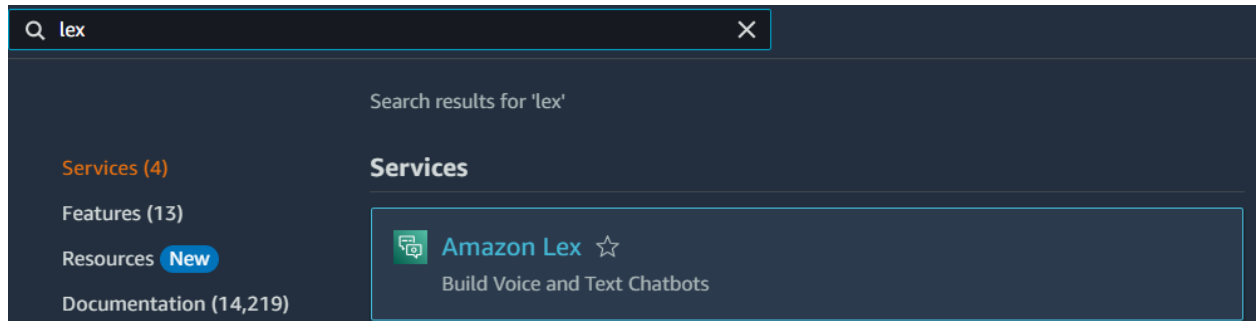


language translation bot using Amazon Lex

4.2A Create an empty chatbot

Create an empty chatbot

1. Login to your **AWS management console** and navigate to **Amazon Lex** from the search bar.



2. Click on '**Create Bot**'.
3. In the Creation method, go with '**Create a blank bot**'.
4. Give the chatbot a suitable name and description.

Configure bot settings [Info](#)

Creation method

☐ ⚡ **Descriptive Bot Builder - GenAI**
Describe the type of bot you would like to create, and Lex will use generative AI to create intents and slot types for you.

☒ **Create a blank bot**
Create a basic bot with no preconfigured languages, intents and slot types.

☐ **Start with an example**
An example bot has preconfigured languages, intents and slot types. You can change these settings.

☐ **Start with transcripts**
Automatically generate intents from conversation transcripts that you upload. Only English (US) language is available when starting with a transcript.

Bot configuration

Bot name

TextTranslator

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Description - *optional*

This description appears on bot list page. It can help you identify the purpose of your bot.

Translates the provided text in different languages using Amazon Translate.

Maximum 200 characters.

5. For the IAM role, choose **'Create a role with basic Amazon Lex permissions'** which will automatically create a role for you.
6. Choose **'No'** for the option Children's Online Privacy Protection Act (COPPA), leave the rest as default and click Next.


IAM permissions [Info](#)

IAM permissions are used to access other services on your behalf.

Runtime role
Choose a role that defines permissions for your bot. To create a customised role, use the IAM console.

☒ Create a role with basic Amazon Lex permissions.

☐ Use an existing role.

 Creating a role takes a few minutes. Don't delete the role or edit the trust or permissions policies in this role until we've finished creating it.

New role
Amazon Lex creates a runtime role with permission to upload to Amazon CloudWatch Logs.

AWSServiceRoleForLexV2Bots_AL31PFB4PBO

Children's Online Privacy Protection Act (COPPA) [Info](#)

Is use of your bot subject to the [Children's Online Privacy Protection Act \(COPPA\)](#) [🔗?](#)

☐ Yes

☒ No

7. You can choose the language you want to communicate with the bot and can also choose between different voice interaction modules.
8. The **Intent classification confidence score threshold** is like a confidence bar that the system uses to decide if it's sure enough about what the user meant. For now let's keep it default and click Done.

Add language to bot [Info](#)

▼ Language: English (US)

Select language

English (US) ▼

Description - *optional*

Maximum 200 characters.

Voice interaction

The text-to-speech voice that your bot uses to interact with users.

Danielle ▼

Voice sample

Hello, my name is Danielle. Let me know how I can assist you. Play

Intent classification confidence score threshold

0.40

Min: 0.00, max: 1.00.

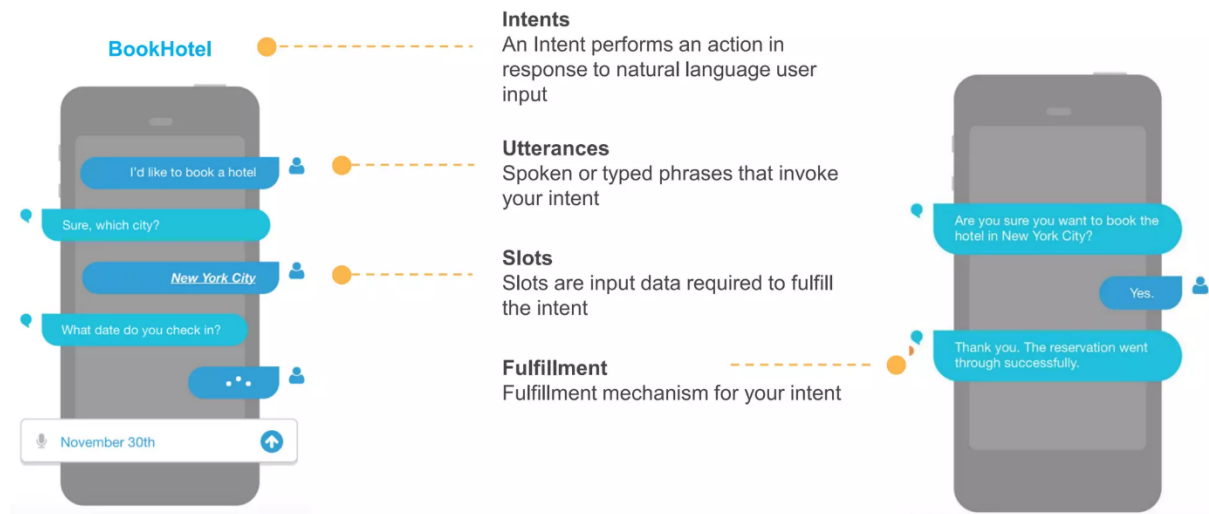
Cancel Add another language Done

9. Your empty chatbot has been created! You will lead on the Intent page where we will work on the conversational flow of the chatbot.

4.2B Specify Intents and slots

Specify Intent and Slots

Lex Bot Structure



1. Let's first understand some terms used in Amazon Lex:

- Intent: The goal or purpose behind what the user is saying.
- Utterance: The actual words or phrases spoken or typed by the user.
- Slots: Specific pieces of information or variables within the user's utterance that the system needs to extract.
- Fulfillment: The action or response that the system takes based on the user's intent and the information provided in the slots.

Take a look at this example:

In a pizza ordering chatbot, the intent reflects what the user wants, like ordering pizza or checking an order. Utterances are what users say, such as "I want a large pepperoni pizza." Slots are specific details in these utterances, like size and toppings. Fulfillment is the chatbot's response, confirming the order and providing information.

2. Now that we are familiar with the basic terms, let's start with building our chatbot.
3. In the Intent details, fill in an appropriate Intent name and description.

▼ Intent details [Info](#)

Intent name

TranslationIntent

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Intent and utterance generation description

Describe the purpose of your intent. This will also be used when generating utterances for your intent.

This intent will translate the text

Maximum 200 characters.



ID: WXFXTF2GQ0


4. In the **sample utterances**, add some inputs that can be asked by the user. Amazon Lex will try to understand the user input by aligning the input with the utterances provided.

Sample utterances (3) [Info](#)

[What's this?](#) [Generate utterances](#)

Representative phrases that you expect a user to speak or type to invoke this intent. Amazon Lex extrapolates based on the sample utterances to interpret any user input that may vary from the samples. The priority order of the sample utterances is not used to determine intent classification output.

 To generate utterances, you must have permissions to Amazon Bedrock. Amazon Lex will make calls to Amazon Bedrock. Additional charges may be incurred based on the usage of Amazon Bedrock. [Learn more](#) 



Sort by added (ascending) ▼

Preview

Plain Text

I want to translate

Can you help me translate

Translate for me

Add utterance

Maximum 250 characters.

We will add some more utterances later on after creation of slots.

- Go to the slots option and click on **Add Slot**. We need to take the language in which the text to be translated as a slot. For that we need to first create the Slot type. Go back and click on **Save Intent**.
- Click on back to Intents list and navigate to slot type, **add Slot type**.
- Choose '**Add blank slot type**' and give a name to the slot type (here '**language**').

Add blank slot type

Create a customisable slot type for your bot.

Slot type name

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Cancel Add

8. Add some languages as Slot type values similar to given below:

Slot type values

Modify the list of values used to train the machine-learning model to recognise values for a slot.

French	×
German	×
Chinese	×
Japanese	×
Spanish	×
Norwegian	×

Add value

Maximum 140 characters. Valid characters: A-Z, a-z, 0-9, @, #, \$

☐ Use slot values as custom vocabulary. [Info](#)

(Let us go with the same slot type values for now instead of adding new ones as there would be modifications required in the Lambda function too)

9. Click on **Save slot type**.

10. Navigate back to the Intents to use this slot type for our slot.

Add slot ×

A slot is used to capture information from the user to fulfil the intent.

☒ **Required for this intent**
The bot will prompt for this slot during the conversation if a value is not provided by the user.

Name	Slot type
<input type="text" value="language"/>	<input data-bbox="899 730 1435 785" type="text" value="language"/>

Prompts

Cancel Add

11. Click on **Add Slot**. Give the slot name and choose the slot type **'language'** previously made. Write the prompt where the chatbot asks for user choice for language translation.

12. Click on **Add**.

13. Create another slot **'text'** which takes the text to be translated as an input. Choose *AMAZON.FreeFormInput* as the slot type option and enter a suitable prompt asking for the text to be translated. Click on **Add**.

Add slot

×

A slot is used to capture information from the user to fulfil the intent.

☒ Required for this intent

The bot will prompt for this slot during the conversation if a value is not provided by the user.

Name

Slot type

AMAZON.FreeFormInput ▼

Prompts

Please input the text you want to translate.

Cancel

Add

14. We can add some more utterances specifying the Slots.

For example : Instead of 'I want to translate' , if user inputs 'I want to translate to French' with the language specified in the starting intent itself, it should not ask for the language again by running the language prompt. Instead we can specify Slots like below:

Sort by added (ascending) ▼

Preview

Plain Text

I want to translate

Can you help me translate

Translate for me

I want to translate in {language}

Translate in {language}

Can you translate in {language} for me?

Add utterance

Maximum 250 characters.

This will automatically understand the language slot type if already specified and ask for the input text to be translated directly.

15. We can also add an **Initial response** to the initial intent as a feedback message.

Initial response [Info](#)

You can provide messages to acknowledge the user's initial request. You can also configure the next step in the conversation and branch based on conditions.

▼ Response to acknowledge the user's request

Message: *Sure, I can help you with that !!*

▼ Message group [Info](#)

You can define a text message group to respond using plain text.

Message - *optional*

► Variations - *optional*

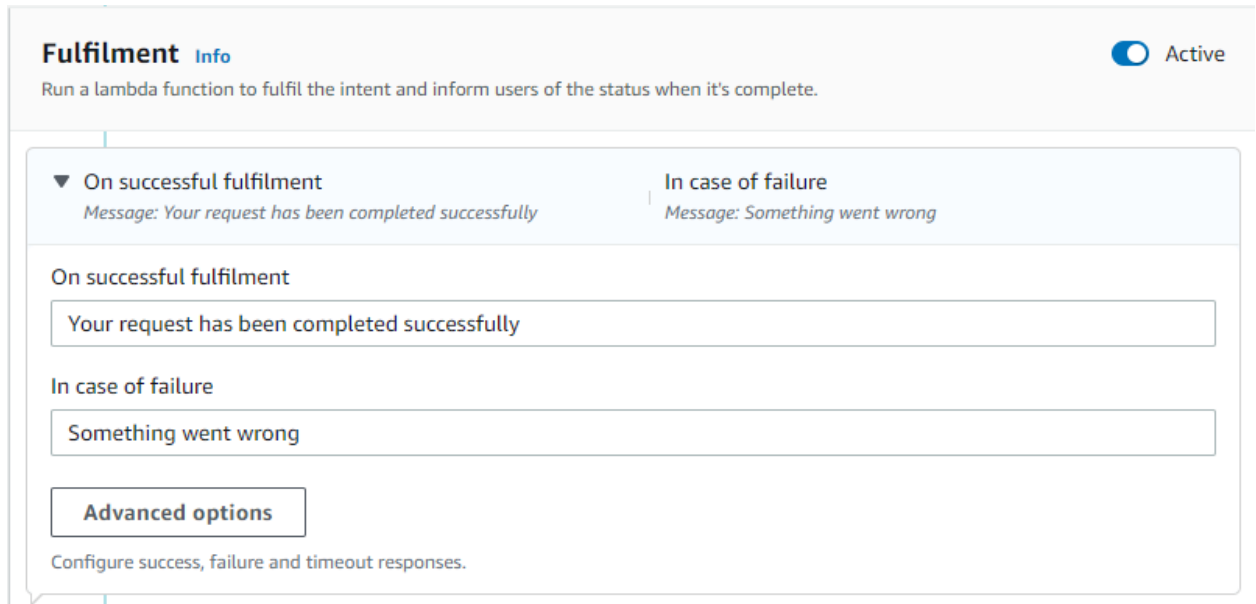
Advanced options

Configure user request acknowledgement response, dialogue code hook and conditional branches.

4.2C Specify Fulfilment

Specify Fulfilment

1. Fulfilment runs the Lambda function to fulfil the intent and informs users about it's status once it is complete.
2. Write a suitable prompt on successful fulfilment and in case of a failure.



The screenshot shows the 'Fulfilment' configuration page in the AWS Lambda console. At the top, there's a header with 'Fulfilment' and an 'Info' link, and a toggle switch labeled 'Active'. Below the header, a description states: 'Run a lambda function to fulfil the intent and inform users of the status when it's complete.' The main configuration area is divided into two sections: 'On successful fulfilment' and 'In case of failure'. Each section has a dropdown menu showing a default message (e.g., 'Your request has been completed successfully' and 'Something went wrong') and a text input field for a custom message. Below these sections is an 'Advanced options' button. At the bottom, a note says 'Configure success, failure and timeout responses.'

3. Click on '**Advanced Options**' and check '**Use a Lambda function for fulfilment**'. Click on **update options**.
4. We can also provide a **Closing message** after completion of the intent.

Closing response [Info](#)

Active

You can define the response when closing the intent.

▼ Response sent to the user after the intent is fulfilled

Message: I hope the translation was helpful!

▼ Message group [Info](#)

You can define a text message group to respond using plain text.

Message

I hope the translation was helpful!

► Variations - optional

More response options

Add customisable payloads, SSML and card groups.

► Set values

-

Next step in conversation

End conversation

+ Add conditional branching

5. Click on 'Save Intent'.

We have setup the conversation flow of our chatbot. Next, we will proceed to creating the Lambda function for actual serverless text translation.

4.2D Create an IAM role

Create an IAM role

1. From your **AWS management console**, navigate to **IAM** from your search bar.
2. Navigate to roles and **Create Role**. This role will be used for the Lambda function to provide basic Lambda execution permissions and access to Amazon Translate.
3. Select the **trusted entity type** as **AWS service** and select **Lambda** as the use case. Click on **Next**.

Select trusted entity [Info](#)

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a use case for the specified service.
 Use case
☒ **Lambda**
 Allows Lambda functions to call AWS services on your behalf.

Cancel **Next**

4. Attach the following policies to the role :
 - a. *TranslateFullAccess*
 - b. *AWSLambdaBasicExecutionRole*
5. Click on **Next**. Enter a suitable name and description for the role and **Create Role**.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and "+=, @-_" characters.

Description
Add a short explanation for this role.

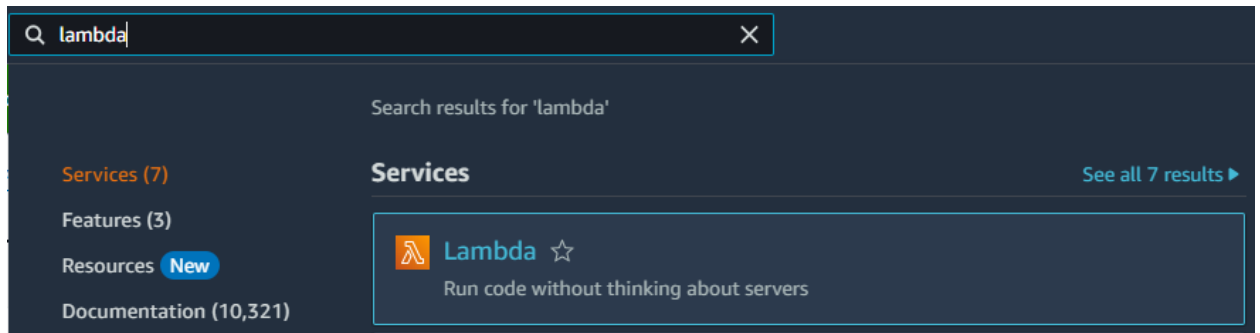
Maximum 1000 characters. Use alphanumeric and "+=, @-_" characters.

6. This role will be used for the Lambda function permission.

4.2E Create a Lambda function

Creating a Lambda function

1. From your AWS management console, search for Lambda from your search bar.



2. Click on **Create Function**. Give the function a suitable name and choose the runtime as **Python 3.12**.
3. Choose the previously created role by toggling the **default execution role** option. Keep the rest of the values as default and '**Create Function**'.
4. Let us first look at the **Lambda code** and we will understand the code below:

PYTHON

```
import boto3

def lambda_handler(event, context):
    try:
        input_text = event['sessionState']['intent']['slots']['text']['value']['interpretedValue'].strip()
        language_slot = event['sessionState']['intent']['slots']['language']['value']['interpretedValue']

        if not input_text:
            raise ValueError("Input text is empty.")

        language_codes = {
            'French': 'fr',
            'Japanese': 'ja',
            'Chinese': 'zh',
            'Spanish': 'es',
            'German': 'de',
            'Norwegian': 'no'
        }

        if language_slot not in language_codes:
            raise ValueError(f"Unsupported language: {language_slot}")

        target_language_code = language_codes[language_slot]

        # Initialize the Amazon Translate client
        translate_client = boto3.client('translate')

        # Call Amazon Translate to perform translation
        response = translate_client.translate_text(
            Text=input_text,
            SourceLanguageCode='auto', # Auto-detect source language

```

```
        response = translate_client.translate_text(
            Text=input_text,
            SourceLanguageCode='auto', # Auto-detect source language
            TargetLanguageCode=target_language_code
        )

        translated_text = response['TranslatedText']

        lex_response = {
            "sessionState": {
                "dialogAction": {
                    "type": "Close"
                },
                "intent": {
                    "name": "TranslateIntent", #Add your Intent Name
                    "state": "Fulfilled"
                }
            },
            "messages": [
                {
                    "contentType": "PlainText",
                    "content": translated_text
                }
            ]
        }

        return lex_response

    except Exception as error:
        error_message = "Lambda execution error: " + str(error)
        print(error_message)
        lex_error_response = {
            "sessionState": {
                "dialogAction": {
```



```

    },
    "messages": [
        {
            "contentType": "PlainText",
            "content": translated_text
        }
    ]
}

return lex_response

except Exception as error:
    error_message = "Lambda execution error: " + str(error)
    print(error_message)
    lex_error_response = {
        "sessionState": {
            "dialogAction": {
                "type": "Close"
            },
            "intent": {
                "name": "TranslateIntent",
                "state": "Fulfilled"
            }
        },
        "messages": [
            {
                "contentType": "PlainText",
                "content": error_message
            }
        ]
    }

return lex_error_response

```

a) **boto3** is used to interact with AWS services through code.

b) **Get Input:**

```

input_text = event['sessionState']['intent']['slots']['text']['value']['interpretedValue'].strip()
language_slot = event['sessionState']['intent']['slots']['language']['value']['interpretedValue']

```

Here, the code extracts the user's input text and the target language from the Lex event. If you have kept different slot names from the ones followed in this documentation, remember to change them accordingly.

c) **Language codes:**

```
language_codes = {
    'French': 'fr',
    'Japanese': 'ja',
    'Chinese': 'zh',
    'Spanish': 'es',
    'German': 'de',
    'Norwegian': 'no'
}
```

These are the language codes format used by Amazon Translate to identify the language to be used. We need to pass this language_code in the translate_text function to translate the input according to the mapped value of the language code.

You can further add more languages as you prefer, however remember to also add them in the slot type language.

d) Check Input:

```
if not input_text:
    raise ValueError("Input text is empty.")

if language_slot not in language_codes:
    raise ValueError(f"Unsupported language: {language_slot}")
```

These lines ensure that the input text is not empty and that the target language is supported.

e) Translate:

```
response = translate_client.translate_text(
    Text=input_text,
    SourceLanguageCode='auto', # Auto-detect source language
    TargetLanguageCode=target_language_code
)
translated_text = response['TranslatedText']
```

This section uses Amazon Translate to translate the input text into the target language specified by the user.

f) Prepare Response:

```
lex_response = {
    "sessionState": {
        "dialogAction": {
            "type": "Close"
        },
        "intent": {
            "name": "TranslateIntent", #Add your intent name
            "state": "Fulfilled"
        }
    },
    "messages": [
        {
            "contentType": "PlainText",
            "content": translated_text
        }
    ]
}
```

After translation, the code constructs a response for the chatbot with the translated text.

g) Handle Errors:

```
except Exception as error:
    error_message = "Lambda execution error: " + str(error)
    print(error_message)
    lex_error_response = {
        "sessionState": {
            "dialogAction": {
                "type": "Close"
            },
            "intent": {
                "name": "TranslateIntent",
                "state": "Fulfilled"
            }
        },
        "messages": [
            {
                "contentType": "PlainText",
                "content": error_message
            }
        ]
    }
    return lex_error_response
```

This part catches any errors that occur during the process and sends an appropriate error message back to the chatbot.

6. Click on '**Deploy**' to deploy the Lambda function.

4.2F Test the Lambda function

Testing the Lambda function

1. To test the Lambda function, click on **Test** and configure the **Test event**.
2. Give a suitable name to the test event and keep the Event JSON in the format below:

Test event action

☒ Create new event

☐ Edit saved event

Event name

translationtest

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

translationtest

Event JSON

Format JSON

```
1 {  
2   "sessionState": {  
3     "intent": {  
4       "name": "TranslateIntent",  
5     "slots": {  
6       "text": {  
7         "value": {  
8           "interpretedValue": "Hello",  
9           "originalValue": "Hello"  
10        }  
11      },  
12     "language": {  
13       "value": {  
14         "interpretedValue": "French",  
15         "originalValue": "French"  
16       }  
17     }  
18   }  
19 }  
20 }  
21 }
```

Cancel

Invoke

Save

```
PYTHON
{
  "sessionState": {
    "intent": {
      "name": "TranslateIntent",
      "slots": {
        "text": {
          "value": {
            "interpretedValue": "Hello",
            "originalValue": "Hello"
          }
        }
      }
    },
    "language": {
      "value": {
        "interpretedValue": "French",
        "originalValue": "French"
      }
    }
  }
}
}
```

Testing the Lambda function

The reason that we are providing this format is because your Lambda function gets an input from the Amazon Lex in this specified format.

3. Save the test event and click on Test again to Invoke the event.

The screenshot shows the AWS Lambda console interface. The 'Test' tab is selected, displaying the execution results for a test event named 'translationTest'. The response is a JSON object representing a session state with intent and language information. The function logs show the start and end of the execution, including the request ID, duration, and memory usage.

Test Event Name	Response	Function Logs
translationTest	<pre>{ "sessionState": { "dialogAction": { "type": "Close" }, "intent": { "name": "TranslationIntent", "state": "Fulfilled" }, "messages": [{ "contentType": "PlainText", "content": "Bonjour" }] } }</pre>	<pre>START RequestId: 720b5156-7a7c-4c96-b61c-fbbf06f6e4da Version: \$LATEST END RequestId: 720b5156-7a7c-4c96-b61c-fbbf06f6e4da REPORT RequestId: 720b5156-7a7c-4c96-b61c-fbbf06f6e4da Duration: 2119.15 ms Billed Duration: 2120 ms Memory Size: 128 MB Max Memory Used: 74 MB Init Duration: 303.98 ms</pre>

The function gives an output in a JSON format which is the required format to give an input for Amazon Lex. It will get the message content and display it to the user.

Next, let's test out the chatbot working.

4.2G Test the chatbot

Testing the chatbot

1. Navigate to the **Intent page** of your previously created Chatbot. Before testing it, we need to specify the Lambda function to be used by Lex.
2. Click on the **settings** option present on your top right of the chatbot window and choose the **Source of Lambda function** as your previous created function.
3. Now you are ready to go! Test out the chatbot, here are some conversations:

Test Draft version

Last build submitted: 2 hours ago



Inspect

Can you help me translate?

Sure, I can help you with that !!

In which language would you like to translate the text?

French

Please input the text you want to translate.

Hello

Bonjour

✓ Ready for complete testing



Type a message

Test Draft version

Last build submitted: 2 hours ago



Inspect

I want to translate in Japanese

Sure, I can help you with that !!

Please input the text you want to translate.

Hello how are you

こんにちは、元気ですか

✓ Ready for complete testing



Type a message