

Resonance

Resonance is a high-performance real-time C# communication library with built-in support for several different transcoding and delivery methods.

This library provides an intuitive API for asynchronous communication between machines and devices by exposing a set of easy to use, pluggable components.

The resonance library might be described by the following layers:

**Transporting**

A transporter responsibility is to provide the API for sending and receiving messages, managing those messages, and propagating the necessary information to other components.

**Transcoding**

Encoders and Decoders are components that can be plugged to a transporter, they determine how outgoing/incoming messages should be encoded and whether the data should be encrypted and/or compressed.

The Following built-in transcoding methods are currently supported by the library:

* Json - (using Json.NET)
* Bson - (using Json.NET)
* Protobuf - (using Google.Protobuf separate NuGet package)
* Xml - (using .NET built-in Xml Serializer)

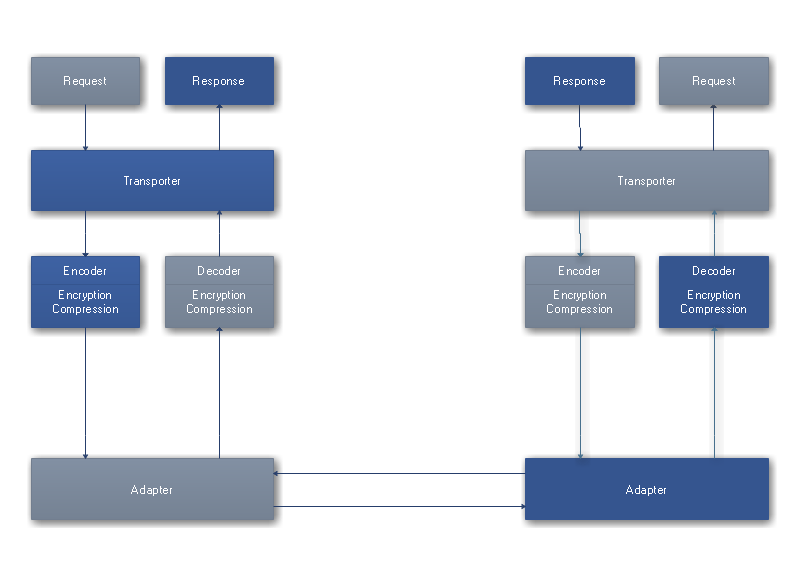
**Adapters**

Adapters can also be plugged to a transporter to determine how outgoing/incoming encoded data is going to be transmitted and where.

The following built-in adapters are currently supported by the library:

* TCP
* UDP
* USB
* HTTP
* In-Memory
* SignalR
* WebRTC
* Named Pipes
* Shared Memory

The following diagram described a simple request-response scenario.



**How it Works**

Resonance is a request-response based communication framework.

This means that for each request that is being sent, a matching response is expected.

This is done by attaching a unique token to each request and expecting the same token from the response.

Although the request-response pattern is the recommended approach, it is not enforced. Sending messages without expecting any response is possible.