

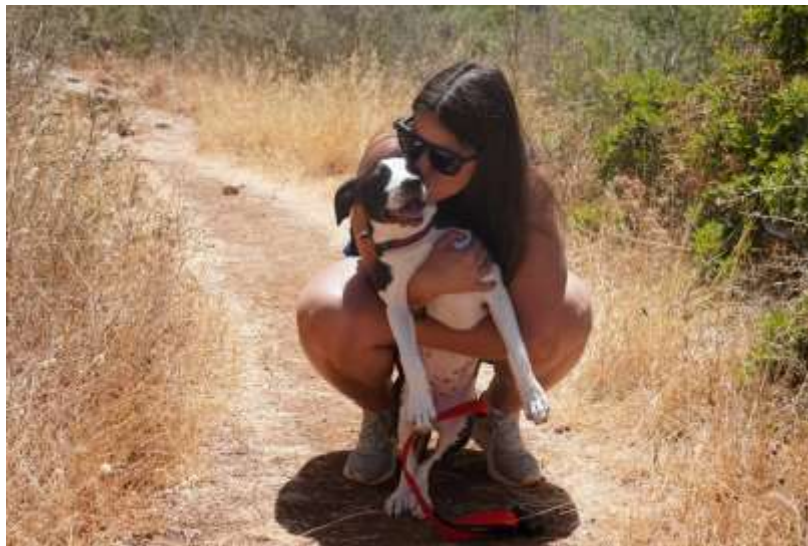
Create your first chat with ACS

Batel Zohar



Batel Zohar

Developer
Advocate



 BatelZohar

 batelzohar

 batelt@jfrog.com



Azure Communication Services

Azure Communication Services is a set of rich communication APIs, video APIs, and SMS APIs for deploying your applications across any device, on any platform.

Learn



Reach customers anywhere with a fully managed platform

Deliver video, voice, chat, text messaging, and telephony experiences anywhere your customers are—across your applications, websites, and mobile platforms.



Scale with a global infrastructure used by Microsoft Teams

Use a reliable global platform trusted by millions of people daily.



Build on a secure and compliant cloud

Reach more customers without compromising trust using a secure and compliant cloud.



Overview

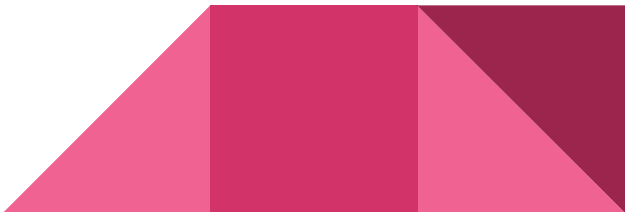
This is a sample application to show how the Chat Web SDK can be used to build a single threaded chat experience. The client-side application is a React based user interface which uses Redux for handling complex state while leveraging Microsoft Fluent UI. Powering this front-end is a C# web application powered by ASP.NET Core to connect this application with Azure Communication Services.

Additional documentation for this sample can be found on [Microsoft Docs](#).



Prerequisites

- Create an Azure account with an active subscription.
- [Node.js \(12.18.4 and above\)](#)
- [Visual Studio \(2019 and above\)](#) or [VScode](#)
- [.NET Core 3.1](#)
- Create an Azure Communication Services resource



Code structure

- `./Chat/ClientApp`: frontend client
 - `./Chat/ClientApp/src`
 - `./Chat/ClientApp/src/Components` : React components to help build the client app chat experience
 - `./Chat/ClientApp/src/Containers` : Connects the redux functionality to the React components
 - `./Chat/ClientApp/src/Core` : Containers a redux wrapper around the Chat SDK
 - `./Chat/ClientApp/src/index.js` : Entry point for the client app
- `./Chat/Controllers` : Server app core logic for client app to get a token to use with the Azure Communication Services Web Chat SDK
- `./Chat/Program.cs` : Server app program logic
- `./Chat/Startup.cs`: Server app startup logic



Let's start !

1. Run the command

\$ git clone https://github.com/Azure-Samples/communication-services-web-chat-hero.git

2. Get the Connection String from the Azure portal:

Communication Services  

Microsoft



Communication Services

 [Add to Favorites](#)

Microsoft

★★★★☆ 3.8 (5 ratings)

Create

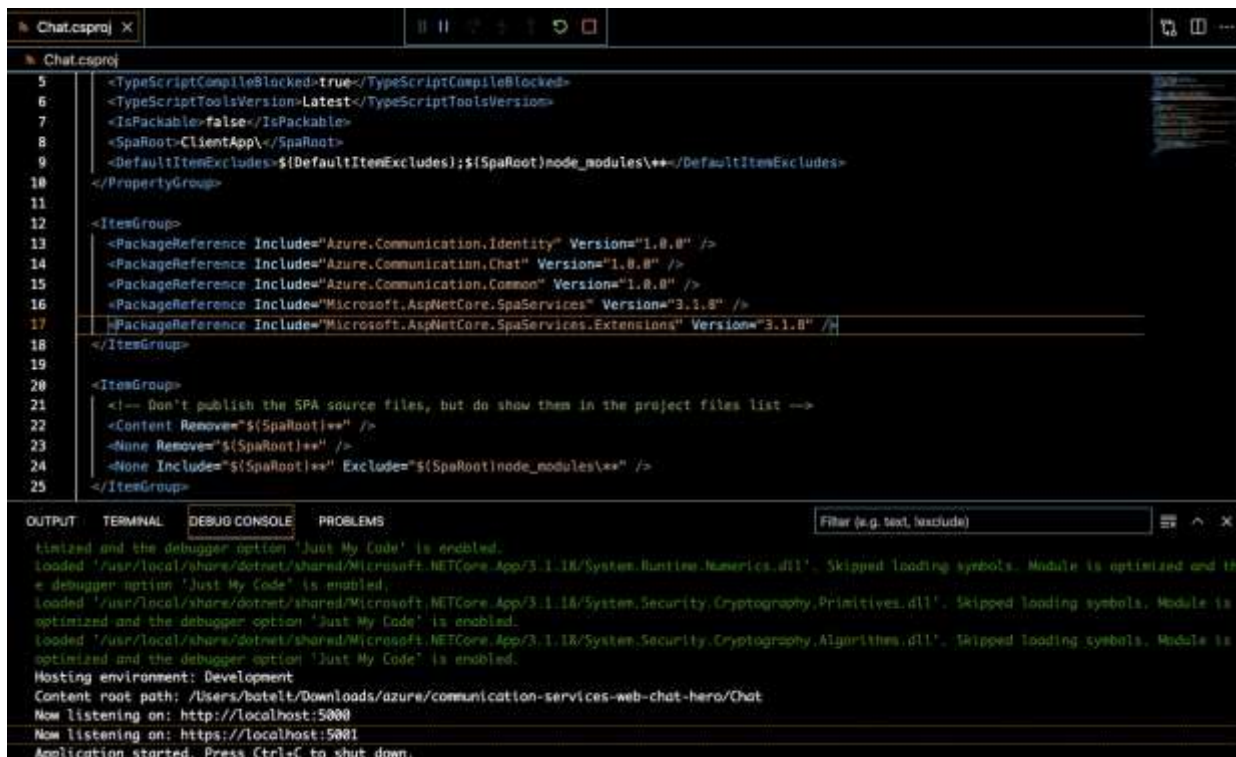


Edit the appsettings.json file



Once you get the Connection String, Add the connection string to the **Chat/appsettings.json** file found under the Chat folder. Input your connection string in the variable: ResourceConnectionString.

Run your application



The screenshot shows the Visual Studio Code interface. The top pane displays the `Chat.csproj` file with the following XML content:

```
5 <TypeScriptCompileBlocked>true</TypeScriptCompileBlocked>
6 <TypeScriptToolsVersion>Latest</TypeScriptToolsVersion>
7 <IsPackable>false</IsPackable>
8 <SpaRoot>ClientApp</SpaRoot>
9 <DefaultItemExcludes>$(DefaultItemExcludes);$(SpaRoot)node_modules\**</DefaultItemExcludes>
10 </PropertyGroup>
11
12 <ItemGroup>
13 <PackageReference Include="Azure.Communication.Identity" Version="1.0.0" />
14 <PackageReference Include="Azure.Communication.Chat" Version="1.0.0" />
15 <PackageReference Include="Azure.Communication.Common" Version="1.0.0" />
16 <PackageReference Include="Microsoft.AspNetCore.SpaServices" Version="3.1.0" />
17 <PackageReference Include="Microsoft.AspNetCore.SpaServices.Extensions" Version="3.1.0" />
18 </ItemGroup>
19
20 <ItemGroup>
21 <!-- Don't publish the SPA source files, but do show them in the project files list -->
22 <Content Remove="$(SpaRoot)**" />
23 <None Remove="$(SpaRoot)**" />
24 <None Include="$(SpaRoot)**" Exclude="$(SpaRoot)node_modules\**" />
25 </ItemGroup>
```

The bottom pane shows the **OUTPUT** window with the following text:

```
timized and the debugger option 'Just My Code' is enabled.
Loaded 'C:\Users\batelt\Downloads\azure\communication-services-web-chat-hero\Chat\bin\Debug\netcoreapp3.1\System.Runtime.Numerics.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
Loaded 'C:\Users\batelt\Downloads\azure\communication-services-web-chat-hero\Chat\bin\Debug\netcoreapp3.1\System.Security.Cryptography.Primitives.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
Loaded 'C:\Users\batelt\Downloads\azure\communication-services-web-chat-hero\Chat\bin\Debug\netcoreapp3.1\System.Security.Cryptography.Algorithms.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.
Hosting environment: Development
Content root path: /Users/batelt/Downloads/azure/communication-services-web-chat-hero/Chat
Now listening on: http://localhost:5000
Now listening on: https://localhost:5001
Application started. Press Ctrl+C to shut down.
```

Troubleshooting

If we have npm installation / build issues you may want to clean the npm cache with the following command:

```
$ npm cache clean --force
```





**KEEP
CALM**
its
**DEMO
TIME**



THANKS

@BatelZohar



batelt@jfrog.com



[linkedin.com/in/batelzohar/](https://www.linkedin.com/in/batelzohar/)

