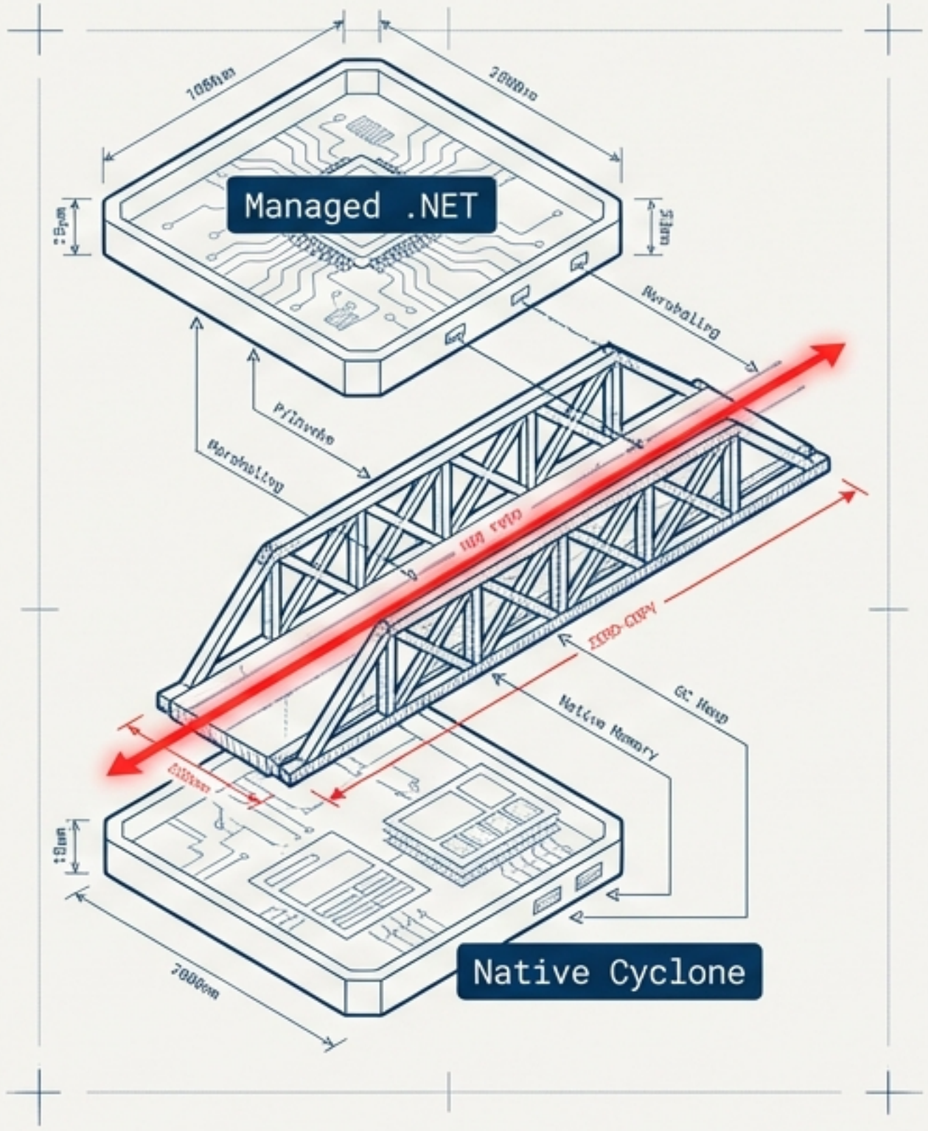
Fast Cyclone DDS C# Bindings

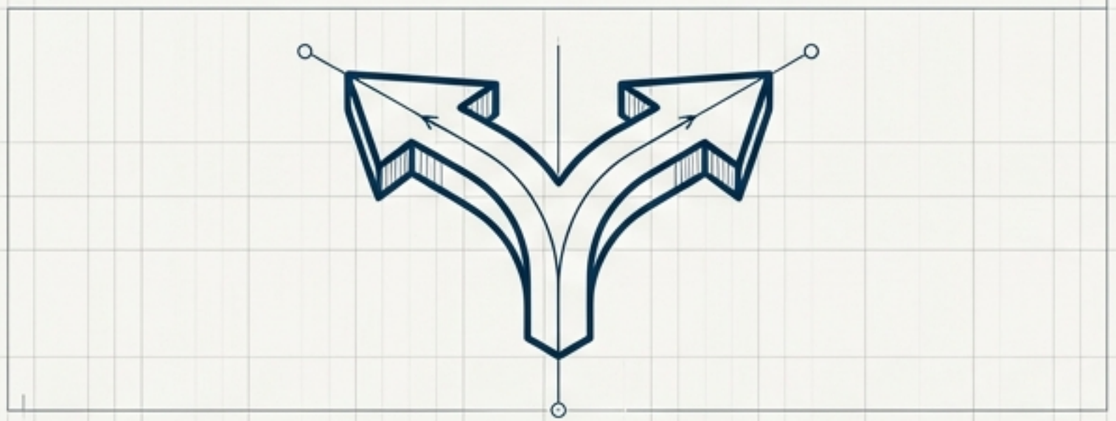
High-Performance .NET Integration for Real-Time Systems

A LIBRARY BRIDGING THE GAP BETWEEN .NET PRODUCTIVITY AND RAW NATIVE SPEED.



The Core Proposition: C# Productivity, C-Like Performance

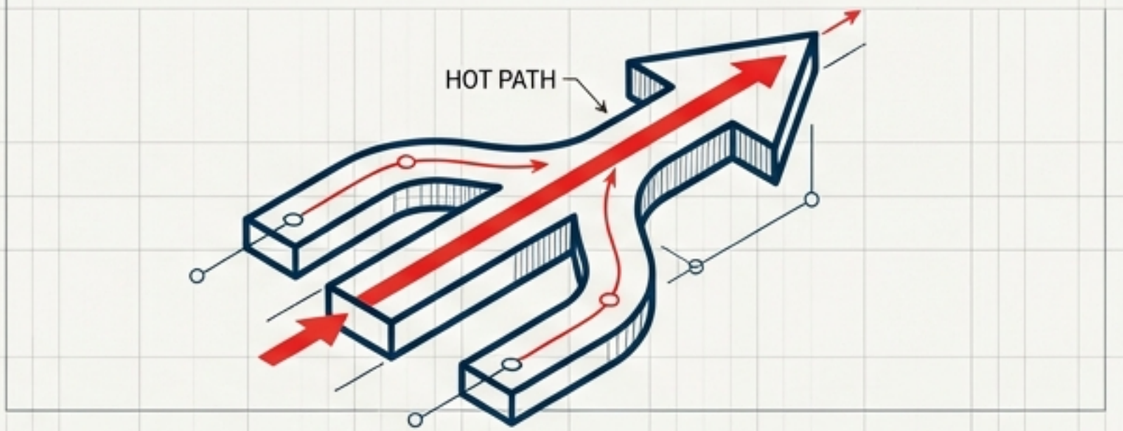
THE HISTORIC TRADE-OFF



Developers of real-time systems have historically faced a binary choice:

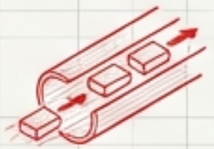
1. **EFFICIENCY:** Write in C/C++ for manual memory management and real-time constraints.
2. **VELOCITY:** Write in C#/.NET for rapid development and safety, but suffer Garbage Collection (GC) pauses.

THE SOLUTION



Fast Cyclone DDS C# Bindings eliminates this trade-off.

1. **ZERO GC OVERHEAD:** Eliminates Garbage Collection allocations in steady-state publishing and receiving.

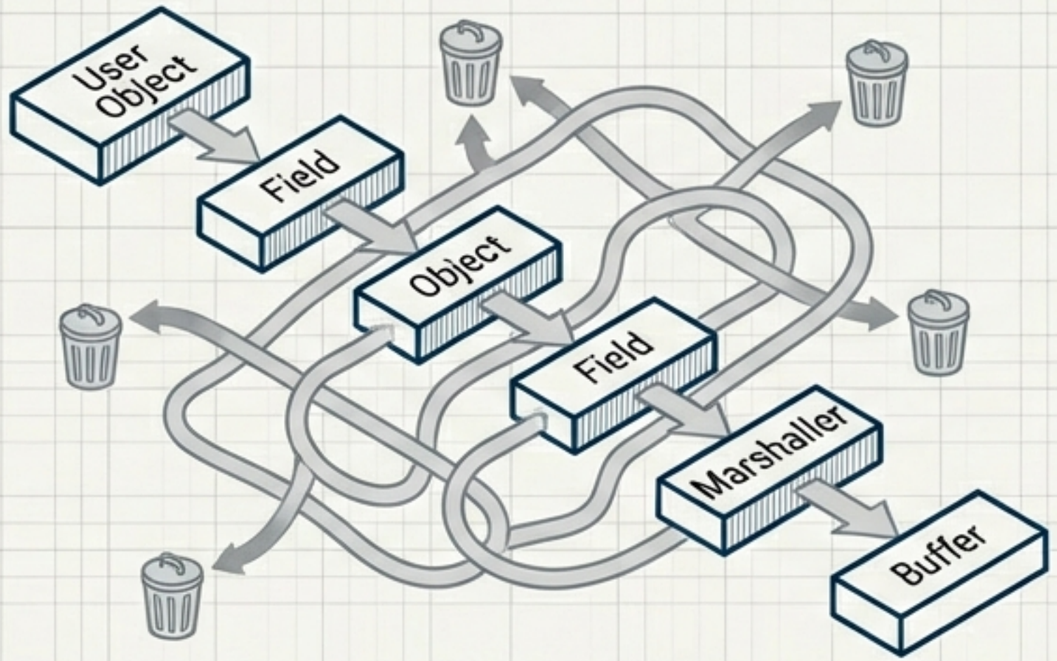


2. **NATIVE SPEED:** Provides a high-performance interface to Eclipse Cyclone DDS using a "Hot Path" architecture.
3. **RESULT:** Applications that run with the speed of C but are written with the safety and speed of .NET.

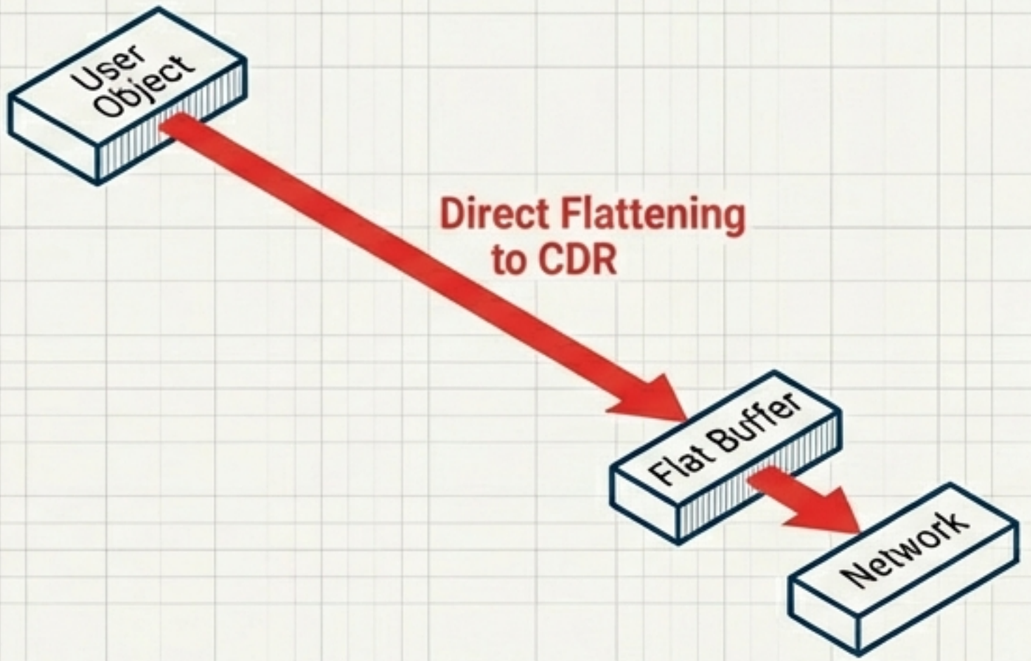
The "Zero-Allocation" Philosophy

MEMORY PATHWAYS

STANDARD MARSHALLING
(High CPU & GC Overhead)



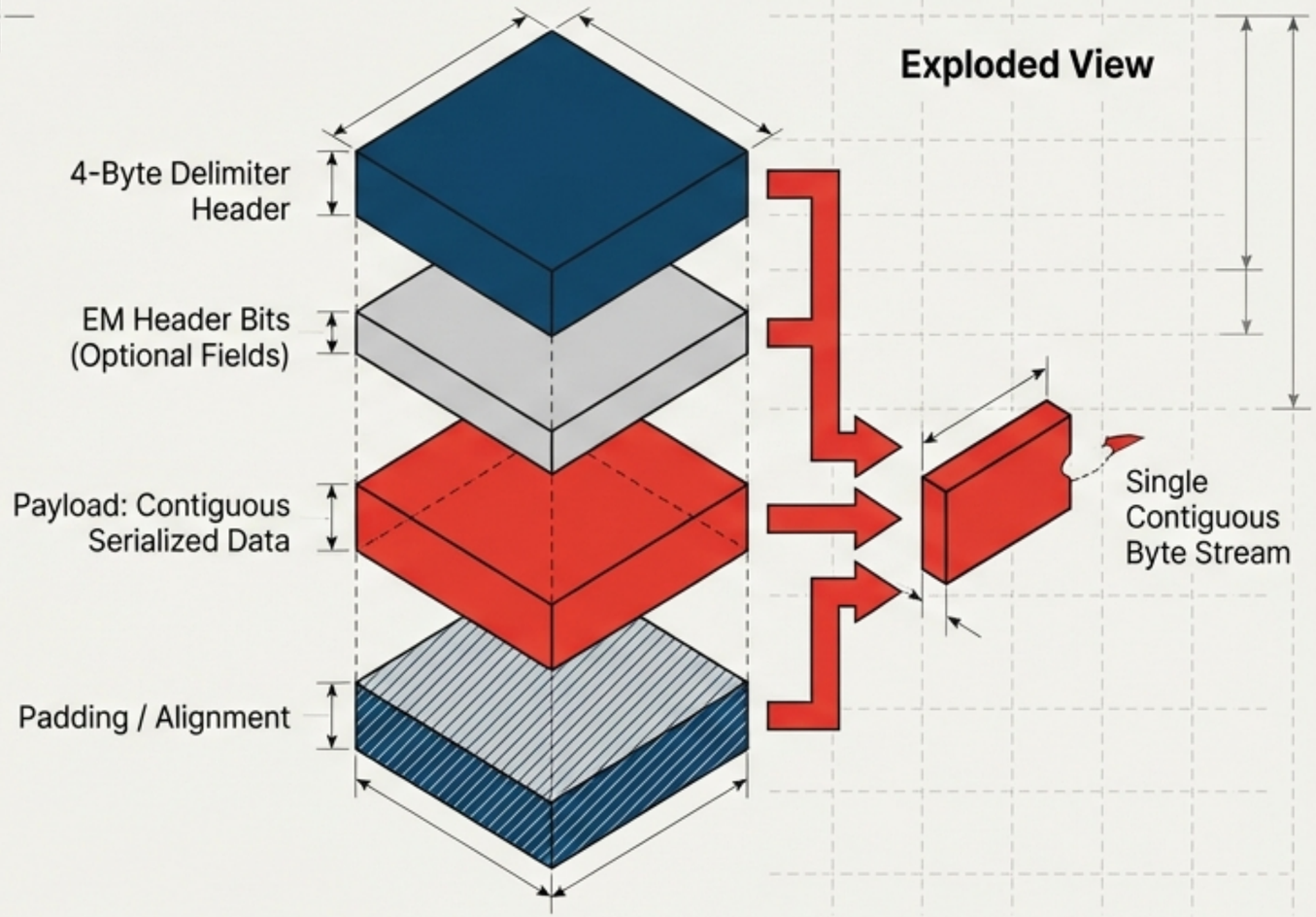
FAST BINDING
(Contiguous Binary / Zero Allocations)



The binding works by flattening samples directly into a contiguous binary representation (CDR), bypassing intermediate C structures or per-field heap objects. Buffers are recycled via pooling.

Advanced Serialization: XCDR2 Appendable

- **Pure C# Implementation**: A custom implementation of the Extended Common Data Representation (XCDR2) standard, written entirely in C#.
- **Source-Generated**: At build time, a code generator emits specific Serialize() / Deserialize() logic. No runtime reflection. No boxing. JIT-optimizable.
- **Extensibility**: Supports “Appendable” types automatically. The serializer handles 4 -byte delimiter headers and optional field bitmaps (EM header bits).
- **Standards Compliant**: Fully conforms to DDS XTypes rules, including Member IDs and consistent ordering.



Exploded View
Data Packet

Defining Topics in C#: The Domain-Specific Language (DSL)

Say goodbye to separate IDL files.

THE OLD WAY (.IDL)

```
struct MyData {
    long key;
    double value;
    string<32> name;
};
```

THE NEW WAY (NATIVE C#)

```
[DdsTopic('MyData')]
public partial struct MyData {
    [DdsKey] public int Key;

    public double Value;

    public FixedString32 Name;
}
```

Define topic directly on struct

Mark unique identifiers via Attributes

Use specialized types for performance

The DSL mirrors IDL capabilities but stays entirely within the familiar C# ecosystem.

Handling Complex Types & Quality of Service

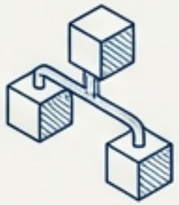
Unions

```
[DdsUnion]
public struct SensorData {
    [DdsDiscriminator] public SensorType Type;
    [DdsCase(SensorType.Temp)] public float Temp;
    [DdsCase(SensorType.GPS)] public GpsCoord Loc;
}
```

Mapped directly to IDL discriminated unions.

[DdsCase(SensorType.Tenp)]
public GpsCoord ::.

Mapped directly to IDL discriminated unions.

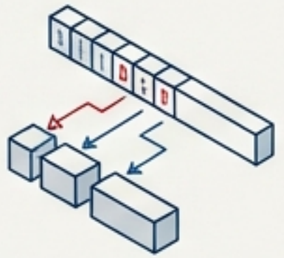


Optional Fields

```
public partial struct Telemetry {
    public int? Altitude; // Nullable value type
    public double? Speed;
}
```

Uses XCDR2 presence flags (bitmaps) on the wire.

Uses XCDR2 presence flags (bitmaps) on the wire.

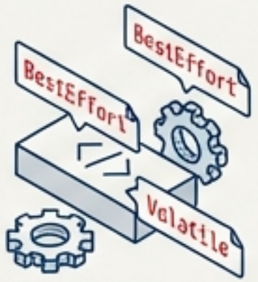


QoS Integration

```
[DdsQos(Reliability = DdsReliability.BestEffort,
        Durability = DdsDurability.Volatile)]
public partial struct CameraFrame { ... }
```

[DdsQos(Reliability = DdsReliability.**BestEffort**,
Durability = DdsDurability.**Volatile**)]

Define policy directly on the type, keeping configuration close to the code.



Swiss Engineering Blueprint

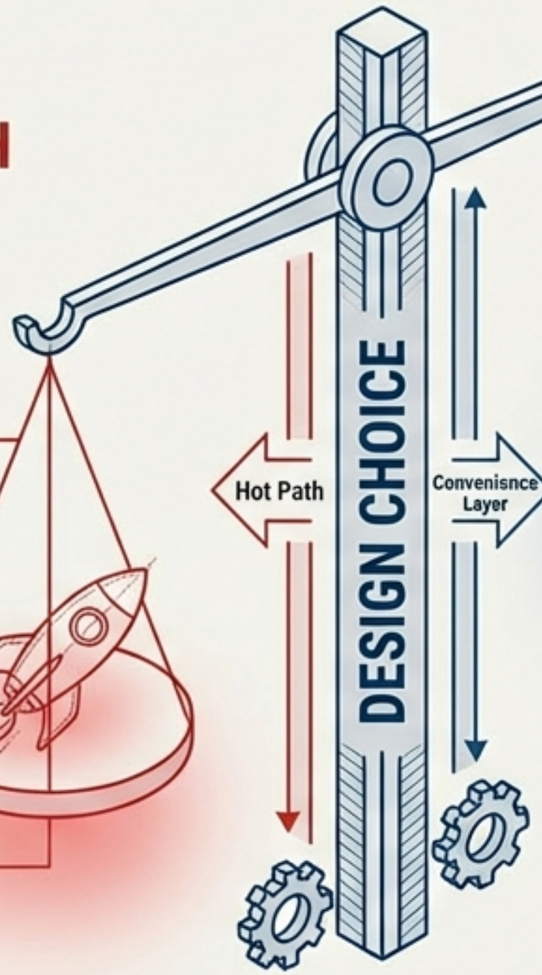
The Hybrid Model: Managed vs. Unmanaged

THE HOT PATH (Unmanaged)

Blittable Structs, FixedString32, Fixed Arrays

0% GC Overhead

High-frequency telemetry, robotics control loops.



THE CONVENIENCE LAYER (Managed)

Standard string, List<T>, Classes.

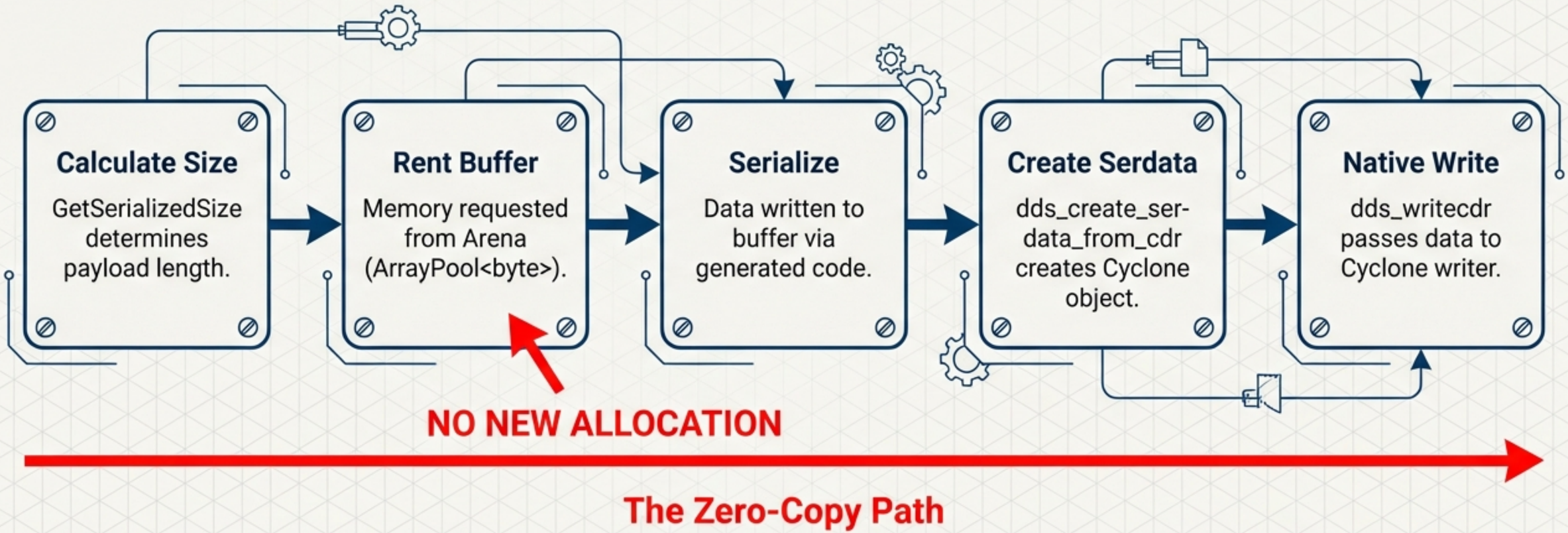
Must be explicitly marked with [DdsManaged].

Incurs standard GC allocation costs.

Configuration data, low-frequency events, UI integration.

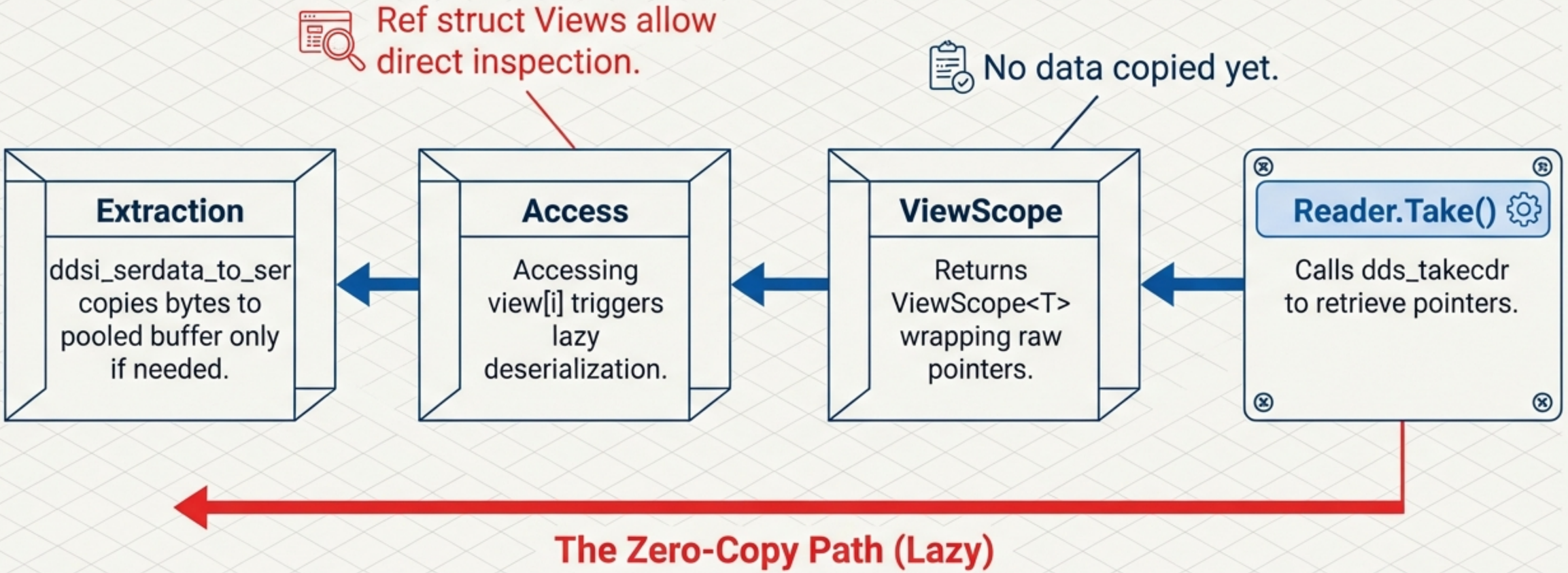
Swiss Engineering Blueprint

Architectural Deep Dive: The Write Path



Swiss Engineering Blueprint

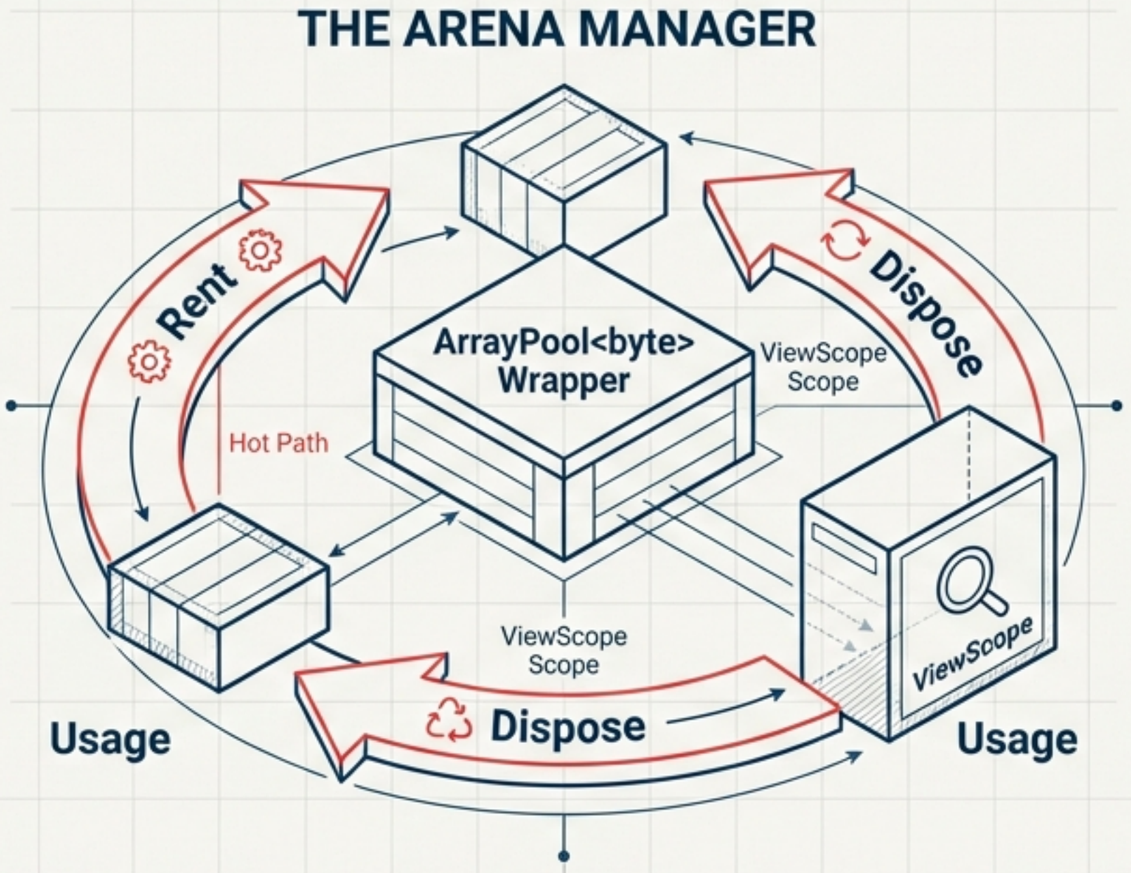
Architectural Deep Dive: The Read Path



Dispose() returns the loan to Cyclone, decrementing references to serdata objects.

Memory Management Strategy

Lifecycle Management: ViewScope ensures buffers cannot be persisted beyond valid lifetime (enforced by compiler ref struct).



Zero Garbage: In steady-state, even variable-size data produces no garbage for GC.

Golden Tests: Byte-for-byte verification ensures custom memory layout matches Cyclone native expectations.

Code Walkthrough: Defining the Data

```
[DdsTopic('ExampleTopic')]
public partial struct ExampleData {
    [DdsKey] public int Id;

    // Unmanaged: Zero allocation
    public FixedString32 Tag;

    // Managed: Dynamic sequence
    [DdsManaged]
    public List<int>? Numbers;
}
```

Key Field Definition.

Unmanaged Type. 32-byte UTF-8 buffer. No Allocations.

Managed Type. Dynamic sequence. Opt-in via [DdsManaged].

Swiss Engineering Blueprint

Code Walkthrough: Publish & Subscribe

PUBLISH (WRITE)

```
var sample = new ExampleData {
    Id = 42,
    Tag = new FixedString32('SensorA')
};
writer.Write(in sample); // Serializes directly to pooled buffer
```

Zero-Copy Flow

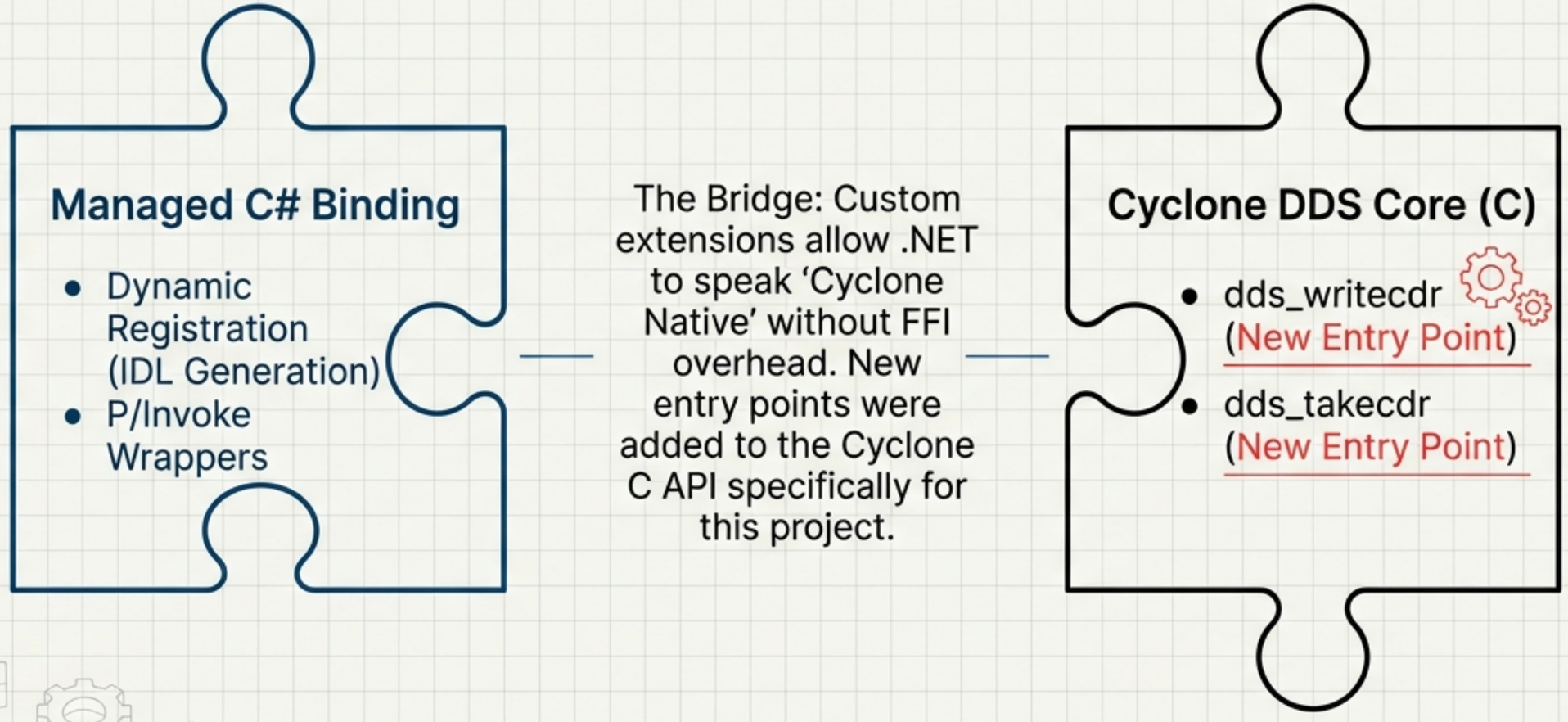
Network

SUBSCRIBE (READ)

```
using var samples = reader.Take(); // Loan samples
foreach (var view in samples) {
    // 'view' is a lightweight struct pointing to raw data
    Console.WriteLine("Received: " + view.Tag);
}
// Loan returned to Cyclone automatically at end of scope
```

Swiss Engineering Blueprint

Native Integration & Extensions



Swiss Engineering Blueprint

Summary of Benefits

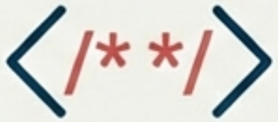
High Throughput

Zero-copy read/write paths bypass standard marshalling bottlenecks.

Low Latency

Elimination of GC pauses ensures deterministic behavior for real-time loops.

Productivity



Use standard C# tools, attributes, and types. No context switching to C++.

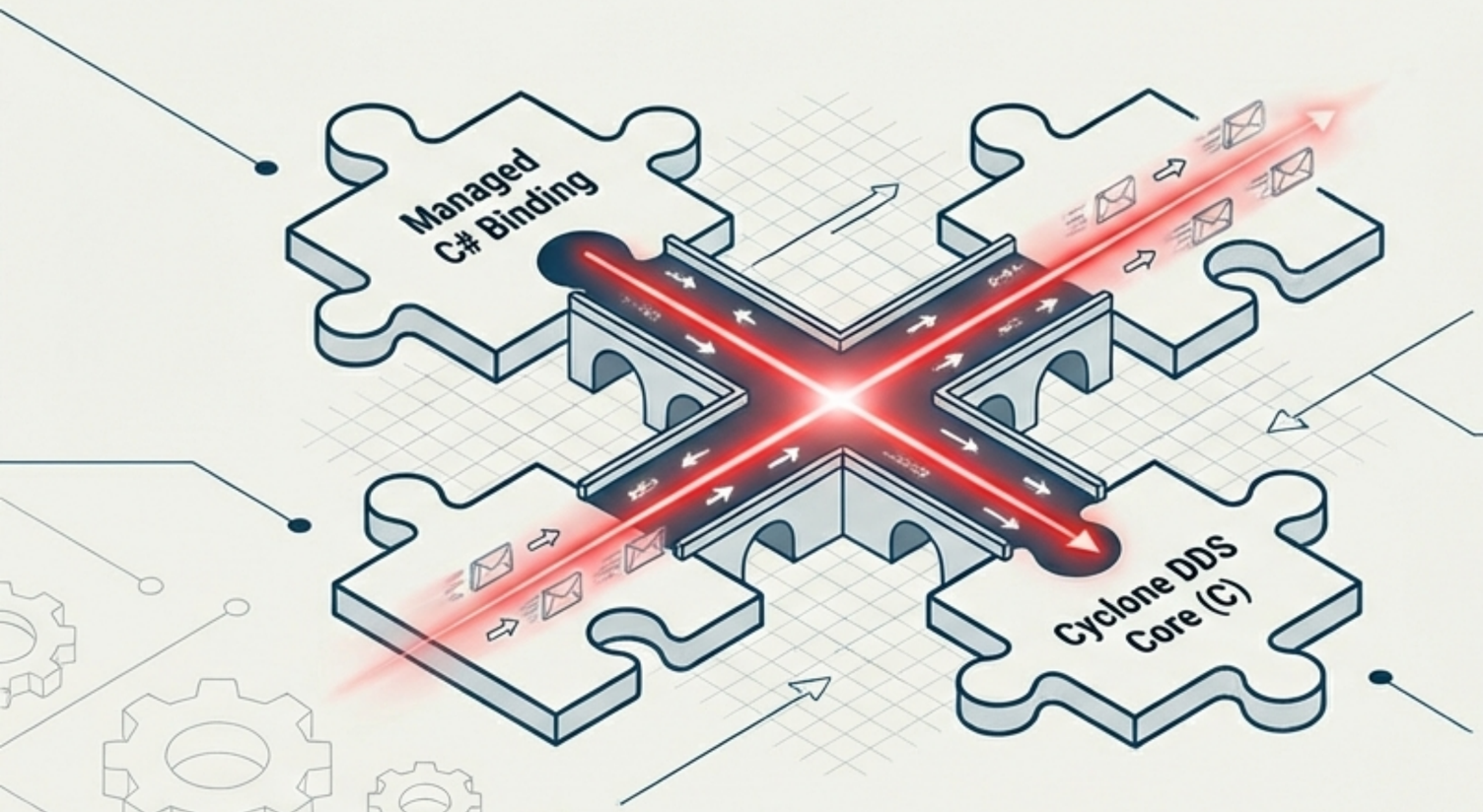
Seamless Interop

“Golden Test” verified. Guaranteed byte-for-byte compatibility with any DDS participant (C++, Python).

Swiss Engineering Blueprint

Real-Time Performance. .NET Convenience.

The power of Eclipse Cyclone DDS, now native to the .NET ecosystem.



IDEAL USE CASES

- Autonomous Vehicles & Robotics
- High-Frequency Telemetry & Sensor Fusion
- Mission-Critical Distributed Systems

Fast Cyclone DDS C# Bindings

Swiss Engineering Blueprint