

ASP.NET Core 용 SignalR에서 MessagePack 허브 프로토콜 사용

04/13/2020 • 읽는 데 15 분 •  +5

이 기사에서

[MessagePack이란 무엇입니까?](#)

[서버에서 MessagePack 구성](#)

[Configure MessagePack on the client](#)

[MessagePack quirks](#)

[Related resources](#)

이 기사에서는 독자가 [시작하기](#) 에서 다루는 주제에 익숙하다고 가정합니다 .

MessagePack이란 무엇입니까?

[MessagePack](#) 은 빠르고 컴팩트 한 이진 직렬화 형식입니다. [JSON](#)에 비해 더 작은 메시지를 생성하므로 성능 및 대역폭이 문제가 될 때 유용합니다 . 바이트가 MessagePack 구문 분석기를 통해 전달되지 않는 한 네트워크 추적 및 로그를 볼 때 바이너리 메시지를 읽을 수 없습니다. SignalR에는 MessagePack 형식에 대한 기본 지원이 있으며 클라이언트와 서버가 사용할 API를 제공합니다.

서버에서 MessagePack 구성

서버에서 MessagePack 허브 프로토콜을 활성화하려면

Microsoft.AspNetCore.SignalR.Protocols.MessagePack 앱에 패키지를 설치하십시오 . 에서 Startup.ConfigureServices 방법, 추가 AddMessagePackProtocol 받는 AddSignalR 서버에 MessagePack 지원을 활성화하기 위해 호출.

❗ 노트

JSON은 기본적으로 활성화되어 있습니다. MessagePack을 추가하면 JSON 및 MessagePack 클라이언트를 모두 지원할 수 있습니다.

씨#

 부

```
services.AddSignalR()
    .AddMessagePackProtocol();
```

MessagePack이 데이터 형식을 지정하는 방법을 사용자 지정하려면

AddMessagePackProtocol 옵션 구성을 위한 위임을받습니다. 해당 대리자에서

FormatterResolvers 속성을 사용하여 MessagePack 직렬화 옵션을 구성 할 수 있습니다.

리졸버 작동 방식에 대한 자세한 내용은 [MessagePack-CSharp](#) 에서 [MessagePack](#) 라이브러리를 참조하세요 . 직렬화하려는 개체에 특성을 사용하여 처리 방법을 정의 할 수 있습니다.

씨#	부
<pre> services.AddSignalR() .AddMessagePackProtocol(options => { options.FormatterResolvers = new List<MessagePack.IFormatterResolver>() { MessagePack.Resolvers.StandardResolver.Instance }; }); </pre>	

⚠ 경고

CVE-2020-5234를 검토 하고 권장 패치를 적용하는 것이 좋습니다. 예를 들어 MessagePackSecurity.Active 정적 속성을 MessagePackSecurity.UntrustedData. 를 설정 MessagePackSecurity.Active 하려면 **1.9.x 버전의 MessagePack**을 수동으로 설치해야 합니다 . MessagePack 1.9.x를 설치 하면 SignalR이 사용하는 버전이 업그레이드 됩니다. 시 MessagePackSecurity.Active 에 설정되지 않은 MessagePackSecurity.UntrustedData 악의적 인 클라이언트는 서비스 거부가 발생할 수 있습니다. 설정 MessagePackSecurity.Active 에서 Program.Main 다음 코드와 같이 :

씨#	부
<pre> public static void Main(string[] args) { MessagePackSecurity.Active = MessagePackSecurity.UntrustedData; CreateHostBuilder(args).Build().Run(); } </pre>	

클라이언트에서 MessagePack 구성

① 노트

JSON은 지원되는 클라이언트에 대해 기본적으로 활성화됩니다. 클라이언트는 단일 프로토콜 만 지원할 수 있습니다. MessagePack 지원을 추가하면 이전에 구성된 모든 프로토콜이 대체됩니다.

.NET 클라이언트

닷넷 클라이언트에 MessagePack을 사용하려면 설치

Microsoft.AspNetCore.SignalR.Protocols.MessagePack 패키지를하고 전화

AddMessagePackProtocol에 HubConnectionBuilder.

씨#	부
<pre>var hubConnection = new HubConnectionBuilder() .WithUrl("/chathub") .AddMessagePackProtocol() .Build();</pre>	

❗ 노트

이 AddMessagePackProtocol 호출은 서버와 같은 옵션을 구성하기 위해 위임을받습니다.

자바 스크립트 클라이언트

JavaScript 클라이언트에 대한 MessagePack 지원은 @microsoft/signalr-protocol-msgpack npm 패키지에서 제공합니다. 명령 셸에서 다음 명령을 실행하여 패키지를 설치합니다.

세계 때리다	부
<pre>npm install @microsoft/signalr-protocol-msgpack</pre>	

npm 패키지를 설치 한 후, 모듈은 JavaScript 모듈 로더를 통해 직접 사용하거나 다음 파일을 참조하여 브라우저로 가져올 수 있습니다.

node_modules \ @microsoft \ signalr-protocol-msgpack \ dist \ browser \ signalr-protocol-msgpack.js

In a browser, the msgpack5 library must also be referenced. Use a <script> tag to create a reference. The library can be found at *node_modules\msgpack5\dist\msgpack5.js*.

ⓘ Note

When using the `<script>` element, the order is important. If *signalr-protocol-msgpack.js* is referenced before *msgpack5.js*, an error occurs when trying to connect with MessagePack. *signalr.js* is also required before *signalr-protocol-msgpack.js*.

HTML

 Copy

```
<script src="/lib/signalr/signalr.js"></script>
<script src="/lib/msgpack5/msgpack5.js"></script>
<script src="/lib/signalr/signalr-protocol-msgpack.js"></script>
```

Adding `.withHubProtocol(new signalR.protocols.msgpack.MessagePackHubProtocol())` to the `HubConnectionBuilder` will configure the client to use the MessagePack protocol when connecting to a server.

JavaScript

 Copy

```
const connection = new signalR.HubConnectionBuilder()
    .withUrl("/chathub")
    .withHubProtocol(new signalR.protocols.msgpack.MessagePackHubProtocol())
    .build();
```

ⓘ Note

At this time, there are no configuration options for the MessagePack protocol on the JavaScript client.

MessagePack quirks

There are a few issues to be aware of when using the MessagePack Hub Protocol.

MessagePack is case-sensitive

The MessagePack protocol is case-sensitive. For example, consider the following C# class:

C#

 Copy

```
public class ChatMessage
{
    public string Sender { get; }
    public string Message { get; }
}
```

When sending from the JavaScript client, you must use `PascalCased` property names, since the casing must match the C# class exactly. For example:

JavaScript

 Copy

```
connection.invoke("SomeMethod", { Sender: "Sally", Message: "Hello!" });
```

Using `camelCased` names won't properly bind to the C# class. You can work around this by using the `Key` attribute to specify a different name for the MessagePack property. For more information, see [the MessagePack-CSharp documentation](#).

DateTime.Kind is not preserved when serializing/deserializing

The MessagePack protocol doesn't provide a way to encode the `Kind` value of a `DateTime`. As a result, when deserializing a date, the MessagePack Hub Protocol assumes the incoming date is in UTC format. If you're working with `DateTime` values in local time, we recommend converting to UTC before sending them. Convert them from UTC to local time when you receive them.

For more information on this limitation, see GitHub issue [aspnet/SignalR#2632](#).

DateTime.MinValue is not supported by MessagePack in JavaScript

The [msgpack5](#) library used by the SignalR JavaScript client doesn't support the `timestamp96` type in MessagePack. This type is used to encode very large date values (either very early in the past or very far in the future). The value of `DateTime.MinValue` is January 1, 0001, which must be encoded in a `timestamp96` value. Because of this, sending `DateTime.MinValue` to a JavaScript client isn't supported. When `DateTime.MinValue` is received by the JavaScript client, the following error is thrown:

 Copy


```
Uncaught Error: unable to find ext type 255 at decoder.js:427
```

Usually, `DateTime.MinValue` is used to encode a "missing" or `null` value. If you need to encode that value in MessagePack, use a nullable `DateTime` value (`DateTime?`) or encode a separate `bool` value indicating if the date is present.

For more information on this limitation, see GitHub issue [aspnet/SignalR#2228](https://github.com/aspnet/SignalR/issues/2228).


MessagePack support in "ahead-of-time" compilation environment

The [MessagePack-CSharp](#) library used by the .NET client and server uses code generation to optimize serialization. As a result, it isn't supported by default on environments that use "ahead-of-time" compilation (such as Xamarin iOS or Unity). It's possible to use MessagePack in these environments by "pre-generating" the serializer/deserializer code. For more information, see [the MessagePack-CSharp documentation](#). Once you have pre-generated the serializers, you can register them using the configuration delegate passed to `AddMessagePackProtocol`:

C#	 Copy
<pre>services.AddSignalR() .AddMessagePackProtocol(options => { options.FormatterResolvers = new List<MessagePack.IFormatterResolver>() { MessagePack.Resolvers.GeneratedResolver.Instance, MessagePack.Resolvers.StandardResolver.Instance }; });</pre>	

Type checks are more strict in MessagePack

JSON 허브 프로토콜은 역 직렬화 중에 유형 변환을 수행합니다. 예를 들어 들어오는 개체에 숫자 ({ foo: 42 }) 인 속성 값이 있지만 .NET 클래스의 속성이 유형 `string` 인 경우 값이 변환됩니다. 그러나 MessagePack은이 변환을 수행하지 않으며 서버 측 로그 (및 콘솔)에서 볼 수 있는 예외를 발생시킵니다.

	 부
<pre>InvalidDataException: Error binding arguments. Make sure that the types of the provided values match the types of the hub method being invoked.</pre>	

이 제한에 대한 자세한 내용은 GitHub 문제 [aspnet / SignalR # 2937](https://github.com/aspnet/SignalR/issues/2937)을 참조하십시오 .

관련 자료

- [시작하다](#)
- [.NET 클라이언트](#)
- [자바 스크립트 클라이언트](#)

Is this page helpful?

 Yes  No