

**DOCUMENTATION: DLL**

Summary

[***I.*** ***Introduction:*** 3](#_Toc190612552)

[**II.** **Definition:** 3](#_Toc190612553)

[**III.** **Setting up a daily log DLL for the EasySave project:** 4](#_Toc190612554)

[- Creating our DLL on Visual Studio 2022: 4](#_Toc190612555)

[- Building our DLL: 4](#_Toc190612556)

[- Tree structure and DLL setup 5](#_Toc190612557)

[ Explanation of each **Logger.cs** file: 5](#_Toc190612558)

[- DLL compilation: 9](#_Toc190612559)

[- How do you use the DLL in a Project? 10](#_Toc190612560)

[- Example of DLL use in a project: 12](#_Toc190612561)

[**IV.** **Troubleshooting and common errors:** 12](#_Toc190612562)

[**VI.** **Conclusion:** 12](#_Toc190612563)

1. ***Introduction:***

As part of EasySave's development, a Dynamic Link Library (DLL) was integrated to optimize log management and ensure detailed monitoring of backups.

This DLL enables:

precise monitoring of backups and encryption

Structured log storage (JSON or XML)

Centralized log management

Independence from the rest of the code

Since version 2.0, an encryption time counter has been added to monitor operation performance.

1. **Definition:**

A Dynamic Link Library is a software library whose functions are loaded into memory by a program, as and when required, during execution, as opposed to static or shared software libraries whose functions are loaded into memory before program execution begins.

.dll is a filename extension used by files containing a Dynamic Link Library.

1. **Setting up a daily log DLL for the EasySave project:**

* Creating our DLL on Visual Studio 2022:

First, from the Visual Studio 2022 interface, we need to set up a new project. This new project must be a .NET 8.0 language “Class Library” (the same as the project in which we need to use it). It will be placed wherever we like, and the name will also be up to us.Une image contenant texte, Police, Logiciel multimédia, logiciel

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

* Building our DLL:

In the case of the EasySave application in our project, we were asked to be able to manage the creation of a daily log file of all the job backups executed that day, from a JSON or XML file “C:\ProgramData\CESI\EasySave\Logs\backup\_log\_yyy-mm-dd.JSON /XML”.Une image contenant texte, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

* Tree structure and DLL setup

For a DLL, we need a class with several methods to build it. This will be the basis of the DLL, plus dynamic attributes that will enable us to link it to the project.Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

* Explanation of each **Logger.cs** file:

**1. General presentation**

The EasySave.Logger namespace contains a Logger class that:

- Generates one JSON or XML log file per day in a defined directory.

- Writes each backup operation to this file.

- Supports logging of encryption time in addition to transfer time.

- Ensures security against concurrent access with a lockObj lock.

- Allows choice between JSON and XML depending on user preferences.

Library used:

- Newtonsoft.Json for handling JSON files.

- System.Xml.Linq to manage XML files.

- System.IO for reading and writing files.

**2. Explanation of code elements**

**a) Imports**

The file begins by importing the essential libraries:

- System.IO: Manages reading and writing of log files.

- Newtonsoft.Json: Serializes objects in JSON.

- System.Xml.Linq: handles XML files.

- System.Threading.Tasks: Secures file access in multitasking mode.

**b) Logger class**

The Logger class contains:

- logFilePath: Log file path.

- logFormat: Log format (json or xml).

- lockObj: Lock guaranteeing write access security.

- WriteLog() methods: Used to record backup operations.

- Encryption time management in logs for improved performance monitoring.

**c) Dynamic log file generation**

**Log file creation process:**

1. **The log file is created every day in a dynamic format:**

backup\_log\_YYYY-MM-DD.json

backup\_log\_YYYY-MM-DD.xml

1. **It is stored in a directory defined by an environment variable EASYSAVE\_LOG\_PATH or, by default:**

C:\ProgramData\CESI\EasySave\Logs\

1. **If the directory does not exist, it is automatically created.**
2. **The file is written in JSON or XML, depending on the format chosen.**

**3. Writing logs**

**a) Creating a log object**

Each log entry contains:

- Name : Name of the backup.

- FileSource : Source file path.

- FileTarget : Destination file path.

- FileSize : File size.

- FileTransferTime: Transfer time (in milliseconds).

- EncryptionTime: Encryption time (in milliseconds).

- Date: Date and time of input.

**b) Writing to a JSON file**

**JSON recording process:**

**1. Check file existence:**

o If it doesn't exist, an empty JSON array is created [ ].

o Otherwise, the existing file is opened.

**2. Add a new JSON entry:**

o the old file is opened and modified without breaking the JSON format.

o the new entry is added to the list with a comma separator.

o the file is updated with the new entry.

**Example of a generated JSON entry:**

{

"Name": "Backup1",

"FileSource": "C:\\Documents\\File.txt",

"FileTarget": "D:\\Backup\\File.txt",

"FileSize": 1024,

"FileTransferTime": 1.5,

"EncryptionTime": 0.8,

"Date": "15/03/2025 14:32:10"

}

**c) Writing to an XML file**

**XML saving process:**

1. **Check that the file exists:**

o If it doesn't exist, it is created with a <Logs> element containing the first <LogEntry>.

o Otherwise, it is loaded and updated.

1. **Adding a new <LogEntry> element:**

o Adds a child element under <Logs>.

o the file is updated.

**Example of a generated XML entry:**

<LogEntry>

<Name>Backup1</Name>

<FileSource>C:\Documents\File.txt</FileSource>

<FileTarget>D:\Backup\File.txt</FileTarget>

<FileSize>1024</FileSize>

<FileTransferTime>1.5</FileTransferTime>

<EncryptionTime>0.8</EncryptionTime>

<Date>15/03/2025 14:32:10</Date>

</LogEntry>

* DLL compilation:

After writing the code:

* **Compile** the project from Visual Studio
* **Check** in “bin/Debug/net8.0/”.

You should see a file in the project directory “bin/Debug/net8.0/”

EasySave.Logger.dll

* Then you’ve compiled successfully!

**Possible error:**

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

If this error occurs during compilation, don't panic, it proves that the DLL can only be used in “console application (.Net)” projects. This shows that your DLL is working as it should!

* How do you use the DLL in a Project?

In a project **(e.g. EasySave),** add the DLL:  
**➜ Right-click on “References or the project.sln”**

**➜ Add a “Project reference”.**

**Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.**

➜ **Select "EasySave.Logger.dll"**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

**You would then have the DLL displayed in the project:**Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

In return, it should appear in “bin/Debug/net8.0” of the project you're working on, otherwise you could get this kind of error: Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

In this case, add it manually in this directory and add the DLL again!

* Example of DLL use in a project:

Une image contenant texte, Police, ligne, capture d’écran

Le contenu généré par l’IA peut être incorrect.

1. **Troubleshooting and common errors:**
2. **Problems and solutions**

**Error: "** **DLL cannot be found"**  
🡪 Solution: Check that the DLL has been added as a reference and is present in bin/Debug/net8.0/.

**Error: "** **Method not found"**  
🡪 Solution: Check the signature of the method called in LogManager.

**Error: "** **JSON/XML format invalid"**  
🡪 Solution: Check that the log JSON file (backup\_log\_yyyy-MM-dd.json/xml) is not corrupted.

* 1. **Conclusion:**

Using a Dynamic Link Library (DLL) allows you to:

- Optimize modularity and code reuse.

- Facilitate maintenance and updates.

- Improve EasySave log management.

Thanks to this DLL, the project is more structured, scalable and easy to maintain.