**What is BSON?**

[**BSON**](http://bsonspec.org/)**is a binary serialization format. “BSON”** stands for “Binary JSON”, but BSON and JSON are serialized very differently. BSON is “JSON-like”, because objects are represented as name-value pairs, similar to JSON. **Unlike JSON, numeric data types are stored as bytes, not strings**

The BSON was designed to be lightweight, easy to scan, and fast to encode/decode.

* BSON is comparable in size to JSON. Depending on the data, a BSON payload may be smaller or larger than a JSON payload. For serializing binary data, such as an image file, BSON is smaller than JSON, because the binary data does is not base64-encoded.
* BSON documents are easy to scan because elements are prefixed with a length field, so a parser can skip elements without decoding them.
* Encoding and decoding are efficient, because numeric data types are stored as numbers, not strings.
* Native clients, such as .NET client apps, can benefit from using BSON in place of text-based formats such as JSON or XML. For browser clients, you will probably want to stick with JSON, because JavaScript can directly convert the JSON payload.

Fortunately, Web API uses [content negotiation](http://www.asp.net/web-api/overview/formats-and-model-binding/content-negotiation), so your API can support both formats and let the client choose.

**Enabling BSON on the Server**

In your Web API configuration, add the **BsonMediaTypeFormatter** to the formatters collection.

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

config.Formatters.Add(new BsonMediaTypeFormatter());

// Other Web API configuration not shown...

}

}

Now if the client requests “application/bson”, Web API will use the BSON formatter.

**To associate BSON with other media types, add them to the SupportedMediaTypes collection. The following code adds “application/vnd.contoso” to the supported media types:**

var bson = new BsonMediaTypeFormatter();

bson.SupportedMediaTypes.Add(new MediaTypeHeaderValue("application/vnd.contoso"));

config.Formatters.Add(bson);

**Using BSON with HttpClient**

.NET clients apps can use the BSON formatter with **HttpClient**. For more information about **HttpClient**, see [Calling a Web API From a .NET Client](http://www.asp.net/web-api/overview/web-api-clients/calling-a-web-api-from-a-net-client).

The following code sends a GET request that accepts BSON, and then deserializes the BSON payload in the response.

static async Task RunAsync()

{

using (HttpClient client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost");

// Set the Accept header for BSON.

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/bson"));

// Send GET request.

result = await client.GetAsync("api/books/1");

result.EnsureSuccessStatusCode();

// Use BSON formatter to deserialize the result.

MediaTypeFormatter[] formatters = new MediaTypeFormatter[] {

new BsonMediaTypeFormatter()

};

var book = await result.Content.ReadAsAsync<Book>(formatters);

}

}

**To request BSON from the server, set the Accept header to “application/bson”:**

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(new

MediaTypeWithQualityHeaderValue("application/bson"));

To deserialize the response body, use the **BsonMediaTypeFormatter**. This formatter is not in the default formatters collection, so you have to specify it when you read the response body:

MediaTypeFormatter[] formatters = new MediaTypeFormatter[] {

new BsonMediaTypeFormatter()

};

var book = await result.Content.ReadAsAsync<Book>(formatters);

The next example shows how to send a POST request that contains BSON.

static async Task RunAsync()

{

using (HttpClient client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:15192");

// Set the Accept header for BSON.

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/bson"));

var book = new Book()

{

Author = "Jane Austen",

Title = "Emma",

Price = 9.95M,

PublicationDate = new DateTime(1815, 1, 1)

};

// POST using the BSON formatter.

MediaTypeFormatter bsonFormatter = new BsonMediaTypeFormatter();

var result = await client.PostAsync("api/books", book, bsonFormatter);

result.EnsureSuccessStatusCode();

}

}

Much of this code is the same as the previous example. But in the **PostAsync** method, specify **BsonMediaTypeFormatter** as the formatter:

MediaTypeFormatter bsonFormatter = new BsonMediaTypeFormatter();

var result = await client.PostAsync("api/books", book, bsonFormatter);