## Sending Welcome Messages via Direct Line / Web Chat

by TJ Durnford & Steven Kanberg

September 23, 2019

When a bot connects with a channel, generally two conversation update events are sent when the conversation is initialized - the first for the bot and another for the user. The second conversation update - the event for the user - is intended to trigger the welcome message. Presently, Web Chat has two different welcome message scenarios that offer a slightly different experience from that of other channels. This experience is determined by how the developer decides to generate the Direct Line token: either with a user ID or without.

### Tokens with User IDs

The first scenario is dependent on the token request including a user ID. If the developer includes a user ID when generating the token, Direct Line will only send one conversation update event to the bot. This conversation update will include two user IDs in the activity's membersAdded property - one for the bot and one for the user. This configuration should trigger the traditional welcome message in the onMembersAdded handler before the user messages the bot.

In the following example, the user ID is added to the token request and the welcome message is sent from the onMembersAdded handler.

### Web Chat

(async function () {

// Note, for the simplicity of this example, we are generating the Direct Line token on client side;

// however, this is not a recommended practice and you should create and manage your tokens from the server.

// You should never put the Direct Line secret in the browser or client app.

// https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-authentication

const secret = '<DIRECT\_LINE\_SECRET> | <WEB\_CHAT\_SECRET>';

const res = await fetch('https://directline.botframework.com/v3/directline/tokens/generate', {

body: JSON.stringify({ user: { id: 'dl\_user\_id', name: 'username' }}),

headers: {

Authorization: `Bearer ${secret}`,

'Content-type': 'application/json'

},

method: 'POST',

});

const { token } = await res.json();

window.WebChat.renderWebChat({

directLine: window.WebChat.createDirectLine({ token })

}, document.getElementById('webchat'));

document.querySelector('#webchat > \*').focus();

})().catch(err => console.error(err));

### Bot Framework SDK v4 (C#)

protected override async Task OnMembersAddedAsync(IList<ChannelAccount> membersAdded, ITurnContext<IConversationUpdateActivity> turnContext, CancellationToken cancellationToken)

{

foreach (var member in membersAdded)

{

if (member.Id != turnContext.Activity.Recipient.Id)

{

await turnContext.SendActivityAsync($"("Welcome Message from `onMembersAdded` handler!", cancellationToken: cancellationToken);

}

}

}

### Bot Framework SDK v4 (Node.js)

this.onMembersAdded(async (context, next) => {

const { membersAdded } = context.activity;

for (let member of membersAdded) {

if (member.id !== context.activity.recipient.id) {

await context.sendActivity("Welcome Message from `onMembersAdded` handler!");

}

}

await next();

});

### Secrets and Tokens without User IDs

In the second scenario, conversations created with tokens that do not include a user ID send two conversation update events. Direct Line sends the first conversation update, the one for the bot, when the connection with the bot is established. Direct Line sends the second conversation update for the user after they send their first message.

Generally, users anticipate the bot to send a welcome message before having to send a message, themselves. To do this, you can dispatch a backchannel welcome event from Web Chat's store middleware when the Direct Line connection is established. Then, in the onEvent handler you can send a welcome message.

Note, in the onMembersAdded handler you should check which channel is sending the event *before* sending the welcome message. If the channel id is "webchat" or "directline" you should not send the traditional welcome message to avoid sending multiple welcome messages, as the first scenario (described above) will already be in use.

### Web Chat

(async function () {

// For the simplicity, the Direct Line token is generated client side.

// In production, you should create and manage your tokens from the server.

// You should never put the Direct Line secret in the browser or client app.

// For more information, please reference the [Direct Line 3.0 Authentication](https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-authentication) doc.

const secret = '<DIRECT\_LINE\_SECRET> | <WEB\_CHAT\_SECRET>';

const res = await fetch('https://directline.botframework.com/v3/directline/tokens/generate', {

headers: {

Authorization: `Bearer ${secret}`,

},

method: 'POST'

});

const { token } = await res.json();

const store = window.WebChat.createStore(

{},

({ dispatch }) => next => action => {

if (action.type === 'DIRECT\_LINE/CONNECT\_FULFILLED') {

dispatch({

type: 'WEB\_CHAT/SEND\_EVENT',

payload: {

name: 'webchat/join',

}

});

}

return next(action);

}

);

window.WebChat.renderWebChat({

directLine: window.WebChat.createDirectLine({ token }),

store

}, document.getElementById('webchat'));

document.querySelector('#webchat > \*').focus();

})().catch(err => console.error(err));

### Bot Framework SDK v4 (C#)

protected override async Task OnEventActivityAsync(ITurnContext<IEventActivity> turnContext, CancellationToken cancellationToken)

{

await turnContext.SendActivityAsync($"("Back Channel Welcome Message", cancellationToken: cancellationToken);

}

protected override async Task OnMembersAddedAsync(IList<ChannelAccount> membersAdded, ITurnContext<IConversationUpdateActivity> turnContext, CancellationToken cancellationToken)

{

foreach (var member in membersAdded)

{

if (member.Id != turnContext.Activity.Recipient.Id)

{

await turnContext.SendActivityAsync($"("Welcome Message from `onMembersAdded` handler!", cancellationToken: cancellationToken);

}

}

}

### Bot Framework SDK v4 (Node.js)

this.onEvent(async (context, next) => {

if (context.activity.name === 'webchat/join') {

await context.sendActivity('Back Channel Welcome Message!');

}

await next();

});

this.onMembersAdded(async (context, next) => {

const { channelId, membersAdded } = context.activity;

if (channelId !== 'directline' && channelId !== 'webchat') {

for (let member of membersAdded) {

if (member.id !== context.activity.recipient.id) {

await context.sendActivity("Welcome Message from `onMembersAdded` handler!");

}

}

}

await next();

});

For more details regarding backchannel welcome events in Web Chat, take a look at this [sample](https://github.com/microsoft/BotFramework-WebChat/tree/master/samples/15.d.backchannel-send-welcome-event).

### Additional Context

* [Bot Framework Direct Line Authentication](https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-authentication?view=azure-bot-service-4.0)
* [Bot Framework Welcome Message Sample - Node SDK](https://github.com/microsoft/BotBuilder-Samples/tree/master/samples/javascript_nodejs/03.welcome-users)
* [Bot Framework Welcome Message Sample - C# SDK](https://github.com/microsoft/BotBuilder-Samples/tree/master/samples/csharp_dotnetcore/03.welcome-user)
* [Web Chat Backchannel Welcome Event Sample](https://github.com/microsoft/BotFramework-WebChat/tree/master/samples/15.d.backchannel-send-welcome-event)