`NetLogger48.dll`** – contains the logging core
- **`LogEnum48.dll`** – central enums for logging and display
- One of the following display modules:
  - **`IODisplay48Con.dll`** (for console applications)
  - **`IODisplay48Win.dll`** (for Windows Forms)

Initialize the Logger

Initialize the logger in your program startup (minimal instructions):

```
Public Shared Sub InitializeLogging(displayMessageCallback As
DisplayExecutionCallback, Optional diagnosticWriterCallback As
DiagnosticLogWriterCallback = Nothing)
 ' === REQUIRED ===
 ' Assignment of the logic policy (display) No diagnosticWriterCallback passed
LogCore.ActiveDisplayPolicy = New DefaultLogDisplayPolicy(
New DisplayExecutionCallback(AddressOf IODisplayServiceCon.ShowMessage))
End Sub
```

Use Logging Functions

```
LogWriter.Information("Programm gestartet")
Dim answer As UserResponse = LogWriter.Question("Backup löschen?",
ResponseButtons.YesNo)
```

Configuration (purely optional, directly in code)

The following options can be configured via code – no XML or `app.config` files are needed:

- **Minimum log level** for file output (`LogCore.MinimumFileLogLevel`)
- **Logfile rotation** (e.g., hourly, daily, etc.) via `LogCore.LogFileRotation`
- **Maximum size** of individual log files (`FileLogSink.MaxLogFileSizeMB`)
- **Control of display and interaction logic** via custom `ILogDisplayPolicy` implementations

Configuration is **not required** – you can start logging productively with just a few lines of code.

FAQ – Frequently Asked Questions

Why not just `Console.WriteLine` or `MessageBox.Show`?

Because NetLogWin unifies both, logs in a structured way, and still allows asking the user questions – in the same call.

Example:

```
Dim result = LogWriter.Question("Datei wirklich löschen?", ResponseButtons.YesNo)
If result = UserResponse.Yes Then …
```

Do I absolutely have to use a display module?

No. A display module (`IODisplay...`) is only required if you want to display user interaction such as questions, warnings, or hints in the console or GUI.
For pure file logging (e.g., error logging), `NetLogger48.dll` alone is sufficient.

If you want to use your own display logic, you can write your own policy that implements the `ILogDisplayPolicy` interface.

Can I disable logging or only capture debug messages?

Yes. The internal `LogCore` class controls behavior via `LogLevel` filters and rotation.

Is NetLogWin suitable for large projects?

NetLogWin is designed for compact tools and lean desktop apps – it does not replace enterprise logging systems, but it can be easily extended.

What happens with logging in background threads?

All `LogWriter` calls are **thread-safe**. User interactions occur within the call context (no hidden `Invoke` or UI magic).

Are other user interactions beyond (Y/N) possible?

Yes. The `LogEnum48.dll` defines various response models via the `ResponseButtons` enum – e.g., (Yes/No/Cancel), (OK/Retry), and many more.

A complete overview can be found in the Sandcastle documentation.

How does automatic log file rotation work?

Log rotation is controlled via the **`LogFileRotation`** enum in `LogEnum48.dll`.

The following modes are available:

- **`SingleFile`**: The same file is always written to.
- **`Hourly`**: A new log file is created every hour.
- **`Daily`**: A new file daily (default).
- **`Weekly`**: Weekly rotation.
- **`Monthly`**: One file per calendar month.

Selection is done directly in code – without XML or AppSettings.

The generated file name automatically includes a time prefix (_H, _D, _W, _M) for clear assignment.

Important: Regardless of the chosen rotation, the **`MaxLogFileSize`** limit also applies. If the maximum size is exceeded, a new file is automatically started – with an incremented suffix (`_01`, `_02`, ...).

Are settings also possible via XML or JSON files?

In the current version: **no**.

NetLogWin was deliberately designed so that all configuration values are set directly in the source code – including `LogLevel`, rotation, display behavior (Policies), and file sizes.

The goal: A lean logging framework for .NET applications with maximum control without external dependencies. No effort with configuration parsers, file monitoring, validation logic, or path resolution – everything is in the hands of the developer.

Planned for a potentially upcoming version

An optional configuration plugin (e.g., based on XML or JSON) is planned for future releases.

Important: Its use will remain completely voluntary. Developers can decide whether to continue working purely via code or integrate configuration files.

Advantages of the planned extension

- Ideal for larger or modular projects
- Allows easy adjustment of logging behavior without rebuild

Trade-off: Flexibility vs. Minimalism

Aspekt	Current Solution (Code-based)	Future Option (Configurable)
Setup Complexity	Very low	Slightly higher (parser + file needed)
Dependencies	Nothing	Potentially needed (plugin)
Modifiable without Rebuild	No	Yes
Maintainability in Larger Projects	Manual in code	More structured via external files
Security against the User	Very Good	Rather low

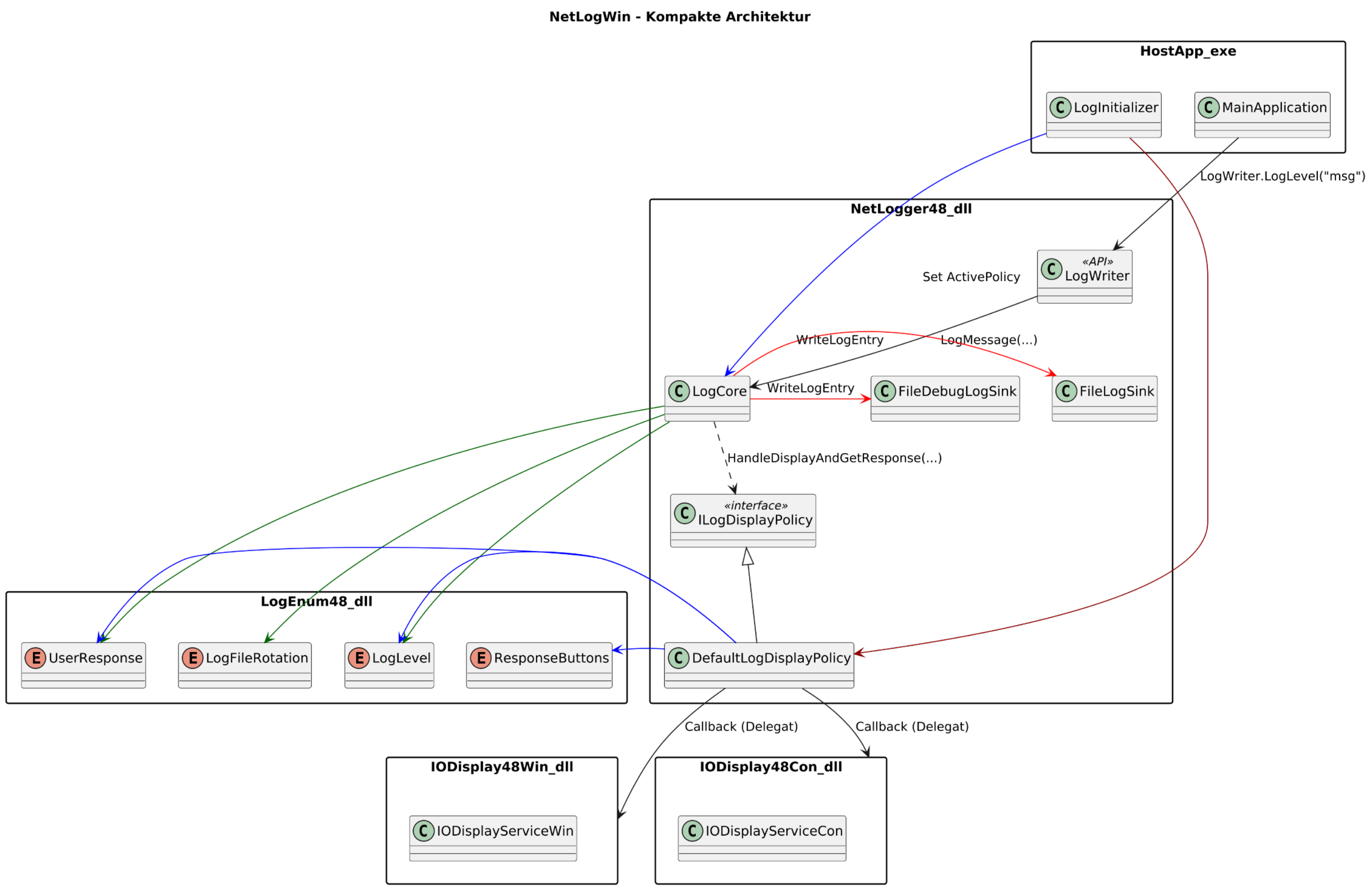
Security with planned extensions

Through configuration files, users – intentionally or unintentionally – can manipulate settings (e.g., activate debug logs, change paths, or influence display behavior).

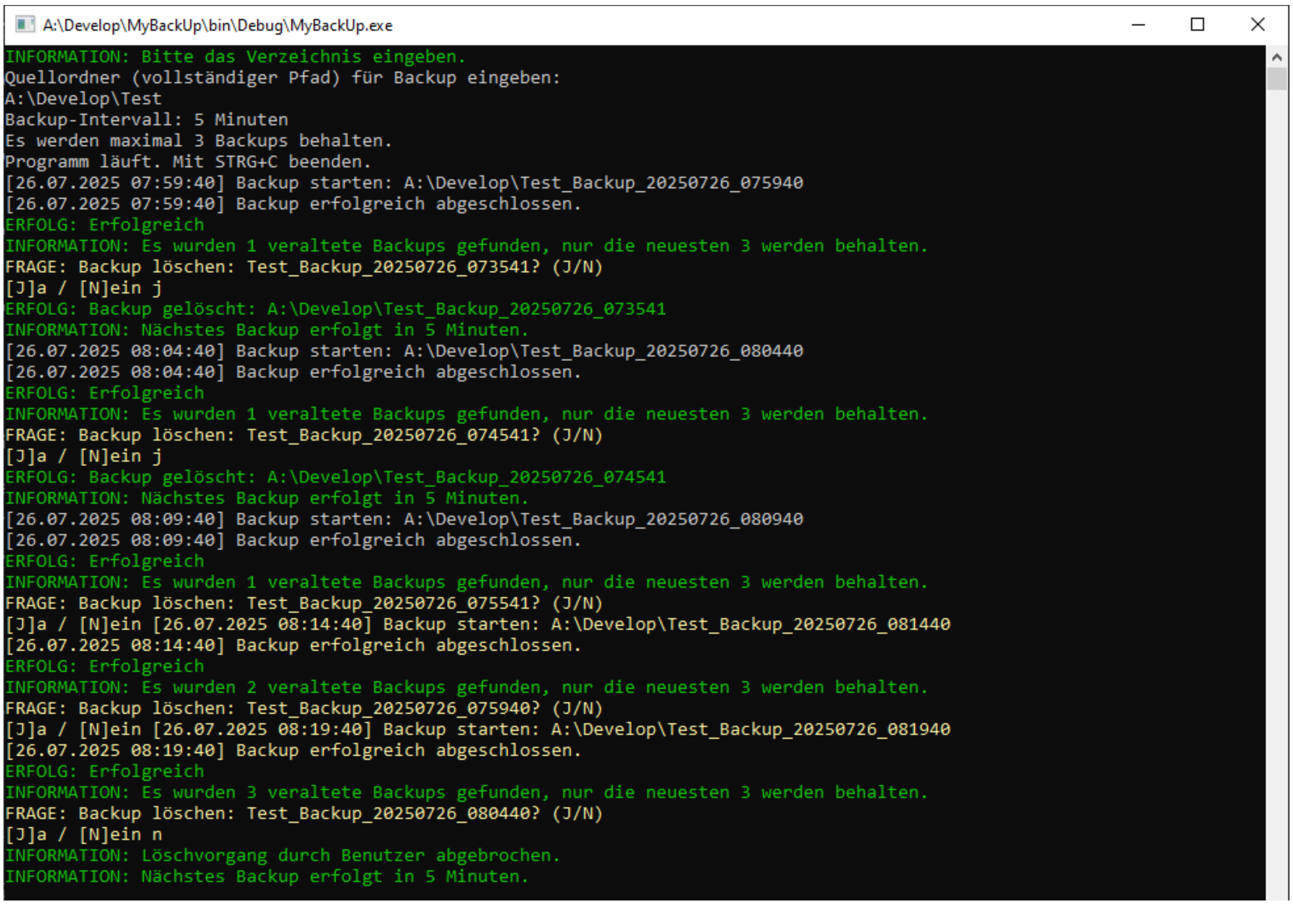
In the purely code-based variant, this is **precluded**, which significantly increases the application's protection – especially in security-critical areas or closed systems.

Images / UML

UML of DLL`s, Classes and Interface



Screen Consolen-App



```
INFORMATION: Bitte das Verzeichnis eingeben.
Quellordner (vollständiger Pfad) für Backup eingeben:
A:\Develop\Test
Backup-Intervall: 5 Minuten
Es werden maximal 3 Backups behalten.
Programm läuft. Mit STRG+C beenden.
[26.07.2025 07:59:40] Backup starten: A:\Develop\Test_Backup_20250726_075940
[26.07.2025 07:59:40] Backup erfolgreich abgeschlossen.
ERFOLG: Erfolgreich
INFORMATION: Es wurden 1 veraltete Backups gefunden, nur die neuesten 3 werden behalten.
FRAGE: Backup löschen: Test_Backup_20250726_073541? (J/N)
[J]a / [N]ein j
ERFOLG: Backup gelöscht: A:\Develop\Test_Backup_20250726_073541
INFORMATION: Nächstes Backup erfolgt in 5 Minuten.
[26.07.2025 08:04:40] Backup starten: A:\Develop\Test_Backup_20250726_080440
[26.07.2025 08:04:40] Backup erfolgreich abgeschlossen.
ERFOLG: Erfolgreich
INFORMATION: Es wurden 1 veraltete Backups gefunden, nur die neuesten 3 werden behalten.
FRAGE: Backup löschen: Test_Backup_20250726_074541? (J/N)
[J]a / [N]ein j
ERFOLG: Backup gelöscht: A:\Develop\Test_Backup_20250726_074541
INFORMATION: Nächstes Backup erfolgt in 5 Minuten.
[26.07.2025 08:09:40] Backup starten: A:\Develop\Test_Backup_20250726_080940
[26.07.2025 08:09:40] Backup erfolgreich abgeschlossen.
ERFOLG: Erfolgreich
INFORMATION: Es wurden 1 veraltete Backups gefunden, nur die neuesten 3 werden behalten.
FRAGE: Backup löschen: Test_Backup_20250726_075541? (J/N)
[J]a / [N]ein [26.07.2025 08:14:40] Backup starten: A:\Develop\Test_Backup_20250726_081440
[26.07.2025 08:14:40] Backup erfolgreich abgeschlossen.
ERFOLG: Erfolgreich
INFORMATION: Es wurden 2 veraltete Backups gefunden, nur die neuesten 3 werden behalten.
FRAGE: Backup löschen: Test_Backup_20250726_075940? (J/N)
[J]a / [N]ein [26.07.2025 08:19:40] Backup starten: A:\Develop\Test_Backup_20250726_081940
[26.07.2025 08:19:40] Backup erfolgreich abgeschlossen.
ERFOLG: Erfolgreich
INFORMATION: Es wurden 3 veraltete Backups gefunden, nur die neuesten 3 werden behalten.
FRAGE: Backup löschen: Test_Backup_20250726_080440? (J/N)
[J]a / [N]ein n
INFORMATION: Löschvorgang durch Benutzer abgebrochen.
INFORMATION: Nächstes Backup erfolgt in 5 Minuten.
```