**1) Introduction**

Bister is a powerful C# binary serializer crafted to efficiently serialize and deserialize complex data structures, serving both modern applications and legacy systems with remarkable ease. Unlike other tools like protobuf-net, which insists on defining types in a restrictive proto definition format, or serializers such as MessagePack and ZeroFormatter, which demand attribute annotations on data classes, Bister eliminates these barriers entirely. It offers a runtime-configurable serialization policy, letting developers dynamically control what gets serialized without needing predefined schemas or code changes. This is especially valuable in enterprise applications where legacy types—like ArrayList, System.Enum, System.Object, and System.Exception—are common, and modifying their source code, such as in COM IDLs, is often impossible. Bister supports these types out of the box, giving it a clear advantage over competitors that rely on altering type definitions. By leveraging runtime code generation, it optimizes performance with tailored serialization logic, and its binary format packs data far more efficiently than text-based JSON or XML, reducing size and boosting speed. This makes Bister an ideal choice for developers seeking high performance, flexibility, and seamless integration across diverse C# projects, from cutting-edge microservices to enterprise legacy systems.

**2) Methods**

Bister’s approach begins with a one-time reflection process that analyzes an object’s type and generates a tailored runtime serialization class, which is then reused for all subsequent operations to avoid repeated reflection overhead. During serialization, it encodes objects into a compact binary stream, adding minimal metadata—only null flags for reference types and fully qualified names for System.Object and System.Enum. Developers can configure what properties or fields to serialize at runtime, such as choosing to include only non-null values or exclude specific fields dynamically, all without altering the input data source code. For deserialization, Bister reconstructs objects using the pre-generated runtime class, ensuring legacy type behaviors, like the order of elements in an ArrayList, are preserved accurately. Implemented as a NuGet package for .NET Standard and Core, Bister is thread-safe and offers a simple two-line API for ease of use. Its performance and flexibility were tested on Windows 10 with .NET 8.0, benchmarked against industry-standard serializers to measure speed, size, and adaptability.