LIVEPERSON

Coding Challenge

ETS Candidate Test

The Challenge

Part 1

Create a JavaScript/NodeJS application that connects to LivePerson's messaging service, requests a new messaging conversation, and publishes a text message to an agent. The first step is to create a free developer account from LivePerson: https://developers.liveperson.com/free-trial-registration/

1. After creating your account, there are a couple of domains that need to be retrieved. These should be fetched programmatically using the Domain API - the domains required are **idp** and **asyncMessagingEnt**

2. Request a consumer JWT and extract the JWT value from the json response. This JWT will be used in the next step. The domain for this service is found under **idp**.

POST

```
https://<domain>/api/account/<youraccount>/signup
```

3. Open the websocket connection. The domain for this service is found under **asyncMessagingEnt**.

```
wss://<domain>/ws_api/account/<youraccount>/messaging/consumer?v=3
```

   If the websocket package being used with your NodeJS app supports custom headers, add the following header to the connection utilising the JWT provided in Step 2:

```
headers: {
 Authorization: "jwt <JWT>"
}
```

NOTE: This header is not mandatory. If you are able to create the connection with this header skip to step 5, otherwise proceed to step 4.

4. The next step is to initiate the connection using the JWT provided in step 2. The initiate call is required if this is a client side app running in a browser, or if it's a server side app and the websocket package doesn't support custom headers.

```
{
    "kind": "req",
    "id": "0",
    "type": "InitConnection",
    "headers": [{
        "type": ".ams.headers.ClientProperties",
        "deviceFamily": "DESKTOP",
        "os": "WINDOWS"
    }, {
        "type": ".ams.headers.ConsumerAuthentication",
        "jwt": "<JWT>"
    }]
}
```

5. Request a new conversation by sending the following message into the open websocket:

```
{"kind":"req","id":1,"type":"cm.ConsumerRequestConversation"}
```

In case of success you should see the following response. Extract the value of conversationId.

```
{
    "kind": "resp",
    "reqId": "1",
    "code": 200,
    "body": {
        "conversationId": "_YOUR_CONVERSATION_ID_"
    },
    "type": "cm.RequestConversationResponse"
}
```

6. To publish content to a conversation, substitute _YOUR_CONVERSATION_ID_ with the conversationId you got in the response of Step 5, and use it with the opened WebSocket.

```
{
    "kind": "req",
    "id": "2",
    "type": "ms.PublishEvent",
    "body": {
        "dialogId": "_YOUR_CONVERSATION_ID_",
        "event": {
            "type": "ContentEvent",
            "contentType": "text/plain",
            "message": "My first message"
        }
    }
}
```

7. The published message should be displayed on the agent side. Verify this by going to the Conversational Cloud Agent Console and checking in the Agent Workspace.
8. Close the conversation

```
{
    "kind": "req",
    "id": "3",
    "type": "cm.UpdateConversationField",
    "body": {
        "conversationId": "_YOUR_CONVERSATION_ID_",
        "conversationField": [{
            "field": "ConversationStateField",
            "conversationState": "CLOSE"
        }]
    }
}
```

TIP: Ensure there is appropriate error handling available at each step.

Part 2

Explain how the solution fits together and any positives / difficulties you had completing the project.

Part 3

Upload the solution to a GitHub repository & provide any build instructions/scripts.

Optional

These items are not required to complete the challenge, but if you have time and want to extend your program, here are a couple of ideas:

- Build a basic unit test framework into your submission
- Build a web page interface that allows users to open the connection, close the connection, and enter their own text messages.

Again, this is only if you have time and want to add additional components - what we are looking for is clean, human readable code that has thought through any considerations that may be needed to write this program.

Timeframe

Please check the cover email for the submission date.

Final Submission Components

- Application source code delivered via GitHub repository
- Email write-up explaining how the solution fits together, and any positives / difficulties you had developing it.

Upon completion, please email details and grant access to the GitHub repo to Chris Ilkiw (ETS APAC Senior Manager) - email address / GitHub profile.