Choosing a Message Encoder

.NET Framework (current version)

This topic discusses criteria for choosing among the message encoders that are included in Windows Communication Foundation (WCF): binary, text, and Message Transmission Optimization Mechanism (MTOM).

In WCF, you specify how to transfer data across a network between endpoints by means of a *binding*, which is made up of a sequence of *binding elements*. A message encoder is represented by a message encoding binding element in the binding stack. A binding includes optional protocol binding elements, such as a security binding element or reliable messaging binding element, a required message encoding binding element, and a required transport binding element.

The message encoding binding element sits below the optional protocol binding elements and above the required transport binding element. On the outgoing side, a message encoder serializes the outgoing Message and passes it to the transport. On the incoming side, a message encoder receives the serialized form of a Message from the transport and passes it to the higher protocol layer, if present, or to the application, if not.

When connecting to a pre-existing client or server, you may not have a choice about using a particular message encoding since you need to encode your messages in a way that the other side is expecting. However, if you are writing an WCF service, you can expose your service through multiple endpoints, each using a different message encoding. This allows clients to choose the best encoding for talking to your service over the endpoint that is best for them, as well as giving your clients the flexibility to choose the encoding that is best for them. Using multiple endpoints also allows you to combine the advantages of different message encodings with other binding elements.

System-Provided Encoders

WCF includes three message encoders, which are represented by the following three classes:

- TextMessageEncodingBindingElement, the text message encoder, supports both plain XML encoding and SOAP encoding. The plain XML encoding mode of the text message encoder is called "plain old XML" (POX) to distinguish it from text-based SOAP encoding. To enable POX, set the MessageVersion property to None. Use the text message encoder to interoperate with non-WCF endpoints.

- BinaryMessageEncodingBindingElement, the binary message encoder, uses a compact binary format and is optimized for WCF to WCF communication, and hence is not interoperable. This is also the most performant encoder of all the encoders WCF provides.

- T:System.ServiceModel.Channels.MTOMMessageEncodingBindingElement, the binding element, specifies the character encoding and message versioning for messages using MTOM encoding. MTOM is an efficient technology for transmitting binary data in WCF messages. The MTOM encoder attempts to create a balance between efficiency and interoperability. The MTOM encoding transmits most XML in textual form, but optimizes large blocks of binary data by transmitting them as-is, without conversion to text. In terms of efficiency, among the encoders WCF provides, MTOM is in-between text (the slowest) and binary (the fastest).

How to Choose a Message Encoder

The following table describes common factors used to choose a message encoder. Prioritize the factors that are important for

your application, and then choose the message encoders that work best with these factors. Be sure to consider any additional factors not listed in this table and any custom message encoders that may be required in your application.

Factor	Description	Encoders that support this factor
Supported Character Sets	TextMessageEncodingBindingElement and MtomMessageEncodingBindingElement support only the UTF8 and UTF16 Unicode (*big-endian* and *little-endian*) encodings. If other encodings are required, such as UTF7 or ASCII, a custom encoder must be used. For a sample custom encoder, see Custom Message Encoder.	Text
Inspection	Inspection is the ability to examine messages during transmission. Text encodings, either with or without the use of SOAP, allow messages to be inspected and analyzed by many applications without the use of specialized tools. Note that the use of transfer security, at either the message or transport level, affects your ability to inspect messages. Confidentiality protects a message from being examined and integrity protects a message from being modified.	Text
Reliability	Reliability is the resiliency of an encoder to transmission errors. Reliability can also be provided at the message, transport, or application layer. All of the standard WCF encoders assume that another layer is providing reliability. The encoder has little ability to recover from a transmission error.	None
Simplicity	Simplicity represents the ease with which you can create encoders and decoders for an encoding specification. Text encodings are particularly advantageous for simplicity, and the POX text encoding has the additional advantage of not requiring support for processing SOAP.	Text (POX)
Size	The encoding determines the amount of overhead imposed on content. The size of encoded messages is directly related to the maximum throughput of service operations. Binary encodings are generally more compact than text encodings. When message size is at a premium, consider also compressing the message contents during encoding. However, compression adds processing costs for both the message sender and receiver.	Binary
Streaming	Streaming allows applications to begin processing a message before the entire message has arrived. Effectively using streaming requires that the important data for a message be available at the beginning of the message so that the receiving application is not required to wait for it to arrive. Moreover, applications that use streamed transfer must organize data in the message incrementally so that the content does not have forward dependencies. In many cases, you must compromise between streaming content and having the smallest possible transfer size for that content.	None
3rd Party Tool Support	Support areas for an encoding include development and diagnosis. Third-party developers have made a large investment in libraries and toolkits for handling messages encoded in the POX format.	Text (POX)
Interoperability	This factor refers to the ability of a WCF encoder to interoperate with non-WCF services.	Text MTOM (partial)

Note: When using the Binary Encoder, using the IgnoreWhitespace setting when creating a XMLReader will have no effect. For example, if you do the following inside a service operation:

C#

```
public void OperationContract(XElement input)
        {
            Console.WriteLine("{0}", input.Value);
            int counter = 0;
            var xreader = input.CreateReader();
            var reader = XmlReader.Create(xreader, new XmlReaderSettings() { IgnoreWhitespace
= true });
            while (reader.Read())
            {
                counter++;
            }

            Console.WriteLine("Read {0} lines with reader", counter);
        }
```

The IgnoreWhitespace setting is ignored.

Compression and the Binary Encoder

Beginning with WCF 4.5 the WCF binary encoder adds support for compression. This enables you to use the gzip/deflate algorithm for sending compressed messages from a WCF client and also respond with compressed messages from a self-hosted WCF service. This feature enables compression on both the HTTP and TCP transports. An IIS hosted WCF service can always be enabled for sending compressed responses by configuring the IIS host server. The type of compression is configured with the P:System.ServiceModel.Channels.BinaryMessageEncodingElement.CompressionFormat property. This property is set to one of the CompressionFormat enum values:

CompressionFormat Value	Description
F:System.ServiceModel.Channels.CompressionFormat.Deflate	Uses Deflate compression.
F:System.ServiceModel.Channels.CompressionFormat.GZip	Uses GZip compression
F:System.ServiceModel.Channels.CompressionFormat.None	No compression will be used

Since this property is only exposed on the binaryMessageEncodingBindingElement, you will need to create a custom binding like the following to use this feature:<customBinding> <binding name="BinaryCompressionBinding"> <binaryMessageEncoding compressionFormat ="GZip"/> <httpTransport /> </binding> </customBinding>Both the client and the service need to agree to send and receive compressed messages and therefore the compressionFormat property must be configured on the binaryMessageEncoding element on both client and service. A ProtocolException is thrown if either the service or client is not configured for compression but the other side is.Enabling compression should be carefully considered. Compression is mostly useful if network bandwidth is a bottleneck. In the case where the CPU is the bottleneck, compression will decrease throughput. Appropriate testing must be done in a simulated environment to find out if this benefits the

application

See Also

Windows Communcation Foundation Bindings

;