
Amazon Lex

V2 Developer Guide



Amazon Lex: V2 Developer Guide

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon Lex?	1
Are You a First-time User of Amazon Lex?	2
How it works	3
Supported languages	4
Supported languages and locales	4
Languages and locales supported by Amazon Lex features	5
Getting started	6
Step 1: Set Up an Account	6
Sign Up for AWS	6
Create an IAM User	7
Next step	7
Step 2: Getting started (Console)	7
Exercise 1: Create a bot from an example	8
Exercise 2: Review the conversation flow	9
Building bots	16
Creating a bot	16
Adding a language	17
Adding intents	17
Adding slot types	18
Testing a bot	19
Creating versions	22
The Draft version	22
Creating a version	23
Updating an Amazon Lex bot	23
Deleting an Amazon Lex bot or version	23
Built-in intents and slot types	23
Built-in intents	23
Built-in slot types	34
Creating custom slot types	41
Using multiple values in a slot	41
Using a Lambda function	43
Attaching a Lambda function to a bot alias	43
Input event format	44
Response format	47
Deploying bots	50
Aliases	50
Deploying on a message platform	51
Integrating with Facebook	52
Integrating with Slack	54
Integrating with Twilio SMS	57
Integrating with a Java application	59
Using bots	62
Managing conversations	62
Managing conversation context	63
Setting intent context	63
Using default slot values	64
Setting session attributes	65
Setting request attributes	66
Setting the session timeout	67
Sharing information between intents	67
Setting complex attributes	68
Managing sessions	69
Starting a new session	70
Switching intents	70

Resuming a prior intent	70
Validating slot values	71
Analyzing sentiment	71
Using confidence scores	72
Session Management	73
Streaming conversations	75
Starting a stream to a bot	76
Time sequence of events for an audio conversation	78
Starting a streaming conversation	80
Enabling your bot to be interrupted	91
Waiting for the user to provide additional information	92
Event stream encoding	93
Monitoring	95
Monitoring with conversation logs	95
Configuring conversation logs	95
Viewing text logs in Amazon CloudWatch Logs	97
Accessing audio logs in Amazon S3	100
Monitoring conversation log status with CloudWatch metrics	100
Obscuring slot values	101
Logging with CloudTrail	101
Amazon Lex information in CloudTrail	102
Understanding Amazon Lex log file entries	102
Monitoring with CloudWatch	103
Importing and exporting bots	108
Exporting a bot or bot locale	108
IAM permissions required to export	109
Exporting a bot (console)	109
Importing a bot or bot locale	110
IAM permissions required to import	111
Importing a bot (console)	111
Using a password when importing or exporting	112
JSON format for importing and exporting	113
Manifest file structure	113
Bot file structure	114
Bot locale file structure	114
Intent file structure	114
Slot file structure	115
Slot type file structure	116
Tagging resources	118
Tagging your resources	118
Tag restrictions	118
Tagging resources (console)	119
Security	120
Data protection	120
Encryption at rest	121
Encryption in transit	121
Identity and access management	122
Audience	122
Authenticating with identities	122
Managing access using policies	124
How Amazon Lex works with IAM	126
Identity-based policy examples	133
Resource-based policy examples	139
Troubleshooting	146
Using service-linked roles	148
Service-linked role permissions for Amazon Lex	148
Creating a service-linked role for Amazon Lex	149

Editing a service-linked role for Amazon Lex	150
Deleting a service-linked role for Amazon Lex	150
Supported regions for Amazon Lex service-linked roles	151
Logging and monitoring	151
Compliance validation	151
Resilience	152
Infrastructure security	152
Migration guide	153
Amazon Lex V2 overview	153
Multiple languages in a bot	153
Simplified information architecture	153
Improved builder productivity	153
Guidelines and quotas	155
Regions	155
General guidelines	155
Quotas	156
Build-time quotas	156
Runtime quotas	157
Document history	159
API reference	161
Actions	161
Amazon Lex Model Building V2	162
Amazon Lex Runtime V2	400
Data Types	431
Amazon Lex Model Building V2	434
Amazon Lex Runtime V2	529
Common Errors	567
Common Parameters	569
AWS glossary	572

What is Amazon Lex?

Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text. Amazon Lex provides the deep functionality and flexibility of natural language understanding (NLU) and automatic speech recognition (ASR) so you can build highly engaging user experiences with lifelike, conversational interactions, and create new categories of products.

Amazon Lex enables any developer to build conversational bots quickly. With Amazon Lex, no deep learning expertise is necessary—to create a bot, you specify the basic conversation flow in the Amazon Lex console. Amazon Lex manages the dialog and dynamically adjusts the responses in the conversation. Using the console, you can build, test, and publish your text or voice chatbot. You can then add the conversational interfaces to bots on mobile devices, web applications, and chat platforms (for example, Facebook Messenger).

Amazon Lex provides integration with AWS Lambda, and you can integrate with many other services on the AWS platform, including Amazon Connect, Amazon Comprehend, and Amazon Kendra. Integration with Lambda provides bots access to pre-built serverless enterprise connectors to link to data in SaaS applications such as Salesforce.

Amazon Lex provides the following benefits:

- **Simplicity** – Amazon Lex guides you through using the console to create your own bot in minutes. You supply a few example phrases, and Amazon Lex builds a complete natural language model through which the bot can interact using voice and text to ask questions, get answers, and complete sophisticated tasks.
- **Democratized deep learning technologies** – Amazon Lex provides ASR and NLU technologies to create a Speech Language Understanding (SLU) system. Through SLU, Amazon Lex takes natural language speech and text input, understands the intent behind the input, and fulfills the user intent by invoking the appropriate business function.

Speech recognition and natural language understanding are some of the most challenging problems to solve in computer science, requiring sophisticated deep learning algorithms to be trained on massive amounts of data and infrastructure. Amazon Lex puts deep learning technologies within reach of all developers. Amazon Lex bots convert incoming speech to text and understand the user intent to generate an intelligent response so you can focus on building your bots with added value for your customers and define entirely new categories of products made possible through conversational interfaces.

- **Seamless deployment and scaling** – With Amazon Lex, you can build, test, and deploy your bots directly from the Amazon Lex console. Amazon Lex enables you to publish your voice or text bots for use on mobile devices, web apps, and chat services (for example, Facebook Messenger). Amazon Lex scales automatically. You don't need to worry about provisioning hardware and managing infrastructure to power your bot experience.
- **Built-in integration with the AWS platform** – Amazon Lex operates natively with other AWS services, such as AWS Lambda and Amazon CloudWatch. You can take advantage of the power of the AWS platform for security, monitoring, user authentication, business logic, storage, and mobile app development.

- **Cost-effectiveness** – With Amazon Lex, there are no upfront costs or minimum fees. You are charged only for the text or speech requests that are made. The pay-as-you-go pricing and the low cost per request make the service a cost-effective way to build conversational interfaces. With the Amazon Lex free tier, you can easily try Amazon Lex without any initial investment.

Are You a First-time User of Amazon Lex?

If you are a first-time user of Amazon Lex, we recommend that you read the following sections in order:

1. [How it works \(p. 3\)](#) – This section introduces Amazon Lex and the features that you use to create a chatbot.
2. [Getting started with Amazon Lex \(p. 6\)](#) – In this section, you set up your account and test Amazon Lex.
3. [API reference \(p. 161\)](#) – This section provides additional examples that you can use to explore Amazon Lex.

How it works

Amazon Lex enables you to build applications using a text or speech interface for a conversation with a user. Following are the typical steps for working with Amazon Lex:

1. Create a bot and add one or more languages. Configure the bot so that it understands the user's goal, engages in conversation with the user to elicit information, and fulfills the user's intent.
2. Test the bot. You can use the test window client provided by the Amazon Lex console.
3. Publish a version and create an alias.
4. Deploy the bot. You can deploy the bot on your own applications or messaging platforms such as Facebook Messenger or Slack

Before you get started, familiarize yourself with the following Amazon Lex core concepts and terminology:

- **Bot** – A bot performs automated tasks such as ordering a pizza, booking a hotel, ordering flowers, and so on. An Amazon Lex bot is powered by automatic speech recognition (ASR) and natural language understanding (NLU) capabilities.

Amazon Lex bots can understand user input provided with text or speech and converse natural language.

- **Language** – An Amazon Lex bot can converse in one or more languages. Each language is independent of the others, you can configure Amazon Lex to converse with a user using native words and phrases. For more information, see [Languages and locales supported by Amazon Lex \(p. 4\)](#).
- **Intent** – An intent represents an action that the user wants to perform. You create a bot to support one or more related intents. For example, you might create an intent that orders pizzas and drinks. For each intent, you provide the following required information:
 - **Intent name** – A descriptive name for the intent. For example, `OrderPizza`.
 - **Sample utterances** – How a user might convey the intent. For example, a user might say "Can I order a pizza" or "I want to order a pizza."
 - **How to fulfill the intent** – How you want to fulfill the intent after the user provides the necessary information. We recommend that you create a Lambda function to fulfill the intent.

You can optionally configure the intent so Amazon Lex returns the information back to the client application for the necessary fulfillment.

In addition to custom intents, Amazon Lex provides built-in intents to quickly set up your bot. For more information, see [Built-in intents and slot types \(p. 23\)](#).

Amazon Lex always includes a fallback intent for each bot. The fallback intent is used whenever Amazon Lex can't deduce the user's intent. For more information, see [AMAZON.FallbackIntent \(p. 24\)](#).

- **Slot** – An intent can require zero or more slots, or parameters. You add slots as part of the intent configuration. At runtime, Amazon Lex prompts the user for specific slot values. The user must provide values for all required slots before Amazon Lex can fulfill the intent.

For example the `OrderPizza` intent requires slots such as size, crust type, and number of pizzas. For each slot, you provide the slot type and one or more prompts that Amazon Lex sends to the client to elicit values from the user. A user can reply with a slot value that contains additional words, such as "large pizza please" or "let's stick with small." Amazon Lex still understands the slot value.

- **Slot type** – Each slot has a type. You can create your own slot type, or you can use built-in slot types. For example, you might create and use the following slot types for the OrderPizza intent:
 - Size – With enumeration values Small, Medium, and Large.
 - Crust – With enumeration values Thick and Thin.
- Amazon Lex also provides built-in slot types. For example, AMAZON.Number is a built-in slot type that you can use for the number of pizzas ordered. For more information, see [Built-in intents and slot types \(p. 23\)](#).
- **Version** – A version is a numbered snapshot of your work that you can publish for use in different parts of your workflow, such as development, beta deployment, and production. Once you create a version, you can use a bot as it existed when the version was made. After you create a version, it stays the same while you continue to work on your application.
- **Alias** – An alias is a pointer to a specific version of a bot. With an alias, you can update the version the your client applications are using. For example, you can point an alias to version 1 of your bot. When you are ready to update the bot, you publish version 2 and change the alias to point to the new version. Because your applications use the alias instead of a specific version, all of your clients get the new functionality without needing to be updated.

For a list of the AWS Regions where Amazon Lex is available, see [Amazon Lex endpoints and quotas](#) in the *Amazon Web Services General Reference*.

Languages and locales supported by Amazon Lex

Amazon Lex supports a variety of languages and locales. The languages supported and the features they support are listed in the following tables.

Supported languages and locales

Amazon Lex supports the following languages and locales.

Code	Language and locale
de_DE	German (Germany)
en_AU	English (Australia)
en_GB	English (UK)
en_IN	English (India)
en_US	English (US)
es_419	Spanish (Latin America)
es_ES	Spanish (Spain)
es_US	Spanish (US)
fr_CA	French (Canada)
fr_FR	French (France)
it_IT	Italian (Italy)
ja_JP	Japanese (Japan)

Languages and locales supported by Amazon Lex features

All Amazon Lex features are supported in all languages and locales except as listed in this table.

Feature	Supported languages and locales
AMAZON.KendraSearchIntent (p. 26)	English (US) (en_US)
Setting intent context (p. 63)	English (US) (en_US)
Using multiple values in a slot (p. 41)	English (US) (en_US)

Getting started with Amazon Lex

Amazon Lex provides API operations that you can integrate with your existing applications. For a list of supported operations, see the [API reference \(p. 161\)](#). You can use any of the following options:

- AWS SDK — When using the SDKs your requests to Amazon Lex are automatically signed and authenticated using the credentials that you provide. We recommend that you use an SDK to build your application.
- AWS CLI — You can use the AWS CLI to access any Amazon Lex feature without having to write any code.
- AWS Console — The console is the easiest way to get started testing and using Amazon Lex

If you are new to Amazon Lex, we recommend that you read [How it works \(p. 3\)](#) first.

Topics

- [Step 1: Set Up an AWS Account and Create an Administrator User \(p. 6\)](#)
- [Step 2: Getting started \(Console\) \(p. 7\)](#)

Step 1: Set Up an AWS Account and Create an Administrator User

Before you use Amazon Lex for the first time, complete the following tasks:

1. [Sign Up for AWS \(p. 6\)](#)
2. [Create an IAM User \(p. 7\)](#)

Sign Up for AWS

If you already have an AWS account, skip this task.

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon Lex. You are charged only for the services that you use.

With Amazon Lex, you pay only for the resources that you use. If you are a new AWS customer, you can get started with Amazon Lex for free. For more information, see [AWS Free Usage Tier](#).

If you already have an AWS account, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Write down your AWS account ID because you'll need it for the next task.

Create an IAM User

Services in AWS, such as Amazon Lex, require that you provide credentials when you access them so that the service can determine whether you have permissions to access the resources owned by that service. The console requires your password. You can create access keys for your AWS account to access the AWS CLI or API.

However, we don't recommend that you access AWS using the credentials for your AWS account. Instead, we recommend that you:

- Use AWS Identity and Access Management (IAM) to create an IAM user
- Add the user to an IAM group with administrative permissions
- Grant administrative permissions to the IAM user that you created.

You can then access AWS using a special URL and the IAM user's credentials.

The Getting Started exercises in this guide assume that you have a user (`adminuser`) with administrator privileges. Follow the procedure to create `adminuser` in your account.

To create an administrator user and sign in to the console

1. Create an administrator user called `adminuser` in your AWS account. For instructions, see [Creating Your First IAM User and Administrators Group](#) in the *IAM User Guide*.
2. As a user, you can sign in to the AWS Management Console using a special URL. For more information, [How Users Sign In to Your Account](#) in the *IAM User Guide*.

For more information about IAM, see the following:

- [AWS Identity and Access Management \(IAM\)](#)
- [Getting started](#)
- [IAM User Guide](#)

Next step

[Step 2: Getting started \(Console\) \(p. 7\)](#)

Step 2: Getting started (Console)

The easiest way to learn how to use Amazon Lex is by using the console. To get you started, we created the following exercises, all of which use the console:

- Exercise 1 — Create an Amazon Lex bot using a blueprint, a predefined bot that provides all of the necessary bot configuration. You do only a minimum of work to test the end-to-end setup.
- Exercise 2 — Review the JSON structures sent between your client application and an Amazon Lex bot.

Topics

- [Exercise 1: Create a bot from an example \(p. 8\)](#)
- [Exercise 2: Review the conversation flow \(p. 9\)](#)

Exercise 1: Create a bot from an example

In this exercise, you create your first Amazon Lex bot and test it in the Amazon Lex console. For this exercise, you use the **OrderFlowers** example.

Example overview

You use the **OrderFlowers** example to create an Amazon Lex bot. For more information about the structure of a bot, see [How it works \(p. 3\)](#).

- **Intent** – OrderFlowers
- **Slot types** – One custom slot type called `FlowerTypes` with enumeration values: `roses`, `lilies`, and `tulips`.
- **Slots** – The intent requires the following information (that is, slots) before the bot can fulfill the intent.
 - `PickupTime` (AMAZON.TIME built-in type)
 - `FlowerType` (`FlowerTypes` custom type)
 - `PickupDate` (AMAZON.DATE built-in type)
- **Utterance** – The following sample utterances indicate the user's intent:
 - "I would like to pick up flowers."
 - "I would like to order some flowers."
- **Prompts** – After the bot identifies the intent, it uses the following prompts to fill the slots:
 - Prompt for the `FlowerType` slot – "What type of flowers would you like to order?"
 - Prompt for the `PickupDate` slot – "What day do you want the {`FlowerType`} to be picked up?"
 - Prompt for the `PickupTime` slot – "At what time do you want the {`FlowerType`} to be picked up?"
 - Confirmation statement – "Okay, your {`FlowerType`} will be ready for pickup by {`PickupTime`} on {`PickupDate`}. Does this sound okay?"

To create an Amazon Lex bot (Console)

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/home>
2. Choose **Create bot**.
3. For the **Creation method**, choose **Start with an example**.
4. In the **Example bots** section, choose **OrderFlowers** from the list.
5. In the **Bot configuration** section give the bot a name and a description. The name must be unique in your account.
6. In the **Permissions** section, choose **Create a new role with basic Amazon Lex permissions**. This will create an AWS Identity and Access Management (IAM) role with the permissions that Amazon Lex needs to run your bot.
7. In the **Children's Online Privacy Protection Act (COPPA)** section, make the appropriate choice.
8. In the **Session timeout** and **Advanced settings** sections, leave the defaults.
9. Choose **Next**. Amazon Lex creates your bot.

After you create your bot, you must add one or more languages that the bot supports. A language contains the intents, slot types, and slots that the bot uses to converse with users.

To add a language to a bot

1. In the **Language** section, choose a supported language, and add a description.

2. Leave the **Voice interaction** and **Intent classification confidence score threshold** fields with their defaults.
3. Choose **Add language** to add the language to the bot.
4. After the language is added, choose **Done** to continue.

After you choose **Done**, the console opens the intent editor. You can use the intent editor to examine the intents used by the bot. When you are done examining the bot, you can test to bot.

To test the OrderFlowers bot

1. From the bottom menu, choose **Build**. Wait for the bot to build.
2. When the build is complete, choose **Test** to open the test window.
3. Test the bot. Start the conversation with one of the sample utterances, such as "I would like to pick up flowers."

Next steps

Now that you've created your first bot using a template, you can use the console to create your own bot. For instruction on creating a custom bot, and for more information about creating bots, see [Building bots \(p. 16\)](#).

Exercise 2: Review the conversation flow

In this exercise you review the JSON structures that are sent between your client application and the Amazon Lex bot that you created in [Exercise 1: Create a bot from an example \(p. 8\)](#). The conversation uses the [RecognizeText \(p. 413\)](#) operation to generate the JSON structures. The [RecognizeUtterance \(p. 419\)](#) returns the same information as HTTP headers in the response.

The JSON structures are divided by each turn of the conversation. A *turn* is a request from the client application and a response from the bot.

Turn 1

During the first turn of the conversation, the client application initiates the conversation with your bot. Both the URI and the body of the request provide information about the request.

```
POST /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId/text HTTP/1.1
Content-type: application/json

{
    "text": "I would like to order flowers"
}
```

- The URI identifies the bot that the client application is communicating with. It also includes a session identifier generated by the client application that identifies a specific conversation between a user and the bot.
- The body of the request contains the text that the user typed to the client application. In this case, only the text is sent, however your application can send additional information, such as request attributes or session state. For more information, see the [RecognizeText \(p. 413\)](#) operation.

From `text`, Amazon Lex detects the user's intent, to order flowers. Amazon Lex chooses one of the intent's slots (`FlowerType`) and one of the prompts for the slot, and then sends the following response to the client application. The client displays the response to the user.

```
{  
    "interpretations": [  
        {  
            "intent": {  
                "confirmationState": "None",  
                "name": "OrderFlowers",  
                "slots": {  
                    "FlowerType": null,  
                    "PickupDate": null,  
                    "PickupTime": null  
                },  
                "state": "InProgress"  
            },  
            "nluConfidence": {  
                "score": 0.95  
            }  
        },  
        {  
            "intent": {  
                "name": "FallbackIntent",  
                "slots": {}  
            }  
        }  
    ],  
    "messages": [  
        {  
            "content": "What type of flowers would you like to order?",  
            "contentType": "PlainText"  
        }  
    ],  
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",  
    "sessionState": {  
        "dialogAction": {  
            "slotToElicit": "FlowerType",  
            "type": "ElicitSlot"  
        },  
        "intent": {  
            "confirmationState": "None",  
            "name": "OrderFlowers",  
            "slots": {  
                "FlowerType": null,  
                "PickupDate": null,  
                "PickupTime": null  
            },  
            "state": "InProgress"  
        },  
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"  
    }  
}
```

Turns 2 and 3

In turn 2 and 3, the user responds to prompts from the Amazon Lex bot with values that fill the `FlowerType` and `PickupDate` slots.

The URI for the second and third turns is the same as the first.

```
{  
    "text": "1 dozen roses"  
}
```

The response for turn 2 shows the `FlowerType` slot filled and provides a prompt to elicit the next slot value.

```
{  
    "interpretations": [  
        {  
            "intent": {  
                "confirmationState": "None",  
                "name": "OrderFlowers",  
                "slots": {  
                    "FlowerType": {  
                        "value": {  
                            "interpretedValue": "dozen roses",  
                            "originalValue": "dozen roses",  
                            "resolvedValues": []  
                        }  
                    },  
                    "PickupDate": null,  
                    "PickupTime": null  
                },  
                "state": "InProgress"  
            },  
            "nluConfidence": {  
                "score": 0.98  
            }  
        },  
        {  
            "intent": {  
                "name": "FallbackIntent",  
                "slots": {}  
            }  
        }  
    ],  
    "messages": [  
        {  
            "content": "What day do you want the dozen roses to be picked up?",  
            "contentType": "PlainText"  
        }  
    ],  
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",  
    "sessionState": {  
        "dialogAction": {  
            "slotToElicit": "PickupDate",  
            "type": "ElicitSlot"  
        },  
        "intent": {  
            "confirmationState": "None",  
            "name": "OrderFlowers",  
            "slots": {  
                "FlowerType": {  
                    "value": {  
                        "interpretedValue": "dozen roses",  
                        "originalValue": "dozen roses",  
                        "resolvedValues": []  
                    }  
                },  
                "PickupDate": null,  
                "PickupTime": null  
            },  
            "state": "InProgress"  
        },  
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"  
    }  
}
```

Turn 4

In turn 4, the user provides the final slot value for the intent, the time that the flowers are picked up.

```
{  
    "text": "5 in the evening"  
}
```

In the response, Amazon Lex sends a confirmation prompt to the user to confirm that the order is correct. The dialogAction is set to ConfirmIntent and the confirmationState is None.

```
{  
    "interpretations": [  
        {  
            "intent": {  
                "confirmationState": "None",  
                "name": "OrderFlowers",  
                "slots": {  
                    "FlowerType": {  
                        "value": {  
                            "interpretedValue": "dozen roses",  
                            "originalValue": "dozen roses",  
                            "resolvedValues": []  
                        }  
                    },  
                    "PickupDate": {  
                        "value": {  
                            "interpretedValue": "2021-01-04",  
                            "originalValue": "next monday",  
                            "resolvedValues": [  
                                "2021-01-04"  
                            ]  
                        }  
                    },  
                    "PickupTime": {  
                        "value": {  
                            "interpretedValue": "17:00",  
                            "originalValue": "5 evening",  
                            "resolvedValues": [  
                                "17:00"  
                            ]  
                        }  
                    },  
                    "state": "InProgress"  
                },  
                "nluConfidence": {  
                    "score": 1.0  
                }  
            },  
            {  
                "intent": {  
                    "name": "FallbackIntent",  
                    "slots": {}  
                }  
            }  
        ],  
        "messages": [  
            {
```

```

        "content": "Okay, your dozen roses will be ready for pickup by 17:00 on
2021-01-04. Does this sound okay?",  

        "contentType": "PlainText"
    }
],  

"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",  

"sessionState": {
    "dialogAction": {
        "type": "ConfirmIntent"
    },
    "intent": {
        "confirmationState": "None",
        "name": "OrderFlowers",
        "slots": {
            "FlowerType": {
                "value": {
                    "interpretedValue": "dozen roses",
                    "originalValue": "dozen roses",
                    "resolvedValues": []
                }
            },
            "PickupDate": {
                "value": {
                    "interpretedValue": "2021-01-04",
                    "originalValue": "next monday",
                    "resolvedValues": [
                        "2021-01-04"
                    ]
                }
            },
            "PickupTime": {
                "value": {
                    "interpretedValue": "17:00",
                    "originalValue": "5 evening",
                    "resolvedValues": [
                        "17:00"
                    ]
                }
            }
        },
        "state": "InProgress"
    },
    "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
}
}

```

Turn 5

In the final turn, the user responds with to the confirmation prompt.

```
{
    "text": "yes"
}
```

In the response, Amazon Lex sends indicates that the intent has been fulfilled by setting the confirmationState to Confirmed and the dialogAction to close. All of the slot values are available to the client application.

```
{
    "interpretations": [

```

```
{  
    "intent": {  
        "confirmationState": "Confirmed",  
        "name": "OrderFlowers",  
        "slots": {  
            "FlowerType": {  
                "value": {  
                    "interpretedValue": "dozen roses",  
                    "originalValue": "dozen roses",  
                    "resolvedValues": []  
                }  
            },  
            "PickupDate": {  
                "value": {  
                    "interpretedValue": "2021-01-04",  
                    "originalValue": "next monday",  
                    "resolvedValues": [  
                        "2021-01-04"  
                    ]  
                }  
            },  
            "PickupTime": {  
                "value": {  
                    "interpretedValue": "17:00",  
                    "originalValue": "5 evening",  
                    "resolvedValues": [  
                        "17:00"  
                    ]  
                }  
            }  
        },  
        "state": "Fulfilled"  
    },  
    "nluConfidence": {  
        "score": 1.0  
    }  
},  
{  
    "intent": {  
        "name": "FallbackIntent",  
        "slots": {}  
    }  
},  
],  
"messages": [  
    {  
        "content": "Thanks. ",  
        "contentType": "PlainText"  
    }  
],  
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",  
"sessionState": {  
    "dialogAction": {  
        "type": "Close"  
    },  
    "intent": {  
        "confirmationState": "Confirmed",  
        "name": "OrderFlowers",  
        "slots": {  
            "FlowerType": {  
                "value": {  
                    "interpretedValue": "dozen roses",  
                    "originalValue": "dozen roses",  
                    "resolvedValues": []  
                }  
            },  
        }  
    }  
},  
14
```

```
"PickupDate": {  
    "value": {  
        "interpretedValue": "2021-01-04",  
        "originalValue": "next monday",  
        "resolvedValues": [  
            "2021-01-04"  
        ]  
    }  
},  
"PickupTime": {  
    "value": {  
        "interpretedValue": "17:00",  
        "originalValue": "5 evening",  
        "resolvedValues": [  
            "17:00"  
        ]  
    }  
},  
"state": "Fulfilled"  
},  
"originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"  
}  
}
```

Building bots

You create an Amazon Lex bot to interact with your users to elicit information to accomplish a task. For example, you can create a bot that gathers the information needed to order a bouquet of flowers or book a hotel room.

To build a bot, you need the following information:

1. The language that the bot uses to interact with the customer. You can choose one or more languages, each language contains independent intents, slots, and slot types.
2. The intents, or goals, that the bot will help the user fulfill. A bot can contain one or more intents, such as ordering flowers, or booking a hotel and rental car. You need to decide which statements, or utterances, that the user makes to trigger the intent.
3. The information, or slots, that you need to gather from the user to fulfill an intent. For example, you might need to get the type of flowers from the user or the start date of a hotel reservation. You need to define one or more prompts that Amazon Lex uses to elicit the slot value from the user.
4. The type of the slots that you need from the user. You may need to create a custom slot type, such as a list of flowers that a user can order, or you can use a built-in slot type, such as using the AMAZON.Date slot type for the start date of a reservation.
5. The interactions between user and intents. Amazon Lex manages the interactions; you can create an AWS Lambda function to validate and fulfill the intent.

Topics

- [Creating a bot \(p. 16\)](#)
- [Adding a language \(p. 17\)](#)
- [Adding intents \(p. 17\)](#)
- [Adding slot types \(p. 18\)](#)
- [Testing a bot using the console \(p. 19\)](#)
- [Creating versions \(p. 22\)](#)
- [Built-in intents and slot types \(p. 23\)](#)
- [Creating custom slot types \(p. 41\)](#)
- [Using multiple values in a slot \(p. 41\)](#)
- [Using an AWS Lambda function \(p. 43\)](#)

Creating a bot

Start creating your bot by defining the name, description and some basic information.

To create a bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/home>
2. Choose **Create bot**.
3. In the **Creation method** section, choose **Create**.
4. In the **Bot configuration** section, give the bot a name and an optional description.

5. In the **IAM permissions** section, choose an AWS Identity and Access Management (IAM) role that provides Amazon Lex permission to access other AWS services, such as Amazon CloudWatch. You can have Amazon Lex create the role, or you can choose an existing role with CloudWatch permissions.
6. In the **Children's Online Privacy Protection Act (COPPA)** section, choose the appropriate response.
7. In the **Idle session timeout** section, choose the duration that Amazon Lex keeps a session with a user open. Amazon Lex maintains session variables for the duration of the session so that your bot can resume a conversation with the same variables.
8. In the **Advanced settings** section add tags that help identify the bot, and can be used to control access and monitor resources.
9. Choose **Next** to create the bot and move to adding a language.

Adding a language

You add one or more languages and locales to your bot to enable it to communicate with users in their languages. You define the intents, slots, and slot types separately for each language so that the utterances, prompts, and slot values are specific to the language.

Your bot must contain at least one language.

To add a language to your bot

1. In the **New language** section, choose the language that you want to use. You can add a description to help identify the language in lists.
2. If your bot supports voice interaction, in the **Voice interaction** section, choose the Amazon Polly voice that Amazon Lex uses to communicate with the user. If your bot doesn't support voice, choose **None**.
3. For the **Intent classification confidence score threshold**, set the value that Amazon Lex uses to determine whether an intent is the correct intent. You can adjust this value after testing your bot.
4. Choose **Add**.

Adding intents

Intents are the goals that your users want to fulfill, such as ordering flowers or booking a hotel. Your bot needs to have at least one intent.

By default, all bots contain a single built-in intent, the fallback intent. This intent is used when Amazon Lex does not recognize any other intent. For example, if a user says "I want to order flowers" to a hotel booking intent, the fallback intent is triggered.

To add an intent

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. From the list of bots, choose the bot that you want to add the intent to, then from **Add languages** choose **View languages**.
3. Choose the language to add the intent to, then choose **Intents**.
4. Choose **Add intent**, give your intent a name, and then choose **Add**.
5. In the intent editor, add the details of your intent.
 - **Conversation flow** – Use the conversation flow diagram to see how a dialog with your bot might look. You can choose different sections of the conversation to jump to that section of the intent editor.

- **Intent details** – Give the intent a name and description to help identify the purpose of the intent. You can also see the unique identifier that Amazon Lex assigned to the intent.
 - **Contexts** – Set the input and output contexts for the intent. A context is a state variable associated with an intent. An output context is set when an intent is fulfilled. An intent with an input context can only be recognized with the context is active. An intent with no input contexts can always be recognized.
 - **Sample utterances** – You should provide 10 or more phrases that you expect your users to use to trigger an intent. Amazon Lex generalizes from these phrases to recognize that the user wants to trigger the intent.
 - **Slots** – Define the slots, or parameters, required to fulfill the intent. Each slot has a type that defines the values that can be entered in the slot. You can choose from your custom slot types, or you can choose a built-in slot type.
 - **Confirmation prompts and declination responses** – These responses are used to end the conversation with a user and to confirm or decline fulfillment of the intent. The confirmation prompt asks the user to review slot values. For example, "I've booked a hotel room for Friday. Is this correct?" The declination response is sent to the user when they decline the confirmation.
 - **Closing responses** – Text sent to the user after the intent is fulfilled. For example, a thank you for booking a hotel room. Or it can prompt the user to start a different intent, such as "Thank you for booking a room, would you like to book a rental car?"
 - **Codehooks** – Indicate whether you are using an AWS Lambda function to initialize the intent, validate user input, and to fulfill the intent. You specify the Lambda function in the alias that you use to run the bot.
6. Choose **Save intent** to save the intent.

Adding slot types

Slot types define the values that users can supply for your intent variables. You define slot types for each language so that the values are specific to that language. For example, for a slot type that lists paint colors, you could include the value "red" in English, "rouge" in French, and "rojo" in Spanish.

This topic describes how to create custom slot types that provide values for your intent's slots. You can also use built-in slot types for standard values. For example, you can use the built-in slot type AMAZON.Country for a list of countries in the world.

To create a slot type

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. From the list of bots, choose the bot that you want to add the language to, then choose **Conversation structure** and then **All languages**.
3. Choose the language to add the slot type to, then choose **Slot types**.
4. Choose **Add slot type**, give your slot type a name, and then choose **Add**.
5. In the slot type editor, add the details of your slot type.
 - **Slot value resolution** – Determines how slot values are resolved. If you choose **Expand values**, Amazon Lex uses the values as representative values for training. If you use **Restrict to slot values** the allowed values for the slot are restricted to the ones that you provide.
 - **Slot type values** – The values for the slot. If you chose **Restrict to slot values**, you can add synonyms for the value. For example, for the value "football" you can add the synonym "soccer." If the user enters "soccer" in a conversation with your bot, the actual value of the slot is "football."
6. Choose **Save slot type**.

Testing a bot using the console

The Amazon Lex console contains a test window that you can use to test the interaction with your bot. You use the test window to have a test conversation with your bot and to see the responses that your application receives from the bot.

There are two types of testing that you can perform with your bot. The first, express testing, enables you to test your bot with the exact phrases that you used for creating the bot. For example, if you added the utterance "I want to pick up flowers" to your intent, you can test the bot using that exact phrase.

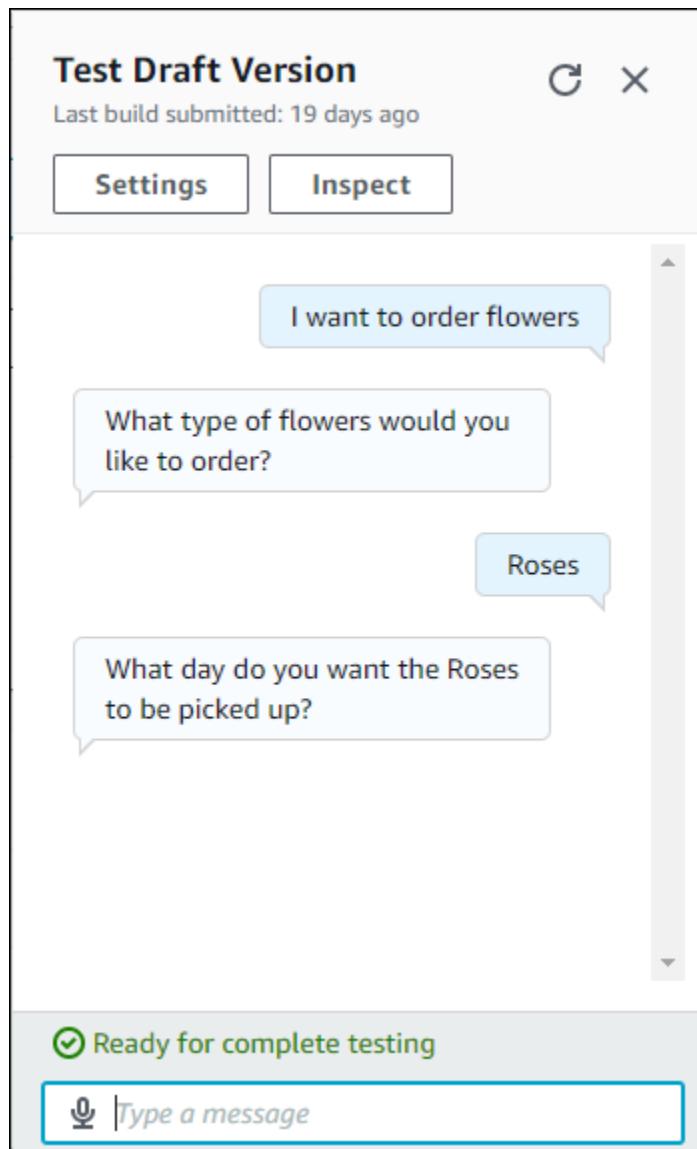
The second type, complete testing, enables you to test your bot using phrases related to the utterances that you configured. For example, you can use the phrase "Can I order flowers" to start a conversation with your bot.

You test a bot using a specific alias and language. If you are testing the development version of the bot, you use the `TestBotAlias` alias for testing.

To open the test window

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. Choose the bot to test from the list of bots.
3. From the left menu, choose **Aliases**.
4. From the list of aliases, choose the alias to test.
5. From **Languages**, choose the radio button of the language to test, and then choose **Test**.

After you choose **Test**, the test window opens in the console. You can use the test window to interact with your bot, as shown in the following graphic.



In addition to the conversation, you can also choose **Inspect** in the test window to see the responses returned from the bot. The first view shows you a summary of the information returned from your bot to the test window.

The screenshot shows the Amazon Lex V2 developer console. On the left, there's an "Inspect" window with tabs for "Summary" (which is selected) and "JSON input and output". The "Summary" tab displays information about the intent and slots. The intent is "OrderFlowers" and the slot "FlowerType" has the value "Roses". There are also entries for "PickupDate" and "PickupTime" with a dash indicating no value. Below this, under "Active contexts", the context "Weather" is listed with a value of "5 turns or 90s". To the right of the inspect window, the "Test Draft Version" section shows a message history. The user says "I want to c...", and the bot responds with "What type of flowers would you like to order?". Another message from the user asks "What day do you want the flowers to be picked up?", and the bot is shown as ready for the next message. A "Type a message" input field is at the bottom.

Slot	Elicitation
FlowerType	Roses
PickupDate	-
PickupTime	-

Active contexts	Number of turns or seconds
Weather	5 turns or 90s

You can also use the test inspection window to see the JSON structures that are sent between the bot and the test window. You can see both the request from the test window and the response from Amazon Lex.

The screenshot shows the 'Inspect' interface for an Amazon Lex bot. On the left, there's a sidebar with tabs for 'Summary' and 'JSON input and output'. The 'JSON input and output' tab is selected, showing 'Request' and 'Response' sections. The 'Request' section contains the following JSON:

```
{  
  "botAliasId": "TSTALIASID",  
  "botId": "Q2NA3VH5E3",  
  "localeId": "en_US",  
  "text": "I want to order flowers"  
  "sessionId": "130772450386735"  
}
```

The 'Response' section contains the following JSON:

```
{  
  "messages": [  
    {  
      "content": "What type of flower",  
      "contentType": "PlainText"  
    }  
  ]  
}
```

To the right of the JSON sections, there's a 'Test Draft Version' panel. It shows a message history:

- I want to order flowers
- What type of flowers would you like to order?
- What day do you want the flowers to be picked up?

At the bottom of the panel, there's a green checkmark icon followed by the text 'Ready for complete testing'.

Creating versions

Amazon Lex supports publishing versions of bots so that you can control the implementation that your client applications use. A *version* is a numbered snapshot of your work that you can publish for use in different parts of your workflow, such as development, beta deployment, and production.

The Draft version

When you create an Amazon Lex bot there is only one version, the *Draft* version.

Draft is the working copy of your bot. You can update only the *Draft* version and until you publish your first version, *Draft* is the only version of the bot that you have.

The `Draft` version of your bot is associated with the `TestBotAlias`. The `TestBotAlias` should only be used for manual testing. Amazon Lex limits the number of runtime requests that you can make to the `TestBotAlias` alias of the bot.

Creating a version

When you version an Amazon Lex bot you create a numbered snapshot of the bot so that you can use the bot as it existed when the version was made. Once you've created a numeric version it will stay the same while you continue to work on the draft version of your application.

When you create a version, you can choose the locales to include in the version. You don't need to choose all of the locales in a bot. Also, when you create a version you can choose a locale from a previous version. For example, if you have three versions of a bot, you can choose one locale from the `Draft` version and one from version two when you create version four.

If you delete a locale from the `Draft` version, it is not deleted from a numbered version.

If a bot version is not used for six months, Amazon Lex will mark the version inactive. When a version is inactive, you can't use runtime operations with the bot. To make the bot active, rebuild all the languages associated with the version.

Updating an Amazon Lex bot

You can update only the `Draft` version of an Amazon Lex bot. Versions can't be changed. You can publish a new version any time after you update a resource in the console or with the [CreateBotVersion \(p. 184\)](#) operation.

Deleting an Amazon Lex bot or version

Amazon Lex supports deleting a bot or version using the console or one of the API operations:

- [DeleteBot \(p. 231\)](#)
- [DeleteBotVersion \(p. 240\)](#)

Built-in intents and slot types

To make it easier to create bots, Amazon Lex allows you to use standard built-in intents and slot types.

Topics

- [Built-in intents \(p. 23\)](#)
- [Built-in slot types \(p. 34\)](#)

Built-in intents

For common actions, you can use the standard built-in intents library. To create an intent from a built-in intent, choose a built-intent in the console, and give it a new name. The new intent has the configuration of the base intent, such as the sample utterances.

In the current implementation, you can't do the following:

- Add or remove sample utterances from the base intent
- Configure slots for built-in intents

To add a built-in intent to a bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. Choose the bot to add the built-in intent to.
3. In the left menu, choose the language and then choose **Intents**.
4. Choose **Add intent**, and then choose **Use built-in intent**.
5. In **Built-in intent**, choose the intent to use.
6. Give the intent a name, and then choose **Add**.
7. Use the intent editor to configure the intent as required for your bot.

Topics

- [AMAZON.CancelIntent \(p. 24\)](#)
- [AMAZON.FallbackIntent \(p. 24\)](#)
- [AMAZON.HelpIntent \(p. 25\)](#)
- [AMAZON.KendraSearchIntent \(p. 26\)](#)
- [AMAZON.PauseIntent \(p. 32\)](#)
- [AMAZON.RepeatIntent \(p. 33\)](#)
- [AMAZON.ResumeIntent \(p. 33\)](#)
- [AMAZON.StartOverIntent \(p. 33\)](#)
- [AMAZON.StopIntent \(p. 33\)](#)

AMAZON.CancelIntent

Responds to words and phrases that indicate the user wants to cancel the current interaction. Your application can use this intent to remove slot type values and other attributes before ending the interaction with the user.

Common utterances:

- cancel
- never mind
- forget it

AMAZON.FallbackIntent

When a user's input to an intent isn't what a bot expects, you can configure Amazon Lex to invoke a *fallback intent*. For example, if the user input "I'd like to order candy" doesn't match an intent in your OrderFlowers bot, Amazon Lex invokes the fallback intent to handle the response.

The built-in `AMAZON.FallbackIntent` intent type is added to your bot automatically when you create a bot using the console, when you use the API you can specify the intent using the [CreateBot \(p. 168\)](#) operation.

Invoking a fallback intent uses two steps. In the first step the fallback intent is matched based on the input from the user. When the fallback intent is matched, the way the bot behaves depends on the number of retries configured for a prompt.

Amazon Lex matches the fallback intent in these situations:

- The user's input to an intent doesn't match the input that the bot expects
- Audio input is noise, or text input isn't recognized as words.
- The user's input is ambiguous and Amazon Lex can't determine which intent to invoke.

The fallback intent is invoked when:

- The bot doesn't recognize the user input as an intent after the configured number of tries for clarification when the conversation is started.
- An intent doesn't recognize the user input as a slot value after the configured number of tries.
- An intent doesn't recognize the user input as a response to a confirmation prompt after the configured number of tries.

You can't add the following to a fallback intent:

- Utterances
- Slots
- A confirmation prompt

Using a Lambda Function with a Fallback Intent

When a fallback intent is invoked, the response depends on the setting of the `fulfillmentCodeHook` parameter to the [CreateIntent \(p. 192\)](#) operation. The bot does one of the following:

- Returns the intent information to the client application.
- Calls the aliases's validation and fulfillment Lambda function. It calls the function with the session variables that are set for the session.

For more information about setting the response when a fallback intent is invoked, see the `fulfillmentCodeHook` parameter of the [CreateIntent \(p. 192\)](#) operation.

If you use the Lambda function with your fallback intent, you can use this function to call another intent or to perform some form of communication with the user, such as collecting a callback number or opening a session with a customer service representative.

A fallback intent can be invoked multiple times in the same session. For example, suppose that your Lambda function uses the `ElicitIntent` dialog action to prompt the user for a different intent. If Amazon Lex can't infer the user's intent after the configured number of tries, it invokes the fallback intent again. It also invokes the fallback intent when the user doesn't respond with a valid slot value after the configured number of tries.

You can configure your Lambda function to keep track of the number of times that the fallback intent is called using a session variable. Your Lambda function can take a different action if it is called more times than the threshold that you set in your Lambda function. For more information about session variables, see [Setting session attributes \(p. 65\)](#).

AMAZON.HelpIntent

Responds to words or phrases that indicate the user needs help while interacting with your bot. When this intent is invoked, you can configure your Lambda function or application to provide information about the your bot's capabilities, ask follow up questions about areas of help, or hand the interaction over to a human agent.

Common utterances:

- help
- help me
- can you help me

AMAZON.KendraSearchIntent

To search documents that you have indexed with Amazon Kendra, use the `AMAZON.KendraSearchIntent` intent. When Amazon Lex can't determine the next action in a conversation with the user, it triggers the search intent.

The `AMAZON.KendraSearchIntent` is available only in the English (US) (en-US) locale.

Amazon Kendra is a machine-learning-based search service that indexes natural language documents such as PDF documents or Microsoft Word files. It can search indexed documents and return the following types of responses to a question:

- An answer
- An entry from a FAQ that might answer the question
- A document that is related to the question

For an example of using the `AMAZON.KendraSearchIntent`, see [Example: Creating a FAQ Bot for an Amazon Kendra Index \(p. 31\)](#).

If you configure an `AMAZON.KendraSearchIntent` intent for your bot, Amazon Lex calls the intent whenever it can't determine the user utterance for a slot or intent. For example, if your bot is eliciting a response for a slot type called "pizza topping" and the user says "What is a pizza?," Amazon Lex calls the `AMAZON.KendraSearchIntent` to handle the question. If there is no response from Amazon Kendra, the conversation continues as configured in the bot.

When you use the `AMAZON.KendraSearchIntent` with the `AMAZON.FallbackIntent` in the same bot, Amazon Lex uses the intents as follows:

1. Amazon Lex calls the `AMAZON.KendraSearchIntent`. The intent calls the Amazon Kendra Query operation.
2. If Amazon Kendra returns a response, Amazon Lex displays the result to the user.
3. If there is no response from Amazon Kendra, Amazon Lex re-prompts the user. The next action depends on response from the user.
 - If the response from the user contains an utterance that Amazon Lex recognizes, such as filling a slot value or confirming an intent, the conversation with the user proceeds as configured for the bot.
 - If the response from the user does not contain an utterance that Amazon Lex recognizes, Amazon Lex makes another call to the `Query` operation.
4. If there is no response after the configured number of retries, Amazon Lex calls the `AMAZON.FallbackIntent` and ends the conversation with the user.

There are three ways to use the `AMAZON.KendraSearchIntent` to make a request to Amazon Kendra:

- Let the search intent make the request for you. Amazon Lex calls Amazon Kendra with the user's utterance as the search string. When you create the intent, you can define a query filter string that limits the number of responses that Amazon Kendra returns. Amazon Lex uses the filter in the query request.
- Add additional query parameters to the request to narrow the search results using your Lambda function. You add a `kendraQueryFilterString` field that contains Amazon Kendra query

parameters to the delegate dialog action. When you add query parameters to the request with the Lambda function, they take precedence over the query filter that you defined when you created the intent.

- Create a new query using the Lambda function. You can create a complete Amazon Kendra query request that Amazon Lex sends. You specify the query in the `kendraQueryRequestPayload` field in the delegate dialog action. The `kendraQueryRequestPayload` field takes precedence over the `kendraQueryFilterString` field.

To specify the `queryFilterString` parameter when you create a bot, or to specify the `kendraQueryFilterString` field when you call the delegate action in a dialog Lambda function, you specify a string that is used as the attribute filter for the Amazon Kendra query. If the string isn't a valid attribute filter, you'll get an `InvalidBotConfigException` exception at runtime. For more information about attribute filters, see [Using document attributes to filter queries in the Amazon Kendra Developer Guide](#).

To have control over the query that Amazon Lex sends to Amazon Kendra, you can specify a query in the `kendraQueryRequestPayload` field in your Lambda function. If the query isn't valid, Amazon Lex returns an `InvalidLambdaResponseException` exception. For more information, see the [Query operation](#) in the [Amazon Kendra Developer Guide](#).

For an example of how to use the `AMAZON.KendraSearchIntent`, see [Example: Creating a FAQ Bot for an Amazon Kendra Index \(p. 31\)](#).

IAM Policy for Amazon Kendra Search

To use the `AMAZON.KendraSearchIntent` intent, you must use a role that provides AWS Identity and Access Management (IAM) policies that enable Amazon Lex to assume a runtime role that has permission to call the Amazon Kendra Query intent. The IAM settings that you use depend on whether you create the `AMAZON.KendraSearchIntent` using the Amazon Lex console, or using an AWS SDK or the AWS Command Line Interface (AWS CLI). When you use the console, you can choose between adding permission to call Amazon Kendra to the Amazon Lex service-linked role or using a role specifically for calling the Amazon Kendra Query operation. When you use the AWS CLI or an SDK to create the intent, you must use a role specifically for calling the `Query` operation.

Attaching Permissions

You can use the console to attach permissions to access the Amazon Kendra `Query` operation to the default Amazon Lex service-linked role. When you attach permissions to the service-linked role, you don't have to create and manage a runtime role specifically to connect to the Amazon Kendra index.

The user, role, or group that you use to access the Amazon Lex console must have permissions to manage role policies. Attach the following IAM policy to the console access role. When you grant these permissions, the role has permissions to change the existing service-linked role policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam:AttachRolePolicy",  
                "iam:PutRolePolicy",  
                "iam:GetRolePolicy"  
            ],  
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/  
AWSServiceRoleForLexBots*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "lambda:InvokeFunction"  
            ],  
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:  
LexBotServiceRoleForKendraIndex"  
        }  
    ]  
}
```

```

        "Action": "iam>ListRoles",
        "Resource": "*"
    }
}

```

Specifying a Role

You can use the console, the AWS CLI, or the API to specify a runtime role to use when calling the Amazon Kendra `Query` operation.

The IAM user, role, or group that you use to specify the runtime role must have the `iam:PassRole` permission. The following policy defines the permission. You can use the `iam:AssociatedResourceArn` and `iam:PassedToService` condition context keys to further limit the scope of the permissions. For more information, see [IAM and AWS STS Condition Context Keys](#) in the [AWS Identity and Access Management User Guide](#).

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account:role/role"
        }
    ]
}

```

The runtime role that Amazon Lex needs to use to call Amazon Kendra must have the `kendra:Query` permissions. When you use an existing IAM role for permission to call the Amazon Kendra `Query` operation, the role must have the following policy attached.

You can use the IAM console, the IAM API, or the AWS CLI to create a policy and attach it to a role. These instructions use the AWS CLI to create the role and policies.

Note

The following code is formatted for Linux and MacOS. For Windows, replace the Linux line continuation character (\) with a caret (^).

To add Query operation permission to a role

1. Create a document called `KendraQueryPolicy.json` in the current directory, add the following code to it, and save it

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kendra:Query"
            ],
            "Resource": [
                "arn:aws:kendra:region:account:index/index_ID"
            ]
        }
    ]
}

```

2. In the AWS CLI, run the following command to create the IAM policy for running the Amazon Kendra `Query` operation.

```
aws iam create-policy \
--policy-name query-policy-name \
--policy-document file://KendraQueryPolicy.json
```

3. Attach the policy to the IAM role that you are using to call the Query operation.

```
aws iam attach-role-policy \
--policy-arn arn:aws:iam::account-id:policy/query-policy-name
--role-name role-name
```

You can choose to update the Amazon Lex service-linked role or to use a role that you created when you create the AMAZON.KendraSearchIntent for your bot. The following procedure shows how to choose the IAM role to use.

To specify the runtime role for AMAZON.KendraSearchIntent

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/home>
2. Choose the bot that you want to add the AMAZON.KendraSearchIntent to.
3. Choose the plus (+) next to **Intents**.
4. In **Add intent**, choose **Search existing intents**.
5. In **Search intents**, enter **AMAZON.KendraSearchIntent** and then choose **Add**.
6. In **Copy built-in intent**, enter a name for the intent, such as **KendraSearchIntent**, and then choose **Add**.
7. Open the **Amazon Kendra query** section.
8. For **IAM role** choose one of the following options:
 - To update the Amazon Lex service-linked role to enable your bot to query Amazon Kendra indexes, choose **Add Amazon Kendra permissions**.
 - To use a role that has permission to call the Amazon Kendra Query operation, choose **Use an existing role**.

Using Request and Session Attributes as Filters

To filter the response from Amazon Kendra to items related to current conversation, use session and request attributes as filters by adding the `queryFilterString` parameter when you create your bot. You specify a placeholder for the attribute when you create the intent, and then Amazon Lex substitutes a value before it calls Amazon Kendra. For more information about request attributes, see [Setting request attributes \(p. 66\)](#). For more information about session attributes, see [Setting session attributes \(p. 65\)](#).

The following is an example of a `queryFilterString` parameter that uses a session attribute called "DocumentType" to filter the Amazon Kendra query.

```
"{\"equalsTo\": {\"key\": \"Type\", \"value\": {\"stringValue\": \"[DocumentType]\"}}}"
```

The following is an example of a `queryFilterString` parameter that uses a request attribute called "DepartmentName" to filter the Amazon Kendra query.

```
"{\"equalsTo\": {\"key\": \"Department\", \"value\": {\"stringValue\": \"((DepartmentName))\"}}}"
```

Using the Search Response

Amazon Kendra returns the response to a search in a response from the intent's `IntentClosingSetting` statement. The intent must have a `closingResponse` statement unless a Lambda function produces a closing response message.

Amazon Kendra has four types of responses.

- `x-amz-lex:kendra-search-response-question_answer-question-<N>` – The question from a FAQ that matches the search.
- `x-amz-lex:kendra-search-response-question_answer-answer-<N>` – The answer from a FAQ that matches the search.
- `x-amz-lex:kendra-search-response-document-<N>` – An excerpt from a document in the index that is related to the text of the utterance.
- `x-amz-lex:kendra-search-response-document-link-<N>` – The URL of a document in the index that is related to the text of the utterance.
- `x-amz-lex:kendra-search-response-answer-<N>` – An excerpt from a document in the index that answers the question.

The responses are returned in `request` attributes. There can be up to five responses for each attribute, numbered 1 through 5. For more information about responses, see [Types of response](#) in the *Amazon Kendra Developer Guide*.

The `closingResponse` statement must have one or more message groups. Each message group contains one or more messages. Each message can contain one or more placeholder variables that are replaced by request attributes in the response from Amazon Kendra. There must be at least one message in the message group where all of the variables in the message are replaced by request attribute values in the runtime response, or there must be a message in the group with no placeholder variables. The request attributes are set off with double parentheses ("((" "))). The following message group messages match any response from Amazon Kendra:

- "I found a FAQ question for you: ((x-amz-lex:kendra-search-response-question_answer-question-1)), and the answer is ((x-amz-lex:kendra-search-response-question_answer-answer-1))"
- "I found an excerpt from a helpful document: ((x-amz-lex:kendra-search-response-document-1))"
- "I think the answer to your questions is ((x-amz-lex:kendra-search-response-answer-1))"

Using a Lambda Function to Manage the Request and Response

The `AMAZON.KendraSearchIntent` intent can use your dialog code hook and fulfillment code hook to manage the request to Amazon Kendra and the response. Use the dialog code hook Lambda function when you want to modify the query that you send to Amazon Kendra, and the fulfillment code hook Lambda function when you want to modify the response.

Creating a Query with the Dialog Code Hook

You can use the dialog code hook to create a query to send to Amazon Kendra. Using the dialog code hook is optional. If you don't specify a dialog code hook, Amazon Lex constructs a query from the user utterance and uses the `queryFilterString` that you provided when you configured the intent, if you provided one.

You can use two fields in the dialog code hook response to modify the request to Amazon Kendra:

- `kendraQueryFilterString` – Use this string to specify attribute filters for the Amazon Kendra request. You can filter the query using any of the index fields defined in your index. For the structure of the filter string, see [Using document attributes to filter queries](#) in the *Amazon Kendra Developer*

Guide. If the specified filter string isn't valid, you will get an `InvalidLambdaResponseException` exception. The `kendraQueryFilterString` string overrides any query string specified in the `queryFilterString` configured for the intent.

- `kendraQueryRequestPayload` – Use this string to specify an Amazon Kendra query. Your query can use any of the features of Amazon Kendra. If you don't specify a valid query, you get a `InvalidLambdaResponseException` exception. For more information, see [Query](#) in the *Amazon Kendra Developer Guide*.

After you have created the filter or query string, you send the response to Amazon Lex with the `dialogAction` field of the response set to `delegate`. Amazon Lex sends the query to Amazon Kendra and then returns the query response to the fulfillment code hook.

Using the Fulfillment Code Hook for the Response

After Amazon Lex sends a query to Amazon Kendra, the query response is returned to the `AMAZON.KendraSearchIntent` fulfillment Lambda function. The input event to the code hook contains the complete response from Amazon Kendra. The query data is in the same structure as the one returned by the Amazon Kendra `Query` operation. For more information, see [Query response syntax](#) in the *Amazon Kendra Developer Guide*.

The fulfillment code hook is optional. If one does not exist, or if the code hook doesn't return a message in the response, Amazon Lex uses the `closingResponse` statement for responses.

Example: Creating a FAQ Bot for an Amazon Kendra Index

This example creates an Amazon Lex bot that uses an Amazon Kendra index to provide answers to users' questions. The FAQ bot manages the dialog for the user. It uses the `AMAZON.KendraSearchIntent` intent to query the index and to present the response to the user. To create the bot, you:

1. Create a bot that your customers will interact with to get answers from your bot.
2. Create a custom intent. Your bot requires at least one intent with at least one utterance. This intent enables your bot to build, but is not used otherwise.
3. Add the `KendraSearchIntent` intent to your bot and configure it to work with your Amazon Kendra index.
4. Test the bot by asking questions that are answered by documents stored in your Amazon Kendra index.

Before you can use this example, you need to create an Amazon Kendra index. For more information, see [Getting started with an S3 bucket \(console\)](#) in the *Amazon Kendra Developer Guide*.

To create a FAQ bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. In the navigation pane, choose **Bots**.
3. Choose **Create**.
4. Choose **Custom bot**. Configure the bot as follows:
 - **Bot name** – Give the bot a name that indicates its purpose, such as `KendraTestBot`.
 - **Output voice** – Choose **None**.
 - **Session timeout** – Enter **5**.
 - **Sentiment analysis** – Choose **No**.
 - **COPPA** – Choose **No**.
 - **User utterance storage** – Choose **Do not store**.

5. Choose **Create**.

To successfully build a bot, you must create at least one intent with at least one sample utterance. This intent is required to build your Amazon Lex bot, but isn't used for the FAQ response. The utterance for the intent must not apply to any of the questions that your customer asks.

To create the required intent

1. On the **Getting started with your bot** page, choose **Create intent**.
2. For **Add intent**, choose **Create intent**.
3. In the **Create intent** dialog box, give the intent a name, such as **RequiredIntent**.
4. For **Sample utterances**, type an utterance, such as **Required utterance**.
5. Choose **Save intent**.

Now, create the intent to search an Amazon Kendra index and the response messages that it should return.

To create an AMAZON.KendraSearchIntent intent and response message

1. In the navigation pane, choose the plus (+) next to **Intents**.
2. For **Add intent**, choose **Search existing intents**.
3. In the **Search intents** box, enter **AMAZON.KendraSearchIntent**, then choose it from the list.
4. For **Copy built-in intent**, give the intent a name, such as **KendraSearchIntent**, and then choose **Add**.
5. In the intent editor, choose **Amazon Kendra query** to open the query options.
6. From the **Amazon Kendra index** menu, choose the index that you want the intent to search.
7. In the **Response** section, add the following three messages:

```
I found a FAQ question for you: ((x-amz-lex:kendra-search-response-question_answer-question-1)) and the answer is ((x-amz-lex:kendra-search-response-question_answer-answer-1)).  
I found an excerpt from a helpful document: ((x-amz-lex:kendra-search-response-document-1)).  
I think the answer to your questions is ((x-amz-lex:kendra-search-response-answer-1)).
```

8. Choose **Save intent**, and then choose **Build** to build the bot.

Finally, use the console test window to test responses from your bot. Your questions should be in the domain that your index supports.

To test your FAQ bot

1. In the console test window, type a question for your index.
2. Verify the answer in the test window's response section.
3. To reset the test window for another question, choose **Clear chat history**.

AMAZON.PauseIntent

Responds to words and phrases that enable the user to pause an interaction with a bot so that they can return to it later. Your Lambda function or application needs to save intent data in session variables, or

you need to use the [GetSession \(p. 404\)](#) operation to retrieve intent data when you resume the current intent.

Common utterances:

- pause
- pause that

AMAZON.RepeatIntent

Responds to words and phrases that enable the user to repeat the previous message. Your application needs to use a Lambda function to save the previous intent information in session variables, or you need to use the [GetSession \(p. 404\)](#) operation to get the previous intent information.

Common utterances:

- repeat
- say that again
- repeat that

AMAZON.ResumeIntent

Responds to words and phrases that enable the user to resume a previously paused intent. Your Lambda function or application must manage the information required to resume the previous intent.

Common utterances:

- resume
- continue
- keep going

AMAZON.StartOverIntent

Responds to words and phrases that enable the user to stop processing the current intent and start over from the beginning. You can use your Lambda function or the [PutSession](#) operation to elicit the first slot value again.

Common utterances:

- start over
- restart
- start again

AMAZON.StopIntent

Responds to words and phrases that indicate that the user wants to stop processing the current intent and end the interaction with a bot. Your Lambda function or application should clear any existing attributes and slot type values and then end the interaction.

Common utterances:

- stop
- off

- shut up

Built-in slot types

Amazon Lex supports built-in slot types that define how data in the slot is recognized and handled. You can create slots of these types in your intents. This eliminates the need to create enumeration values for commonly used slot data such as date, time, and location. Built-in slot types do not have versions.

Slot Type	Short Description	Supported Locales	
AMAZON.AlphaNumeric (p. 35)	Recognizes words made up of letters and numbers.	All locales	
AMAZON.City (p. 36)	Recognizes words that represent a city.	All locales	
AMAZON.Country (p. 36)	Recognizes words that represent a country.	All locales	
AMAZON.Date (p. 36)	Recognizes words that represent a date and converts them to a standard format.	All locales	
AMAZON.Duration (p. 37)	Recognizes words that represent duration and converts them to a standard format.	All locales	
AMAZON.EmailAddress (p. 37)	Recognizes words that represent an email address and converts them into a standard email address.	All locales	
AMAZON.FirstName (p. 37)	Recognizes words that represent a first name.	All locales	
AMAZON.LastName (p. 37)	Recognizes words that represent a last name.	All locales	
AMAZON.Number (p. 38)	Recognizes numeric words and converts them into digits.	All locales	
AMAZON.Percentage (p. 38)	Recognizes words that represent a percentage and converts them to a number and a percent sign (%).	All locales	
AMAZON.PhoneNumber (p. 39)	Recognizes words that represent a phone number and converts them into a numeric string.	All locales	

Slot Type	Short Description	Supported Locales	
AMAZON.State (p. 39)	Recognizes words that represent a state.	All locales	
AMAZON.StreetName (p. 39)	Recognizes words that represent a street name.	All locales	
AMAZON.Time (p. 39)	Recognizes words that indicate times and converts them into a time format.	All locales	
AMAZON.UKPostalCode (p. 40)	Recognizes words that represent a UK post code and converts them to a standard form.	en-GB only	

AMAZON.AlphaNumeric

Recognizes strings made up of letters and numbers, such as **APQ123**.

You can use the `AMAZON.AlphaNumeric` slot type for strings that contain:

- Alphabetical characters, such as **ABC**
- Numeric characters, such as **123**
- A combination of alphanumeric characters, such as **ABC123**

You can add a regular expression to the `AMAZON.AlphaNumeric` slot type to validate values entered for the slot. For example, you can use a regular expression to validate:

- Canadian postal codes
- Driver's license numbers
- Vehicle identification numbers

Use a standard regular expression. Amazon Lex supports the following characters in the regular expression:

- A-Z, a-z
- 0-9

Amazon Lex also supports Unicode characters in regular expressions. The form is \uUnicode. Use four digits to represent Unicode characters. For example, [\u0041-\u005A] is equivalent to [A-Z].

The following regular expression operators are not supported:

- Infinite repeaters: *, +, or {x,} with no upper bound.
- Wild card (.)

The maximum length of the regular expression is 100 characters. The maximum length of a string stored in an `AMAZON.AlphaNumeric` slot type that uses a regular expression is 30 characters.

The following are some example regular expressions.

- Alphanumeric strings, such as **APQ123** or **APQ1: [A-Z]{3}[0-9]{1,3}** or a more constrained **[A-DP-T]{3} [1-5]{1,3}**
- US Postal Service Priority Mail International format, such as **CP123456789US: CP[0-9]{9}US**
- Bank routing numbers, such as **123456789: [0-9]{9}**

To set the regular expression for a slot type, use the console or the [CreateSlotType \(p. 224\)](#) operation. The regular expression is validated when you save the slot type. If the expression isn't valid, Amazon Lex returns an error message.

When you use a regular expression in a slot type, Amazon Lex checks input to slots of that type against the regular expression. If the input matches the expression, the value is accepted for the slot. If the input does not match, Amazon Lex prompts the user to repeat the input.

AMAZON.City

Provides a list of local and world cities. The slot type recognizes common variations of city names. Amazon Lex doesn't convert from a variation to an official name.

Examples:

- New York
- Reykjavik
- Tokyo
- Versailles

AMAZON.Country

The names of countries around the world. Examples:

- Australia
- Germany
- Japan
- United States
- Uruguay

AMAZON.Date

Converts words that represent dates into a date format.

The date is provided to your intent in ISO-8601 date format. The date that your intent receives in the slot can vary depending on the specific phrase uttered by the user.

- Utterances that map to a specific date, such as "today," "now," or "November twenty-fifth," convert to a complete date: 2020-11-25. This defaults to dates *on or after* the current date.
- Utterances that map to a future week, such as "next week," convert to the date of the last day of the current week. In ISO-8601 format, the week starts on Monday and ends on Sunday. For example, if today is 2020-11-25, "next week" converts to 2020-11-29. Dates that map to the current or previous week convert to the first day of the week. For example, if today is 2020-11-25, "last week" converts to 2020-11-16.
- Utterances that map to a future month, but not a specific day, such as "next month," convert to the last day of the month. For example, if today is 2020-11-25, "next month" converts to 2020-12-31. For

dates that map to the current or previous month convert to the first day of the month. For example, if today is 2020-11-25, "this month" maps to 2020-11-01.

- Utterances that map to a future year, but not a specific month or day, such as "next year," convert to the last day of the following year. For example, if today is 2020-11-25, "next year" converts to 2021-12-31. For dates that map to the current or previous year convert to the first day of the year. For example, if today is 2020-11-25, "last year" converts to 2019-01-01.

AMAZON.Duration

Converts words that indicate durations into a numeric duration.

The duration is resolved to a format based on the [ISO-8601 duration format](#), PnYnMnWnDTnHnMsS. The P indicates that this is a duration, the n is a numeric value, and the capital letter following the n is the specific date or time element. For example, P3D means 3 days. A T is used to indicate that the remaining values represent time elements rather than date elements.

Examples:

- "ten minutes": PT10M
- "five hours": PT5H
- "three days": P3D
- "forty five seconds": PT45S
- "eight weeks": P8W
- "seven years": P7Y
- "five hours ten minutes": PT5H10M
- "two years three hours ten minutes": P2YT3H10M

AMAZON.EmailAddress

Recognizes words that represent an email address provided as username@domain. Addresses can include the following special characters in a user name: underscore (_), hyphen (-), period (.), and the plus sign (+).

AMAZON.FirstName

Commonly used first names. This slot type recognizes both formal names and informal nicknames. The name sent to your intent is the value sent by the user. Amazon Lex doesn't convert from the nick name to the formal name.

For first names that sound alike but are spelled differently, Amazon Lex sends your intent a single common form.

Examples:

- Emily
- John
- Sophie

AMAZON.LastName

Commonly used last names. For names that sound alike that are spelled differently, Amazon Lex sends your intent a single common form.

Examples:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON.Number

Converts words or numbers that express a number into digits, including decimal numbers. The following table shows how the `AMAZON.Number` slot type captures numeric words.

Input	Response
one hundred twenty three point four five	123.45
one hundred twenty three dot four five	123.45
point four two	0.42
point forty two	0.42
232.998	232.998
50	50
-15	-15
minus 15	-15

AMAZON.Percentage

Converts words and symbols that represent a percentage into a numeric value with a percent sign (%).

If the user enters a number without a percent sign or the word "percent," the slot value is set to the number. The following table shows how the `AMAZON.Percentage` slot type captures percentages.

Input	Response
50 percent	50%
0.4 percent	0.4%
23.5%	23.5%
twenty five percent	25%

AMAZON.PhoneNumber

Converts the numbers or words that represent a phone number into a string format without punctuation as follows.

Type	Description	Input	Result
International number with leading plus (+) sign	11-digit number with leading plus sign.	+61 7 4445 1061 +1 (509) 555-1212	+61744431061 +15095551212
International number without leading plus (+) sign	11-digit number without leading plus sign	1 (509) 555-1212 61 7 4445 1061	15095551212 61744451061
National number	10-digit number without international code	(03) 5115 4444 (509) 555-1212	0351154444 5095551212
Local number	7-digit phone number without an international code or an area code	555-1212	5551212

AMAZON.State

The names of geographical and political regions within countries.

Examples:

- Bavaria
- Fukushima Prefecture
- Pacific Northwest
- Queensland
- Wales

AMAZON.StreetName

The names of streets within a typical street address. This includes just the street name, not the house number.

Examples:

- Canberra Avenue
- Front Street
- Market Road

AMAZON.Time

Converts words that represent times into time values. Includes resolutions for ambiguous times. When a user enters an ambiguous time, Amazon Lex uses the `slots` attribute of a Lambda event to pass resolutions for the ambiguous times to your Lambda function. For example, if your bot prompts the user for a delivery time, the user can respond by saying "10 o'clock." This time is ambiguous. It means either 10:00 AM or 10:00 PM. In this case, the value in the `interpretedValue` field is `null`, and the `resolvedValues` field contains the two possible resolutions of the time. Amazon Lex inputs the following into the Lambda function:

```
"slots": {
```

```
"deliveryTime": {  
    "value": {  
        "originalValue": "10 o'clock",  
        "interpretedValue": null,  
        "resolvedValues": [  
            "10:00", "22:00"  
        ]  
    }  
}
```

When the user responds with an unambiguous time, Amazon Lex sends the time to your Lambda function in the `interpretedValue` field of the `slots` attribute of the Lambda event. For example, if your user responds to the prompt for a delivery time with "10:00 AM," Amazon Lex inputs the following into the Lambda function:

```
"slots": {  
    "deliveryTime": {  
        "value": {  
            "originalValue": "10 AM",  
            "interpretedValue": 10:00,  
            "resolvedValues": [  
                "10:00"  
            ]  
        }  
    }  
}
```

For more information about the data sent from Amazon Lex to a Lambda function, see [Input event format \(p. 44\)](#).

AMAZON.UKPostalCode

Converts words that represent a UK postal code to a standard format. The `AMAZON.UKPostalCode` slot type validates and resolves the post code to a set of standardized formats, but it doesn't check to make sure that the post code is valid. Your application must validate the post code.

The `AMAZON.UKPostalCode` slot type is available only in the English (UK) (en-GB) locale.

The slot type recognizes all valid post code formats, including British overseas territories. The valid formats are (A represents a letter, 9 represents a digit):

- AA9A 9AA
- A9A 9AA
- A9 9AA
- A99 9AA
- AA9 9AA
- AA99 9AA

For text input, the user can enter any mix of upper and lower case letters. The user can use or omit the space in the post code. The resolved value will always include the space in the proper location for the post code.

For spoken input, the user can speak the individual characters, or they can use double letter pronunciations, such as "double A" or "double 9". They can also use double-digit pronunciations, such as "ninety-nine" for "99".

Creating custom slot types

For each intent, you can specify parameters that indicate the information that the intent needs to fulfill the user's request. These parameters, or slots, have a type. A *slot type* is a list of values that Amazon Lex uses to train the machine learning model to recognize values for a slot. For example, you can define a slot type called "Genres." Each value in the slot type is the name of a genre, "comedy," "adventure," "documentary," etc. You can define a synonym for a slot type value. For example, you can define the synonyms "funny" and "humorous" for the value "comedy."

You can configure the slot type to restrict resolution to the slot values. The slot values will be used as an enumeration and the value entered by the user will be resolved to the slot value only if it is the same as one of the slot values or a synonym. A synonym is resolved to the corresponding slot value. For example, if the user enters "funny" it will resolve to the slot value "comedy."

Alternately, you can configure the slot type to expand the values. Slot values will be used as training data and the slot is resolved to the value provided by the user if it is similar to the slot values and synonyms. This is the default behavior.

Amazon Lex maintains a list of possible resolutions for a slot. Each entry in the list provides a *resolved value* that Amazon Lex recognized as additional possibilities for the slot. A resolved value is the best effort to match the slot value. The list contains up to five values.

When the value entered by the user is a synonym, the first entry in the list of `resolvedValues` is the slot type value. For example, if the user enters "funny," the `originalValue` field contains "funny" and the first entry in the `resolvedValues` field is "comedy." You configure the `valueSelectionStrategy` when you create or update a slot type with the [CreateSlotType \(p. 224\)](#) operation so that the slot value is filled with the first value in the resolution list.

If you are using a Lambda function, the input event to the function includes a resolution list called `resolvedValues`. The following example shows the slot section of the input to a Lambda function:

```
"slots": {  
    "MovieGenre": {  
        "value": {  
            "originalValue": "funny",  
            "interpretedValue": "comedy",  
            "resolvedValues": [  
                "comedy"  
            ]  
        }  
    }  
}
```

For each slot type, you can define a maximum of 10,000 values and synonyms. Each bot can have a total number of 50,000 slot type values and synonyms. For example, you can have 5 slot types, each with 5,000 values and 5,000 synonyms, or you can have 10 slot types, each with 2,500 values and 2,500 synonyms.

Using multiple values in a slot

Note

Multiple value slots are only supported in the English (US) language.

For some intents, you might want to capture multiple values for a single slot. For example, a pizza ordering bot might have an intent with the following utterance:

I want a pizza with {toppings}

The intent expects that the {toppings} slot contains a list of the toppings that the customer wants on their pizza, for example "pepperoni and pineapple".

To configure a slot to capture multiple values, you set the `allowMultipleValues` field on the slot to true. You can set the field using the console or with the [CreateSlot \(p. 212\)](#) or [UpdateSlot \(p. 383\)](#) operation.

You can only mark slots with custom slot types as multi-value slots.

For a multi-value slot, Amazon Lex returns a list of slot values in the response to the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation. The following is the slot information returned for the utterance "I want a pizza with pepperoni and pineapple" from the OrderPizza bot.

```
  "slots": {
    "toppings": {
      "shape": "List",
      "value": {
        "interpretedValue": "pepperoni and pineapple",
        "originalValue": "pepperoni and pineapple",
        "resolvedValues": [
          "pepperoni and pineapple"
        ]
      },
      "values": [
        {
          "shape": "Scalar",
          "value": {
            "interpretedValue": "pepperoni",
            "originalValue": "pepperoni",
            "resolvedValues": [
              "pepperoni"
            ]
          }
        },
        {
          "shape": "Scalar",
          "value": {
            "interpretedValue": "pineapple",
            "originalValue": "pineapple",
            "resolvedValues": [
              "pineapple"
            ]
          }
        }
      ]
    }
  }
```

Multi-valued slots always return a list of values. When the utterance only contains one value, the list of values returned only contains one response.

Amazon Lex recognizes multiple values separated by spaces, commas (,), and the conjunction "and". Multi-value slots work with both text and voice input.

You can use multi-valued slots in prompts. For example, you can set the confirmation prompt for an intent to

Would you like me to order your {toppings} pizza?

When Amazon Lex sends the prompt to the user, it sends "Would you like me to order your pepperoni and pineapple pizza?"

Multi-valued slots support default values. For more information, see [Using default slot values \(p. 64\)](#).

You can use slot obfuscation to mask the values of a multi-value slot in conversation logs. When you obfuscate slot values, the value of each of the slot values is replaced with the name of the slot. For more information, see [Obscuring slot values in logs \(p. 101\)](#).

Using an AWS Lambda function

This section describes how to attach a Lambda function to a bot alias and the structure of the event data that Amazon Lex provides to a Lambda function. Use this information to parse the input to your Lambda code. It also explains the format of the response that Amazon Lex expects your Lambda function to return.

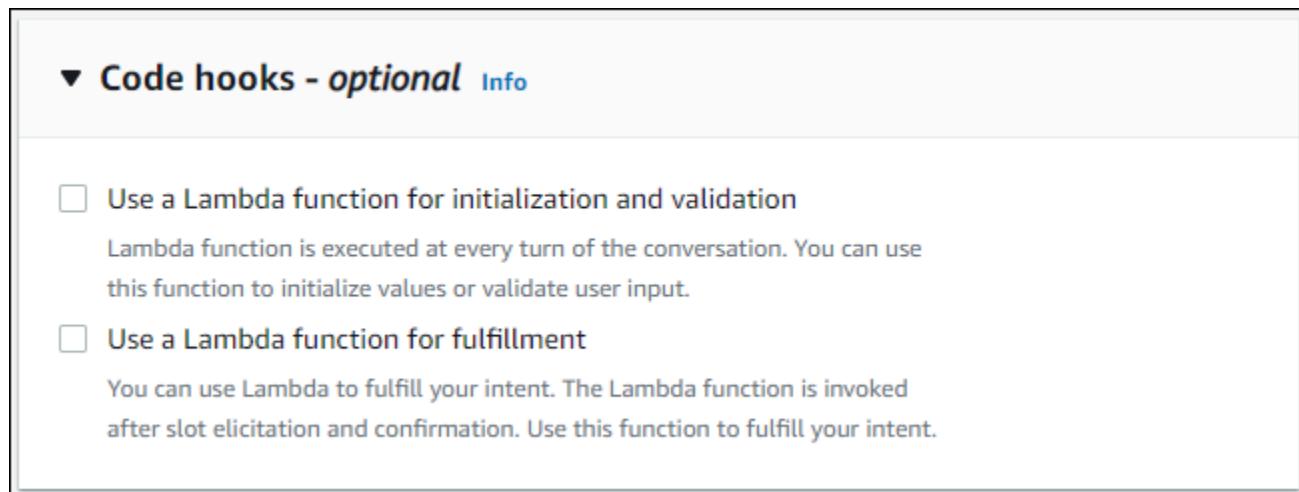
Topics

- [Attaching a Lambda function to a bot alias \(p. 43\)](#)
- [Input event format \(p. 44\)](#)
- [Response format \(p. 47\)](#)

Attaching a Lambda function to a bot alias

With Amazon Lex you indicate that an intent should use a Lambda function for each intent, but you assign the Lambda function that your intents use for each language supported by a bot alias. That way you can create a Lambda function tailored to each language that your bot alias supports.

You indicate that an intent should use a Lambda function using the console or the [CreateIntent \(p. 192\)](#) or [UpdateIntent \(p. 368\)](#) operation. In the intent editor, you turn on Lambda functions in the **Code hooks** section of the editor.



You define the Lambda function to use in each bot alias. The same Lambda function is used for all intents in a language supported by the bot.

To choose a Lambda function to use with a bot alias

1. Open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>.

2. From the list of bots, choose the name of the bot that you want to use.
3. From **Create versions and aliases for deployment**, choose **View aliases**.
4. From the list of aliases, choose the name of the alias that you want to use.
5. From the list of supported languages, choose the language that the Lambda function is used for.
6. Choose the name of the Lambda function to use, then choose the version or alias of the function.
7. Choose **Save** to save your changes.

Input event format

The following is the general format of an Amazon Lex event that is passed to a Lambda function. Use this information when you're writing your Lambda function.

Note

The input format may change without a corresponding change to the `messageVersion`. Your code shouldn't throw an error if new fields are present.

```
{  
    "messageVersion": "1.0",  
    "invocationSource": "DialogCodeHook | FulfillmentCodeHook",  
    "inputMode": "DTMF | Speech | Text",  
    "responseContentType": "CustomPayload | ImageResponseCard | PlainText | SSML",  
    "sessionId": "string",  
    "inputTranscript": "string",  
    "bot": {  
        "id": "string",  
        "name": "string",  
        "aliasId": "string",  
        "localeId": "string",  
        "version": "string"  
    },  
    "interpretations": [  
        {  
            "intent": {  
                "confirmationState": "Confirmed | Denied | None",  
                "name": "string",  
                "slots": {  
                    "string": {  
                        "value": {  
                            "interpretedValue": "string",  
                            "originalValue": "string",  
                            "resolvedValues": [  
                                "string"  
                            ]  
                        }  
                    }  
                }  
            },  
            "string": {  
                "shape": "List",  
                "value": {  
                    "interpretedValue": "string",  
                    "originalValue": "string",  
                    "resolvedValues": [  
                        "string"  
                    ]  
                }  
            },  
            "values": [  
                {  
                    "shape": "Scalar",  
                    "value": {  
                        "originalValue": "string",  
                    }  
                }  
            ]  
        }  
    ]  
}
```

```

                "interpretedValue": "string",
                "resolvedValues": [
                    "string"
                ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": "string",
                "interpretedValue": "string",
                "resolvedValues": [
                    "string"
                ]
            }
        }
    ]
},
"state": "Failed | Fulfilled | InProgress | ReadyForFulfillment",
"kendraResponse": {
    // Only present when intent is KendraSearchIntent. For details, see
    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
}
},
"nluConfidence": {
    "score": number
},
"sentimentResponse": {
    "sentiment": "string",
    "sentimentScore": {
        "mixed": number,
        "negative": number,
        "neutral": number,
        "positive": number
    }
}
],
"requestAttributes": {
    "string": "string"
},
"sessionState": {
    "activeContexts": [
        {
            "name": "string",
            "contextAttributes": {
                "string": "string"
            },
            "timeToLive": {
                "timeToLiveInSeconds": number,
                "turnsToLive": number
            }
        }
    ],
    "sessionAttributes": {
        "string": "string"
    },
    "dialogAction": {
        "slotToElicit": "string",
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        "confirmationState": "Confirmed | Denied | None",
        "name": "string",

```

```
"slots": {
    "string": {
        "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [
                "string"
            ]
        }
    },
    "string": {
        "shape": "List",
        "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [
                "string"
            ]
        }
    },
    "values": [
        {
            "shape": "Scalar",
            "value": {
                "originalValue": "string",
                "interpretedValue": "string",
                "resolvedValues": [
                    "string"
                ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": "string",
                "interpretedValue": "string",
                "resolvedValues": [
                    "string"
                ]
            }
        }
    ]
},
"state": "Failed | Fulfilled | InProgress | ReadyForFulfillment",
"kendraResponse": {
    // Only present when intent is KendraSearchIntent. For details, see
    // https://docs.aws.amazon.com/kendra/latest/dg/
    API_Query.html#API_Query_ResponseSyntax
},
"originatingRequestId": "string"
}
```

Note the following additional information about the event fields:

- **invocationSource** – Indicates the action that called the Lambda function. When the source is `DialogCodeHook`, the Lambda function was called after input from the user. When the source is `FulfillmentCodeHook` the Lambda function was called after all required slots have been filled and the intent is ready for fulfillment.
- **inputTranscript** – The text that was used to process the input from the user. For text or DTMF input, this is the text that the user typed. For speech input, this is the text that was recognized from the speech.

- **interpretations** – One or more intents that Amazon Lex considers possible matches to the user's utterance. For more information, see [Interpretation \(p. 550\)](#).
- **requestAttributes** – Request-specific attributes that the client sends in the request. Use request attributes to pass information that doesn't need to persist for the entire session.
- **sessionState** – The current state of the conversation between the user and your Amazon Lex bot. For more information about the session state, see [SessionState \(p. 556\)](#).

Response format

Amazon Lex expects a response from your Lambda function in the following format:

```
{  
    "sessionState": {  
        "activeContexts": [  
            {  
                "name": "string",  
                "contextAttributes": {  
                    "key": "value"  
                },  
                "timeToLive": {  
                    "timeToLiveInSeconds": number,  
                    "turnsToLive": number  
                }  
            }  
        ],  
        "sessionAttributes": {  
            "string": "string"  
        },  
        "dialogAction": {  
            "slotToElicit": "string",  
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"  
        },  
        "intent": {  
            "confirmationState": "Confirmed | Denied | None",  
            "name": "string",  
            "slots": {  
                "string": {  
                    "value": {  
                        "interpretedValue": "string",  
                        "originalValue": "string",  
                        "resolvedValues": [  
                            "string"  
                        ]  
                    }  
                }  
            },  
            "string": {  
                "shape": "List",  
                "value": {  
                    "originalValue": "string",  
                    "interpretedValue": "string",  
                    "resolvedValues": [  
                        "string"  
                    ]  
                }  
            },  
            "values": [  
                {  
                    "shape": "Scalar",  
                    "value": {  
                        "originalValue": "string",  
                        "interpretedValue": "string",  
                        "resolvedValues": [  
                            "string"  
                        ]  
                    }  
                }  
            ]  
        }  
    }  
}
```

```
        "string"
    ]
}
{
    "shape": "Scalar",
    "value": {
        "originalValue": "string",
        "interpretedValue": "string",
        "resolvedValues": [
            "string"
        ]
    }
}
]
}
},
"state": "Failed | Fulfilled | InProgress | ReadyForFulfillment"
},
"messages": [
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": "string",
    "imageResponseCard": {
        "title": "string",
        "subtitle": "string",
        "imageUrl": "string",
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ]
    }
},
"requestAttributes": {
    "string": "string"
}
}
```

Note the following additional information about the response fields:

- **sessionState** – Required. The current state of the conversation with the user. The actual contents of the structure depends on the type of dialog action.
- **dialogAction** – Determines the type of action that Amazon Lex should take in response to the Lambda function. The `type` field is always required, the `slotToElicit` field is only required when `dialogAction.type` is `ElicitSlot`.
- **intent** – The name of the intent that Amazon Lex should use. Not required when `dialogAction.type` is `Delegate` or `ElicitIntent`.
- **state** – Required. The state can only be `ReadyForFulfillment` if `delegateAction.type` is `Delegate`.
- **messages** – Required if `dialogAction.type` is `ElicitIntent`. One or more messages that Amazon Lex shows to the customer to perform the next turn of the conversation. If you don't supply messages, Amazon Lex uses the appropriate message defined when the bot was created. For more information, see the [Message \(p. 551\)](#) data type.
- **contentType** – The type of message to use.
- **content** – If the message type is `PlainText`, `CustomPayload`, or `SSML`, the `content` field contains the message to send to the user.

- **imageResponseCard** – If the message type is `ImageResponseCard`, contains the definition of the response card to show to the user. For more information, see the [ImageResponseCard \(p. 545\)](#) data type.

Deploying bots

This section provides examples of deploying Amazon Lex bots on messaging platforms, mobile applications, and Web sites.

Topics

- [Aliases \(p. 50\)](#)
- [Deploying an Amazon Lex bot on a messaging platform \(p. 51\)](#)
- [Using a Java application to interact with an Amazon Lex bot \(p. 59\)](#)

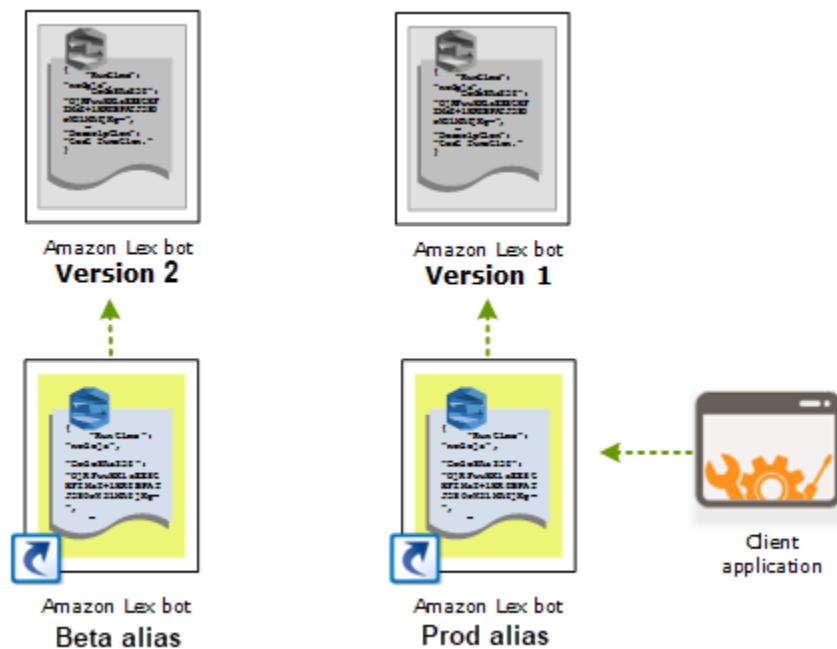
Aliases

Amazon Lex bots support aliases. An *alias* is a pointer to a specific version of a bot. With an alias, you can easily update the version that your client applications are using. For example, you can point an alias to version 1 of your bot. When you are ready to update the bot, you publish version 2 and change the alias to point to the new version. Because your applications use the alias instead of a specific version, all of your clients get the new functionality without needing to be updated.

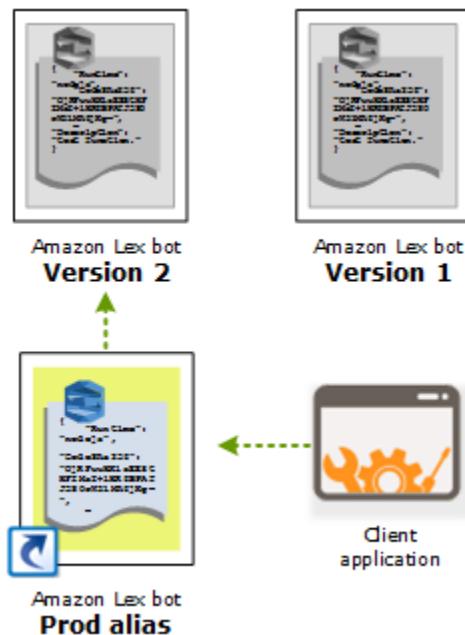
An alias is a pointer to a specific version of an Amazon Lex bot. Use an alias to allow client applications to use a specific version of the bot without requiring the application to track which version that is.

When you create a bot, Amazon Lex creates an alias called `TestBotAlias` that you can use for testing your bot. The `TestBotAlias` alias is always associated with the `Draft` version of your bot. You should only use the `TestBotAlias` alias for testing. Amazon Lex limits the number of runtime requests that you can make to the alias.

The following example shows two versions of an Amazon Lex bot, version version 1 and version 2. Each of these bot versions has an associated alias, `BETA` and `PROD`, respectively. Client applications use the `PROD` alias to access the bot.



When you create a second version of the bot, you can update the alias to point to the new version of the bot using the console or the [UpdateBotAlias \(p. 355\)](#) operation. When you change the alias, all of your client applications use the new version. If there is a problem with the new version, you can roll back to the previous version by simply changing the alias to point to that version.



Note

Although you can test the `Draft` version of a bot in the console, we recommend that when you integrate a bot with your client application, you first publish a version and create an alias that points to that version. Use the alias in your client application for the reasons explained in this section. When you update an alias, Amazon Lex will use the current version for all in-progress sessions. New sessions use the new version.

Deploying an Amazon Lex bot on a messaging platform

This section explains how to deploy Amazon Lex bots on the Facebook, Slack, and Twilio messaging platforms.

Note

When storing your Facebook, Slack, or Twilio configurations, Amazon Lex uses AWS Key Management Service customer master keys (CMK) to encrypt information. The first time that you create a channel to one of these messaging platforms, Amazon Lex creates a default CMK (`aws/lex`) in your AWS account, or you can select your own CMK. Amazon Lex supports only symmetric keys.

When a messaging platform sends a request to Amazon Lex it includes platform-specific information as a request attribute to your Lambda function. Use this attribute to customize the way that your bot behaves. For more information, see [Setting request attributes \(p. 66\)](#).

Common request attribute

Attribute	Description
<code>x-amz-lex:channels:platform</code>	One of the following values:

Attribute	Description
	<ul style="list-style-type: none">• Facebook• Slack• Twilio

Integrating an Amazon Lex bot with Facebook Messenger

You can host your Amazon Lex bot in Facebook Messenger. When you do, Facebook users can interact with your bot to fulfill intents.

Before you start, you need to sign up for a Facebook developer account at <https://developers.facebook.com>.

You need to perform the following steps:

Topics

- [Step 1: Create an Amazon Lex bot \(p. 52\)](#)
- [Step 2: Create a Facebook application \(p. 52\)](#)
- [Step 3: Integrate Facebook Messenger with the Amazon Lex bot \(p. 53\)](#)
- [Step 4: Complete Facebook integration \(p. 53\)](#)
- [Step 5: Test the integration \(p. 54\)](#)

Step 1: Create an Amazon Lex bot

If you don't already have an Amazon Lex bot, create one. In this topic, we assume that you are using the bot that you created in [Exercise 1: Create a bot from an example \(p. 8\)](#). However, you can use any bot.

Next step

[Step 2: Create a Facebook application \(p. 52\)](#)

Step 2: Create a Facebook application

On the Facebook developer portal, create a Facebook application and a Facebook page.

To create a Facebook application

1. Open <https://developers.facebook.com/apps>
2. Choose **Create App**.
3. In the **Create an App**, choose **Manage Business Integrations**, then choose **Continue**.
4. For the **App Display Name**, **App Contact Email**, and **App Purpose** make the appropriate choices for your app. Choose **Create App** to continue.
5. From **Add Products to Your App**, choose **Set Up** from the **Messenger** tile.
6. In the **Access Tokens** section, choose **Add or Remove pages**.
7. Choose a page to use with your app, then choose **Next**.
8. For **What is app allowed to do**, leave the defaults then choose **Done**.

9. On the confirmation page, choose **OK**.
10. In the **Access Tokens** section, choose **Generate Token**, then copy the token. You enter this token in the Amazon Lex console.
11. From the left menu, choose **Settings** and then choose **Basic**.
12. For **App Secret**, choose **Show** and then copy the secret. You enter this token in the Amazon Lex console.

Next step

[Step 3: Integrate Facebook Messenger with the Amazon Lex bot \(p. 53\)](#)

Step 3: Integrate Facebook Messenger with the Amazon Lex bot

In this step you link your Amazon Lex bot with Facebook.

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. From the list of bots, choose the Amazon Lex bot that you created in step 1.
3. In the left menu, choose **Channel integrations** and then choose **Add channel**.
4. In **Create channel**, do the following:
 - a. For **Platform**, choose **Facebook**.
 - b. For **Identity policies**, choose the AWS KMS key to protect channel information. The default key is provided by Amazon Lex.
 - c. For **Integration configuration**, give the channel a name and an optional description. Choose the alias that points to the version of the bot to use, and choose the language that the channel supports.
 - d. For **Additional configuration**, enter the following:
 - **Alias** – A string that identifies the app that is calling Amazon Lex. You can use any string. Record this string, you enter it in the Facebook developer console.
 - **Page access token** – The page access token that you copied from the Facebook developer console.
 - **App secret key** – The secret key that you copied from the Facebook developer console.
 - e. Choose **Create**
 - f. Amazon Lex shows the list of channels for your bot. From the list, choose the channel that you just created.
 - g. From **Callback URL**, record the callback URL. You enter this URL in the Facebook developer console.

Next step

[Step 4: Complete Facebook integration \(p. 53\)](#)

Step 4: Complete Facebook integration

In this step, use the Facebook developer console to complete integration with Amazon Lex.

To complete Facebook Messenger integration

1. Open <https://developers.facebook.com/apps>

2. From the list of apps, choose the app that you are integrating with Facebook Messenger.
3. In the left menu, choose **Messenger**, then choose **Settings**.
4. In the **Webhooks** section, choose **Add Callback URL**.
5. In **Edit Callback URL**, enter the following:
 - **Callback URL** – Enter the callback URL that you recorded from the Amazon Lex console.
 - **Verify Token** – Enter the alias that you entered in the Amazon Lex console.
6. Choose **Verify and Save**.

Next step

[Step 5: Test the integration \(p. 54\)](#)

Step 5: Test the integration

You can now start a conversation from Facebook Messenger with your Amazon Lex bot.

To test the integration between Facebook Messenger and an Amazon Lex bot

1. Open the Facebook page that you associated with your bot in step 2.
2. In the Messenger window, use the test utterances provided in [Exercise 1: Create a bot from an example \(p. 8\)](#).

Integrating an Amazon Lex bot with Slack

This topic provides instructions for integrating an Amazon Lex bot with the Slack messaging application. You perform the following steps:

Topics

- [Step 1: Create an Amazon Lex bot \(p. 54\)](#)
- [Step 2: Sign up for Slack and create a Slack team \(p. 54\)](#)
- [Step 3: Create a Slack application \(p. 55\)](#)
- [Step 4: Integrate the Slack application with the Amazon Lex bot \(p. 55\)](#)
- [Step 5: Complete Slack integration \(p. 56\)](#)
- [Step 6: Test the integration \(p. 57\)](#)

Step 1: Create an Amazon Lex bot

If you don't already have an Amazon Lex bot, create one. In this topic, we assume that you are using the bot that you created in [Exercise 1: Create a bot from an example \(p. 8\)](#). However, you can use any bot.

Next step

[Step 2: Sign up for Slack and create a Slack team \(p. 54\)](#)

Step 2: Sign up for Slack and create a Slack team

Sign up for a Slack account and create a Slack team. For instructions, see [Using Slack](#). In the next section you create a Slack application, which any Slack team can install.

Next step

[Step 3: Create a Slack application \(p. 55\)](#)

Step 3: Create a Slack application

In this section, you do the following:

1. Create a Slack application in the Slack API Console.
2. Configure the application to add interactive messaging to your bot.

At the end of this section, you get application credentials (Client ID, Client Secret, and Verification Token). In the next step, you use this information to integrate the bot in the Amazon Lex console.

To create a Slack application

1. Sign in to the Slack API Console at <http://api.slack.com>.
2. Create an application.

After you have successfully created the application, Slack displays the **Basic Information** page for the application.

3. Configure the application features as follows:

- In the left menu, choose **Interactivity & Shortcuts**.
 - Choose the toggle to turn interactive components on.
 - In the **Request URL** box, specify any valid URL. For example, you can use `https://slack.com`.

Note

For now, enter any valid URL to get the verification token that you need in the next step. You will update this URL after you add the bot channel association in the Amazon Lex console.

- Choose **Save Changes**.

4. In the left menu, in **Settings**, choose **Basic Information**. Record the following application credentials:

- Client ID
- Client Secret
- Verification Token

Next step

[Step 4: Integrate the Slack application with the Amazon Lex bot \(p. 55\)](#)

Step 4: Integrate the Slack application with the Amazon Lex bot

To integrate the Slack application with your Amazon Lex bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. From the list of bots, choose the Amazon Lex bot that you created in step 1.
3. In the left menu, choose **Channel integrations** and then choose **Add channel**.
4. In **Create channel**, do the following:

- a. For **Platform**, choose **Slack**.
- b. For **Identity policies**, choose the AWS KMS key to protect channel information. The default key is provided by Amazon Lex.
- c. For **Integration configuration**, give the channel a name and an optional description. Choose the alias that points to the version of the bot to use, and choose the language that the channel supports.
- d. For **Additional configuration**, enter the following:
 - **Client ID** – enter the client ID from Slack.
 - **Client secret** – enter the client secret from Slack.
 - **Verification token** – enter the verification token from Slack.
 - **Success page URL** – The URL of the page that Slack should open when the user is authenticated. Typically you leave this blank.
5. Choose **Create** to create the channel.
6. Amazon Lex shows the list of channels for your bot. From the list, choose the channel that you just created.
7. From **Callback URL**, record the endpoint and the OAuth endpoint.

Next step

[Step 5: Complete Slack integration \(p. 56\)](#)

Step 5: Complete Slack integration

In this section, use the Slack API console to complete integration with the Slack application.

To complete Slack application integration

1. Sign in to the Slack API console at <http://api.slack.com>. Choose the app that you created in [Step 3: Create a Slack application \(p. 55\)](#).
2. Update the **OAuth & Permissions** feature as follows:
 - a. In the left menu, choose **OAuth & Permissions**.
 - b. In the **Redirect URLs** section, add the OAuth endpoint that Amazon Lex provided in the preceding step. Choose **Add a new Redirect URL**, and then choose **Save URLs**.
 - c. In the **Bot Token Scopes** section, add two permissions with the **Add an OAuth Scope** button. Filter the list with the following text:
 - **chat:write**
 - **team:read**
3. Update the **Interactivity & Shortcuts** feature by updating the **Request URL** value to the endpoint that Amazon Lex provided in the preceding step. Enter the endpoint that you saved in step 4, and then choose **Save Changes**.
4. Subscribe to the **Event Subscriptions** feature as follows:
 - Enable events by choosing the **On** option.
 - Set the **Request URL** value to the endpoint that Amazon Lex provided in the preceding step.
 - In the **Subscribe to Bot Events** section, subscribe to the `message.im` bot event to enable direct messaging between the end user and the Slack bot.
 - Save the changes.
5. Enable sending messages from the messages tab as follows:

- From the left menu, choose **App Home**.
- In the **Show Tabs** section, choose **Allow users to send Slash commands and messages from the messages tab**.

Next step

[Step 6: Test the integration \(p. 57\)](#)

Step 6: Test the integration

Now use a browser window to test the integration of Slack with your Amazon Lex bot.

To test your Slack application

1. Open the Slack console at <http://api.slack.com>. From the list of apps, choose the app to test.
2. Choose **Manage Distribution** under **Settings**. Choose **Add to Slack** to install the application. Choose **Allow** to authorize the bot to respond to messages.
3. You are redirected to your Slack team. From the left menu, in the **Direct Messages** section, choose your bot. If you don't see your bot, choose the plus icon (+) next to **Direct Messages** to search for it.
4. Engage in a chat with your Slack application. Your bot responds to messages.

If you created the bot using Getting started exercise 1, you can use the example conversations from that exercise.

Integrating an Amazon Lex bot with Twilio SMS

This topic provides instructions for integrating an Amazon Lex bot with the Twilio simple message service (SMS). You perform the following steps:

Topics

- [Step 1: Create an Amazon Lex bot \(p. 57\)](#)
- [Step 2: Create a Twilio SMS account \(p. 57\)](#)
- [Step 3: Integrate the Twilio message service endpoint with the Amazon Lex bot \(p. 58\)](#)
- [Step 4: Complete Twilio integration \(p. 58\)](#)
- [Step 5: Test the integration \(p. 58\)](#)

Step 1: Create an Amazon Lex bot

If you don't already have an Amazon Lex bot, create one. In this topic, we assume that you are using the bot that you created in [Exercise 1: Create a bot from an example \(p. 8\)](#). However, you can use any bot.

Next step

[Step 2: Create a Twilio SMS account \(p. 57\)](#)

Step 2: Create a Twilio SMS account

Sign up for a Twilio account and record the following account information:

- **ACCOUNT SID**
- **AUTH TOKEN**

For sign-up instructions, see <https://www.twilio.com/console>.

Next step

[Step 3: Integrate the Twilio message service endpoint with the Amazon Lex bot \(p. 58\)](#)

Step 3: Integrate the Twilio message service endpoint with the Amazon Lex bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. From the list of bots, choose the Amazon Lex bot that you created in step 1.
3. In the left menu, choose **Channel integrations** and then choose **Add channel**.
4. In **Create channel**, do the following:
 - a. For **Platform**, choose **Twilio**.
 - b. For **Identity policies**, choose the AWS KMS key to protect channel information. The default key is provided by Amazon Lex.
 - c. For **Integration configuration**, give the channel a name and an optional description. Choose the alias that points to the version of the bot to use, and choose the language that the channel supports.
 - d. For **Additional configuration**, enter the account SID and authentication token from the Twilio dashboard.
5. Choose **Create**.
6. From the list of channels, choose the channel that you just created.
7. Copy the **Callback URL**.

Next step

[Step 4: Complete Twilio integration \(p. 58\)](#)

Step 4: Complete Twilio integration

Use the Twilio console to complete the integration of your Amazon Lex bot with Twilio SMS.

1. Open the Twilio console at <https://www.twilio.com/console>.
2. From the left menu, choose **All Products & Services**, then choose **Phone Number**.
3. If you have a phone number, choose it. If you don't have a phone number, choose **Buy a Number** to get one.
4. In the **Messaging** section, in **A MESSAGE COMES IN**, enter the callback URL from the Amazon Lex console.
5. Choose **Save**.

Next step

[Step 5: Test the integration \(p. 58\)](#)

Step 5: Test the integration

Use your mobile phone to test the integration between Twilio SMS and your bot. Using your mobile phone, send messages to the Twilio number.

If you created the bot using Getting started exercise 1, you can use the example conversations from that exercise.

Using a Java application to interact with an Amazon Lex bot

The AWS SDK for Java provides an interface that you can use from your Java applications to interact with your bots. Use the SDK for Java to build client applications for users.

The following application interacts with the OrderFlowers bot that you created in [Exercise 1: Create a bot from an example \(p. 8\)](#). It uses the `LexRuntimeV2Client` from the SDK for Java to call the `RecognizeText` ([p. 413](#)) operation to conduct a conversation with the bot.

The output from the conversation looks like this:

```
User : I would like to order flowers
Bot  : What type of flowers would you like to order?
User : 1 dozen roses
Bot  : What day do you want the dozen roses to be picked up?
User : Next Monday
Bot  : At what time do you want the dozen roses to be picked up?
User : 5 in the evening
Bot  : Okay, your dozen roses will be ready for pickup by 17:00 on 2021-01-04. Does this sound okay?
User : Yes
Bot  : Thanks.
```

For the JSON structures that are sent between the client application and the Amazon Lex bot, see [Exercise 2: Review the conversation flow \(p. 9\)](#).

To run the application, you must provide the following information:

- `botId` – The identifier assigned to the bot when you created it. You can see the bot ID in the Amazon Lex console on the bot **Settings** page.
- `botAliasId` – The identifier assigned to the bot alias when you created it. You can see the bot alias ID in the Amazon Lex console on the **Aliases** page. If you can't see the alias ID in the list, choose the gear icon on the upper right and turn on **Alias ID**.
- `localeId` – The identifier of the locale that you used for your bot. For a list of locales, see [Languages and locales supported by Amazon Lex \(p. 4\)](#).
- `accessKey` and `secretKey` – The authentication keys for your account. If you don't have a set of keys, create them using the AWS Identity and Access Management console.
- `sessionId` – An identifier for the session with the Amazon Lex bot. In this case, the code uses a random UUID.
- `region` – If your bot is not in the US East (N. Virginia) Region, make sure that you change the Region.

The applications uses a function called `getRecognizeTextRequest` to create individual requests to the bot. The function builds a request with the required parameters to send to Amazon Lex.

```
package com.lex.recognizetext.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
```

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2Client;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextRequest;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextResponse;

import java.net.URISyntaxException;
import java.util.UUID;

/**
 * This is a sample application to interact with a bot using RecognizeText API.
 */
public class OrderFlowersSampleApplication {

    public static void main(String[] args) throws URISyntaxException, InterruptedException
    {
        String botId = "";
        String botAliasId = "";
        String localeId = "en_US";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.US_EAST_1; // pick an appropriate region

        AwsBasicCredentials awsCreds = AwsBasicCredentials.create(accessKey, secretKey);
        AwsCredentialsProvider awsCredentialsProvider =
        StaticCredentialsProvider.create(awsCreds);

        LexRuntimeV2Client lexV2Client = LexRuntimeV2Client
            .builder()
            .credentialsProvider(awsCredentialsProvider)
            .region(region)
            .build();

        // utterance 1
        String userInput = "I would like to order flowers";
        RecognizeTextRequest recognizeTextRequest = getRecognizeTextRequest(botId,
        botAliasId, localeId, sessionId, userInput);
        RecognizeTextResponse recognizeTextResponse =
        lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 2
        userInput = "1 dozen roses";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
        sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 3
        userInput = "next monday";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
        sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });
    }
}

```

```
});

// utterance 4
userInput = "5 in evening";
recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

System.out.println("User : " + userInput);
recognizeTextResponse.messages().forEach(message -> {
    System.out.println("Bot : " + message.content());
});

// utterance 5
userInput = "Yes";
recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

System.out.println("User : " + userInput);
recognizeTextResponse.messages().forEach(message -> {
    System.out.println("Bot : " + message.content());
});
}

private static RecognizeTextRequest getRecognizeTextRequest(String botId, String
botAliasId, String localeId, String sessionId, String userInput) {
    RecognizeTextRequest recognizeTextRequest = RecognizeTextRequest.builder()
        .botAliasId(botAliasId)
        .botId(botId)
        .localeId(localeId)
        .sessionId(sessionId)
        .text(userInput)
        .build();
    return recognizeTextRequest;
}
}
```

Using bots

Topics

- [Managing conversations \(p. 62\)](#)
- [Managing conversation context \(p. 63\)](#)
- [Managing sessions with the Amazon Lex API \(p. 69\)](#)
- [Analyzing the sentiment of user utterances \(p. 71\)](#)
- [Using intent confidence scores \(p. 72\)](#)

Managing conversations

After you build a bot, you integrate your client application with the Amazon Lex runtime operations to hold conversations with your bot.

When a user starts a conversation with your bot, Amazon Lex creates a *session*. A session encapsulates the information exchanged between your application and the bot. For more information, see [Managing sessions with the Amazon Lex API \(p. 69\)](#).

A typical conversation involves a back and forth flow between the user and a bot. For example:

```
User : I'd like to make an appointment
Bot : What type of appointment would you like to schedule?
User : dental
Bot : When should I schedule your dental appointment?
User : Tomorrow
Bot : At what time do you want to schedule the dental appointment on 2021-01-01?
User : 9 am
Bot : 09:00 is available, should I go ahead and book your appointment?
User : Yes
Bot : Thank you. Your appointment has been set successfully.
```

When you use the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation, you must manage the conversation in your client application. When you use the [StartConversation \(p. 426\)](#) operation, Amazon Lex manages the conversation for you.

To manage the conversation, you must send user utterances to the bot until the conversation reaches a logical end. The current conversation is captured in session state. The session state is updated after each user utterance. The session state contains the current state of the conversation and is returned by the bot in a response. to each user utterance.

A conversation can be in any of the following states:

- **ElicitIntent** – Indicates that the bot has not yet determined the user's intent.
- **ElicitSlot** – Indicates that the bot has detected the user's intent and is gathering the required information to fulfill the intent.
- **ConfirmIntent** – Indicates that the bot is waiting for the user to confirm that the information collected is correct.
- **Closed** – Indicates that the user's intent is complete and that the conversation with the bot reached a logical end.

A user can specify a new intent after the first intent is completed. For more information, see [Managing conversation context \(p. 63\)](#).

An intent can have the one of the following states:

- **InProgress** – Indicates that the bot is gathering information necessary to complete the intent. This is in conjunction with the `ElicitSlot` conversation state.
- **Waiting** – Indicates that the user requested the bot to wait when the bot asked for information for a specific slot.
- **Fulfilled** – Indicates that the business logic in a Lambda function associated with the intent ran successfully.
- **ReadyForFulfillment** – Indicates that the bot gathered all of the information required to fulfill the intent and that the client application can run fulfillment business logic.
- **Failed** – Indicates that an intent has failed.

Managing conversation context

Conversation context is information that the user, your client application, or a Lambda function provides to a Amazon Lex bot to fulfill an intent. Conversation context includes slot data that the user provides, request attributes set by the client application, and session attributes that the client application and Lambda functions create.

Topics

- [Setting intent context \(p. 63\)](#)
- [Using default slot values \(p. 64\)](#)
- [Setting session attributes \(p. 65\)](#)
- [Setting request attributes \(p. 66\)](#)
- [Setting the session timeout \(p. 67\)](#)
- [Sharing information between intents \(p. 67\)](#)
- [Setting complex attributes \(p. 68\)](#)

Setting intent context

You can have Amazon Lex trigger intents based on *context*. A *context* is a state variable that can be associated with an intent when you define a bot. You configure the contexts for an intent when you create the intent using the console or using the [CreateIntent \(p. 192\)](#) operation. You can only use context in the English (US) (en-US) locale.

There are two types of relationships for contexts, output contexts and input contexts. An *output context* becomes active when an associated intent is fulfilled. An output context is returned to your application in the response from the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation, and it is set for the current session. After a context is activated, it stays active for the number of turns or time limit configured when the context was defined.

An *input context* specifies conditions under which an intent can be recognized. An intent can only be recognized during a conversation when all of its input contexts are active. An intent with no input contexts is always eligible for recognition.

Amazon Lex automatically manages the lifecycle of contexts that are activated by fulfilling intents with output contexts. You can also set active contexts in a call to the `RecognizeText` or `RecognizeUtterance` operation.

You can also set the context of a conversation using the Lambda function for the intent. The output context from Amazon Lex is sent to the Lambda function input event. The Lambda function can send contexts in its response. For more information, see [Using an AWS Lambda function \(p. 43\)](#).

For example, suppose you have an intent to book a rental car that is configured to return an output context called "book_car_fulfilled". When the intent is fulfilled, Amazon Lex sets the output context variable "book_car_fulfilled". Since "book_car_fulfilled" is an active context, an intent with the "book_car_fulfilled" context set as an input context is now considered for recognition, as long as a user utterance is recognized as an attempt to elicit that intent. You can use this for intents that only make sense after booking a car, such as emailing a receipt or modifying a reservation.

Output context

Amazon Lex makes an intent's output contexts active when the intent is fulfilled. You can use the output context to control the intents eligible to follow up the current intent.

Each context has a list of parameters that are maintained in the session. The parameters are the slot values for the fulfilled intent. You can use these parameters to pre-populate slot values for other intents. For more information, see [Using default slot values \(p. 64\)](#).

You configure the output context when you create an intent with the console or with the [CreateIntent \(p. 192\)](#) operation. You can configure an intent with more than one output context. When the intent is fulfilled, all of the output contexts are activated and returned in the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) response.

When you define an output context you also define its *time to live*, the length of time or number of turns that the context is included in responses from Amazon Lex. A *turn* is one request from your application to Amazon Lex. Once the number of turns or the time has expired, the context is no longer active.

Your application can use the output context as needed. For example, your application can use the output context to:

- Change the behavior of the application based on the context. For example, a travel application could have a different action for the context "book_car_fulfilled" than "rental_hotel_fulfilled."
- Return the output context to Amazon Lex as the input context for the next utterance. If Amazon Lex recognizes the utterance as an attempt to elicit an intent, it uses the context to limit the intents that can be returned to ones with the specified context.

Input context

You set an input context to limit the points in the conversation where the intent is recognized. Intents without an input context are always eligible to be recognized.

You set the input contexts that an intent responds to using the console or the [CreateIntent](#) operation. An intent can have more than one input context.

For an intent with more than one input context, all contexts must be active to trigger the intent. You can set an input context when you call the [RecognizeText \(p. 413\)](#), [RecognizeUtterance \(p. 419\)](#), or [PutSession \(p. 408\)](#) operation.

You can configure the slots in an intent to take default values from the current active context. Default values are used when Amazon Lex recognizes a new intent but doesn't receive a slot value. You specify the context name and slot name in the form `#context-name.parameter-name` when you define the slot. For more information, see [Using default slot values \(p. 64\)](#).

Using default slot values

When you use a default value, you specify a source for a slot value to be filled for new intents when no slot is provided by the user's input. This source can be previous dialog, request or session attributes, or a fixed value that you set at build-time.

You can use the following as the source for your default values.

- Previous dialog (contexts) – #context-name.parameter-name
- Session attributes – [attribute-name]
- Request attributes – <attribute-name>
- Fixed value – Any value that doesn't match the previous

When you use the [CreateIntent \(p. 192\)](#) operation to add slots to an intent, you can add a list of default values. Default values are used in the order that they are listed. For example, suppose you have an intent with a slot with the following definition:

```
"slots": [
  {
    "botId": "string",
    "defaultValueSpec": {
      "defaultValueList": [
        {
          "defaultValue": "#book-car-fulfilled.startDate"
        },
        {
          "defaultValue": "[reservationStartDate]"
        }
      ]
    },
    Other slot configuration settings
  }
]
```

When the intent is recognized, the slot named "reservation-start-date" has its value set to one of the following.

1. If the "book-car-fulfilled" context is active, the value of the "startDate" parameter is used as the default value.
2. If the "book-car-fulfilled" context is not active, or if the "startDate" parameter is not set, the value of the "reservationStartDate" session attribute is used as the default value.
3. If neither of the first two default values are used, then the slot doesn't have a default value and Amazon Lex will elicit a value as usual.

If a default value is used for the slot, the slot is not elicited even if it is required.

Setting session attributes

Session attributes contain application-specific information that is passed between a bot and a client application during a session. Amazon Lex passes session attributes to all Lambda functions configured for a bot. If a Lambda function adds or updates session attributes, Amazon Lex passes the new information back to the client application.

Use session attributes in your Lambda functions to initialize a bot and to customize prompts and response cards. For example:

- Initialization — In a pizza ordering bot, the client application passes the user's location as a session attribute in the first call to the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation. For example, "Location": "111 Maple Street". The Lambda function uses this information to find the closest pizzeria to place the order.
- Personalize prompts — Configure prompts and response cards to refer to session attributes. For example, "Hey [FirstName], what toppings would you like?" If you pass the user's first name as

a session attribute (`{"FirstName": "Vivian"}`), Amazon Lex substitutes the name for the placeholder. It then sends a personalized prompt to the user, "Hey Vivian, which toppings would you like?"

Session attributes persist for the duration of the session. Amazon Lex stores them in an encrypted data store until the session ends. The client can create session attributes in a request by calling either the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation with the `sessionAttributes` field set to a value. A Lambda function can create a session attribute in a response. After the client or a Lambda function creates a session attribute, the stored attribute value is used any time that the client application doesn't include `sessionAttribute` field in a request to Amazon Lex.

For example, suppose you have two session attributes, `{"x": "1", "y": "2"}`. If the client calls the `RecognizeText` or `RecognizeUtterance` operation without specifying the `sessionAttributes` field, Amazon Lex calls the Lambda function with the stored session attributes (`{"x": 1, "y": 2}`). If the Lambda function doesn't return session attributes, Amazon Lex returns the stored session attributes to the client application.

If either the client application or a Lambda function passes session attributes, Amazon Lex updates the stored session attributes. Passing an existing value, such as `{"x": 2}`, updates the stored value. If you pass a new set of session attributes, such as `{"z": 3}`, the existing values are removed and only the new value is kept. When an empty map, `{}`, is passed, stored values are erased.

To send session attributes to Amazon Lex, you create a string-to-string map of the attributes. The following shows how to map session attributes:

```
{  
    "attributeName": "attributeValue",  
    "attributeName": "attributeValue"  
}
```

For the `RecognizeText` operation, you insert the map into the body of the request using the `sessionAttributes` field of the `sessionState` structure, as follows:

```
"sessionState": {  
    "sessionAttributes": {  
        "attributeName": "attributeValue",  
        "attributeName": "attributeValue"  
    }  
}
```

For the `PostContent` operation, you base64 encode the map, and then send it as part of the `x-amz-lex-session-state` header.

If you are sending binary or structured data in a session attribute, you must first transform the data to a simple string. For more information, see [Setting complex attributes \(p. 68\)](#).

Setting request attributes

Request attributes contain request-specific information and apply only to the current request. A client application sends this information to Amazon Lex. Use request attributes to pass information that doesn't need to persist for the entire session. You can create your own request attributes or you can use predefined attributes. To send request attributes, use the `x-amz-lex-request-attributes` header in a [RecognizeUtterance \(p. 419\)](#) or the `requestAttributes` field in a [RecognizeText \(p. 413\)](#) request. Because request attributes don't persist across requests like session attributes do, they are not returned in `RecognizeUtterance` or `RecognizeText` responses.

Note

To send information that persists across requests, use session attributes.

Setting user-defined request attributes

A *user-defined request attribute* is data that you send to your bot in each request. You send the information in the `amz-lex-request-attributes` header of a `RecognizeUtterance` request or in the `requestAttributes` field of a `RecognizeText` request.

To send request attributes to Amazon Lex, you create a string-to-string map of the attributes. The following shows how to map request attributes:

```
{  
    "attributeName": "attributeValue",  
    "attributeName": "attributeValue"  
}
```

For the `PostText` operation, you insert the map into the body of the request using the `requestAttributes` field, as follows:

```
"requestAttributes": {  
    "attributeName": "attributeValue",  
    "attributeName": "attributeValue"  
}
```

For the `PostContent` operation, you base64 encode the map, and then send it as the `x-amz-lex-request-attributes` header.

If you are sending binary or structured data in a request attribute, you must first transform the data to a simple string. For more information, see [Setting complex attributes \(p. 68\)](#).

Setting the session timeout

Amazon Lex retains context information—slot data and session attributes—until a conversation session ends. To control how long a session lasts for a bot, set the session timeout. By default, session duration is 5 minutes, but you can specify any duration between 0 and 1,440 minutes (24 hours).

For example, suppose that you create a `ShoeOrdering` bot that supports intents such as `OrderShoes` and `GetOrderStatus`. When Amazon Lex detects that the user's intent is to order shoes, it asks for slot data. For example, it asks for shoe size, color, brand, etc. If the user provides some of the slot data but doesn't complete the shoe purchase, Amazon Lex remembers all of the slot data and session attributes for the entire session. If the user returns to the session before it expires, they can provide the remaining slot data, and complete the purchase.

In the Amazon Lex console, you set the session timeout when you create a bot. With the AWS command line interface (AWS CLI) or API, you set the timeout when you create a bot with the [CreateBot \(p. 168\)](#) operation by setting the `idleSessionTTLInSeconds` field.

Sharing information between intents

Amazon Lex supports sharing information between intents. To share between intents, use output contexts or session attributes.

To use output contexts, you define an output context when you create or update an intent. When the intent is fulfilled, responses from Amazon Lex contain the context and slot values from the intent as context parameters. You can use these parameters as default values in subsequent intents or in your application code or Lambda functions.

To use session attributes, you set the attributes in your Lambda or application code. For example, a user of the `ShoeOrdering` bot starts by ordering shoes. The bot engages in a conversation with

the user, gathering slot data, such as shoe size, color, and brand. When the user places an order, the Lambda function that fulfills the order sets the `orderNumber` session attribute, which contains the order number. To get the status of the order, the user uses the `GetOrderStatus` intent. The bot can ask the user for slot data, such as order number and order date. When the bot has the required information, it returns the status of the order.

If you think that your users might switch intents during the same session, you can design your bot to return the status of the latest order. Instead of asking the user for order information again, you use the `orderNumber` session attribute to share information across intents and fulfill the `GetOrderStatus` intent. The bot does this by returning the status of the last order that the user placed.

Setting complex attributes

Session and request attributes are string-to-string maps of attributes and values. In many cases, you can use the string map to transfer attribute values between your client application and a bot. In some cases, however, you might need to transfer binary data or a complex structure that can't be easily converted to a string map. For example, the following JSON object represents an array of the three most populous cities in the United States:

```
{  
  "cities": [  
    {  
      "city": {  
        "name": "New York",  
        "state": "New York",  
        "pop": "8537673"  
      }  
    },  
    {  
      "city": {  
        "name": "Los Angeles",  
        "state": "California",  
        "pop": "3976322"  
      }  
    },  
    {  
      "city": {  
        "name": "Chicago",  
        "state": "Illinois",  
        "pop": "2704958"  
      }  
    }  
  ]  
}
```

This array of data doesn't translate well to a string-to-string map. In such a case, you can transform an object to a simple string so that you can send it to your bot with the [RecognizeText \(p. 413\)](#) and [RecognizeUtterance \(p. 419\)](#) operations.

For example, if you are using JavaScript, you can use the `JSON.stringify` operation to convert an object to JSON, and the `JSON.parse` operation to convert JSON text to a JavaScript object:

```
// To convert an object to a string.  
var jsonString = JSON.stringify(object, null, 2);  
// To convert a string to an object.  
var obj = JSON.parse(JSON string);
```

To send attributes with the `RecognizeUtterance` operation, you must base64 encode the attributes before you add them to the request header, as shown in the following JavaScript code:

```
var encodedAttributes = new Buffer(attributeString).toString("base64");
```

You can send binary data to the `RecognizeText` and `RecognizeUtterance` operations by first converting the data to a base64-encoded string, and then sending the string as the value in the session attributes:

```
"sessionAttributes" : {  
    "binaryData": "base64 encoded data"  
}
```

Managing sessions with the Amazon Lex API

When a user starts a conversation with your bot, Amazon Lex creates a *session*. The information exchanged between your application and Amazon Lex makes up the session state for the conversation. When you make a request, the session is identified by an identifier that you specify. For more information about the session identifier, see the `sessionId` field in the [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation.

You can modify the session state sent between your application and your bot. For example, you can create and modify session attributes that contain custom information about the session, and you can change the flow of the conversation by setting the dialog context to interpret the next utterance.

There are three ways that you can update session state.

- Pass the session information inline as part of a call to the `RecognizeText` or `RecognizeUtterance` operation.
- Use a Lambda function with the `RecognizeText` or `RecognizeUtterance` operation that is called after each turn of the conversation. For more information, see [Using an AWS Lambda function \(p. 43\)](#). The other is to use the Amazon Lex runtime API in your application to make changes to the session state.
- Use operations that enable you to manage session information for a conversation with your bot. The operations are the [PutSession \(p. 408\)](#) operation, the [GetSession \(p. 404\)](#) operation, and the [DeleteSession \(p. 401\)](#) operation. You use these operations to get information about the state of your user's session with your bot, and to have fine-grained control over the state.

Use the `GetSession` operation when you want to get the current state of the session. The operation returns the current state of the session, including the state of the dialog with your user, any session attributes that have been set and slot values for the current intent and any other intents that Amazon Lex has identified as possible intents that match the user's utterance.

The `PutSession` operation enables you to directly manipulate the current session state. You can set the session, including the type of dialog action that the bot will perform next and the messages that Amazon Lex sends to the user. This gives you control over the flow of the conversation with the bot. Set the `dialogActionType` field to `Delegate` to have Amazon Lex determine the next action for the bot.

You can use the `PutSession` operation to create a new session with a bot and set the intent that the bot should start with. You can also use the `PutSession` operation to change from one intent to another. When you create a session or change the intent you also can set session state, such as slot values and session attributes. When the new intent is finished, you have the option of restarting the prior intent.

The response from the `PutSession` operation contains the same information as the `RecognizeUtterance` operation. You can use this information to prompt the user for the next piece of information, just as you would with the response from the `RecognizeUtterance` operation.

Use the `DeleteSession` operation to remove an existing session and start over with a new session. For example, when you are testing your bot you can use the `DeleteSession` operation to remove test sessions from your bot.

The session operations work with your fulfillment Lambda functions. For example, if your Lambda function returns `Failed` as the fulfillment state you can use the `PutSession` operation to set the dialog action type to `close` and `fulfillmentState` to `ReadyForFulfillment` to retry the fulfillment step.

Here are some things that you can do with the session operations:

- Have the bot start a conversation instead of waiting for the user.
- Switch intents during a conversation.
- Return to a previous intent.
- Start or restart a conversation in the middle of the interaction.
- Validate slot values and have the bot re-prompt for values that are not valid.

Each of these are described further below.

Starting a new session

If you want to have the bot start the conversation with your user, you can use the `PutSession` operation.

- Create a welcome intent with no slots and a conclusion message that prompts the user to state an intent. For example, "What would you like to order? You can say 'Order a drink' or 'Order a pizza.'"
- Call the `PutSession` operation. Set the intent name to the name of your welcome intent and set the dialog action to `Delegate`.
- Amazon Lex will respond with the prompt from your welcome intent to start the conversation with your user.

Switching intents

You can use the `PutSession` operation to switch from one intent to another. You can also use it to switch back to a previous intent. You can use the `PutSession` operation to set session attributes or slot values for the new intent.

- Call the `PutSession` operation. Set the intent name to the name of the new intent and set the dialog action to `Delegate`. You can also set any slot values or session attributes required for the new intent.
- Amazon Lex will start a conversation with the user using the new intent.

Resuming a prior intent

To resume a prior intent you use the `GetSession` operation to get the state of the intent, perform the needed interaction, and then use the `PutSession` operation to set the intent to its previous dialog state.

- Call the `GetSession` operation. Store the state of the intent.
- Perform another interaction, such as fulfilling a different intent.
- Using the information saved information for the previous intent, call the `PutSession` operation. This will return the user to the previous intent in the same place in the conversation.

In some cases it may be necessary to resume your user's conversation with your bot. For example, say that you have created a customer service bot. Your application determines that the user needs to talk to a customer service representative. After talking to the user, the representative can direct the conversation back to the bot with the information that they collected.

To resume a session, use steps similar to these:

- Your application determines that the user needs to speak to a customer service representative.
- Use the `GetSession` operation to get the current dialog state of the intent.
- The customer service representative talks to the user and resolves the issue.
- Use the `PutSession` operation to set the dialog state of the intent. This may include setting slot values, setting session attributes, or changing the intent.
- The bot resumes the conversation with the user.

Validating slot values

You can validate responses to your bot using your client application. If the response isn't valid, you can use the `PutSession` operation to get a new response from your user. For example, suppose that your flower ordering bot can only sell tulips, roses, and lilies. If the user orders carnations, your application can do the following:

- Examine the slot value returned from the `PostText` or `PostContent` response.
- If the slot value is not valid, call the `PutSession` operation. Your application should clear the slot value, set the `slotToElicit` field, and set the `dialogAction.type` value to `elicitSlot`. Optionally, you can set the `message` and `messageFormat` fields if you want to change the message that Amazon Lex uses to elicit the slot value.

Analyzing the sentiment of user utterances

You can use sentiment analysis to determine the sentiments expressed in a user utterance. With the sentiment information you can manage conversation flow or perform post-call analysis. For example, if the user sentiment is negative you can create a flow to hand over a conversation to a human agent.

Amazon Lex integrates with Amazon Comprehend to detect user sentiment. The response from Amazon Comprehend indicates whether the overall sentiment of the text is positive, neutral, negative, or mixed. The response contains the most likely sentiment for the user utterance and the scores for each of the sentiment categories. The score represents the likelihood that the sentiment was correctly detected.

You enable sentiment analysis for a bot using the console or by using the Amazon Lex API. You enable sentiment analysis on an alias for the bot. On the Amazon Lex console:

1. Choose an alias.
2. In **Details**, choose **Edit**.
3. Choose **Enable sentiment analysis** to sentiment analysis on or off.
4. Choose **Confirm** to save your changes.

If you are using the API, call the [CreateBotAlias \(p. 173\)](#) operation with the `detectSentiment` field set to `true`.

When sentiment analysis is enabled, the response from the [RecognizeText \(p. 413\)](#) and [RecognizeUtterance \(p. 419\)](#) operations return a field called `sentimentResponse` in the `interpretations` structure with other metadata. The `sentimentResponse` field has two fields,

`sentiment` and `sentimentScore`, that contain the result of the sentiment analysis. If you are using a Lambda function, the `sentimentResponse` field is included in the event data sent to your function.

The following is an example of the `sentimentResponse` field returned as part of the `RecognizeText` or `RecognizeUtterance` response.

```
sentimentResponse {  
    "sentimentScore": {  
        "mixed": 0.030585512690246105,  
        "positive": 0.94992071056365967,  
        "neutral": 0.0141543131828308,  
        "negative": 0.00893945890665054  
    },  
    "sentiment": "POSITIVE"  
}
```

Amazon Lex calls Amazon Comprehend on your behalf to determine the sentiment in every utterance processed by the bot. By enabling sentiment analysis, you agree to the service terms and agreements for Amazon Comprehend. For more information about pricing for Amazon Comprehend, see [Amazon Comprehend Pricing](#).

For more information about how Amazon Comprehend sentiment analysis works, see [Determine the sentiment](#) in the *Amazon Comprehend Developer Guide*.

Using intent confidence scores

When a user makes an utterance, Amazon Lex uses natural language understanding (NLU) to understand the user's request and return the proper intent. By default Amazon Lex returns the most likely intent defined by your bot.

In some cases it may be difficult for Amazon Lex to determine the most likely intent. For example, the user might make an ambiguous utterance, or there might be two intents that are similar. To help determine the proper intent, you can combine your domain knowledge with the *NLU confidence scores* in a list of alternative intents. A confidence score is a rating that Amazon Lex provides that shows how confident it is that an intent is the correct intent.

To determine the difference between two alternative intents, you can compare their confidence scores. For example, if one intent has a confidence score of 0.95 and another has a score of 0.65, the first intent is probably correct. However, if one intent has a score of 0.75 and another has a score of 0.72, there is ambiguity between the two intents that you may be able to discriminate using domain knowledge in your application.

You can also use confidence scores to create test applications that determine if changes to an intent's utterances make a difference in the behavior of the bot. For example, you can get the confidence scores for a bot's intents using a set of utterances, then update the intents with new utterances. You can then check the confidence scores to see if there was an improvement.

The confidence scores that Amazon Lex returns are comparative values. You should not rely on them as an absolute score. The values may change based on improvements to Amazon Lex.

Amazon Lex returns the most likely intent and up to 4 alternative intents with their associated scores in the `interpretations` structure in each response. If all of the confidence scores are less than a threshold, Amazon Lex includes the `AMAZON.FallbackIntent`. If you have the `AMAZON.KendraSearchIntent` configured, Amazon Lex returns it as well. You can use the default threshold or you can set your own threshold.

The following JSON code shows the `interpretations` structure in the response from the [RecognizeText \(p. 413\)](#) operation.

```
"interpretations": [  
    {  
        "intent": {  
            "confirmationState": "string",  
            "name": "string",  
            "slots": {  
                "string" : {  
                    "value": {  
                        "interpretedValue": "string",  
                        "originalValue": "string",  
                        "resolvedValues": [ "string" ]  
                    }  
                }  
            },  
            "state": "string"  
        },  
        "nluConfidence": {  
            "score": number  
        }  
    }  
]
```

To change the confidence threshold, set it in the console or using the [UpdateBotLocale \(p. 361\)](#) operation. The threshold must be a number between 1.00 and 0.00.

To use the console, set the confidence threshold when you create or update your bot. You set the confidence threshold for each language that you add to your bot.

To set the confidence threshold when creating a bot (Console)

- On **Add language**, enter a value in the **Confidence score threshold** field.

To set or update the confidence threshold (SDK)

- Set the `nluIntentConfidenceThreshold` parameter of the [CreateBotLocale \(p. 179\)](#) operation. The following JSON code shows the parameter being set.

```
"nluIntentConfidenceThreshold": 0.75,
```

Session Management

To change the intent that Amazon Lex uses in a conversation with the user, you can use the response from your dialog code hook Lambda function, or you can use the session management APIs in your custom application.

Using a Lambda function

When you use a Lambda function, Amazon Lex calls it with a JSON structure that contains the input to the function. The JSON structure contains a field called `currentIntent` that contains the intent that Amazon Lex has identified as the most likely intent for the user's utterance. The JSON structure also includes an `alternativeIntents` field that contains up to four additional intents that may satisfy the user's intent. Each intent includes a field called `nluIntentConfidenceScore` that contains the confidence score that Amazon Lex assigned to the intent.

To use an alternative intent, you specify it in the `ConfirmIntent` or the `ElicitSlot` dialog action in your Lambda function.

For more information, see [Using an AWS Lambda function \(p. 43\)](#).

Using the Session Management API

To use a different intent from the current intent, use the [PutSession \(p. 408\)](#) operation. For example, if you decide that the first alternative is preferable to the intent that Amazon Lex chose, you can use the PutSession operation to change intents so that the next intent that the user interacts with is the one that you selected.

For more information, see [Managing sessions with the Amazon Lex API \(p. 69\)](#).

Streaming to an Amazon Lex bot

You can use the Amazon Lex streaming API to start a bidirectional stream between an Amazon Lex bot and your application. Starting a stream enables the bot to manage the conversation between the bot and the user. The bot responds to user input without you writing code to handle responses from the user. The bot can:

- Handle interruptions from the user while it's playing a prompt. For more information, see [Enabling your bot to be interrupted by your user \(p. 91\)](#).
- Wait for the user to provide input. For example, the bot can wait for the user to gather credit card information. For more information, see [Enabling the bot to wait for the user to provide additional information \(p. 92\)](#).
- Take both dual-tone multiple-frequency (DTMF) and audio input in the same stream.
- Handle pauses in user input better than if you were managing the conversation from your application.

Not only does the Amazon Lex bot respond to data sent from your application, but it also sends information about the state of the conversation to your application. You can use this information to change how your application responds to customers.

The Amazon Lex bot also monitors the connection between the bot and your application. It can determine if the connection has timed out.

To use the API to start a stream to an Amazon Lex bot, see [Starting a stream to a bot \(p. 76\)](#).

When you start streaming to an Amazon Lex bot from your application, you can configure the bot to accept either audio input or text input from the user. You can also choose whether the user receives audio or text in response to their input.

If you've configured the Amazon Lex bot to accept audio input from the user, it can't take text input. If you've configured the bot to accept text input, the user can only use written text to communicate with it.

When an Amazon Lex bot takes a streaming audio input, the bot determines when a user begins speaking and when they stop speaking. It handles any pauses or any interruptions from the user. It can also take both DTMF (dual-tone multi-frequency) input and speech input in the same stream. This enables the user to interact with the bot more naturally. You can present users with welcome messages and prompts. You can also enable users to interrupt those messages and prompts.

When you start a bidirectional stream, Amazon Lex uses the [HTTP/2 protocol](#). Your application and the bot exchange data in a single stream as a series of *events*. An event can be one of the following:

- Text, audio, or DTMF input from the user.
- Signals from the application to the Amazon Lex bot. These include an indication that audio playback of a message has been completed, or that the user has disconnected from the session.

For more information about events, see [Starting a stream to a bot \(p. 76\)](#). For information about how to encode events, see [Event stream encoding \(p. 93\)](#).

Topics

- [Starting a stream to a bot \(p. 76\)](#)

- [Event stream encoding \(p. 93\)](#)

Starting a stream to a bot

You use the [StartConversation \(p. 426\)](#) operation to start a stream between the user and the Amazon Lex bot in your application. The POST request from the application establishes a connection between your application and the Amazon Lex bot. This enables your application and the bot to start exchanging information with each other through events.

The StartConversation operation is supported only in the following SDKs:

- [AWS SDK for C++](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

The first event your application must send to the Amazon Lex bot is a [ConfigurationEvent \(p. 539\)](#). This event includes information such as the response type format. The following are the parameters that you can use in a configuration event:

- **responseContentType** – Determines whether the bot responds to user input with text or speech.
- **sessionState** – Information relating to the streaming session with the bot such as predetermined intent or dialog state.
- **welcomeMessages** – Specifies the welcome messages that play for the user at the beginning of their conversation with a bot. These messages play before the user provides any input. To enable a welcome message, you must also specify values for the **sessionState** and **dialogAction** parameters.
- **disablePlayback** – Determines whether the bot should wait for a cue from the client before it starts listening for caller input. By default, playback is enabled, so the value of this field is `false`.
- **requestAttributes** – Provides additional information for the request.

For information about how to specify values for the preceding parameters, see the [ConfigurationEvent \(p. 539\)](#) data type of the [StartConversation \(p. 426\)](#) operation.

Each stream between a bot and your application can only have one configuration event. After your application has sent a configuration event, the bot can take additional communication from your application.

If you've specified that your user is using audio to communicate with the Amazon Lex bot, your application can send the following events to the bot during the course of that conversation:

- [AudioInputEvent \(p. 534\)](#) – Contains an audio chunk that has maximum size of 320 bytes. Your application must use multiple audio input events to send a message from the server to the bot. Every audio input event in the stream must have the same audio format.
- [DTMFInputEvent \(p. 543\)](#) – Sends a DTMF input to the bot. Each DTMF key press corresponds to a single event.
- [PlaybackCompletionEvent \(p. 552\)](#) – Informs the server that a response from the user's input has been played back to them. You must use a playback completion event if you're sending an audio response to the user. If `disablePlayback` of your configuration event is `true`, you can't use this feature.
- [DisconnectionEvent \(p. 542\)](#) – Informs the bot that the user has disconnected from the conversation.

If you've specified that the user is using text to communicate with the bot, your application can send the following events to the bot during the course of that conversation:

- [TextInputEvent \(p. 564\)](#) – Text that is sent from your application to the bot. You can have up to 512 characters in a text input event.
- [PlaybackCompletionEvent \(p. 552\)](#) – Informs the server that a response from the user's input has been played back to them. You must use this event if you're playing audio back to the user. If `disablePlayback` of your configuration event is `true`, you can't use this feature.
- [DisconnectionEvent \(p. 542\)](#) – Informs the bot that the user has disconnected from the conversation.

You must encode every event that you send to an Amazon Lex bot in the correct format. For more information, see [Event stream encoding \(p. 93\)](#).

Every event has an event ID. To help troubleshoot any issues that might occur in the stream, assign a unique event ID to each input event. This enables you to troubleshoot any processing failures with the bot.

Amazon Lex also uses timestamps for each event. You can use these timestamps in addition to the event ID to help troubleshoot any network transmission issues.

During the course of the conversation between the user and the Amazon Lex bot, the bot can send the following outbound events in response to the user:

- [IntentResultEvent \(p. 548\)](#) – Contains the intent that Amazon Lex determined from the user utterance. Each internal result event includes:
 - **InputMode** – The type of user utterance. Valid values are `Speech`, `DTMF`, or `Text`.
 - **Interpretations** – Interpretations that Amazon Lex determines from the user utterance.
 - **RequestAttributes** – If you haven't modified the request attributes by using a lambda function, these are the same attributes that have been passed in at the start of the conversation.
 - **SessionId** – Session identifier used for the conversation.
 - **SessionState** – The state of the user's session with Amazon Lex.
- [TranscriptEvent \(p. 566\)](#) – If the user provides an input to your application, this event contains the transcript of the user's utterance to the bot. Your application does not receive a `TranscriptEvent` when there is no user input.

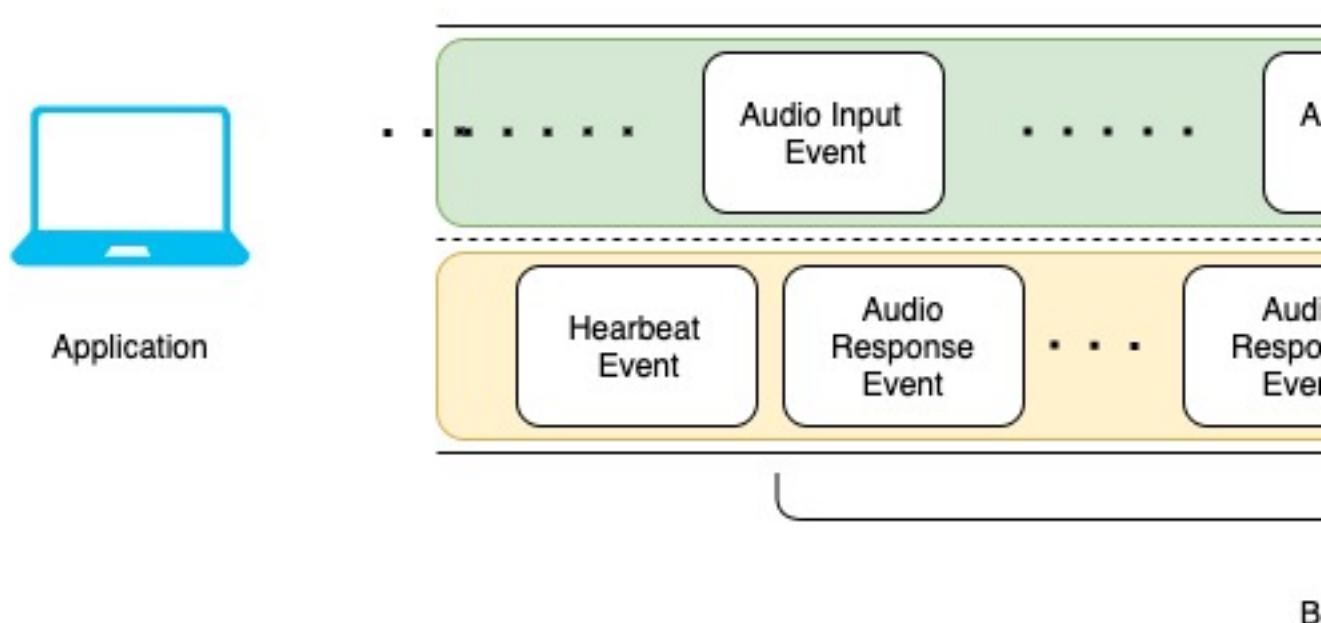
The value of the transcript event sent to your application depends on whether you've specified audio (speech and DMTF) or text as a conversation mode:

- Transcript of speech input – If the user is speaking with the bot, the transcript event is the transcription of the user's audio. It's a transcript of all the speech from the time the user begins speaking to the time they end speaking.
- Transcript of DTMF input – If the user is typing on a keypad, the transcript event contains all the digits the user pressed in their input.
- Transcript of text input – If the user is providing text input, the transcript event contains all of the text in the user's input.
- [TextResponseEvent \(p. 565\)](#) – Contains the bot response in text format. A text response is returned by default. If you've configured Amazon Lex to return an audio response, this text is used to generate an audio response. Each text response event contains an array of message objects that the bot returns to the user.
- [AudioResponseEvent \(p. 536\)](#) – Contains the audio response synthesized from the text generated in the `TextResponseEvent`. To receive audio response events, you must configure Amazon Lex to provide an audio response. All audio response events have the same audio format. Each event contains audio chunks that have a maximum size of 100 bytes. Amazon Lex sends an empty audio chunk with the `bytes` field set to `null` to indicate that the end of the audio response event to your application.
- [PlaybackInterruptionEvent \(p. 553\)](#) – When a user interrupts a response that the bot has sent to your application, Amazon Lex triggers this event to stop the playback of the response.
- [HeartbeatEvent \(p. 544\)](#) – Amazon Lex sends this event back periodically to keep the connection between your application and the bot from timing out.

Time sequence of events for an audio conversation

The following diagrams show a streaming audio conversation between a user and an Amazon Lex bot. The application continuously streams audio to the bot, and the bot looks for user input from the audio. In this example, both the user and the bot are using speech to communicate. Each diagram corresponds to a user utterance and the response of the bot to that utterance.

The following diagram shows the beginning of a conversation between the application and the bot. The stream begins at time zero (t0).

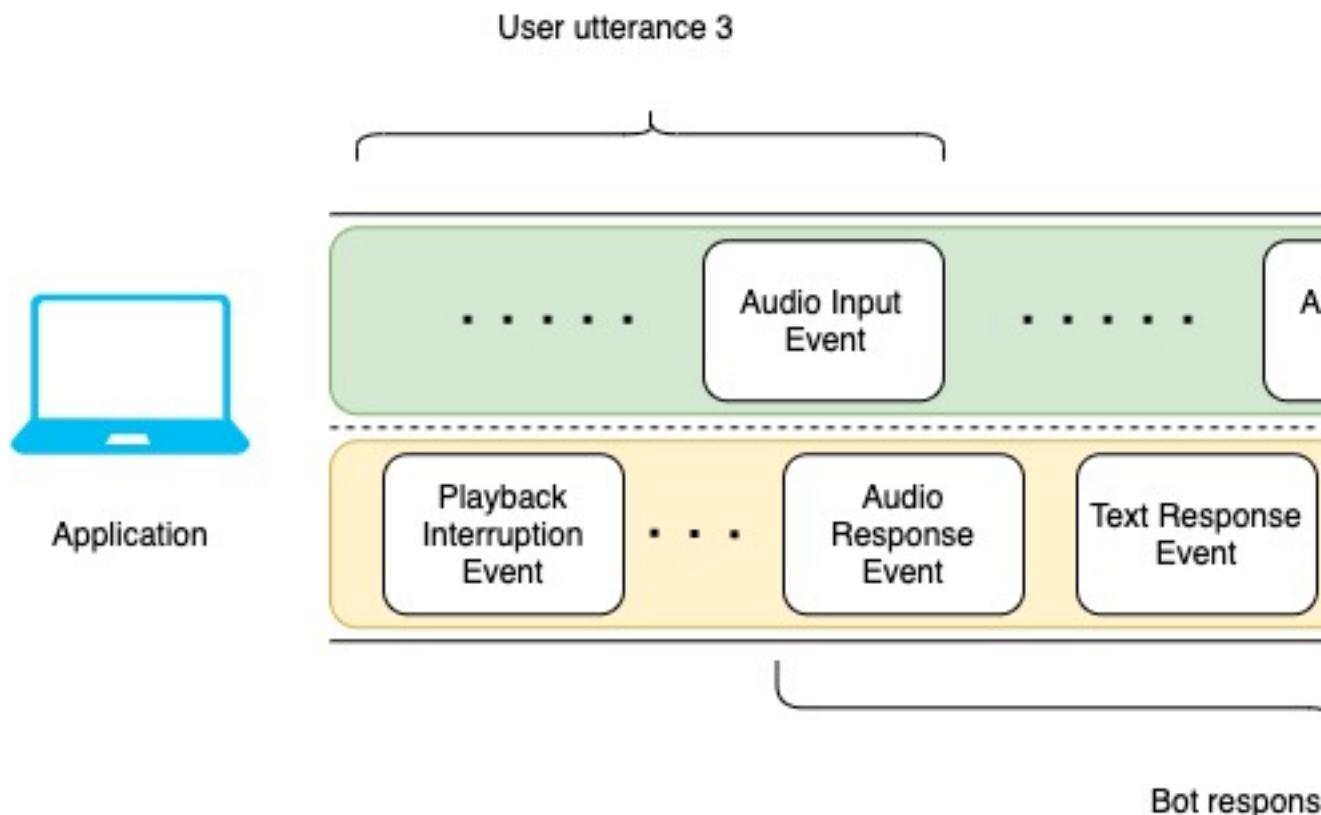


The following list describes the events of the preceding diagram.

- t0: The application sends a configuration event to the bot to start the stream.
- t1: The application streams audio data. This data is broken up into a series of input events from the application.
- t2: For *user utterance 1*, the bot detects an audio input event when the user begins speaking.
- t2: While the user is speaking, the bot sends a heartbeat event to maintain the connection. It sends these events intermittently to make sure the connection doesn't time out.
- t3: The bot detects the end of the user's utterance.
- t4: The bot sends back a transcript event that contains a transcript of the user's speech to the application. This is the beginning of *Bot response to user utterance 1*.
- t5: The bot sends an intent result event to indicate the action that the user wants to perform.

- t6: The bot begins providing its response as text in a text response event.
- t7: The bot sends a series of audio response events to the application to play for the user.
- t8: The bot sends another heartbeat event to intermittently maintain the connection.

The following diagram is a continuation of the previous diagram. It shows the application sending a playback completion event to the bot to indicate that it has stopped playing the audio response for the user. The application plays back *Bot response to user utterance 1* to the user. The user responds to *Bot response to user utterance 1* with *User utterance 2*.

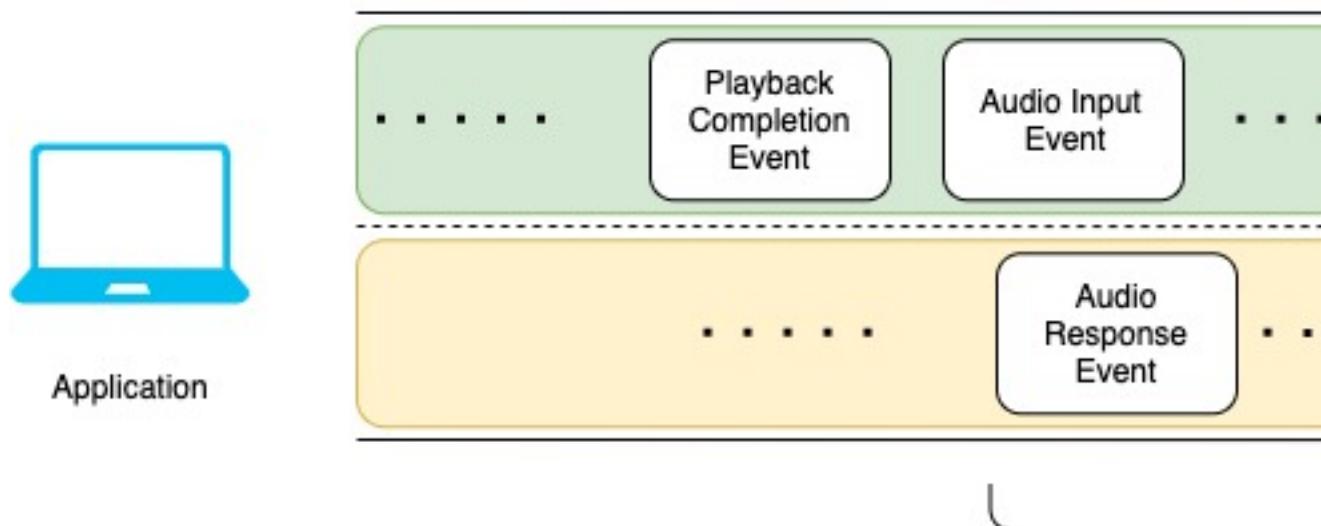


The following list describes the events of the preceding diagram:

- t10: The application sends a playback completion event to indicate that it has finished playing the bot's message to the user.
- t11: The application sends the user response back to the bot as *User utterance 2*.
- t12: For *Bot response to user utterance 2*, the bot waits for the user to stop speaking and then begins to provide an audio response.
- t13: While the bot sends *Bot response to user utterance 2* to the application, the bot detects the start of *User utterance 3*. The bot stops *Bot response to user utterance 2* and sends a playback interruption event.

- t14: The bot sends a playback interruption event to the application to signal that the user has interrupted the prompt.

The following diagram shows the *Bot response to user utterance 3*, and that the conversation continues after the bot responds to the user utterance.



Using the API to start a streaming conversation

When you start a stream to an Amazon Lex bot, you accomplish the following tasks:

1. Create an initial connection to the server.
2. Configure the security credentials and bot details. Bot details include whether the bot takes DTMF and audio input, or text input.
3. Send events to the server. These events are text data or audio data from the user.
4. Process events sent from the server. In this step, you determine whether the bot output is presented to the user as text or speech.

The following code examples initialize a streaming conversation with an Amazon Lex bot and your local machine. You can modify the code to meet your needs.

The following code is an example request using the AWS SDK for Java to start the connection to a bot and configure the bot details and credentials.

```

package com.lex.streaming.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2AsyncClient;
import software.amazon.awssdk.services.lexruntimev2.model.ConversationMode;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequest;

import java.net.URISyntaxException;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * The following code creates a connection with the Amazon Lex bot and configures the bot
 * details and credentials.
 * Prerequisite: To use this example, you must be familiar with the Reactive streams
 * programming model.
 * For more information, see
 * https://github.com/reactive-streams/reactive-streams-jvm.
 * This example uses AWS SDK for Java for Amazon Lex V2.
 * <p>
 * The following sample application enables you to interact with an Amazon Lex bot with the
 * streaming API. It uses the Audio
 * conversation mode to return audio responses to the user's input.
 * <p>
 * The code in this example accomplishes the following:
 * <p>
 * 1. Configure details about the conversation between the user and the Amazon Lex bot.
 * These details include the conversation mode and the specific bot the user is speaking
 * with.
 * 2. Create an events publisher that passes the audio events to the Amazon Lex bot after
 * you establish the connection. The code we provide in this example tells your computer to
 * pick up the audio from
 * your microphone and send that audio data to Amazon Lex.
 * 3. Create a response handler that handles the audio responses from the Amazon Lex bot
 * and plays back the audio to you.
 */
public class LexBidirectionalStreamingExample {

    public static void main(String[] args) throws URISyntaxException, InterruptedException
    {
        String botId = "";
        String botAliasId = "";
        String localeId = "";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.region_name; // Choose an AWS Region where the Amazon Lex
        Streaming API is available.

        AwsCredentialsProvider awsCredentialsProvider = StaticCredentialsProvider
            .create(AwsBasicCredentials.create(accessKey, secretKey));

        // Create a new SDK client. You need to use an asynchronous client.
        System.out.println("step 1: creating a new Lex SDK client");
        LexRuntimeV2AsyncClient lexRuntimeServiceClient = LexRuntimeV2AsyncClient.builder()
            .region(region)
            .credentialsProvider(awsCredentialsProvider)
            .build();
    }
}

```

```
// Configure the bot, alias and locale that you'll use to have a conversation.
System.out.println("step 2: configuring bot details");
StartConversationRequest.Builder startConversationRequestBuilder =
StartConversationRequest.builder()
    .botId(botId)
    .botAliasId(botAliasId)
    .localeId(localeId);

// Configure the conversation mode of the bot. By default, the
// conversation mode is audio.
System.out.println("step 3: choosing conversation mode");
startConversationRequestBuilder =
startConversationRequestBuilder.conversationMode(ConversationMode.AUDIO);

// Assign a unique identifier for the conversation.
System.out.println("step 4: choosing a unique conversation identifier");
startConversationRequestBuilder =
startConversationRequestBuilder.sessionId(sessionId);

// Start the initial request.
StartConversationRequest startConversationRequest =
startConversationRequestBuilder.build();

// Create a stream of audio data to the Amazon Lex bot. The stream will start after
the connection is established with the bot.
EventsPublisher eventsPublisher = new EventsPublisher();

// Create a class to handle responses from bot. After the server processes the user
data you've streamed, the server responds
// on another stream.
BotResponseHandler botResponseHandler = new BotResponseHandler(eventsPublisher);

// Start a connection and pass in the publisher that streams the audio and process
the responses from the bot.
System.out.println("step 5: starting the conversation ...");
CompletableFuture<Void> conversation = lexRuntimeServiceClient.startConversation(
    startConversationRequest,
    eventsPublisher,
    botResponseHandler);

// Wait until the conversation finishes. The conversation finishes if the dialog
state reaches the "Closed" state.
// The client stops the connection. If an exception occurs during the conversation,
the
// client sends a disconnection event.
conversation.whenComplete((result, exception) -> {
    if (exception != null) {
        eventsPublisher.disconnect();
    }
});

// The conversation finishes when the dialog state is closed and last prompt has
been played.
while (!botResponseHandler.isConversationComplete()) {
    Thread.sleep(100);
}

// Randomly sleep for 100 milliseconds to prevent JVM from exiting.
// You won't need this in your production code because your JVM is
// likely to always run.
// When the conversation finishes, the following code block stops publishing more
data and informs the Amazon Lex bot that there is no more data to send.
if (botResponseHandler.isConversationComplete()) {
    System.out.println("conversation is complete.");
}
```

```
        eventsPublisher.stop();
    }
}
```

The following code is an example request using the AWS SDK for Java to send events to the bot. The code in this example uses the microphone on your computer to send audio events.

```
package com.lex.streaming.sample;

import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import
software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

/**
 * You use the Events publisher to send events to the Amazon Lex bot. When you establish a
connection, the bot uses the
 * subscribe() method and enables the events publisher starts sending events to
 * your computer. The bot uses the "request" method of the subscription to make more
requests. For more information on the request method, see https://github.com/reactive-streams/reactive-streams-jvm.
 */
public class EventsPublisher implements Publisher<StartConversationRequestEventStream> {

    private AudioEventsSubscription audioEventsSubscription;

    @Override
    public void subscribe(Subscriber<? super StartConversationRequestEventStream>
subscriber) {
        if (audioEventsSubscription == null) {

            audioEventsSubscription = new AudioEventsSubscription(subscriber);
            subscriber.onSubscribe(audioEventsSubscription);

        } else {
            throw new IllegalStateException("received unexpected subscription request");
        }
    }

    public void disconnect() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.disconnect();
        }
    }

    public void stop() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.stop();
        }
    }

    public void playbackFinished() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.playbackFinished();
        }
    }
}
```

The following code is an example request using the AWS SDK for Java to handle responses from the bot. The code in this example configures Amazon Lex to play an audio response back to you.

```
package com.lex.streaming.sample;

import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.lexruntimev2.model.AudioResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DialogActionType;
import software.amazon.awssdk.services.lexruntimev2.model.IntentResultEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackInterruptionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponse;
import
software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseStream;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseHandler;
import software.amazon.awssdk.services.lexruntimev2.model.TextResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.TranscriptEvent;

import java.io.IOException;
import java.io.UncheckedIOException;
import java.util.concurrent.CompletableFuture;

/**
 * The following class is responsible for processing events sent from the Amazon Lex bot.
 * The bot sends multiple audio events,
 * so the following code concatenates those audio events and uses a publicly available Java
 * audio player to play out the message to
 * the user.
 */
public class BotResponseHandler implements StartConversationResponseHandler {

    private final EventsPublisher eventsPublisher;

    private boolean lastBotResponsePlayedBack;
    private boolean isDialogStateClosed;
    private AudioResponse audioResponse;

    public BotResponseHandler(EventsPublisher eventsPublisher) {
        this.eventsPublisher = eventsPublisher;
        this.lastBotResponsePlayedBack = false; // At the start, we have not played back
last response from bot.
        this.isDialogStateClosed = false; // At the start, the dialog state is open.
    }

    @Override
    public void responseReceived(StartConversationResponse startConversationResponse) {
        System.out.println("successfully established the connection with server. request
id:" + startConversationResponse.responseMetadata().requestId()); // would have 2XX,
request id.
    }

    @Override
    public void onEventStream(SdkPublisher<StartConversationResponseEventStream>
sdkPublisher) {

        sdkPublisher.subscribe(event -> {
            if (event instanceof PlaybackInterruptionEvent) {
                handle((PlaybackInterruptionEvent) event);
            } else if (event instanceof TranscriptEvent) {
                handle((TranscriptEvent) event);
            }
        });
    }
}
```

```

        } else if (event instanceof IntentResultEvent) {
            handle((IntentResultEvent) event);
        } else if (event instanceof TextResponseEvent) {
            handle((TextResponseEvent) event);
        } else if (event instanceof AudioResponseEvent) {
            handle((AudioResponseEvent) event);
        }
    });
}

@Override
public void exceptionOccurred(Throwable throwable) {
    System.err.println("got an exception:" + throwable);
}

@Override
public void complete() {
    System.out.println("on complete");
}

private void handle(PlaybackInterruptionEvent event) {
    System.out.println("Got a PlaybackInterruptionEvent: " + event);
}

private void handle(TranscriptEvent event) {
    System.out.println("Got a TranscriptEvent: " + event);
}

private void handle(IntentResultEvent event) {
    System.out.println("Got an IntentResultEvent: " + event);
    isDialogStateClosed =
        DialogActionType.CLOSE.equals(event.sessionState().dialogAction().type());
}

private void handle(TextResponseEvent event) {
    System.out.println("Got an TextResponseEvent: " + event);
    event.messages().forEach(message -> {
        System.out.println("Message content type:" + message.contentType());
        System.out.println("Message content:" + message.content());
    });
}

private void handle(AudioResponseEvent event) //Synthesize speech
// System.out.println("Got a AudioResponseEvent: " + event);
if (audioResponse == null) {
    audioResponse = new AudioResponse();
    //Start an audio player in a different thread.
    CompletableFuture.runAsync(() -> {
        try {
            AdvancedPlayer audioPlayer = new AdvancedPlayer(audioResponse);

            audioPlayer.setPlayBackListener(new PlaybackListener() {
                @Override
                public void playbackFinished(PlaybackEvent evt) {
                    super.playbackFinished(evt);

                    // Inform the Amazon Lex bot that the playback has finished.
                    eventsPublisher.playbackFinished();
                    if (isDialogStateClosed) {
                        lastBotResponsePlayedBack = true;
                    }
                }
            });
            audioPlayer.play();
        } catch (JavaLayerException e) {

```

```

        throw new RuntimeException("got an exception when using audio player",
e);
    }
}
}

if (event.audioChunk() != null) {
    audioResponse.write(event.audioChunk().asByteArray());
} else {
    // The audio audio prompt has ended when the audio response has no
    // audio bytes.
    try {
        audioResponse.close();
        audioResponse = null; // Prepare for the next audio prompt.
    } catch (IOException e) {
        throw new UncheckedIOException("got an exception when closing the audio
response", e);
    }
}

// The conversation with the Amazon Lex bot is complete when the bot marks the Dialog
as DialogActionType.CLOSE
// and any prompt playback is finished. For more information, see
// https://docs.aws.amazon.com/lexv2/latest/dg/API\_runtime\_DialogAction.html.
public boolean isConversationComplete() {
    return isDialogStateClosed && lastBotResponsePlayedBack;
}

}

```

To configure a bot to respond to input events with audio, you must first subscribe to audio events from Amazon Lex and then configure the bot to provide an audio response to the input events from the user.

The following code is a AWS SDK for Java example for subscribing to audio events from Amazon Lex.

```

package com.lex.streaming.sample;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lexruntimev2.model.AudioInputEvent;
import software.amazon.awssdk.services.lexruntimev2.model.ConfigurationEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DisconnectionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackCompletionEvent;
import
software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
import java.io.IOException;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.LinkedBlockingQueue;

```

```

import java.util.concurrent.atomic.AtomicLong;

public class AudioEventsSubscription implements Subscription {
    private static final AudioFormat MIC_FORMAT = new AudioFormat(8000, 16, 1, true,
        false);
    private static final String AUDIO_CONTENT_TYPE = "audio/lpcm; sample-rate=8000; sample-
    size-bits=16; channel-count=1; is-big-endian=false";
    //private static final String RESPONSE_TYPE = "audio/pcm; sample-rate=8000";
    private static final String RESPONSE_TYPE = "audio/mpeg";
    private static final int BYTES_IN_AUDIO_CHUNK = 320;
    private static final AtomicLong eventIdGenerator = new AtomicLong(0);

    private final AudioInputStream audioInputStream;
    private final Subscriber<? super StartConversationRequestEventStream> subscriber;
    private final EventWriter eventWriter;
    private CompletableFuture eventWriterFuture;

    public AudioEventsSubscription(Subscriber<? super StartConversationRequestEventStream>
        subscriber) {
        this.audioInputStream = getMicStream();
        this.subscriber = subscriber;
        this.eventWriter = new EventWriter(subscriber, audioInputStream);
        configureConversation();
    }

    private AudioInputStream getMicStream() {
        try {
            DataLine.Info dataLineInfo = new DataLine.Info(TargetDataLine.class,
                MIC_FORMAT);
            TargetDataLine targetDataLine = (TargetDataLine)
                AudioSystem.getLine(dataLineInfo);

            targetDataLine.open(MIC_FORMAT);
            targetDataLine.start();

            return new AudioInputStream(targetDataLine);
        } catch (LineUnavailableException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void request(long demand) {
        // If a thread to write events has not been started, start it.
        if (eventWriterFuture == null) {
            eventWriterFuture = CompletableFuture.runAsync(eventWriter);
        }
        eventWriter.addDemand(demand);
    }

    @Override
    public void cancel() {
        subscriber.onError(new RuntimeException("stream was cancelled"));
        try {
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
    }

    public void configureConversation() {
        String eventId = "ConfigurationEvent-" +
            String.valueOf(eventIdGenerator.incrementAndGet());
        ConfigurationEvent configurationEvent = StartConversationRequestEventStream

```

```

        .configurationEventBuilder()
        .eventId(eventId)
        .clientTimestampMillis(System.currentTimeMillis())
        .responseContentType(RESPONSE_TYPE)
        .build();

    System.out.println("writing config event");
    eventWriter.writeConfigurationEvent(configurationEvent);
}

public void disconnect() {

    String eventId = "DisconnectionEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

    DisconnectionEvent disconnectionEvent = StartConversationRequestEventStream
        .disconnectionEventBuilder()
        .eventId(eventId)
        .clientTimestampMillis(System.currentTimeMillis())
        .build();

    eventWriter.writeDisconnectEvent(disconnectionEvent);

    try {
        audioInputStream.close();
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }
}

//Notify the subscriber that we've finished.
public void stop() {
    subscriber.onComplete();
}

public void playbackFinished() {
    String eventId = "PlaybackCompletion-" +
String.valueOf(eventIdGenerator.incrementAndGet());

    PlaybackCompletionEvent playbackCompletionEvent =
StartConversationRequestEventStream
        .playbackCompletionEventBuilder()
        .eventId(eventId)
        .clientTimestampMillis(System.currentTimeMillis())
        .build();

    eventWriter.writePlaybackFinishedEvent(playbackCompletionEvent);
}

private static class EventWriter implements Runnable {
    private final BlockingQueue<StartConversationRequestEventStream> eventQueue;
    private final AudioInputStream audioInputStream;
    private final AtomicLong demand;
    private final Subscriber subscriber;

    private boolean conversationConfigured;

    public EventWriter(Subscriber subscriber, AudioInputStream audioInputStream) {
        this.eventQueue = new LinkedBlockingQueue<>();

        this.demand = new AtomicLong(0);
        this.subscriber = subscriber;
        this.audioInputStream = audioInputStream;
    }

    public void writeConfigurationEvent(ConfigurationEvent configurationEvent) {

```

```

        eventQueue.add(configurationEvent);
    }

    public void writeDisconnectEvent(DisconnectionEvent disconnectionEvent) {
        eventQueue.add(disconnectionEvent);
    }

    public void writePlaybackFinishedEvent(PlaybackCompletionEvent playbackCompletionEvent) {
        eventQueue.add(playbackCompletionEvent);
    }

    void addDemand(long l) {
        this.demand.addAndGet(l);
    }

    @Override
    public void run() {
        try {

            while (true) {
                long currentDemand = demand.get();

                if (currentDemand > 0) {
                    // Try to read from queue of events.
                    // If nothing is in queue at this point, read the audio events
                    directly from audio stream.
                    for (long i = 0; i < currentDemand; i++) {

                        if (eventQueue.peek() != null) {
                            subscriber.onNext(eventQueue.take());
                            demand.decrementAndGet();
                        } else {
                            writeAudioEvent();
                        }
                    }
                }
            }
        } catch (InterruptedException e) {
            throw new RuntimeException("interrupted when reading data to be sent to
server");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void writeAudioEvent() {
        byte[] bytes = new byte[BYTES_IN_AUDIO_CHUNK];

        int numBytesRead = 0;
        try {
            numBytesRead = audioInputStream.read(bytes);
            if (numBytesRead != -1) {
                byte[] byteArrayCopy = Arrays.copyOf(bytes, numBytesRead);

                String eventId = "AudioEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

                AudioInputEvent audioInputEvent = StartConversationRequestEventStream
                    .audioInputEventBuilder()

                .audioChunk(SdkBytes.fromByteBuffer(ByteBuffer.wrap(byteArrayCopy)))
                    .contentType(AUDIO_CONTENT_TYPE)
                    .clientTimestampMillis(System.currentTimeMillis())
                    .eventId(eventId).build();
            }
        }
    }
}

```

```
//System.out.println("sending audio event:" + audioInputEvent);
subscriber.onNext(audioInputEvent);
demand.decrementAndGet();
//System.out.println("sent audio event:" + audioInputEvent);
} else {
    subscriber.onComplete();
    System.out.println("audio stream has ended");
}

} catch (IOException e) {
    System.out.println("got an exception when reading from audio stream");
    System.err.println(e);
    subscriber.onError(e);
}
}
}
```

The following AWS SDK for Java example configures the Amazon Lex bot to provide an audio response to the input events.

```
package com.lex.streaming.sample;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.util.Optional;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

public class AudioResponse extends InputStream{

    // Used to convert byte, which is signed in Java, to positive integer (unsigned)
    private static final int UNSIGNED_BYTE_MASK = 0xFF;
    private static final long POLL_INTERVAL_MS = 10;

    private final LinkedBlockingQueue<Integer> byteQueue = new LinkedBlockingQueue<>();

    private volatile boolean closed;

    @Override
    public int read() throws IOException {
        try {
            Optional<Integer> maybeInt;
            while (true) {
                maybeInt = Optional.ofNullable(this.byteQueue.poll(POLL_INTERVAL_MS,
                TimeUnit.MILLISECONDS));
                // If we get an integer from the queue, return it.
                if (maybeInt.isPresent()) {
                    return maybeInt.get();
                }
                // If the stream is closed and there is nothing queued up, return -1.
                if (this.closed) {
                    return -1;
                }
            }
        } catch (InterruptedException e) {
            throw new IOException(e);
        }
    }
}
```

```

    }

    /**
     * Writes data into the stream to be offered on future read() calls.
     */
    public void write(byte[] byteArray) {
        // Don't write into the stream if it is already closed.
        if (this.closed) {
            throw new UncheckedIOException(new IOException("Stream already closed when
attempting to write into it."));
        }

        for (byte b : byteArray) {
            this.byteQueue.add(b & UNSIGNED_BYTE_MASK);
        }
    }

    @Override
    public void close() throws IOException {
        this.closed = true;
        super.close();
    }
}

```

Enabling your bot to be interrupted by your user

When you start a bidirectional audio stream between an Amazon Lex bot and your application, you can configure the bot to listen for user input while it is sending back a prompt. This enables the user to interrupt the prompt before the bot has finished playing it back. You can use this configuration for situations where the user might already know the answer to a question, such as when they're being prompted to provide a CVV code.

A bot knows when the user interrupts a prompt when it detects user input before your application can send back a `PlaybackCompletion` event. When the user interrupts a bot, the bot sends a `PlaybackInterruptionEvent`.

By default, the user can interrupt any prompt that the bot is streaming to your application. You can change this setting in the Amazon Lex console.

You can change how a user can respond to a prompt by editing a slot. A slot is part of an intent, and it is the means by which the user provides you the information you want. Each slot has a prompt for the user to provide you with that information. To learn more about slots, see [How it works \(p. 3\)](#).

To change whether the user can interrupt a prompt (console)

1. Sign in to AWS Management Console and open the Amazon Lex console at [Amazon Lex console](#).
2. Under **Bots**, select a bot.
3. Under **Language**, select the language of the bot.
4. Choose **View intents**.
5. Choose the intent.
6. For **Slots**, choose a slot.
7. Under **Advanced options**, choose **Wait and continue**.
8. Choose **More prompt options**.
9. Select or deselect **Users can interrupt the prompt when it is being read**.

You can test this functionality by creating a bot with two slots and specifying that users can't interrupt a prompt for one slot. If you interrupt the prompt that is interruptible, the bot sends a playback interruption event. If you interrupt the prompt that is not interruptible, the prompt continues to play.

Enabling the bot to wait for the user to provide additional information

When you start a bidirectional stream from an Amazon Lex bot to your application, you can configure the bot to wait for the user to provide additional information. There are circumstances when a user might not be ready to respond to a prompt. For example, a user might not be ready to provide their credit card information because their wallet is in another room.

By using the *Wait and continue* behavior of the Amazon Lex bot, users can say phrases such as "hold on a second" to make the bot wait for them to find the information and provide it. When you enable this behavior, the bot sends periodic reminders to the user to provide the information. It does not send back transcript events because there are no user utterances for it to transcribe.

The Amazon Lex bot automatically manages a streaming conversation. You don't have to write any additional code to enable this functionality. When a bot is prompted to wait by the user, the state of the Intent is Waiting and the type of the DialogAction is ElicitSlot. You can use this information to help customize your application for your needs. For example, you can configure your application to play music when the user is looking for their credit card.

You enable the wait and continue behavior for an individual slot. To learn more about slots, see [How it works \(p. 3\)](#).

To enable wait and continue

1. Sign in to AWS Management Console and open the Amazon Lex console at [Amazon Lex console](#).
2. Under **Bots**, select a bot.
3. Under **Language**, select the language of the bot.
4. Choose **View intents**.
5. Choose the intent.
6. Under **Slots**, choose a slot.
7. Under **Advanced options**, choose **Wait and continue**.
8. Under **Wait and continue** specify the following fields:
 - **Response when user wants the bot to wait** – This is how the bot responds when the user asks it to wait for the additional information.
 - **Response if the user needs the bot to continue waiting** – This is the response the bot sends to remind the user that it's still waiting for the information. You can change how frequently the bot reminds the user.
 - **Response when the user wants to continue** – this is the bot's response when the user has the requested information.

For every bot prompt, you can give multiple variations of the prompt, and one is presented to the user at random. You can also choose whether these prompts can be interrupted by the user.

To test the wait and continue functionality, configure your bot to wait for user input and start a stream to an Amazon Lex bot. For information on streaming to a bot, see [Using the API to start a streaming conversation \(p. 80\)](#).

Event stream encoding

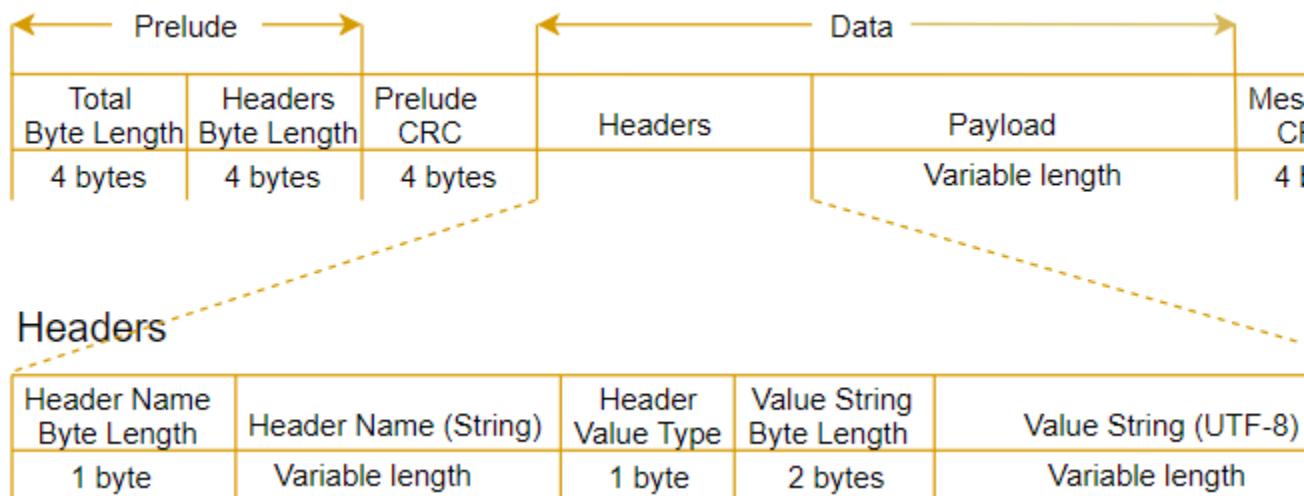
Event stream encoding provides bidirectional communication using messages between a client and a server. Data frames sent to the Amazon Lex streaming service are encoded in this format. The response from Amazon Lex also uses this encoding.

Each message consists of two sections: the prelude and the data. The prelude section contains the total byte length of the message and the combined byte length of all of the headers. The data section contains the headers and a payload.

Each section ends with a 4-byte big-endian integer CRC checksum. The message CRC checksum is for both the prelude section and the data section. Amazon Lex uses CRC32 (often referred to as GZIP CRC32) to calculate both CRCs. For more information about CRC32, see [GZIP file format specification version 4.3](#).

Total message overhead, including the prelude and both checksums, is 16 bytes.

The following diagram shows the components that make up a message and a header. There are multiple headers per message.



Each message contains the following components:

- **Prelude:** Always a fixed size of 8 bytes, two fields of 4 bytes each.
 - *First 4 bytes:* The total byte-length. This is the big-endian integer byte-length of the entire message, including the 4-byte length field itself.
 - *Second 4 bytes:* The headers byte-length. This is the big-endian integer byte-length of the headers portion of the message, excluding the headers length field itself.
- **Prelude CRC:** The 4-byte CRC checksum for the prelude portion of the message, excluding the CRC itself. The prelude has a separate CRC from the message CRC to ensure that Amazon Lex can detect corrupted byte-length information immediately without causing errors such as buffer overruns.
- **Headers:** Metadata annotating the message, such as the message type, content type, and so on. Messages have multiple headers. Headers are key-value pairs where the key is a UTF-8 string. Headers can appear in any order in the headers portion of the message and any given header can appear only once. For the required header types, see the following sections.
- **Payload:** The audio or text content being sent to Amazon Lex.
- **Message CRC:** The 4-byte CRC checksum from the start of the message to the start of the checksum. That is, everything in the message except the CRC itself.

Each header contains the following components. There are multiple headers per frame.

- **Header name byte-length:** The byte-length of the header name.
- **Header name:** The name of the header indicating the header type. For valid values, see the following frame descriptions.
- **Header value type:** An enumeration indicating the header value type.
- **Value string byte length:** The byte-length of the header value string.
- **Header value:** The value of the header string. Valid values for this field depend on the type of header. For valid values, see the following frame descriptions.

Monitoring Amazon Lex

Monitoring is important for maintaining the reliability, availability, and performance of your Amazon Lex chatbots. This topic describes how to use Amazon CloudWatch Logs and AWS CloudTrail to monitor Amazon Lex and describes the Amazon Lex runtime and channel association metrics.

Topics

- [Monitoring with conversation logs \(p. 95\)](#)
- [Obscuring slot values in logs \(p. 101\)](#)
- [Logging Amazon Lex API calls with AWS CloudTrail \(p. 101\)](#)
- [Monitoring Amazon Lex with Amazon CloudWatch \(p. 103\)](#)

Monitoring with conversation logs

You enable *conversation logs* to store bot interactions. You can use these logs to review the performance of your bot and to troubleshoot issues with conversations. You can log text for the [RecognizeText \(p. 413\)](#) operation. You can log both text and audio for the [RecognizeUtterance \(p. 419\)](#) operation. By enabling conversation logs you get a detailed view of conversations that users have with your bot.

For example, a session with your bot has a session ID. You can use this ID to get the transcript of the conversation including user utterances and the corresponding bot responses. You also get metadata such as intent name and slot values for an utterance.

Note

You can't use conversation logs with a bot subject to the Children's Online Privacy Protection Act (COPPA).

Conversation logs are configured for an alias. Each alias can have different settings for their text and audio logs. You can enable text logs, audio logs, or both for each alias. Text logs store text input, transcripts of audio input, and associated metadata in CloudWatch Logs. Audio logs store audio input in Amazon S3. You can enable encryption of text and audio logs using AWS KMS customer managed CMKs.

To configure logging, use the console or the [CreateBotAlias \(p. 173\)](#) or [UpdateBotAlias \(p. 355\)](#) operation. After enabling conversation logs for an alias, [RecognizeText \(p. 413\)](#) or [RecognizeUtterance \(p. 419\)](#) operation for that alias logs the text or audio utterances in the configured CloudWatch Logs log group or S3 bucket.

Topics

- [Configuring conversation logs \(p. 95\)](#)
- [Viewing text logs in Amazon CloudWatch Logs \(p. 97\)](#)
- [Accessing audio logs in Amazon S3 \(p. 100\)](#)
- [Monitoring conversation log status with CloudWatch metrics \(p. 100\)](#)

Configuring conversation logs

You enable and disable conversation logs using the console or the `conversationLogSettings` field of the `CreateBotAlias` or `UpdateBotAlias` operation. You can turn on or turn off audio logs, text logs, or both. Logging starts on new bot sessions. Changes to log settings aren't reflected for active sessions.

To store text logs, use an Amazon CloudWatch Logs log group in your AWS account. You can use any valid log group. The log group must be in the same region as the Amazon Lex bot. For more information about creating a CloudWatch Logs log group, see [Working with Log Groups and Log Streams](#) in the [Amazon CloudWatch Logs User Guide](#).

To store audio logs, use an Amazon S3 bucket in your AWS account. You can use any valid S3 bucket. The bucket must be in the same region as the Amazon Lex bot. For more information about creating an S3 bucket, see [Creating a bucket](#) in the [Amazon Simple Storage Service Getting Started Guide](#).

When you manage conversation logs using the console, the console updates your service role so that it has access to the log group and S3 bucket. If you are not using the console, you must provide an IAM role with policies that enable Amazon Lex to write to the configured log group or bucket.

The IAM role that you use to enable conversation logs must have the `iam:PassRole` permission. The following policy should be attached to the role.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam::account:role/role"  
        }  
    ]  
}
```

Enabling conversation logs

To turn on logs using the console

1. Open the Amazon Lex console <https://console.aws.amazon.com/lexv2>.
2. From the list, choose a bot.
3. From the left menu, choose **Aliases**.
4. In the list of aliases, choose the alias for which you want to configure conversation logs.
5. In the **Conversation logs** section, choose **Manage conversation logs**.
6. For text logs, choose **Enable** then enter the Amazon CloudWatch Logs log group name.
7. For audio logs, choose **Enable** then enter the S3 bucket information.
8. Optional. To encrypt audio logs, choose the AWS KMS key to use for encryption.
9. Choose **Save** to start logging conversations. If necessary, Amazon Lex will update your service role with permissions to access the CloudWatch Logs log group and selected S3 bucket.

Disabling conversation logs

To turn off logs using the console

1. Open the Amazon Lex console <https://console.aws.amazon.com/lexv2>.
2. From the list, choose a bot.
3. From the left menu, choose **Aliases**.
4. In the list of aliases, choose the alias for which you want to configure conversation logs.
5. In the **Conversation logs** section, choose **Manage conversation logs**.
6. Disable text logging, audio logging, or both to turn off logging.
7. Choose **Save** to stop logging conversations.

Viewing text logs in Amazon CloudWatch Logs

Amazon Lex stores text logs for your conversations in Amazon CloudWatch Logs. To view the logs, you can use the CloudWatch Logs console or API. For more information, see [Search Log Data Using Filter Patterns](#) and [CloudWatch Logs Insights Query Syntax](#) in the *Amazon CloudWatch Logs User Guide*.

To view logs using the Amazon Lex console

1. Open the Amazon Lex console <https://console.aws.amazon.com/lexv2>.
2. From the list, choose a bot.
3. From the left menu, choose **Analytics** and then choose **CloudWatch metrics**.
4. View metrics for your bot on the **CloudWatch metrics** page.

You can also use the CloudWatch console or API to view your log entries. To find the log entries, navigate to the log group that you configured for the alias. You find the log stream prefix for your logs in the Amazon Lex console or by using the [DescribeBotAlias \(p. 262\)](#) operation.

Log entries for a user utterance is in multiple log streams. An utterance in the conversation has an entry in one of the log streams with the specified prefix. An entry in the log stream contains the following information.

```
{  
    "message-version": "2.0",  
    "bot": {  
        "id": "string",  
        "name": "string",  
        "aliasId": "string",  
        "aliasName": "string",  
        "localeId": "string",  
        "version": "string"  
    },  
    "messages": [  
        {  
            "contentType": "PlainText | SSML | CustomPayload | ImageResponseCard",  
            "content": "string",  
            "imageResponseCard": {  
                "title": "string",  
                "subtitle": "string",  
                "imageUrl": "string",  
                "buttonsList": [  
                    {  
                        "text": "string",  
                        "value": "string"  
                    }  
                ]  
            }  
        }  
    ],  
    "sessionState": {  
        "dialogAction": {  
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",  
            "slotToElicit": "string"  
        },  
        "intent": {  
            "name": "string",  
            "slots": {  
                "string" : {  
                    "value": {  
                        "interpretedValue": "string",  
                        "originalValue": "string"  
                    }  
                }  
            }  
        }  
    }  
}
```

```
        "originalValue": "string",
        "resolvedValues": [ "string" ]
    }
},
"string": {
    "shape": "List",
    "value": {
        "originalValue": "string",
        "interpretedValue": "string",
        "resolvedValues": [ "string" ]
    },
    "values": [
        {
            "shape": "Scalar",
            "value": {
                "originalValue": "string",
                "interpretedValue": "string",
                "resolvedValues": [ "string" ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": "string",
                "interpretedValue": "string",
                "resolvedValues": [ "string" ]
            }
        }
    ]
},
"kendraResponse": {
    // Only present when intent is KendraSearchIntent. For details, see
    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
},
"state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
"confirmationState": "Confirmed | Denied | None"
},
"originatingRequestId": "string",
"sessionAttributes": {
    "string": "string"
}
},
"interpretations": [
{
    "nluConfidence": "string",
    "intent": {
        "name": "string",
        "slots": {
            "string": {
                "value": {
                    "originalValue": "string",
                    "interpretedValue": "string",
                    "resolvedValues": [ "string" ]
                }
            },
            "string": {
                "shape": "List",
                "value": {
                    "interpretedValue": "string",
                    "originalValue": "string",
                    "resolvedValues": [ "string" ]
                }
            },
            "values": [
                {

```

```
        "shape": "Scalar",
        "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [ "string" ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [ "string" ]
        }
    }
]
},
"kendraResponse": {
    // Only present when intent is KendraSearchIntent. For details, see
    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
},
    "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
    "confirmationState": "Confirmed | Denied | None"
},
    "sentimentResponse": {
        "sentiment": "string",
        "sentimentScore": {
            "positive": "string",
            "negative": "string",
            "neutral": "string",
            "mixed": "string"
        }
    }
},
],
"sessionId": "string",
"inputTranscript": "string",
"missedUtterance": "bool",
"requestId": "string",
"timestamp": "string",
"developerOverride": "bool",
"inputMode": "DTMF | Speech | Text",
"requestAttributes": {
    "string": "string"
},
"audioProperties": {
    "contentType": "string",
    "s3Path": "string",
    "duration": {
        "total": "integer",
        "voice": "integer",
        "silence": "integer"
    }
},
"bargeIn": "string",
"responseReason": "string",
"operationName": "string"
}
```

The contents of the log entry depends on the result of a transaction and the configuration of the bot and request.

- The `intent`, `slots`, and `slotToElicit` fields don't appear in an entry if the `missedUtterance` field is `true`.
- The `s3PathForAudio` field doesn't appear if audio logs are disabled or if the `inputDialogMode` field is `Text`.
- The `responseCard` field only appears when you have defined a response card for the bot.
- The `requestAttributes` map only appears if you have specified request attributes in the request.
- The `kendraResponse` field is only present when the `AMAZON.KendraSearchIntent` makes a request to search an Amazon Kendra index.
- The `developerOverride` field is `true` when an alternative intent was specified in the bot's Lambda function.
- The `sessionAttributes` map only appears if you have specified session attributes in the request.
- The `sentimentResponse` map only appears if you configure the bot to return sentiment values.

Note

The input format may change without a corresponding change in the `messageVersion`. Your code should not throw an error if new fields are present.

Accessing audio logs in Amazon S3

Amazon Lex stores audio logs for your conversations in an S3 bucket.

You can use the Amazon S3 console or API to access audio logs. You can see the S3 object key prefix of the audio files in the Amazon Lex console, or in the `conversationLogSettings` field in the `DescribeBotAlias` operation response.

Monitoring conversation log status with CloudWatch metrics

Use Amazon CloudWatch to monitor delivery metrics of your conversation logs. You can set alarms on metrics so that you are aware of issues with logging if they should occur.

Amazon Lex provides four metrics in the `AWS/Lex` namespace for conversation logs:

- `ConversationLogsAudioDeliverySuccess`
- `ConversationLogsAudioDeliveryFailure`
- `ConversationLogsTextDeliverySuccess`
- `ConversationLogsTextDeliveryFailure`

The success metrics show that Amazon Lex has successfully written your audio or text logs to their destinations.

The failure metrics show that Amazon Lex couldn't deliver audio or text logs to the specified destination. Typically, this is a configuration error. When your failure metrics are above zero, check the following:

- Make sure that Amazon Lex is a trusted entity for the IAM role.
- For text logging, make sure that the CloudWatch Logs log group exists. For audio logging, make sure that the S3 bucket exists.
- Make sure that the IAM role that Amazon Lex uses to access the CloudWatch Logs log group or S3 bucket has write permission for the log group or bucket.
- Make sure that the S3 bucket exists in the same region as the Amazon Lex bot and belongs to your account.

Obscuring slot values in logs

Amazon Lex enables you to obfuscate, or hide, the contents of slots so that the content is not visible. To protect sensitive data captured as slot values, you can enable slot obfuscation to mask those values for logging.

When you choose to obfuscate slot values, Amazon Lex replaces the value of the slot with the name of the slot in conversation logs. For a slot called `full_name`, the value of the slot would be obfuscated as follows:

```
Before:  
  My name is John Stiles  
After:  
  My name is {full_name}
```

If an utterance contains bracket characters ({}), Amazon Lex escapes the bracket characters with two back slashes (\\\). For example, the text `{John Stiles}` is obfuscated as follows:

```
Before:  
  My name is {John Stiles}  
After:  
  My name is \\\{{full_name}\\\\}
```

Slot values are obfuscated in conversation logs. The slot values are still available in the response from the `RecognizeText` and `RecognizeUtterance` operations, and the slot values are available to your validation and fulfillment Lambda functions. If you are using slot values in your prompts or responses, those slot values are not obfuscated in conversation logs.

In the first turn of a conversation, Amazon Lex obfuscates slot values if it recognizes a slot and slot value in the utterance. If no slot value is recognized, Amazon Lex does not obfuscate the utterance.

On the second and later turns, Amazon Lex knows the slot to elicit and if the slot value should be obfuscated. If Amazon Lex recognizes the slot value, the value is obfuscated. If Amazon Lex does not recognize a value, the entire utterance is obfuscated. Any slot values in missed utterances won't be obfuscated.

Amazon Lex also doesn't obfuscate slot values that you store in request or session attributes. If you are storing slot values that should be obfuscated as an attribute, you must encrypt or otherwise obfuscate the value.

Amazon Lex doesn't obfuscate the slot value in audio. It does obfuscate the slot value in the audio transcription.

All slots are obfuscated by default. However, you don't need to obfuscate all of the slots in a bot. You can choose which slots obfuscate using the console or by using the Amazon Lex API. In the console, choose **Slot obfuscation** in the settings for a slot. If you are using the API, set the `obfuscationSetting` field of the slot to `DEFAULT_OBFUSCATION` when you call the [CreateSlot](#) (p. 212) or [UpdateSlot](#) (p. 383) operation.

Logging Amazon Lex API calls with AWS CloudTrail

Amazon Lex is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Lex. CloudTrail captures API calls for Amazon Lex as events. The calls captured include calls from the Amazon Lex console and code calls to the Amazon Lex API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3

bucket, including events for Amazon Lex. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Lex, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

Amazon Lex information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon Lex, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing events with CloudTrail Event history](#).

For an ongoing record of events in your AWS account, including events for Amazon Lex, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

Amazon Lex supports logging for all of the actions listed in [Amazon Lex Model Building V2 \(p. 162\)](#).

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity element](#).

Understanding Amazon Lex log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the [CreateBotAlias \(p. 173\)](#) action.

```
{  
    "eventVersion": "1.05",  
    "userIdentity": {  
        "type": "AssumedRole",  
        "principalId": "ID of caller:temporary credentials",  
        "arn": "ARN of assumed role",  
        "accountId": "Account ID of assumed role",  
        "accessKeyId": "Access key ID of assumed role"  
    },  
    "versionId": "Version ID of log entry",  
    "awsRegion": "Region where log entry was created",  
    "sourceIPAddress": "IP address of requester",  
    "userAgent": "User agent of requester",  
    "version": "1",  
    "eventTime": "2018-09-18T18:38:12Z",  
    "eventType": "AwsApiCall",  
    "recipientAccountId": "Account ID of target account",  
    "requestParameters": {  
        "Action": "CreateBotAlias",  
        "BotName": "MyBot",  
        "BotAliasName": "MyBotAlias",  
        "Description": "A new bot alias.",  
        "FunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-lambda-function",  
        "Locale": "en-US",  
        "Name": "MyBotAlias",  
        "Tags": [{}],  
        "Type": "SINGLE_VALUE_TEXT"  
    },  
    "responseElements": {  
        "BotAlias": {  
            "BotName": "MyBot",  
            "BotAliasName": "MyBotAlias",  
            "BotArn": "arn:aws:lex:us-east-1:123456789012:bot/MyBot/MyBotAlias",  
            "LastUpdated": "2018-09-18T18:38:12Z",  
            "Status": "ACTIVE",  
            "Version": "1",  
            "Created": "2018-09-18T18:38:12Z",  
            "LastModified": "2018-09-18T18:38:12Z",  
            "Locale": "en-US",  
            "Name": "MyBotAlias",  
            "Type": "SINGLE_VALUE_TEXT"  
        }  
    },  
    "awsService": "Lex",  
    "requestId": "Request ID of log entry",  
    "macAddress": "MAC address of requester",  
    "eventTypeDetails": {}  
}
```

```

"arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
"accountId": "111122223333",
"accessKeyId": "AKIAIOSFODNN7EXAMPLE",
"sessionContext": {
    "sessionIssuer": {
        "type": "Role",
        "principalId": "ID of caller",
        "arn": "arn:aws:iam::111122223333:role/role name",
        "accountId": "111122223333",
        "userName": "role name"
    },
    "webIdFederationData": {},
    "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "creation date"
    }
},
"eventTime": "event timestamp",
"eventSource": "lex.amazonaws.com",
"eventName": "CreateBotAlias",
"awsRegion": "Region",
"sourceIPAddress": "192.0.2.0",
"userAgent": "user agent",
"requestParameters": {
    "botAliasLocaleSettingsMap": {
        "en_US": {
            "enabled": true
        }
    },
    "botId": "bot ID",
    "botAliasName": "bot alias name",
    "botVersion": "1"
},
"responseElements": {
    "botAliasLocaleSettingsMap": {
        "en_US": {
            "enabled": true
        }
    },
    "botAliasId": "bot alias ID",
    "botAliasName": "bot alias name",
    "botId": "bot ID",
    "botVersion": "1",
    "creationDateTime": creation timestamp
},
"requestID": "unique request ID",
"eventId": "unique event ID",
"readOnly": false,
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

Monitoring Amazon Lex with Amazon CloudWatch

You can monitor Amazon Lex using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the [Amazon CloudWatch User Guide](#).

The Amazon Lex service reports the following metrics in the AWS/Lex namespace.

Metric	Description
KendraIndexAccessErrors	<p>The number of times that Amazon Lex could not access your Amazon Kendra index.</p> <ul style="list-style-type: none"> • Operation, BotId, BotAliasId, LocaleId <p>Unit: Count</p>
KendraLatency	<p>The amount of time that it takes Amazon Kendra to respond to a request from the <code>AMAZON.KendraSearchIntent</code>.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotVersion, LocaleId • Operation, BotId, BotAliasId, LocaleId <p>Unit: Milliseconds</p>
KendraSuccess	<p>The number of times that Amazon Lex couldn't access your Amazon Kendra index.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotVersion, LocaleId • Operation, BotId, BotAliasId, LocaleId <p>Unit: Count</p>
KendraSystemErrors	<p>The number of times that Amazon Lex couldn't query the Amazon Kendra index.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotAliasId, InputMode, LocaleId <p>Unit: Count</p>
KendraThrottledEvents	<p>The number of times Amazon Kendra throttled requests from the <code>AMAZON.KendraSearchIntent</code>.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotAliasId, InputMode, LocaleId <p>Unit: Count</p>
MissedUtteranceCount	<p>The number of utterances that were not recognized in the specified time period.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotVersion, InputMode, LocaleId • Operation, BotId, BotAliasId, InputMode, LocaleId

Metric	Description
	Unit: Count
RuntimeInvalidLambdaResponses	<p>The number of invalid AWS Lambda responses in the specified period.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotAliasId, InputMode, LocaleId
	Unit: Count
RuntimeLambdaErrors	<p>The number of Lambda runtime errors in the specified time period.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotAliasId, InputMode, LocaleId
	Unit: Count
RuntimePollyErrors	<p>The number of invalid Amazon Polly responses in the specified time period.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotAliasId, InputMode, LocaleId
	Unit: Count
RuntimeRequestCount	<p>The number of runtime requests in the specified time period.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • Operation, BotId, BotVersion, InputMode, LocaleId • Operation, BotId, BotAliasId, InputMode, LocaleId
	Unit: Count
RuntimeRequestLength	<p>Total length of a conversation with a Amazon Lex bot. Only applicable to the StartConversation (p. 426) operation.</p> <p>Valid dimensions:</p> <ul style="list-style-type: none"> • BotAliasId, BotId, LocaleId, Operation • BotId, BotAliasId, LocaleId, Operation
	Unit: milliseconds

Metric	Description
RuntimeSuccessfulRequests	The latency for successful requests between the time the request was made and the response was passed back. Valid dimensions: <ul style="list-style-type: none">• Operation, BotId, BotVersion, InputMode, LocaleId• Operation, BotId, BotAliasId, InputMode, LocaleId Unit: milliseconds
RuntimeSystemErrors	The number of system errors in the specified period. The response code range for a system error is 500 to 599. Valid dimensions: <ul style="list-style-type: none">• Operation, BotId, BotAliasId, InputMode, LocaleId Unit: Count
RuntimeThrottledEvents	The number of throttled events. Amazon Lex throttles an event when it receives more requests than the limit of transactions per second set for your account. If the limit set for your account is frequently exceeded, you can request a limit increase. To request an increase, see AWS service limits . Valid dimensions: <ul style="list-style-type: none">• Operation, BotId, BotAliasId, InputMode, LocaleId Unit: Count
RuntimeUserErrors	The number of user errors in the specified period. The response code range for a user error is 400 to 499. Valid dimensions: <ul style="list-style-type: none">• Operation, BotId, BotAliasId, InputMode, LocaleId Unit: Count

The following dimensions are supported for the Amazon Lex metrics.

Dimension	Description
Operation	The name of the Amazon Lex operation – RecognizeText, RecognizeUtterance, StartConversation, GetSession, PutSession, DeleteSession – that generated the entry.
BotId	The alphanumeric unique identifier for the bot.
BotAliasId	The alphanumeric unique identifier for the bot alias.
BotVersion	The numeric version of the bot.

Dimension	Description
InputMode	The type of input to the bot – speech, text, or DTMF.
LocaleId	The identifier of the bot's locale, such as en-US or fr-CA.

Importing and exporting bots

You can export a bot definition or a bot locale and then import it back to create a new bot or to overwrite an existing bot in an AWS account. For example, you can export a bot from a test account and then create a copy of the bot in your production account. You can also copy a bot from one AWS Region to another Region.

You can change the resources of the exported bot or bot locale before importing it. For example, you can export a bot and then edit the JSON file for a slot to add or remove slot value elicitation utterances from a specific slot. After you finish editing the definition, you can import the modified file.

Topics

- [Exporting a bot or bot locale \(p. 108\)](#)
- [Importing a bot or bot locale \(p. 110\)](#)
- [Using a password when importing or exporting \(p. 112\)](#)
- [JSON format for importing and exporting \(p. 113\)](#)

Exporting a bot or bot locale

You export a bot or bot locale using the console or the `StartExport` operation. You specify the resource to export, and you can provide an optional password to help protect the .zip file when you start an export. After you download the .zip file, you must use the password to access the file before you can use it. For more information, see [Using a password when importing or exporting \(p. 112\)](#).

Exporting is an asynchronous operation. Once you have started the export, you can use the console or the `DescribeExport` operation to monitor the progress of the export. Once the export is complete, the console or the `DescribeExport` operation shows a status of `COMPLETED`, and the console downloads the export .zip file to your browser. If you use the `DescribeExport` operation, Amazon Lex provides a pre-signed Amazon S3 URL where you can download the results of the export. The download URL is available for only five minutes, but you can get a new URL by calling the `DescribeExport` operation again.

You can see the history of exports for a bot or bot locale with the console or with the `ListExports` operation. The results show the exports along with their current status. An export is available in the history for seven days.

When you export the `Draft` version of a bot or a bot locale, it is possible for the definition in the JSON file to be in an inconsistent state because the `Draft` version of a bot or bot locale can be changed while an export is in progress. If the `Draft` version is changed while it is being exported, the changes may not be included in the export file.

When you export a bot locale, Amazon Lex exports all of the information that defines the locale, including the locale, intents, slot types, and slots.

When you export a bot, Amazon Lex exports all of the locales defined for the bot, including the intents, slot types, and slots. The following items are not exported with a bot:

- Bot aliases
- Role ARN associated with a bot
- Tags associated with bots and bot aliases

- Lambda code hooks associated with a bot alias

The role ARN and tags are entered as request parameters when you import a bot. You need to create bot aliases and assign Lambda code hooks after importing, if necessary.

You can remove an export and the associated .zip file using the console or the `DeleteExport` operation.

For an example of exporting a bot using the console, see [Exporting a bot \(console\) \(p. 109\)](#).

IAM permissions required to export

To export bots and bot locales, the user running the export must have the following IAM permissions.

API	• Required IAM actions	Resource
CreateExport (p. 188)	• <code>CreateExport</code>	Bot
UpdateExport (p. 365)	• <code>UpdateExport</code>	Bot
DescribeExport (p. 274)	<ul style="list-style-type: none"> • <code>DescribeExport</code> • <code>DescribeBot</code> • <code>DescribeLocale</code> • <code>DescribeIntent</code> • <code>DescribeSlot</code> • <code>DescribeSlotType</code> • <code>ListLocale</code> • <code>ListIntent</code> • <code>ListSlot</code> • <code>ListSlotType</code> 	Bot
DeleteExport (p. 243)	• <code>DeleteExport</code>	Bot
ListExports (p. 320)	• <code>ListExports</code>	*

For an example IAM policy, see [Allow a user to export bots and bot locales \(p. 136\)](#).

Exporting a bot (console)

You can export a bot from the bot list, from the list of versions, or from the version details page. When you choose a version, Amazon Lex exports that version. The following instructions assume that you start exporting the bot from the list of bots, but when you start with a version the steps are the same.

To export a bot using the console

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. From the list of bots, choose the bot to export.
3. From **Action**, choose **Export**.
4. Choose the bot version, platform, and export format.
5. (Optional) Enter a password for the .zip file. Providing a password helps protect the output archive.
6. Choose **Export**.

After you start the export, you return to the list of bots. To monitor the progress of the export, use the **Import/export history** list. When the status of the export is **Complete**, the console automatically downloads the .zip file to your computer.

To download the export again, on the import/export list, choose the export, and then choose **Download**. You can provide a password for the downloaded .zip file.

To export a bot language

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. From the list of bots, choose the bot whose language you want to export.
3. From **Add languages**, choose **View languages**.
4. From the **All languages** list, choose the language to export.
5. From **Action**, choose **Export**.
6. Choose the bot version, platform, and format.
7. (Optional) Enter a password for the .zip file. Providing a password helps protect the output archive.
8. Choose **Export**.

After you start the export, you return to the list of languages. To monitor the progress of the export, use the **Import/export history** list. When the status of the export is **Complete**, the console automatically downloads the .zip file to your computer.

To download the export again, on the import/export list, choose the export, and then choose **Download**. You can provide a password for the downloaded .zip file.

Importing a bot or bot locale

To use the console to import a previously exported bot or bot locale, you provide the file location on your local computer and the optional password to unlock the file. For an example, see [Importing a bot \(console\) \(p. 111\)](#).

When you use the API, importing a bot or bot locale is a three step process:

1. Create an upload URL using the `CreateUploadUrl` operation. You don't need to create an upload URL when you are using the console.
2. Upload the .zip file that contains the bot or bot locale definition.
3. Start the import with the `StartImport` operation.

The upload URL is a pre-signed Amazon S3 URL with write permission. The URL is available for five minutes after it is generated. If you password protect the .zip file, you must provide the password when you start the import. For more information, see [Using a password when importing or exporting \(p. 112\)](#).

An import is an asynchronous process. You can monitor the progress of an import using the console or the `DescribeImport` operation.

When you import a bot or bot locale, there may be conflicts between resource names in the import file and existing resource names in Amazon Lex. Amazon Lex can handle the conflict in two ways:

- **Fail on conflict** – The import stops and no resources are imported from the import .zip file.
- **Overwrite** – Amazon Lex imports all of the resources from the import .zip file, replacing any existing resource with the definition from the import file.

You can see a list of the imports to a bot or bot locale using the console or the `ListImports` operation. Imports remain in the list for seven days. You can use the console or the `DescribeImport` operation to see details about a specific import.

You can also remove an import and the associated .zip file using the console or the `DeleteImport` operation.

For an example of importing a bot using the console, see [Importing a bot \(console\) \(p. 111\)](#).

IAM permissions required to import

To import bots and bot locales, the user running the import must have the following IAM permissions.

API	Required IAM actions	Resource
CreateUploadUrl (p. 229)	<ul style="list-style-type: none"> • <code>CreateUploadUrl</code> 	*
StartImport (p. 343)	<ul style="list-style-type: none"> • <code>StartImport</code> • <code>iam:PassRole</code> • <code>CreateBot</code> • <code>CreateLocale</code> • <code>CreateIntent</code> • <code>CreateSlot</code> • <code>CreateSlotType</code> • <code>UpdateBot</code> • <code>UpdateLocale</code> • <code>UpdateIntent</code> • <code>UpdateSlot</code> • <code>UpdateSlotType</code> • <code>DeleteBot</code> • <code>DeleteLocale</code> • <code>DeleteIntent</code> • <code>DeleteSlot</code> • <code>DeleteSlotType</code> 	1. To import a new bot: bot, bot alias. 2. To overwrite an existing bot: bot. 3. To import a new locale: bot.
DescribeImport (p. 277)	<ul style="list-style-type: none"> • <code>DescribeImport</code> 	Bot
DeleteImport (p. 245)	<ul style="list-style-type: none"> • <code>DeleteImport</code> 	Bot
ListImports (p. 324)	<ul style="list-style-type: none"> • <code>ListImports</code> 	*

For an example IAM policy, see [Allow a user to import bots and bot locales \(p. 137\)](#).

Importing a bot (console)

To import a bot using the console

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. From **Action**, choose **Import**.
3. In **Input file**, give the bot a name and then choose the .zip file that contains the JSON files that define the bot.

4. If the .zip file is password protected, enter the password for the .zip file. Password protecting the archive is optional, but it helps protect the contents.
5. Create or enter the IAM role that defines permissions for your bot.
6. Indicate whether your bot is subject to the Children's Online Privacy Protection Act (COPPA).
7. Provide an idle timeout setting for your bot. If you don't provide a value, the value from the zip file is used. If the .zip file does not contain a timeout setting, Amazon Lex uses the default of 300 seconds (five minutes).
8. (Optional) Add tags for your bot.
9. Choose whether to warn about overwriting existing bots with the same name. If you enable warnings, if the bot you are importing would overwrite an existing bot, you receive a warning and the bot is not imported. If you disable warnings, the imported bot replaces the existing bot with the same name.
10. Choose **Import**.

After you start the import, you return to the list of bots. To monitor the progress of the import, use the **Import/export history** list. When the status of the import is **Complete**, you can choose the bot from the list of bots to modify or build the bot.

To import a bot language

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. From the list of bots, choose the bot to which you want to import a language.
3. From **Add languages**, choose **View languages**.
4. From **Action**, choose **import**.
5. In **Input file**, choose the file that contains the language to import. If you protected the .zip file, provide the password in **Password**.
6. In **Language**, choose the language to import as. The language doesn't have to match the language in the import file. You can copy the intents from one language to another.
7. In **Voice**, choose the Amazon Polly voice to use for voice interaction, or choose **None** for a text-only bot.
8. In **Confidence score threshold**, enter the threshold where Amazon Lex inserts the AMAZON.FallbackIntent, the AMAZON.KendraSearchIntent, or both when returning alternative intents.
9. Choose whether to warn about overwriting an existing language. If you enable warnings, if the language you are importing would overwrite an existing language, you receive a warning and the language is not imported. If you disable warnings, the imported language replaces the existing language.
10. Choose **Import** to start importing the language.

After you start the import, you return to the list of languages. To monitor the progress of the import, use the **Import/export history** list. When the status of the import is **Complete**, you can choose the language from the list of bots to modify or build the bot.

Using a password when importing or exporting

Amazon Lex can password protect your export archives or read your protected import archvies using standard .zip file compression. You should always password protect your import and export archives.

Amazon Lex sends your export archive to an S3 bucket, and it is available to you with a pre-signed S3 URL. The URL is only available for five minutes. The archive is available to anyone with access to

the download URL. To help protect the data in the archive, provide a password when you export the bot or bot locale. If you need to get the archive after the URL expires, you can use the console or the `DescribeExport` operation to get a new URL.

If you lose the password for an export archive, you can create a new password for an existing file by choosing **Download** from the import/export history table or by using the `UpdateExport` operation. If you choose **Download** in the history table for an export and you don't provide a password, Amazon Lex downloads an unprotected zip file.

JSON format for importing and exporting

You import and export bots and bot locales from Amazon Lex using a .zip file that contains JSON structures that describe the parts of the bot or alias. When you export a bot, Amazon Lex creates the .zip file and makes it available to you using an Amazon S3 pre-signed URL. When you import a bot, you must create a .zip file that contains the JSON structures and upload it to an S3 pre-signed URL.

Amazon Lex creates the following directory structure in the .zip file when you export a bot.

```
BotName_BotVersion_ExportID_LexJson.zip
-or-
BotName_BotVersion_LocaleId_ExportId_LEX_JSON.zip
    --> manifest.json
    --> BotName
    ----> Bot.json
    ----> BotLocales
    -----> Locale_A
    -----> BotLocale.json
    -----> Intents
    -----> Intent_A
    -----> Intent.json
    -----> Slots
    -----> Slot_A
    -----> Slot.json
    -----> Slot_B
    -----> Slot.json
    -----> Intent_B
    ...
    -----> SlotTypes
    -----> SlotType_A
    -----> SlotType.json
    -----> SlotType_B
    ...
    -----> Locale_B
    ...
```

Manifest file structure

The manifest file contains metadata for the export file.

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale"
    }
}
```

Bot file structure

The bot file contains the configuration information for the bot.

```
{  
    "name": "BotName",  
    "identifier": "identifier",  
    "version": "number",  
    "description": "description",  
    "dataPrivacy": {  
        "childDirected": true | false  
    },  
    "idleSessionTTLInSeconds": seconds  
}
```

Bot locale file structure

The bot locale file contains a description of the locale or language of a bot. When you export a bot, there can be more than one bot locale file in the .zip file. When you export a bot locale, there is only one locale in the zip file.

```
{  
    "name": "locale name",  
    "identifier": "locale ID",  
    "version": "number",  
    "description": "description",  
    "voiceSettings": {  
        "voiceId": "voice"  
    },  
    "nluConfidenceThreshold": number  
}
```

Intent file structure

The intent file contains the configuration information for an intent. There is one intent file in the .zip file for each intent in a specific locale.

The following is an example of a JSON structure for the BookCar intent in the sample BookTrip bot. For a complete example of the JSON structure for an intent, see the [CreateIntent \(p. 192\)](#) operation.

```
{  
    "name": "BookCar",  
    "identifier": "891RWHHICO",  
    "description": "Intent to book a car.",  
    "parentIntentSignature": null,  
    "sampleUtterances": [  
        {  
            "utterance": "Book a car"  
        },  
        {  
            "utterance": "Reserve a car"  
        },  
        {  
            "utterance": "Make a car reservation"  
        }  
    ],  
    "intentConfirmationSetting": {  
        "confirmationPrompt": {  
            "messageGroupList": [  
                {  
                    "text": "Would you like to proceed with the booking?"  
                }  
            ]  
        }  
    }  
}
```

```

        "message": {
            "plainTextMessage": {
                "value": "OK, I have you down for a {CarType} hire in
{PickUpCity} from {PickUpDate} to {ReturnDate}. Should I book the reservation?"
            },
            "ssmlMessage": null,
            "customPayload": null,
            "imageResponseCard": null
        },
        "variations": null
    }
],
"maxRetries": 2
},
"declinationResponse": {
    "messageGroupList": [
        {
            "message": {
                "plainTextMessage": {
                    "value": "OK, I have cancelled your reservation in progress."
                },
                "ssmlMessage": null,
                "customPayload": null,
                "imageResponseCard": null
            },
            "variations": null
        }
    ]
},
"intentClosingSetting": null,
"inputContexts": null,
"outputContexts": null,
"kendraConfiguration": null,
"dialogCodeHook": null,
"fulfillmentCodeHook": null,
"slotPriorities": [
    {
        "slotName": "DriverAge",
        "priority": 4
    },
    {
        "slotName": "PickUpDate",
        "priority": 2
    },
    {
        "slotName": "ReturnDate",
        "priority": 3
    },
    {
        "slotName": "PickUpCity",
        "priority": 1
    },
    {
        "slotName": "CarType",
        "priority": 5
    }
]
}

```

Slot file structure

The slot file contains the configuration information for a slot in an intent. There is one slot file in the .zip file for each slot defined for an intent in a specific locale.

The following example is the JSON structure of a slot that enables the customer to choose the type of car they wish to rent in the BookCar intent in the BookTrip example bot. For a complete example of the JSON structure for an slot, see the [CreateSlot \(p. 212\)](#) operation.

```
{
    "name": "CarType",
    "identifier": "KDHJWNGZGC",
    "description": "Type of car being reserved.",
    "multipleValuesSetting": {
        "allowMultipleValues": false
    },
    "slotTypeName": "CarTypeValues",
    "obfuscationSetting": null,
    "slotConstraint": "Required",
    "defaultValueSpec": null,
    "slotValueElicitationSetting": {
        "promptSpecification": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "What type of car would you like to rent? Our most popular options are economy, midsize, and luxury"
                        }
                    }
                }
            ],
            "maxRetries": 2
        },
        "sampleValueElicitingUtterances": null,
        "waitAndContinueSpecification": null,
    }
}
```

Slot type file structure

The slot type file contains the configuration information for a custom slot type used in a language or locale. There is one slot type file in the .zip file for each custom slot type in a specific locale.

The following is the JSON structure for the slot type that lists the types of cars available in the BookTrip example bot. For a complete example of the JSON structure for a slot type, see the [CreateSlotType \(p. 224\)](#) operation.

```
{
    "name": "CarTypeValues",
    "identifier": "BROZ1QFWSH",
    "description": "Enumeration representing possible types of cars available for rental",
    "slotTypeValues": [
        {
            "sampleValue": {
                "value": "economy"
            },
            "synonyms": null
        },
        {
            "sampleValue": {
                "value": "standard"
            },
            "synonyms": null
        }
    ]
}
```

```
        "synonyms": null
    },
{
    "sampleValue": {
        "value": "midsize"
    },
    "synonyms": null
},
{
    "sampleValue": {
        "value": "full size"
    },
    "synonyms": null
},
{
    "sampleValue": {
        "value": "luxury"
    },
    "synonyms": null
},
{
    "sampleValue": {
        "value": "minivan"
    },
    "synonyms": null
}
],
"parentSlotTypeSignature": null,
"slotTypeConfiguration": null,
"valueSelectionStrategy": "ORIGINAL_VALUE"
}
```

Tagging resources

To help you manage your Amazon Lex bots and bot aliases, you can assign metadata to each resource as *tags*. A tag is a label that you assign to an AWS resource. Each tag consists of a key and a value.

Tags enable you to categorize your AWS resources in different ways, for example, by purpose, owner, or application. Tags help you to:

- Identify and organize your AWS resources. Many AWS resources support tagging, so you can assign the same tag to resources in different services to indicate that the resources are the same. For example, you can tag a bot and the Lambda functions that it uses with the same tag.
- Allocate costs. You activate tags on the AWS Billing and Cost Management dashboard. AWS uses the tags to categorize your costs and deliver a monthly cost allocation report to you. For Amazon Lex, you can allocate costs for each alias using tags specific to the alias. For more information, see [Use cost allocation tags](#) in the *AWS Billing and Cost Management User Guide*.
- Control access to your resources. You can use tags with Amazon Lex to create policies to control access to Amazon Lex resources. These policies can be attached to an IAM role or user to enable tag-based access control.

You can work with tags using the AWS Management Console, the AWS Command Line Interface, or the Amazon Lex API.

Tagging your resources

If you are using the Amazon Lex console, you can tag resources when you create them, or you can add the tags later. You can also use the console to update or remove existing tags.

If you are using the AWS CLI or Amazon Lex API, you use the following operations to manage tags for your resource:

- [CreateBot \(p. 168\)](#) and [CreateBotAlias \(p. 173\)](#) – apply tags when you create a bot or a bot alias.
- [ListTagsForResource \(p. 341\)](#) – view the tags associated with a resource.
- [TagResource \(p. 347\)](#) – add and modify tags on an existing resource.
- [UntagResource \(p. 349\)](#) – remove tags from a resource.

The following resources in Amazon Lex support tagging:

- Bots – use an Amazon Resource Name (ARN) like the following:
 - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Bot aliases – use an ARN like the following:
 - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

The `bot-id` and `bot-alias-id` values are capitalized alphanumeric strings 10 characters long.

Tag restrictions

The following basic restrictions apply to tags on Amazon Lex resources:

- Maximum number of keys – 50 using the console, 200 using the API
- Maximum key length – 128 characters
- Maximum value length – 256 characters
- Valid characters for key and value – a-z, A-Z, 0-9, space, and the following characters: _.:/=+- and @
- Keys and values are case-sensitive
- Don't use aws : as a prefix for keys, it's reserved for AWS use

Tagging resources (console)

You can use the console to manage tags on a bot or bot alias. You can add tags when you create the resource, or you can add, modify, or remove tags from existing resources.

To add a tag when you create a bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>.
2. Choose **Create bot**.
3. In the **Advanced settings** section of **Configure bot settings**, choose **Add new tag**. You can add tags to the bot and to the TestBotAlias alias.
4. Choose **Next** to continue creating your bot.

To add a tag when you create a bot alias

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>.
2. Choose the bot that you want to add the bot alias to.
3. From the left menu, choose **Aliases** and then choose **Create alias**.
4. In **General info**, choose **Add new tag** from **Tags**.
5. Choose **Create**.

To add, remove, or modify a tag on an existing bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>.
2. Choose the bot that you want to modify.
3. From the left menu, choose **Settings**, and then choose **Edit**.
4. In **Tags**, make your changes.
5. Choose **Save** to save your changes to the bot.

To add, remove, or modify a tag on an existing alias

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>.
2. Choose the bot that you want to modify.
3. From the left menu, choose **Aliases** and then from the list of aliases, choose the alias to modify.
4. From **Alias details**, in **Tags**, choose **Modify tags**.
5. In **Manage tags**, make your changes.
6. Choose **Save** to save your changes to the alias.

Security in Amazon Lex

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to Amazon Lex, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Lex. The following topics show you how to configure Amazon Lex to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Lex resources.

Topics

- [Data protection in Amazon Lex \(p. 120\)](#)
- [Identity and access management for Amazon Lex \(p. 122\)](#)
- [Using service-linked roles for Amazon Lex \(p. 148\)](#)
- [Logging and monitoring in Amazon Lex \(p. 151\)](#)
- [Compliance validation for Amazon Lex \(p. 151\)](#)
- [Resilience in Amazon Lex \(p. 152\)](#)
- [Infrastructure security in Amazon Lex \(p. 152\)](#)

Data protection in Amazon Lex

Amazon Lex conforms to the AWS [shared responsibility model](#), which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.

- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon Lex or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Lex or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog](#).

Encryption at rest

Amazon Lex encrypts user utterances and other information that it stores.

Topics

- [Sample utterances \(p. 121\)](#)
- [Session attributes \(p. 121\)](#)
- [Request attributes \(p. 121\)](#)

Sample utterances

When you develop a bot, you can provide sample utterances for each intent and slot. You can also provide custom values and synonyms for slots. This information is encrypted at rest, and it is used only to build the bot and create the customer experience.

Session attributes

Session attributes contain application-specific information that is passed between Amazon Lex and client applications. Amazon Lex passes session attributes to all AWS Lambda functions configured for a bot. If a Lambda function adds or updates session attributes, Amazon Lex passes the new information back to the client application.

Session attributes persist in an encrypted store for the duration of the session. You can configure the session to remain active for a minimum of 1 minute and up to 24 hours after the last user utterance. The default session duration is 5 minutes.

Request attributes

Request attributes contain request-specific information and apply only to the current request. A client application uses request attributes to send information to Amazon Lex at runtime.

You use request attributes to pass information that doesn't need to persist for the entire session. Because request attributes don't persist across requests, they aren't stored.

Encryption in transit

Amazon Lex uses the HTTPS protocol to communicate with your client application. It uses HTTPS and AWS signatures to communicate with other services, such as Amazon Polly and AWS Lambda on your application's behalf.

Identity and access management for Amazon Lex

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Lex resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience \(p. 122\)](#)
- [Authenticating with identities \(p. 122\)](#)
- [Managing access using policies \(p. 124\)](#)
- [How Amazon Lex works with IAM \(p. 126\)](#)
- [Identity-based policy examples for Amazon Lex \(p. 133\)](#)
- [Resource-based policy examples for Amazon Lex \(p. 139\)](#)
- [Troubleshooting Amazon Lex identity and access \(p. 146\)](#)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon Lex.

Service user – If you use the Amazon Lex service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Lex features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Lex, see [Troubleshooting Amazon Lex identity and access \(p. 146\)](#).

Service administrator – If you're in charge of Amazon Lex resources at your company, you probably have full access to Amazon Lex. It's your job to determine which Amazon Lex features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Lex, see [How Amazon Lex works with IAM \(p. 126\)](#).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Lex. To view example Amazon Lex identity-based policies that you can use in IAM, see [Identity-based policy examples for Amazon Lex \(p. 133\)](#).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see [Signing in to the AWS Management Console as an IAM user or root user](#) in the [IAM User Guide](#).

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the [AWS Management Console](#), use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request

using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using [Signature Version 4](#), a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 signing process](#) in the [AWS General Reference](#).

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Using multi-factor authentication \(MFA\) in AWS](#) in the [IAM User Guide](#).

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see [Managing access keys for IAM users](#) in the [IAM User Guide](#). When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the [IAM User Guide](#).

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the [IAM User Guide](#).

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an *identity provider*. For more information about federated users, see [Federated users and roles](#) in the [IAM User Guide](#).
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access.

However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
- **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, resources, and condition keys for Amazon Lex](#) in the *Service Authorization Reference*.
- **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies.

Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How Amazon Lex works with IAM

Before you use IAM to manage access to Amazon Lex, learn what IAM features are available to use with Amazon Lex.

IAM features you can use with Amazon Lex

IAM feature	Amazon Lex support
Identity-based policies (p. 126)	Yes
Resource-based policies (p. 127)	Yes
Policy actions (p. 129)	Yes
Policy resources (p. 130)	Yes
Policy condition keys (p. 131)	No
ACLs (p. 131)	No
ABAC (tags in policies) (p. 131)	Yes
Temporary credentials (p. 132)	No
Principal permissions (p. 132)	Yes
Service roles (p. 132)	Yes
Service-linked roles (p. 133)	Partial

To get a high-level view of how Amazon Lex and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Identity-based policies for Amazon Lex

Supports identity-based policies	Yes
----------------------------------	-----

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based

policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

Identity-based policy examples for Amazon Lex

To view examples of Amazon Lex identity-based policies, see [Identity-based policy examples for Amazon Lex \(p. 133\)](#).

Resource-based policies within Amazon Lex

Supports resource-based policies	Yes
----------------------------------	-----

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include users, roles, federated users, or AWS services.

You can't use cross-account or cross-region policies with Amazon Lex. If you create a policy for a resource with a cross-account or cross-region ARN, Amazon Lex returns an error.

The Amazon Lex service supports resource-based policies called a *bot policy* and a *bot alias policy*, which are attached to a bot or a bot alias. These policies define which principals can perform actions on the bot or bot alias.

Actions can only be used on specific resources. For example, the `UpdateBot` action can only be used on bot resources, the `UpdateBotAlias` action can only be used on bot alias resources. If you specify an action in a policy that can't be used on the resource specified in the policy, Amazon Lex returns an error. For the list of actions and the resources that they can be used with, see the following table.

Action	Supports resource-based policy	Resource
<code>BuildBotLocale</code>	Supported	<code>BotId</code>
<code>CreateBot</code>	No	
<code>CreateBotAlias</code>	No	
<code>CreateBotChannel [permission only]</code>	Supported	<code>BotId</code>
<code>CreateBotLocale</code>	Supported	<code>BotId</code>
<code>CreateBotVersion</code>	Supported	<code>BotId</code>
<code>CreateExport</code>	Supported	<code>BotId</code>
<code>CreateIntent</code>	Supported	<code>BotId</code>
<code>CreateResourcePolicy</code>	Supported	<code>BotId, BotAliasId</code>
<code>CreateSlot</code>	Supported	<code>BotId</code>
<code>CreateSlotType</code>	Supported	<code>BotId</code>
<code>CreateUploadUrl</code>	No	

Action	Supports resource-based policy	Resource
DeleteBot	Supported	BotId, BotAliasId
DeleteBotAlias	Supported	BotAliasId
DeleteBotChannel [permission only]	Supported	BotId
DeleteBotLocale	Supported	BotId
DeleteBotVersion	Supported	BotId
DeleteExport	Supported	BotId
DeleteImport	Supported	BotId
DeleteIntent	Supported	BotId
DeleteResourcePolicy	Supported	BotId, BotAliasId
DeleteSession	Supported	BotAliasId
DeleteSlot	Supported	BotId
DeleteSlotType	Supported	BotId
DescribeBot	Supported	BotId
DescribeBotAlias	Supported	BotAliasId
DescribeBotChannel [permission only]	Supported	BotId
DescribeBotLocale	Supported	BotId
DescribeBotVersion	Supported	BotId
DescribeExport	Supported	BotId
DescribeImport	Supported	BotId
DescribeIntent	Supported	BotId
DescribeResourcePolicy	Supported	BotId, BotAliasId
DescribeSlot	Supported	BotId
DescribeSlotType	Supported	BotId
GetSession	Supported	BotAliasId
ListBotAliases	Supported	BotAliasId
ListBotChannels [permission only]	Supported	BotId
ListBotLocales	Supported	BotId
ListBots	No	
ListBotVersions	Supported	BotId

Action	Supports resource-based policy	Resource
ListBuiltInIntents	No	
ListBuiltInSlotTypes	No	
ListExports	No	
ListImports	No	
ListIntents	Supported	BotId
ListSlots	Supported	BotId
ListSlotTypes	Supported	BotId
PutSession	Supported	BotAliasId
RecognizeText	Supported	BotAliasId
RecognizeUtterance	Supported	BotAliasId
StartConversation	Supported	BotAliasId
StartImport	Supported	BotId, BotAliasId
TagResource	No	
UpdateBot	Supported	BotId
UpdateBotAlias	Supported	BotAliasId
UpdateBotLocale	Supported	BotId
UpdateBotVersion	Supported	BotId
UpdateExport	Supported	BotId
UpdateIntent	Supported	BotId
UpdateResourcePolicy	Supported	BotId, BotAliasId
UpdateSlot	Supported	BotId
UpdateSlotType	Supported	BotId
UntagResource	No	

To learn how to attach a resource-based policy to a bot or bot alias, see [Resource-based policy examples for Amazon Lex \(p. 139\)](#).

Resource-based policy examples within Amazon Lex

To view examples of Amazon Lex resource-based policies, see [Resource-based policy examples for Amazon Lex \(p. 139\)](#).

Policy actions for Amazon Lex

Supports policy actions	Yes
-------------------------	-----

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Lex actions, see [Actions defined by Amazon Lex](#) in the *Service Authorization Reference*.

Policy actions in Amazon Lex use the following prefix before the action:

```
lex
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [  
    "lex:action1",  
    "lex:action2"  
]
```

To view examples of Amazon Lex identity-based policies, see [Identity-based policy examples for Amazon Lex \(p. 133\)](#).

Policy resources for Amazon Lex

Supports policy resources	Yes
---------------------------	-----

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon Lex resource types and their ARNs, see [Resources defined by Amazon Lex](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions defined by Amazon Lex](#).

To view examples of Amazon Lex identity-based policies, see [Identity-based policy examples for Amazon Lex \(p. 133\)](#).

Policy condition keys for Amazon Lex

Supports policy condition keys	No
--------------------------------	----

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The **Condition** element (or **Condition block**) lets you specify conditions in which a statement is in effect. The **Condition** element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple **Condition** elements in a statement, or multiple keys in a single **Condition** element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

To see a list of Amazon Lex condition keys, see [Condition keys for Amazon Lex](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions defined by Amazon Lex](#).

To view examples of Amazon Lex identity-based policies, see [Identity-based policy examples for Amazon Lex \(p. 133\)](#).

Access control lists (ACLs) in Amazon Lex

Supports ACLs	No
---------------	----

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Amazon Lex

Supports ABAC (tags in policies)	Yes
----------------------------------	-----

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

For more information about ABAC, see [What is ABAC?](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

Using Temporary credentials with Amazon Lex

Supports temporary credentials	No
--------------------------------	----

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switching to a role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

Cross-service principal permissions for Amazon Lex

Supports principal permissions	Yes
--------------------------------	-----

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, resources, and condition keys for Amazon Lex](#) in the *Service Authorization Reference*.

Service roles for Amazon Lex

Supports service roles	Yes
------------------------	-----

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

Warning

Changing the permissions for a service role might break Amazon Lex functionality. Edit service roles only when Amazon Lex provides guidance to do so.

Service-linked roles for Amazon Lex

Supports service-linked roles	Partial
-------------------------------	---------

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see [AWS services that work with IAM](#). Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the Yes link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon Lex

By default, IAM users and roles don't have permission to create or modify Amazon Lex resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating policies on the JSON tab](#) in the *IAM User Guide*.

Topics

- [Policy best practices \(p. 133\)](#)
- [Using the Amazon Lex console \(p. 134\)](#)
- [Allow users to add functions to a bot \(p. 134\)](#)
- [Allow users to add channels to a bot \(p. 134\)](#)
- [Allow users to create and update bots \(p. 135\)](#)
- [Allow users to delete bots \(p. 135\)](#)
- [Allow users to have a conversation with a bot \(p. 136\)](#)
- [Allow a specific user to manage resource-based policies \(p. 136\)](#)
- [Allow a user to export bots and bot locales \(p. 136\)](#)
- [Allow a user to import bots and bot locales \(p. 137\)](#)
- [Allow a user to migrate a bot from Amazon Lex V1 to Amazon Lex V2 \(p. 137\)](#)
- [Allow users to view their own permissions \(p. 138\)](#)

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon Lex resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using Amazon Lex quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see [Get started using permissions with AWS managed policies](#) in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as

necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see [Grant least privilege](#) in the *IAM User Guide*.

- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.

Using the Amazon Lex console

To access the Amazon Lex console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Lex resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

To ensure that users and roles can still use the Amazon Lex console, users need to have Console access. For more information about creating a user with Console access, see [Creating an IAM user in your AWS account](#) in the *IAM User Guide*.

Allow users to add functions to a bot

This example shows a policy that allows IAM users to add Amazon Comprehend, sentiment analysis and Amazon Kendra query permissions to an Amazon Lex bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/AWSServiceRoleForLexV2Bots*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/AWSServiceRoleForLexV2Bots*"
        }
    ]
}
```

Allow users to add channels to a bot

This example is a policy that allows IAM users to add a messaging channel to a bot. A user must have this policy in place before they can deploy a bot on a messaging platform.

```
{
    "Version": "2012-10-17",
```

```

"Statement": [
  {
    "Sid": "Id1",
    "Effect": "Allow",
    "Action": "iam:PutRolePolicy",
    "Resource": "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
  },
  {
    "Sid": "Id2",
    "Effect": "Allow",
    "Action": "iam:GetRolePolicy",
    "Resource": "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
  }
]
}

```

Allow users to create and update bots

This example shows an example policy that allows IAM users to create and update any bot. The policy includes permissions to complete this action on the console or using the AWS CLI or AWS API.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex>CreateBot",
        "lex:UpdateBot",
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*"]
    }
  ]
}

```

Allow users to delete bots

This example shows an example policy that allows IAM users to delete any bot. The policy includes permissions to complete this action on the console or using the AWS CLI or AWS API.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex>DeleteBot",
        "lex>DeleteBotLocale",
        "lex>DeleteBotAlias",
        "lex>DeleteIntent",
        "lex>DeleteSlot",
        "lex>DeleteSlottype"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*",
                  "arn:aws:lex:Region:123412341234:bot-alias/*"]
    }
  ]
}

```

Allow users to have a conversation with a bot

This example shows an example policy that allows IAM users have a conversation with any bot. The policy includes permissions to complete this action on the console or using the AWS CLI or AWS API.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "lex:StartConversation",  
                "lex:RecognizeText",  
                "lex:RecognizeUtterance",  
                "lex:GetSession",  
                "lex:PutSession",  
                "lex:DeleteSession"  
            ],  
            "Effect": "Allow",  
            "Resource": "arn:aws:lex:Region:123412341234:bot-alias/*"  
        }  
    ]  
}
```

Allow a specific user to manage resource-based policies

The following example grants permission for a specific user to manage the resource-based policies. It allows console and API access to the policies associated with bots and bot aliases.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "ResourcePolicyEditor",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::123456789012:role/ResourcePolicyEditor"  
            },  
            "Action": [  
                "lex>CreateResourcePolicy",  
                "lex:UpdateResourcePolicy",  
                "lex>DeleteResourcePolicy",  
                "lex:DescribeResourcePolicy"  
            ]  
        }  
    ]  
}
```

Allow a user to export bots and bot locales

The following IAM permission policy enables a user to create, update, and get an export for a bot or bot locale.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "lex>CreateExport",  
                "lex:UpdateExport",  
                "lex:DescribeExport",  
                "lex:DescribeBot",  
                "lex:DeleteExport"  
            ]  
        }  
    ]  
}
```

```
        "lex:DescribeBotLocale",
        "lex>ListBotLocales",
        "lex:DescribeIntent",
        "lex>ListIntents",
        "lex:DescribeSlotType",
        "lex>ListSlotTypes",
        "lex:DescribeSlot",
        "lex>ListSlots"
    ],
    "Effect": "Allow",
    "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
}
]
}
```

Allow a user to import bots and bot locales

The following IAM permission policy allows a user to import a bot or bot locale and to check the status of an import.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex>CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex>CreateBot",
                "lex:UpdateBot",
                "lex>DeleteBot",
                "lex>CreateBotLocale",
                "lex:UpdateBotLocale",
                "lex>DeleteBotLocale",
                "lex>CreateIntent",
                "lex:UpdateIntent",
                "lex>DeleteIntent",
                "lex>CreateSlotType",
                "lex:UpdateSlotType",
                "lex>DeleteSlotType",
                "lex>CreateSlot",
                "lex:UpdateSlot",
                "lex>DeleteSlot",
                "iam:PassRole",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*",
                "arn:aws:lex:Region:123456789012:bot-alias/*"
            ]
        }
    ]
}
```

Allow a user to migrate a bot from Amazon Lex V1 to Amazon Lex V2

The following IAM permission policy allows a user to start migrating a bot from Amazon Lex V1 to Amazon Lex V2.

```
{
```

```

"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "startMigration",
        "Effect": "Allow",
        "Action": "lex:StartMigration",
        "Resource": "arn:aws:lex:>Region<:>123456789012<:bot:*"
    },
    {
        "Sid": "passRole",
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "arn:aws:iam:>123456789012<:role/>v2 bot role<"
    },
    {
        "Sid": "allowOperations",
        "Effect": "Allow",
        "Action": [
            "lex:CreateBot",
            "lex:CreateIntent",
            "lex:UpdateSlot",
            "lex:DescribeBotLocale",
            "lex:UpdateBotAlias",
            "lex:CreateSlotType",
            "lex:DeleteBotLocale",
            "lex:DescribeBot",
            "lex:UpdateBotLocale",
            "lex:CreateSlot",
            "lex:DeleteSlot",
            "lex:UpdateBot",
            "lex:DeleteSlotType",
            "lex:DescribeBotAlias",
            "lex:CreateBotLocale",
            "lex:DeleteIntent",
            "lex:StartImport",
            "lex:UpdateSlotType",
            "lex:UpdateIntent",
            "lex:DescribeImport"
        ],
        "Resource": [
            "arn:aws:lex:>Region<:>123456789012<:bot/*",
            "arn:aws:lex:>Region<:>123456789012<:bot-alias/*/*"
        ]
    },
    {
        "Sid": "showBots",
        "Effect": "Allow",
        "Action": [
            "lex>CreateUploadUrl",
            "lex>ListBots"
        ],
        "Resource": "*"
    }
]
}

```

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
```

```
"Statement": [
    {
        "Sid": "ViewOwnUserInfo",
        "Effect": "Allow",
        "Action": [
            "iam:GetUserPolicy",
            "iam>ListGroupsForUser",
            "iam>ListAttachedUserPolicies",
            "iam>ListUserPolicies",
            "iam GetUser"
        ],
        "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
        "Sid": "NavigateInConsole",
        "Effect": "Allow",
        "Action": [
            "iam:GetGroupPolicy",
            "iam:GetPolicyVersion",
            "iam GetPolicy",
            "iam>ListAttachedGroupPolicies",
            "iam>ListGroupPolicies",
            "iam>ListPolicyVersions",
            "iam>ListPolicies",
            "iam>ListUsers"
        ],
        "Resource": "*"
    }
]
```

Resource-based policy examples for Amazon Lex

A *resource-based policy* is attached to a resource, such as a bot or a bot alias. With a resource-based policy you can specify who has access to the resource and the actions that they can perform on it. For example, you can add resource-based policies that enable a user to modify a specific bot, or to allow a user to use runtime operations on a specific bot alias.

When you use a resource-based policy you can allow other AWS services to access resources in your account. For example, you can allow Amazon Connect to access an Amazon Lex bot.

To learn how to create a bot or bot alias, see [Building bots \(p. 16\)](#).

Topics

- [Use the console to specify a resource-based policy \(p. 139\)](#)
- [Use the API to specify a resource-based policy \(p. 142\)](#)
- [Allow an IAM role to update a bot and list bot aliases \(p. 144\)](#)
- [Allow a user to have a conversation with a bot \(p. 144\)](#)
- [Allow an AWS service to use a specific Amazon Lex bot \(p. 145\)](#)

Use the console to specify a resource-based policy

You can use the Amazon Lex console to manage the resource-based policies for your bots and bot aliases. You enter the JSON structure of a policy and the console associates it with the resource. If there is a policy already associated with a resource, you can use the console to view and modify the policy.

When you save a policy with the policy editor, the console checks the syntax of the policy. If the policy contains errors, such as a non-existent user or an action that is not supported by the resource, it returns an error and doesn't save the policy.

The following shows the resource-based policy editor for a bot in the console. The policy editor for a bot alias is similar.

Resource-based policy

You can use a resource-based policy to grant access permission to other AWS services, IAM users, and roles.

Resource ARN

arn:aws:lex:us-west-2: [REDACTED]:bot/AKWB8PVLD2

Policy

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Sid": "botRunners",  
6       "Effect": "Allow",  
7       "Principal": {  
8         "AWS": "arn:aws:iam::123456789012:user/botRunner"  
9       },  
10      "Action": [  
11        "lex:RecognizeText",  
12        "lex:RecognizeUtterance",  
13        "lex:StartConversaion"  
14      ],  
15      "Resource": [  
16        "arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"  
17      ]  
18    }  
19  ]  
20 }
```

Cancel

Save

To open the policy editor for a bot

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. From the **Bots** list, choose the bot whose policy you want to edit.
3. In the **Resource-based policy** section, choose **Edit**.

To open the policy editor for a bot alias

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lex/>.
2. From the **Bots** list, choose the bot that contains the alias that you want to edit.
3. From the left menu, choose **Aliases**, then choose the alias to edit.
4. In the **Resource-based policy** section, choose **Edit**.

Use the API to specify a resource-based policy

You can use API operations to manage the resource-based policies for your bots and bot aliases. There are operations to create, update and delete policies.

[CreateResourcePolicy \(p. 205\)](#)

Adds a new resource policy with the specified policy statements to a bot or bot alias.

[CreateResourcePolicyStatement \(p. 208\)](#)

Adds a new resource policy statement to a bot or bot alias.

[DeleteResourcePolicy \(p. 249\)](#)

Removes a resource policy from a bot or bot alias.

[DeleteResourcePolicyStatement \(p. 251\)](#)

Removes a resource policy statement from a bot or bot alias.

[DescribeResourcePolicy \(p. 287\)](#)

Gets a resource policy and the policy revision.

[UpdateResourcePolicy \(p. 380\)](#)

Replaces the existing resource policy for a bot or bot alias with a new one.

Examples

Java

The following example shows how to use the resource-based policy operations to manage a resource-based policy.

```
/*
 * Create a new policy for the specified bot alias
 * that allows a role to invoke lex:UpdateBotAlias on it.
 * The created policy will have revision id 1.
 */

CreateResourcePolicyRequest createPolicyRequest =
    CreateResourcePolicyRequest.builder()
```

```

        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
        .policy("{\"Version\": \"2012-10-17\", \"Statement\":
[{\\"Sid\\": \\"BotAliasEditor\\\", \\"Effect\\": \\"Allow\\\", \\"Principal\\": \\"AWS\":
\\arn:aws:iam::123456789012:role/BotAliasEditor\\\"}, \\"Action\\": [\\\"lex:UpdateBotAlias
\\\"], \\"Resource\\": [\\\"arn:aws:lex:Region:123456789012:bot-alias/MYBOTALIAS/TSTALIASID
\\\"]}]})")

lexmodelsv2Client.createResourcePolicy(createPolicyRequest);

/*
 * Overwrite the policy for the specified bot alias with a new policy.
 * Since no expectedRevisionId is provided, this request overwrites the current
revision.
 * After this update, the revision id for the policy is 2.
 */
UpdateResourcePolicyRequest updatePolicyRequest =
UpdateResourcePolicyRequest.builder()
        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
        .policy("{\"Version\": \"2012-10-17\", \"Statement\":
[{\\"Sid\\": \\"BotAliasEditor\\\", \\"Effect\\": \\"Deny\\\", \\"Principal\\": \\"AWS\":
\\arn:aws:iam::123456789012:role/BotAliasEditor\\\"}, \\"Action\\": [\\\"lex:UpdateBotAlias
\\\"], \\"Resource\\": [\\\"arn:aws:lex:Region:123456789012:bot-alias/MYBOTALIAS/TSTALIASID
\\\"]}]})")

lexmodelsv2Client.updateResourcePolicy(updatePolicyRequest);

/*
 * Creates a statement in an existing policy for the specified bot alias
 * that allows a role to invoke lex:RecognizeText on it.
 * This request expects to update revision 2 of the policy. The request will
fail
 * if the current revision of the policy is no longer revision 2.
 * After this request, the revision id for this policy will be 3.
 */
CreateResourcePolicyStatementRequest createStatementRequest =
CreateResourcePolicyStatementRequest.builder()
        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
        .effect("Allow")

.principal(Principal.builder().arn("arn:aws:iam::123456789012:role/
BotRunner").build())
        .action("lex:RecognizeText")
        .statementId("BotRunnerStatement")
        .expectedRevisionId(2)
        .build();

lexmodelsv2Client.createResourcePolicyStatement(createStatementRequest);

/*
 * Deletes a statement from an existing policy for the specified bot alias by
statementId.
 * Since no expectedRevisionId is supplied, the request will remove the
statement from
 * the current revision of the policy for the bot alias.
 * After this request, the revision id for this policy will be 4.
 */
DeleteResourcePolicyRequest deleteStatementRequest =
DeleteResourcePolicyRequest.builder()
        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
        .statementId("BotRunnerStatement")
        .build();

```

```

lexmodelsv2Client.deleteResourcePolicy(deleteStatementRequest);

/*
 * Describe the current policy for the specified bot alias
 * It always returns the current revision.
 */
DescribeResourcePolicyRequest describePolicyRequest =
    DescribeResourcePolicyRequest.builder()
        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
        .build();

lexmodelsv2Client.describeResourcePolicy(describePolicyRequest);

/*
 * Delete the current policy for the specified bot alias
 * This request expects to delete revision 3 of the policy. Since the revision
id for
 * this policy is already at 4, this request will fail.
 */
DeleteResourcePolicyRequest deletePolicyRequest =
    DeleteResourcePolicyRequest.builder()
        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
        .expectedRevisionId(3);
        .build();

lexmodelsv2Client.deleteResourcePolicy(deletePolicyRequest);

```

Allow an IAM role to update a bot and list bot aliases

The following example grants permissions for a specific IAM role to call Amazon Lex model building API operations to modify an existing bot. The user can list aliases for a bot and update the bot, but can't delete the bot or bot aliases.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botBuilders",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/BotBuilder"
            },
            "Action": [
                "lex>ListBotAliases",
                "lex:UpdateBot"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/MYBOT"
            ]
        }
    ]
}
```

Allow a user to have a conversation with a bot

The following example grants permission for a specific user to call Amazon Lex runtime API operations on a single alias of a bot.

The user is specifically denied permission to update or delete the bot alias.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botRunners",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation",
                "lex>DeleteSession",
                "lex:GetSession",
                "lex:PutSession"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        },
        {
            "Sid": "botRunners",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:UpdateBotAlias",
                "lex>DeleteBotAlias"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        }
    ]
}
```

Allow an AWS service to use a specific Amazon Lex bot

The following example grants permission for AWS Lambda and Amazon Connect to call Amazon Lex runtime API operations.

The condition block is required for service principals, and must use the global context keys `AWS:SourceAccount` and `AWS:SourceArn`.

The `AWS:SourceAccount` is the account ID that is calling the Amazon Lex bot.

The `AWS:SourceArn` is the resource ARN of the Amazon Connect service instance or Lambda function that the call to the Amazon Lex bot alias originates from.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "connect-bot-alias",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "connect.amazonaws.com"
                ]
            }
        }
    ]
}
```

```
        ],
    },
    "Action": [
        "lex:RecognizeText",
        "lex:StartConversation"
    ],
    "Resource": [
        "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
    ],
    "Condition": {
        "StringEquals": {
            "AWS:SourceAccount": "123456789012"
        },
        "ArnEquals": {
            "AWS:SourceArn":
                "arn:aws:connect:Region:123456789012:instance/instance-id"
        }
    },
    {
        "Sid": "lambda-function",
        "Effect": "Allow",
        "Principal": {
            "Service": [
                "lambda.amazonaws.com"
            ]
        },
        "Action": [
            "lex:RecognizeText",
            "lex:StartConversation"
        ],
        "Resource": [
            "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
        ],
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "123456789012"
            },
            "ArnEquals": {
                "AWS:SourceArn": "arn:aws:lambda:Region:123456789012:function/function-name"
            }
        }
    }
]
```

Troubleshooting Amazon Lex identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Lex and IAM.

Topics

- [I am not authorized to perform an action in Amazon Lex \(p. 147\)](#)
- [I am not authorized to perform iam:PassRole \(p. 147\)](#)
- [I want to view my access keys \(p. 147\)](#)
- [I'm an administrator and want to allow others to access Amazon Lex \(p. 148\)](#)
- [I want to allow people outside of my AWS account to access my Amazon Lex resources \(p. 148\)](#)

I am not authorized to perform an action in Amazon Lex

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional `my-example-widget` resource but does not have the fictional `lex:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
lex:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-example-widget` resource using the `lex:GetWidget` action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Amazon Lex.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Lex. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing access keys](#) in the *IAM User Guide*.

I'm an administrator and want to allow others to access Amazon Lex

To allow others to access Amazon Lex, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon Lex.

To get started right away, see [Creating your first IAM delegated user and group](#) in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my Amazon Lex resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Lex supports these features, see [How Amazon Lex works with IAM \(p. 126\)](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

Using service-linked roles for Amazon Lex

Amazon Lex uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to Amazon Lex. Service-linked roles are predefined by Amazon Lex and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon Lex easier because you don't have to manually add the necessary permissions. Amazon Lex defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon Lex can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your Amazon Lex resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Amazon Lex

Amazon Lex uses these service-linked roles.

- **AWSServiceRoleForLexV2Bots_** – Gives permission to connect your bot to other required services..
- **AWSServiceRoleForLexV2Channels_** – Gives permission to list bots in an account and to call conversation APIs for a bot..

The AWSServiceRoleForLexV2Bots_ service-linked role trusts the following service to assume the role:

- `lexv2.amazonaws.com`

The AWSServiceRoleForLexV2Channels_ service linked role trusts the following service to assume the role:

- `channels.lexv2.amazonaws.com`

The AWSServiceRoleForLexV2Bots_ role allows Amazon Lex to complete the following actions on the specified resources:

- Action: Use Amazon Polly to synthesize speech on all Amazon Lex resources that the action supports.
- Action: If a bot is configured to use Amazon Comprehend sentiment analysis, detect sentiment on all Amazon Lex resources that the action supports.
- Action: If a bot is configured to store audio logs in an S3 bucket, put object in a specified bucket.
- Action: If a bot is configured to store audio and text logs, create a log stream in and put logs into a specified log group.
- Action: If a bot is configured to use a AWS KMS key to encrypt data, generate a specific data key.
- Action: If a bot is configured to use the `KendraSearchIntent` intent, query access to a specified Amazon Kendra index.

If a bot is configured to use a channel to communicate with a messaging service, the AWSServiceRoleForLexV2Channels_ role permissions policy allows Amazon Lex to complete the following actions on the specified resources:

- Action: List permissions on all bots in an account.
- Action: Recognize text, get session and put session permissions on a specified bot alias.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see [Service-Linked Role Permissions](#) in the *IAM User Guide*.

Creating a service-linked role for Amazon Lex

You don't need to manually create a service-linked role. When you create or modify or bot, or create a channel integration with a messaging service in the AWS Management Console, the AWS CLI, or the AWS API, Amazon Lex creates the service-linked role for you.

Amazon Lex creates a new service-linked role in your account for each bot. When you create a channel integration to deploy a bot on a messaging platform, Amazon Lex creates a new service-linked role in your account for each channel.

If you delete this service-linked role, and then need to create one again, you can use the same process to create a new role in your account. When you create or modify or bot, or create a channel integration with a messaging service, Amazon Lex creates the service-linked role with a new random suffix.

Editing a service-linked role for Amazon Lex

Amazon Lex does not allow you to edit the `AWSServiceRoleForLexV2Bots_` or `AWSServiceRoleForLexV2Channels_` service-linked roles. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. For `AWSServiceRoleForLexV2Bots_`, Amazon Lex modifies the role when you add additional capabilities to a bot. For example, if you add Amazon Comprehend sentiment analysis to a bot, Amazon Lex adds permission for the `DetectSentiment` action to `AWSServiceRoleForLexV2Bots_`. You can edit the description of either role using IAM. For more information, see [Editing a Service-Linked Role](#) in the *IAM User Guide*.

Deleting a service-linked role for Amazon Lex

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

Note

If the Amazon Lex service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete Amazon Lex resources used by `AWSServiceRoleForLexV2Bots_`

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. Choose to bot to delete.
3. From the left menu, choose **Settings**.
4. From **IAM permissions**, record the **Runtime role identifier**.
5. From the left menu, choose **Bots**.
6. From the list of bots, choose the radio button next to the bot to delete.
7. From the **Action**, choose **Delete**.

To delete Amazon Lex resources used by `AWSServiceRoleForLexV2Channels_`

1. Sign in to the AWS Management Console and open the Amazon Lex console at <https://console.aws.amazon.com/lexv2/>
2. Choose to bot to delete.
3. From the left menu, choose **Bot versions**, then choose **Channel integrations**.
4. Choose the channel to delete.
5. From **General configuration**, choose the choose the role name. The IAM management console opens in a new tab with the role chosen.
6. In the Amazon Lex console, choose **Delete**, then choose **Delete** again to delete the channel.
7. In the IAM management console, choose **DeleteRole** to remove the channel integration role.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForLexV2Bots_` or `AWSServiceRoleForLexV2Channels_` service-linked role. For more information, see [Deleting a Service-Linked Role](#) in the *IAM User Guide*.

Supported regions for Amazon Lex service-linked roles

Amazon Lex supports using service-linked roles in all of the regions where the service is available. For more information, see [AWS Regions and Endpoints](#).

Logging and monitoring in Amazon Lex

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Lex and your other AWS solutions. AWS provides the following monitoring tools to watch Amazon Lex, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the [Amazon CloudWatch User Guide](#).
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the [AWS CloudTrail User Guide](#).

Compliance validation for Amazon Lex

Third-party auditors assess the security and compliance of Amazon Lex as part of multiple AWS compliance programs. Amazon Lex is a HIPAA eligible service. It is PCI, SOC, and ISO compliant.

To learn whether Amazon Lex or other AWS services are in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

Note

Not all services are compliant with HIPAA.

- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon Lex

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

In addition to the AWS global infrastructure, Amazon Lex offers several features to help support your data resiliency and backup needs.

Infrastructure security in Amazon Lex

As a managed service, Amazon Lex is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access Amazon Lex through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Amazon Lex V1 to V2 migration guide

The Amazon Lex V2 console and APIs make it easier to build and manage bots. Use this guide to learn about the improvements in the Amazon Lex V2 API as you migrate bots.

You migrate a bot using the Amazon Lex V1 console or API. For more information see [Migrating a bot](#) in the *Amazon Lex developer guide*.

Amazon Lex V2 overview

Multiple languages can be added to a bot so you can manage them as a single resource. A simplified information architecture lets you efficiently manage your bot versions. Capabilities such as a 'conversation flow', partial saving of bot configuration and bulk upload of utterances give you more flexibility.

Multiple languages in a bot

You can add multiple languages with the Amazon Lex V2 API. You add, modify, and build each language independently. Resources such as slot types are scoped at the language level. You can quickly move between different languages to compare and refine the conversations. You can use one dashboard in the console to review utterances for all languages for faster analysis and iterations. A bot operator can manage permissions and logging operations for all languages with one bot configuration. You must provide a language as a runtime parameter to converse with a Amazon Lex V2 bot. For more information, see [Languages and locales supported by Amazon Lex \(p. 4\)](#).

Simplified information architecture

The Amazon Lex V2 API follows a simplified information architecture (IA) with intent and slot types scoped to a language. You version at the bot level so that resources such as intents and slot types aren't versioned individually. By default, a bot is created with a *Draft* version that is mutable and used for testing changes. You can create numbered snapshots from the draft version. You choose the languages to include in a version. All resources within the bot (languages, intents, slot types) are archived as part of creating a bot version. For more information, see [Creating versions \(p. 22\)](#).

Improved builder productivity

You have additional builder productivity tools and capabilities that give you more flexibility and control of your bot design process.

Save partial configuration

The Amazon Lex V2 API enables you to save partial changes during development. For example, you can save a slot that references a deleted slot type. This flexibility enables you to save your work and return to it later. You can resolve these changes before building the bot. In Amazon Lex the partial save can be applied to slots, versions, and aliases.

Renaming resources

With Amazon Lex V2 you can rename a resource after it's created. Use a resource name to associate user-friendly metadata with each resource. The Amazon Lex V2 API assigns every resource a unique 10-character resource ID. All resources have a resource name. You can rename the following resources:

- Bot
- Intent
- Slot type
- Slot
- Alias

You can use resource IDs to read and modify your resources. If you are using the AWS Command Line Interface or the Amazon Lex V2 API to work with Amazon Lex, resource IDs are required for certain commands.

Simplified management of Lambda functions

In the Amazon Lex V2 API you define one Lambda function per language instead of a function for each intent. The Lambda function is configured in the alias for the language and is used for both the dialog and fulfillment code hook. You can still choose to enable or disable the dialog and fulfillment code hooks independently for each intent. For more information, see [Using an AWS Lambda function \(p. 43\)](#).

Granular settings

The Amazon Lex V2 API moves the voice and intent classification confidence score threshold from the bot to the language scope. The sentiment analysis flag moves from bot scope to alias scope. Session time out and privacy settings at the bot scope, and conversation logs at the alias scope, remain unchanged.

Default fallback intent

The Amazon Lex V2 API adds a default fallback intent when you create a language. Use it to configure error handling for your bot instead of specific error-handling prompts.

Optimized session variable update

With the Amazon Lex V2 API you can update session state directly with the [RecognizeText \(p. 413\)](#) and [RecognizeUtterance \(p. 419\)](#) operations without any dependency on session APIs.

Guidelines and quotas

Topics

- [Regions \(p. 155\)](#)
- [General guidelines \(p. 155\)](#)
- [Quotas \(p. 156\)](#)

Regions

For a list of AWS Regions where Amazon Lex is available, see [AWS regions and endpoints](#) in the [AWS general reference](#).

General guidelines

This topic describes general guidelines when using Amazon Lex.

- **Signing requests** – All Amazon Lex model-building and runtime requests in the [API reference \(p. 161\)](#) use signature V4 for authenticating requests. For more information about authenticating requests, see [Signature Version 4 signing process](#) in the [Amazon Web Services general reference](#).
- Note the following about how Amazon Lex captures slot values from user utterances:

Amazon Lex uses the enumeration values that you provide in a slot type definition to train its machine learning models. Suppose that you define an intent called `GetPredictionIntent` with the following sample utterance:

```
"Tell me the prediction for {sign}"
```

where `{sign}` is a slot with the custom type `ZodiacSign`. It has 12 enumeration values, `Aries` through `Pisces`. From the user utterance "Tell me the prediction for ..." Amazon Lex understands that the following is a zodiac sign.

When the `valueSelectionStrategy` field is set to `OriginalValue` using the [CreateSlotType \(p. 224\)](#) operation, or if **Expand values** is selected in the console, if the user says "Tell me the prediction for earth", Amazon Lex infers that "earth" is a `ZodiacSign` value and passes it to your client application or Lambda function. You must check that slot values are valid values before using them in your fulfillment activity.

If you set the `valueSelectionStrategy` field to `TopResolution` using the `CreateSlotType` operation or if **Restrict to slot values and synonyms** is selected in the console, the values that are returned are limited to the values that you defined for the slot type. For example, if the user says "Tell me the prediction for earth", the value would not be recognized because it is not one of the values defined for the slot type. When you define synonyms for slot values, they are recognized the same as a slot value, however, the slot value is returned instead of the synonym.

When Amazon Lex calls a Lambda function or returns the result of a speech interaction with your client, the case of the slot values is not guaranteed. In text interactions, the case of the slot values matches the text entered or the slot value, depending on the value of the `valueResolutionStrategy` field.

- When defining slot values that contain acronyms, use the following patterns:
 - Capital letters separated by periods (D.V.D.)

- Capital letters separated by spaces (D V D)
- The [AMAZON.Date \(p. 36\)](#) and [AMAZON.Time \(p. 39\)](#) built-on slot types capture both absolute and relative dates and times. Relative dates and times are resolved in the region where Amazon Lex is processing the request.

For the `AMAZON.Time` built-in slot type, if the user doesn't specify that a time is before or after noon, the time is ambiguous and Amazon Lex will prompt the user again. We recommend prompts that elicit an absolute time. For example, use a prompt such as "When do you want your pizza delivered? You can say 6 PM or 6 in the evening."

- Providing confusable training data in your bot reduces the ability of Amazon Lex to understand user input. Consider these examples:

Suppose you have two intents (`OrderPizza` and `OrderDrink`) in your bot and both are configured with an "I want to order" utterance. This utterance does not map to a specific intent that Amazon Lex can learn from while building the language model for the bot at build time. As a result, when a user inputs this utterance at runtime, Amazon Lex can't pick an intent with a high degree of confidence.

When you have two intents with the same utterance, use input contexts to help Amazon Lex distinguish between the two intents at runtime. For more information, see [Setting intent context](#).

- When you use the `TSTALIASID` alias, keep in mind the following:
 - The `TSTALIASID` alias of your bot points to the Draft version and should only be used for manual testing. Amazon Lex limits the number of runtime requests that you can make to the `TSTALIASID` alias of the bot.
 - When you update the Draft version of the bot, Amazon Lex terminates any in-progress conversations for any client application using the `TSTALIASID` alias of the bot. Generally, you should not use the `TSTALIASID` alias of a bot in production because the Draft version can be updated. You should publish a version and an alias and use them instead.
 - When you update an alias, Amazon Lex takes a few minutes to pick up the changes. When you modify the Draft version of the bot, the change is picked up by the `TSTALIASID` alias immediately.
 - The runtime API operations [RecognizeText \(p. 413\)](#) and [RecognizeUtterance \(p. 419\)](#) take a session ID as a required parameter. Developers can set this to any value that meets the constraints described in the API. We recommend that you don't use this parameter to send any confidential information such as user logins, emails, or social security numbers. This ID is primarily used to uniquely identify a conversation with a bot.

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources for your AWS account. For more information, see [AWS service quotas](#) in the [AWS general reference](#).

Service quotas can be adjusted or increased. Contact AWS customer support to increase a quota. It can take a few days to increase a service quota. If you're increasing your quota as part of a larger project, be sure to add this time to your plan.

Build-time quotas

The following maximum quotas are enforced when you are creating a bot.

Description	Default
Bots per AWS account	100
Bot channel associations per AWS account	5,000

Description	Default
Versions per bot	100
Intents per locale in each bot	<ul style="list-style-type: none"> • 1,000 in en-AU, en-GB, and en-US • 100 in all other locales
Slots per locale in each bot	<ul style="list-style-type: none"> • 4,000 in en-AU, en-GB, and en-US • 2,000 in all other locales
Custom slot types per bot locale	<ul style="list-style-type: none"> • 250 in en-AU, en-GB, and en-US • 100 in all other locales
Custom slot type values and synonyms per locale in each bot	50,000
Total characters in sample utterances per locale in each bot	<ul style="list-style-type: none"> • 2,000,000 in en-AU, en-GB, and en-US • 200,000 in all other locales
Channel associations per bot alias	10
Slots per intent	100
Sample utterances per intent	1,500
Characters per sample utterance	500
Text response length	4,000
Sample utterances per slot	10
Characters per sample slot utterance	500
Prompts per slot	30
Values and synonyms per custom slot type	10,000
Characters per custom slot type value	500
Characters in a channel association name	100

Runtime quotas

The following maximum quotas are enforced at runtime.

Description	Default
Input text size for RecognizeText (p. 413) and RecognizeUtterance (p. 419)	1024 Unicode characters
Speech input length for <code>RecognizeUtterance</code> operation	15 seconds
Size of <code>RecognizeUtterance</code> headers	16 KB
Size of combined request and session headers for <code>RecognizeUtterance</code>	12 KB

Description	Default
Maximum number of concurrent conversations for <code>RecognizeText</code> , <code>RecognizeUtterance</code> , or <code>StartConversation</code> for the <code>TestBotAlias</code>	2
Maximum number of concurrent conversations for <code>RecognizeText</code> , <code>RecognizeUtterance</code> , or <code>StartConversation</code> for other aliases	25
Maximum number of concurrent session management operations (<code>PutSession</code> , <code>GetSession</code> , or <code>DeleteSession</code>) when using the <code>TestBotAlias</code>	2
Maximum number of concurrent session management operations (<code>PutSession</code> , <code>GetSession</code> , or <code>DeleteSession</code>) when using other aliases	25
Maximum input size to a Lambda function	12 KB
Maximum output size of a Lambda function	25 KB
Maximum size of session attributes in Lambda function output	12 KB

Document history for Amazon Lex

- **Latest documentation update:** July 27, 2021

The following table describes important changes in each release of Amazon Lex. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
New feature	Amazon Lex now provides a built-in slot type for UK postal codes. For more information, see AMAZON.UKPostalCode .	July 27, 2021
New feature	Amazon Lex now supports the English (Indian) locale. For more information, see Languages and locales supported by Amazon Lex .	July 15, 2021
New feature	Amazon Lex now provides a tool to migrate a bot from Amazon Lex V1 to the Amazon Lex V2 API. For more information, see Migrating a bot in the Amazon Lex Developer Guide .	July 13, 2021
New feature	Amazon Lex now enables you to accept multiple values for a single slot in the English (US) language. For more information, see Using multiple values in a slot .	June 15, 2021
New feature	You can now build larger bots for English languages. For more information, see Quotas .	June 11, 2021
New feature	Use Amazon Lex resource-based policies to manage access to your bots and bot aliases. For more information, see Resource-based policies within Amazon Lex .	May 20, 2021
New feature	Amazon Lex now enables you to import and export bots and bot locales. You can use this feature to copy bots and bot locales between accounts and AWS Regions. For more information, see Importing and exporting bots .	May 18, 2021

Region expansion	Amazon Lex is now available in Canada (Central) (ca-central-1).	May 17, 2021
New feature	Amazon Lex now supports the Japanese (Japan) locale. For more information, see Languages and locales supported by Amazon Lex .	April 1, 2021
New feature	Amazon Lex now supports three new built-in slot types: AMAZON.City, AMAZON.Country, and AMAZON.State.	March 12, 2021
New guide (p. 159)	This is the first release of the <i>Amazon Lex V2 user guide</i> .	January 21, 2021

API reference

This section contains the API reference documentation. See the left menu for a list of actions and data types.

Note

If you use a custom HTTP client to call Amazon Lex Model Building V2 operations, you must set the "Content-Type" HTTP header to "application/x-amz-json-1.1". Otherwise, you receive an HTTP 404 - UnknownOperationException in the response.

Amazon Lex Model Building V2 operations return the responses with the "application/x-amz-json-1.1" content type.

Actions

The following actions are supported by Amazon Lex Model Building V2:

- [BuildBotLocale \(p. 165\)](#)
- [CreateBot \(p. 168\)](#)
- [CreateBotAlias \(p. 173\)](#)
- [CreateBotLocale \(p. 179\)](#)
- [CreateBotVersion \(p. 184\)](#)
- [CreateExport \(p. 188\)](#)
- [CreateIntent \(p. 192\)](#)
- [CreateResourcePolicy \(p. 205\)](#)
- [CreateResourcePolicyStatement \(p. 208\)](#)
- [CreateSlot \(p. 212\)](#)
- [CreateSlotType \(p. 224\)](#)
- [CreateUploadUrl \(p. 229\)](#)
- [DeleteBot \(p. 231\)](#)
- [DeleteBotAlias \(p. 234\)](#)
- [DeleteBotLocale \(p. 237\)](#)
- [DeleteBotVersion \(p. 240\)](#)
- [DeleteExport \(p. 243\)](#)
- [DeleteImport \(p. 245\)](#)
- [DeleteIntent \(p. 247\)](#)
- [DeleteResourcePolicy \(p. 249\)](#)
- [DeleteResourcePolicyStatement \(p. 251\)](#)
- [DeleteSlot \(p. 254\)](#)
- [DeleteSlotType \(p. 257\)](#)
- [DescribeBot \(p. 259\)](#)
- [DescribeBotAlias \(p. 262\)](#)
- [DescribeBotLocale \(p. 266\)](#)
- [DescribeBotVersion \(p. 270\)](#)
- [DescribeExport \(p. 274\)](#)
- [DescribeImport \(p. 277\)](#)

- [DescribeIntent \(p. 280\)](#)
- [DescribeResourcePolicy \(p. 287\)](#)
- [DescribeSlot \(p. 289\)](#)
- [DescribeSlotType \(p. 297\)](#)
- [ListBotAliases \(p. 301\)](#)
- [ListBotLocales \(p. 304\)](#)
- [ListBots \(p. 308\)](#)
- [ListBotVersions \(p. 311\)](#)
- [ListBuiltInIntents \(p. 314\)](#)
- [ListBuiltInSlotTypes \(p. 317\)](#)
- [ListExports \(p. 320\)](#)
- [ListImports \(p. 324\)](#)
- [ListIntents \(p. 328\)](#)
- [ListSlots \(p. 332\)](#)
- [ListSlotTypes \(p. 337\)](#)
- [ListTagsForResource \(p. 341\)](#)
- [StartImport \(p. 343\)](#)
- [TagResource \(p. 347\)](#)
- [UntagResource \(p. 349\)](#)
- [UpdateBot \(p. 351\)](#)
- [UpdateBotAlias \(p. 355\)](#)
- [UpdateBotLocale \(p. 361\)](#)
- [UpdateExport \(p. 365\)](#)
- [UpdateIntent \(p. 368\)](#)
- [UpdateResourcePolicy \(p. 380\)](#)
- [UpdateSlot \(p. 383\)](#)
- [UpdateSlotType \(p. 396\)](#)

The following actions are supported by Amazon Lex Runtime V2:

- [DeleteSession \(p. 401\)](#)
- [GetSession \(p. 404\)](#)
- [PutSession \(p. 408\)](#)
- [RecognizeText \(p. 413\)](#)
- [RecognizeUtterance \(p. 419\)](#)
- [StartConversation \(p. 426\)](#)

Amazon Lex Model Building V2

The following actions are supported by Amazon Lex Model Building V2:

- [BuildBotLocale \(p. 165\)](#)
- [CreateBot \(p. 168\)](#)
- [CreateBotAlias \(p. 173\)](#)
- [CreateBotLocale \(p. 179\)](#)
- [CreateBotVersion \(p. 184\)](#)

- [CreateExport \(p. 188\)](#)
- [CreateIntent \(p. 192\)](#)
- [CreateResourcePolicy \(p. 205\)](#)
- [CreateResourcePolicyStatement \(p. 208\)](#)
- [CreateSlot \(p. 212\)](#)
- [CreateSlotType \(p. 224\)](#)
- [CreateUploadUrl \(p. 229\)](#)
- [DeleteBot \(p. 231\)](#)
- [DeleteBotAlias \(p. 234\)](#)
- [DeleteBotLocale \(p. 237\)](#)
- [DeleteBotVersion \(p. 240\)](#)
- [DeleteExport \(p. 243\)](#)
- [DeleteImport \(p. 245\)](#)
- [DeleteIntent \(p. 247\)](#)
- [DeleteResourcePolicy \(p. 249\)](#)
- [DeleteResourcePolicyStatement \(p. 251\)](#)
- [DeleteSlot \(p. 254\)](#)
- [DeleteSlotType \(p. 257\)](#)
- [DescribeBot \(p. 259\)](#)
- [DescribeBotAlias \(p. 262\)](#)
- [DescribeBotLocale \(p. 266\)](#)
- [DescribeBotVersion \(p. 270\)](#)
- [DescribeExport \(p. 274\)](#)
- [DescribeImport \(p. 277\)](#)
- [DescribeIntent \(p. 280\)](#)
- [DescribeResourcePolicy \(p. 287\)](#)
- [DescribeSlot \(p. 289\)](#)
- [DescribeSlotType \(p. 297\)](#)
- [ListBotAliases \(p. 301\)](#)
- [ListBotLocales \(p. 304\)](#)
- [ListBots \(p. 308\)](#)
- [ListBotVersions \(p. 311\)](#)
- [ListBuiltInIntents \(p. 314\)](#)
- [ListBuiltInSlotTypes \(p. 317\)](#)
- [ListExports \(p. 320\)](#)
- [ListImports \(p. 324\)](#)
- [ListIntents \(p. 328\)](#)
- [ListSlots \(p. 332\)](#)
- [ListSlotTypes \(p. 337\)](#)
- [ListTagsForResource \(p. 341\)](#)
- [StartImport \(p. 343\)](#)
- [TagResource \(p. 347\)](#)
- [UntagResource \(p. 349\)](#)
- [UpdateBot \(p. 351\)](#)
- [UpdateBotAlias \(p. 355\)](#)
- [UpdateBotLocale \(p. 361\)](#)

- [UpdateExport \(p. 365\)](#)
- [UpdateIntent \(p. 368\)](#)
- [UpdateResourcePolicy \(p. 380\)](#)
- [UpdateSlot \(p. 383\)](#)
- [UpdateSlotType \(p. 396\)](#)

BuildBotLocale

Service: Amazon Lex Model Building V2

Builds a bot, its intents, and its slot types into a specific locale. A bot can be built into multiple locales. At runtime the locale is used to choose a specific build of the bot.

Request Syntax

```
POST /bots/botId/botversions/botVersion/botlocales/localeId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 165)

The identifier of the bot to build. The identifier is returned in the response from the [CreateBot](#) (p. 168) operation.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 165)

The version of the bot to build. This can only be the draft version of the bot.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId (p. 165)

The identifier of the language and locale that the bot will be used in. The string must match one of the supported locales. All of the intents, slot types, and slots used in the bot must have the same locale. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botId": "string",
    "botLocaleStatus": "string",
    "botVersion": "string",
    "lastBuildSubmittedDateTime": number,
    "localeId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 165)

The identifier of the specified bot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botLocaleStatus (p. 165)

The bot's build status. When the status is ReadyExpressTesting you can test the bot using the utterances defined for the intents and slot types. When the status is Built, the bot is ready for use and can be tested using any utterance.

Type: String

Valid Values: Creating | Building | Built | ReadyExpressTesting | Failed | Deleting | NotBuilt | Importing

botVersion (p. 165)

The version of the bot that was built. This is only the draft version of the bot.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

lastBuildSubmittedDateTime (p. 165)

A timestamp indicating the date and time that the bot was last built for this locale.

Type: Timestamp

localeId (p. 165)

The language and locale specified of where the bot can be used.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateBot

Service: Amazon Lex Model Building V2

Creates an Amazon Lex conversational bot.

Request Syntax

```
PUT /bots/ HTTP/1.1
Content-type: application/json

{
    "botName": "string",
    "botTags": {
        "string" : "string"
    },
    "dataPrivacy": {
        "childDirected": boolean
    },
    "description": "string",
    "idleSessionTTLInSeconds": number,
    "roleArn": "string",
    "testBotAliasTags": {
        "string" : "string"
    }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

[botName \(p. 168\)](#)

The name of the bot. The bot name must be unique in the account that creates the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

[botTags \(p. 168\)](#)

A list of tags to add to the bot. You can only add tags when you create a bot. You can't use the `UpdateBot` operation to update tags. To update tags, use the `TagResource` operation.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Required: No

[dataPrivacy \(p. 168\)](#)

Provides information on additional privacy protections Amazon Lex should use with the bot's data.

Type: [DataPrivacy \(p. 471\)](#) object

Required: Yes

[description \(p. 168\)](#)

A description of the bot. It appears in lists to help you identify a particular bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[idleSessionTTLInSeconds \(p. 168\)](#)

The time, in seconds, that Amazon Lex should keep information about a user's conversation with the bot.

A user interaction remains active for the amount of time specified. If no conversation occurs during this time, the session expires and Amazon Lex deletes any data provided before the timeout.

You can specify between 60 (1 minute) and 86,400 (24 hours) seconds.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

Required: Yes

[roleArn \(p. 168\)](#)

The Amazon Resource Name (ARN) of an IAM role that has permission to access the bot.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:iam::[0-9]{12}:role/.+\$

Required: Yes

[testBotAliasTags \(p. 168\)](#)

A list of tags to add to the test alias for a bot. You can only add tags when you create a bot. You can't use the `UpdateAlias` operation to update tags. To update tags on the test alias, use the `TagResource` operation.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Required: No

Response Syntax

```
HTTP/1.1 202
```

```
Content-type: application/json

{
    "botId": "string",
    "botName": "string",
    "botStatus": "string",
    "botTags": {
        "string" : "string"
    },
    "creationDateTime": number,
    "dataPrivacy": {
        "childDirected": boolean
    },
    "description": "string",
    "idleSessionTTLInSeconds": number,
    "roleArn": "string",
    "testBotAliasTags": {
        "string" : "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 169)

A unique identifier for a particular bot. You use this to identify the bot when you call other Amazon Lex API operations.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botName (p. 169)

The name specified for the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

botStatus (p. 169)

Shows the current status of the bot. The bot is first in the `Creating` status. Once the bot is ready for use, it changes to the `Available` status. After the bot is created, you can use the `Draft` version of the bot.

Type: String

Valid Values: `Creating` | `Available` | `Inactive` | `Deleting` | `Failed` | `Versioning` | `Importing`

botTags (p. 169)

A list of tags associated with the bot.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

creationDateTime (p. 169)

A timestamp indicating the date and time that the bot was created.

Type: Timestamp

dataPrivacy (p. 169)

The data privacy settings specified for the bot.

Type: [DataPrivacy \(p. 471\)](#) object

description (p. 169)

The description specified for the bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

idleSessionTTLInSeconds (p. 169)

The session idle time specified for the bot.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

roleArn (p. 169)

The IAM role specified for the bot.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:iam::[0-9]{12}:role/.+\$

testBotAliasTags (p. 169)

A list of tags associated with the test alias for the bot.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateBotAlias

Service: Amazon Lex Model Building V2

Creates an alias for the specified version of a bot. Use an alias to enable you to change the version of a bot without updating applications that use the bot.

For example, you can create an alias called "PROD" that your applications use to call the Amazon Lex bot.

Request Syntax

```
PUT /bots/botId/botaliases/ HTTP/1.1
Content-type: application/json

{
    "botAliasLocaleSettings": {
        "string": {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "string",
                    "lambdaARN": "string"
                }
            },
            "enabled": boolean
        }
    },
    "botAliasName": "string",
    "botVersion": "string",
    "conversationLogSettings": {
        "audioLogSettings": [
            {
                "destination": {
                    "s3Bucket": {
                        "kmsKeyArn": "string",
                        "logPrefix": "string",
                        "s3BucketArn": "string"
                    }
                },
                "enabled": boolean
            }
        ],
        "textLogSettings": [
            {
                "destination": {
                    "cloudWatch": {
                        "cloudWatchLogGroupArn": "string",
                        "logPrefix": "string"
                    }
                },
                "enabled": boolean
            }
        ]
    },
    "description": "string",
    "sentimentAnalysisSettings": {
        "detectSentiment": boolean
    },
    "tags": {
        "string": "string"
    }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 173)

The unique identifier of the bot that the alias applies to.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

botAliasLocaleSettings (p. 173)

Maps configuration information to a specific locale. You can use this parameter to specify a specific Lambda function to run different functions in different locales.

Type: String to [BotAliasLocaleSettings \(p. 440\)](#) object map

Map Entries: Maximum number of items.

Required: No

botAliasName (p. 173)

The alias to create. The name must be unique for the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

botVersion (p. 173)

The version of the bot that this alias points to. You can use the [UpdateBotAlias \(p. 355\)](#) operation to change the bot version associated with the alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Required: No

conversationLogSettings (p. 173)

Specifies whether Amazon Lex logs text and audio for a conversation with the bot. When you enable conversation logs, text logs store text input, transcripts of audio input, and associated metadata in Amazon CloudWatch Logs. Audio logs store audio input in Amazon S3.

Type: [ConversationLogSettings \(p. 469\)](#) object

Required: No

[description \(p. 173\)](#)

A description of the alias. Use this description to help identify the alias.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[sentimentAnalysisSettings \(p. 173\)](#)

Determines whether Amazon Lex will use Amazon Comprehend to detect the sentiment of user utterances.

Type: [SentimentAnalysisSettings \(p. 507\)](#) object

Required: No

[tags \(p. 173\)](#)

A list of tags to add to the bot alias. You can only add tags when you create an alias, you can't use the `UpdateBotAlias` operation to update the tags on a bot alias. To update tags, use the `TagResource` operation.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botAliasId": "string",
    "botAliasLocaleSettings": {
        "string" : {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "string",
                    "lambdaARN": "string"
                }
            },
            "enabled": boolean
        }
    },
    "botAliasName": "string",
    "botAliasStatus": "string",
    "botId": "string",
    "botVersion": "string",
    "conversationLogSettings": {
        "audioLogSettings": [
            {
                "destination": {

```

```
        "s3Bucket": {
            "kmsKeyArn": "string",
            "logPrefix": "string",
            "s3BucketArn": "string"
        }
    },
    "enabled": boolean
}
],
"textLogSettings": [
{
    "destination": {
        "cloudWatch": {
            "cloudWatchLogGroupArn": "string",
            "logPrefix": "string"
        }
    },
    "enabled": boolean
}
]
},
"creationDateTime": number,
"description": "string",
"sentimentAnalysisSettings": {
    "detectSentiment": boolean
},
"tags": {
    "string" : "string"
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[botAliasId \(p. 175\)](#)

The unique identifier of the bot alias.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

[botAliasLocaleSettings \(p. 175\)](#)

Configuration information for a specific locale.

Type: String to [BotAliasLocaleSettings \(p. 440\)](#) object map

Map Entries: Maximum number of items.

[botAliasName \(p. 175\)](#)

The name specified for the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[botAliasStatus \(p. 175\)](#)

The current status of the alias. The alias is first put into the `Creating` state. When the alias is ready to be used, it is put into the `Available` state. You can use the `DescribeBotAlias` operation to get the current state of an alias.

Type: String

Valid Values: `Creating` | `Available` | `Deleting` | `Failed`

[botId \(p. 175\)](#)

The unique identifier of the bot that this alias applies to.

Type: String

Length Constraints: Fixed length of 10.

Pattern: `^[0-9a-zA-Z]+$`

[botVersion \(p. 175\)](#)

The version of the bot associated with this alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: `^[0-9]+$`

[conversationLogSettings \(p. 175\)](#)

The conversation log settings specified for the alias.

Type: [ConversationLogSettings \(p. 469\)](#) object

[creationDateTime \(p. 175\)](#)

A Unix timestamp indicating the date and time that the bot alias was created.

Type: Timestamp

[description \(p. 175\)](#)

The description specified for the bot alias.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[sentimentAnalysisSettings \(p. 175\)](#)

Determines whether Amazon Lex will use Amazon Comprehend to detect the sentiment of user utterances.

Type: [SentimentAnalysisSettings \(p. 507\)](#) object

[tags \(p. 175\)](#)

A list of tags associated with the bot alias.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateBotLocale

Service: Amazon Lex Model Building V2

Creates a locale in the bot. The locale contains the intents and slot types that the bot uses in conversations with users in the specified language and locale. You must add a locale to a bot before you can add intents and slot types to the bot.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/ HTTP/1.1
Content-type: application/json

{
  "description": "string",
  "localeId": "string",
  "nluIntentConfidenceThreshold": number,
  "voiceSettings": {
    "voiceId": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 179)

The identifier of the bot to create the locale for.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 179)

The version of the bot to create the locale for. This can only be the draft version of the bot.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

description (p. 179)

A description of the bot locale. Use this to help identify the bot locale in lists.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[localeId \(p. 179\)](#)

The identifier of the language and locale that the bot will be used in. The string must match one of the supported locales. All of the intents, slot types, and slots used in the bot must have the same locale. For more information, see [Supported languages](#).

Type: String

Required: Yes

[nluIntentConfidenceThreshold \(p. 179\)](#)

Determines the threshold where Amazon Lex will insert the `AMAZON.FallbackIntent`, `AMAZON.KendraSearchIntent`, or both when returning alternative intents. `AMAZON.FallbackIntent` and `AMAZON.KendraSearchIntent` are only inserted if they are configured for the bot.

For example, suppose a bot is configured with the confidence threshold of 0.80 and the `AMAZON.FallbackIntent`. Amazon Lex returns three alternative intents with the following confidence scores: IntentA (0.70), IntentB (0.60), IntentC (0.50). The response from the PostText operation would be:

- `AMAZON.FallbackIntent`
- IntentA
- IntentB
- IntentC

Type: Double

Valid Range: Minimum value of 0. Maximum value of 1.

Required: Yes

[voiceSettings \(p. 179\)](#)

The Amazon Polly voice ID that Amazon Lex uses for voice interaction with the user.

Type: [VoiceSettings \(p. 528\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
  "botId": "string",
  "botLocaleStatus": "string",
  "botVersion": "string",
  "creationDateTime": number,
  "description": "string",
  "localeId": "string",
  "localeName": "string",
  "nluIntentConfidenceThreshold": number,
  "voiceSettings": {
    "voiceId": "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 180)

The specified bot identifier.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botLocaleStatus (p. 180)

The status of the bot.

When the status is `Creating` the bot locale is being configured. When the status is `Building` Amazon Lex is building the bot for testing and use.

If the status of the bot is `ReadyExpressTesting`, you can test the bot using the exact utterances specified in the bots' intents. When the bot is ready for full testing or to run, the status is `Built`.

If there was a problem with building the bot, the status is `Failed`. If the bot was saved but not built, the status is `NotBuilt`.

Type: String

Valid Values: `Creating` | `Building` | `Built` | `ReadyExpressTesting` | `Failed` | `Deleting` | `NotBuilt` | `Importing`

botVersion (p. 180)

The specified bot version.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

creationDateTime (p. 180)

A timestamp specifying the date and time that the bot locale was created.

Type: Timestamp

description (p. 180)

The specified description of the bot locale.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

localeId (p. 180)

The specified locale identifier.

Type: String

localeName (p. 180)

The specified locale name.

Type: String

nluIntentConfidenceThreshold (p. 180)

The specified confidence threshold for inserting the AMAZON.FallbackIntent and AMAZON.KendraSearchIntent intents.

Type: Double

Valid Range: Minimum value of 0. Maximum value of 1.

voiceSettings (p. 180)

The Amazon Polly voice ID that Amazon Lex uses for voice interaction with the user.

Type: [VoiceSettings \(p. 528\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateBotVersion

Service: Amazon Lex Model Building V2

Creates a new version of the bot based on the `DRAFT` version. If the `DRAFT` version of this resource hasn't changed since you created the last version, Amazon Lex doesn't create a new version, it returns the last created version.

When you create the first version of a bot, Amazon Lex sets the version to 1. Subsequent versions increment by 1.

Request Syntax

```
PUT /bots/botId/botversions/ HTTP/1.1
Content-type: application/json

{
  "botVersionLocaleSpecification": {
    "string": {
      "sourceBotVersion": "string"
    }
  },
  "descriptionstring"
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 184)

The identifier of the bot to create the version for.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

botVersionLocaleSpecification (p. 184)

Specifies the locales that Amazon Lex adds to this version. You can choose the `Draft` version or any other previously published version for each locale. When you specify a source version, the locale data is copied from the source version to the new version.

Type: String to [BotVersionLocaleDetails](#) (p. 458) object map

Map Entries: Maximum number of items.

Required: Yes

description (p. 184)

A description of the version. Use the description to help identify the version in lists.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botId": "string",
    "botStatus": "string",
    "botVersion": "string",
    "botVersionLocaleSpecification": {
        "string" : {
            "sourceBotVersion": "string"
        }
    },
    "creationDateTime": number,
    "description": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 185)

The bot identifier specified in the request.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botStatus (p. 185)

When you send a request to create or update a bot, Amazon Lex sets the status response element to `Creating`. After Amazon Lex builds the bot, it sets status to `Available`. If Amazon Lex can't build the bot, it sets status to `Failed`.

Type: String

Valid Values: `Creating` | `Available` | `Inactive` | `Deleting` | `Failed` | `Versioning` | `Importing`

botVersion (p. 185)

The version number assigned to the version.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

botVersionLocaleSpecification (p. 185)

The source versions used for each locale in the new version.

Type: String to [BotVersionLocaleDetails \(p. 458\)](#) object map

Map Entries: Maximum number of items.

creationDateTime (p. 185)

A timestamp of the date and time that the version was created.

Type: Timestamp

description (p. 185)

The description of the version specified in the request.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateExport

Service: Amazon Lex Model Building V2

Creates a zip archive containing the contents of a bot or a bot locale. The archive contains a directory structure that contains JSON files that define the bot.

You can create an archive that contains the complete definition of a bot, or you can specify that the archive contain only the definition of a single bot locale.

For more information about exporting bots, and about the structure of the export archive, see [Importing and exporting bots](#)

Request Syntax

```
PUT /exports/ HTTP/1.1
Content-type: application/json

{
    "fileFormat": "string",
    "filePassword": "string",
    "resourceSpecification": {
        "botExportSpecification": {
            "botId": "string",
            "botVersion": "string"
        },
        "botLocaleExportSpecification": {
            "botId": "string",
            "botVersion": "string",
            "localeId": "string"
        }
    }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

[fileFormat \(p. 188\)](#)

The file format of the bot or bot locale definition files.

Type: String

Valid Values: `LexJson`

Required: Yes

[filePassword \(p. 188\)](#)

An password to use to encrypt the exported archive. Using a password is optional, but you should encrypt the archive to protect the data in transit between Amazon Lex and your local computer.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

[resourceSpecification \(p. 188\)](#)

Specifies the type of resource to export, either a bot or a bot locale. You can only specify one type of resource to export.

Type: [ExportResourceSpecification \(p. 474\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
  "creationDateTime": number,
  "exportId": "string",
  "exportStatus": "string",
  "fileFormat": "string",
  "resourceSpecification": {
    "botExportSpecification": {
      "botId": "string",
      "botVersion": "string"
    },
    "botLocaleExportSpecification": {
      "botId": "string",
      "botVersion": "string",
      "localeId": "string"
    }
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[creationDateTime \(p. 189\)](#)

The date and time that the request to export a bot was created.

Type: Timestamp

[exportId \(p. 189\)](#)

An identifier for a specific request to create an export.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[exportStatus \(p. 189\)](#)

The status of the export. When the status is `Completed`, you can use the [DescribeExport \(p. 274\)](#) operation to get the pre-signed S3 URL link to your exported bot or bot locale.

Type: String

Valid Values: `InProgress` | `Completed` | `Failed` | `Deleting`

[fileFormat \(p. 189\)](#)

The file format used for the bot or bot locale definition files.

Type: String

Valid Values: `LexJson`

[resourceSpecification \(p. 189\)](#)

A description of the type of resource that was exported, either a bot or a bot locale.

Type: [ExportResourceSpecification \(p. 474\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateIntent

Service: Amazon Lex Model Building V2

Creates an intent.

To define the interaction between the user and your bot, you define one or more intents. For example, for a pizza ordering bot you would create an `OrderPizza` intent.

When you create an intent, you must provide a name. You can optionally provide the following:

- Sample utterances. For example, "I want to order a pizza" and "Can I order a pizza." You can't provide utterances for built-in intents.
- Information to be gathered. You specify slots for the information that your bot requests from the user. You can specify standard slot types, such as date and time, or custom slot types for your application.
- How the intent is fulfilled. You can provide a Lambda function or configure the intent to return the intent information to your client application. If you use a Lambda function, Amazon Lex invokes the function when all of the intent information is available.
- A confirmation prompt to send to the user to confirm an intent. For example, "Shall I order your pizza?"
- A conclusion statement to send to the user after the intent is fulfilled. For example, "I ordered your pizza."
- A follow-up prompt that asks the user for additional activity. For example, "Do you want a drink with your pizza?"

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/intents/ HTTP/1.1
Content-type: application/json
```

```
{
  "description": "string",
  "dialogCodeHook": {
    "enabled": boolean
  },
  "fulfillmentCodeHook": {
    "enabled": boolean
  },
  "inputContexts": [
    {
      "name": "string"
    }
  ],
  "intentClosingSetting": {
    "closingResponse": {
      "allowInterrupt": boolean,
      "messageGroups": [
        {
          "message": {
            "customPayload": {
              "value": "string"
            },
            "imageResponseCard": {
              "buttons": [
                {
                  "text": "string",
                  "value": "string"
                }
              ],
              "imageUrl": "string",
              "title": "string"
            }
          }
        }
      ]
    }
  }
}
```

```

        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
},
"variations": [
{
    "customPayload": {
        "value": "string"
    },
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
},
"intentConfirmationSetting": {
    "declinationResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
{
        "message": {
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            },
            "plainTextMessage": {
                "value": "string"
            },
            "ssmlMessage": {
                "value": "string"
            }
        },
        "variations": [

```

```
{
    "customPayload": {
        "value": "string"
    },
    "imageResponseCard": [
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
},
"promptSpecification": {
    "allowInterrupt": boolean,
    "maxRetries": number,
    "messageGroups": [
        {
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": [
                    "buttons": [
                        {
                            "text": "string",
                            "value": "string"
                        }
                    ],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
                },
                "plainTextMessage": {
                    "value": "string"
                },
                "ssmlMessage": {
                    "value": "string"
                }
            },
            "variations": [
                {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": [
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    }
                }
            ]
        }
    ]
}
```

```
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
}
},
"intentName": "string",
"kendraConfiguration": {
    "kendraIndex": "string",
    "queryFilterString": "string",
    "queryFilterStringEnabled": boolean
},
"outputContexts": [
{
    "name": "string",
    "timeToLiveInSeconds": number,
    "turnsToLive": number
}
],
"parentIntentSignature": "string",
"sampleUtterances": [
{
    "utterance": "string"
}
]
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 192)

The identifier of the bot associated with this intent.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 192)

The identifier of the version of the bot associated with this intent.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId (p. 192)

The identifier of the language and locale where this intent is used. All of the bots, slot types, and slots used by the intent must have the same locale. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

[description \(p. 192\)](#)

A description of the intent. Use the description to help identify the intent in lists.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[dialogCodeHook \(p. 192\)](#)

Specifies that Amazon Lex invokes the alias Lambda function for each user input. You can invoke this Lambda function to personalize user interaction.

For example, suppose that your bot determines that the user's name is John. You Lambda function might retrieve John's information from a backend database and prepopulate some of the values. For example, if you find that John is gluten intolerant, you might set the corresponding intent slot, `glutenIntolerant` to `true`. You might find John's phone number and set the corresponding session attribute.

Type: [DialogCodeHookSettings \(p. 472\)](#) object

Required: No

[fulfillmentCodeHook \(p. 192\)](#)

Specifies that Amazon Lex invokes the alias Lambda function when the intent is ready for fulfillment. You can invoke this function to complete the bot's transaction with the user.

For example, in a pizza ordering bot, the Lambda function can look up the closest pizza restaurant to the customer's location and then place an order on the customer's behalf.

Type: [FulfillmentCodeHookSettings \(p. 478\)](#) object

Required: No

[inputContexts \(p. 192\)](#)

A list of contexts that must be active for this intent to be considered by Amazon Lex.

When an intent has an input context list, Amazon Lex only considers using the intent in an interaction with the user when the specified contexts are included in the active context list for the session. If the contexts are not active, then Amazon Lex will not use the intent.

A context can be automatically activated using the `outputContexts` property or it can be set at runtime.

For example, if there are two intents with different input contexts that respond to the same utterances, only the intent with the active context will respond.

An intent may have up to 5 input contexts. If an intent has multiple input contexts, all of the contexts must be active to consider the intent.

Type: Array of [InputContext \(p. 486\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

Required: No

[intentClosingSetting \(p. 192\)](#)

Sets the response that Amazon Lex sends to the user when the intent is closed.

Type: [IntentClosingSetting \(p. 487\)](#) object

Required: No

[intentConfirmationSetting \(p. 192\)](#)

Provides prompts that Amazon Lex sends to the user to confirm the completion of an intent. If the user answers "no," the settings contain a statement that is sent to the user to end the intent.

Type: [IntentConfirmationSetting \(p. 488\)](#) object

Required: No

[intentName \(p. 192\)](#)

The name of the intent. Intent names must be unique in the locale that contains the intent and cannot match the name of any built-in intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

[kendraConfiguration \(p. 192\)](#)

Configuration information required to use the AMAZON.KendraSearchIntent intent to connect to an Amazon Kendra index. The AMAZON.KendraSearchIntent intent is called when Amazon Lex can't determine another intent to invoke.

Type: [KendraConfiguration \(p. 493\)](#) object

Required: No

[outputContexts \(p. 192\)](#)

A lists of contexts that the intent activates when it is fulfilled.

You can use an output context to indicate the intents that Amazon Lex should consider for the next turn of the conversation with a customer.

When you use the `outputContextsList` property, all of the contexts specified in the list are activated when the intent is fulfilled. You can set up to 10 output contexts. You can also set the number of conversation turns that the context should be active, or the length of time that the context should be active.

Type: Array of [OutputContext \(p. 499\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

[parentIntentSignature \(p. 192\)](#)

A unique identifier for the built-in intent to base this intent on.

Type: String

Required: No

[sampleUtterances \(p. 192\)](#)

An array of strings that a user might say to signal the intent. For example, "I want a pizza", or "I want a {PizzaSize} pizza".

In an utterance, slot names are enclosed in curly braces ("{"}, "}") to indicate where they should be displayed in the utterance shown to the user..

Type: Array of [SampleUtterance \(p. 505\)](#) objects

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botVersion": "string",
  "creationDateTime": number,
  "description": "string",
  "dialogCodeHook": {
    "enabled": boolean
  },
  "fulfillmentCodeHook": {
    "enabled": boolean
  },
  "inputContexts": [
    {
      "name": "string"
    }
  ],
  "intentClosingSetting": {
    "closingResponse": {
      "allowInterrupt": boolean,
      "messageGroups": [
        {
          "message": {
            "customPayload": {
              "value": "string"
            },
            "imageResponseCard": {
              "buttons": [
                {
                  "text": "string",
                  "value": "string"
                }
              ],
              "imageUrl": "string",
              "subtitle": "string",
              "title": "string"
            }
          },
          "plainTextMessage": {
            "value": "string"
          },
          "ssmlMessage": {
            "value": "string"
          }
        },
        "variations": [
          {
            "text": "string"
          }
        ]
      ]
    }
  }
}
```

```

    "customPayload": {
      "value": "string"
    },
    "imageResponseCard": {
      "buttons": [
        {
          "text": "string",
          "value": "string"
        }
      ],
      "imageUrl": "string",
      "subtitle": "string",
      "title": "string"
    },
    "plainTextMessage": {
      "value": "string"
    },
    "ssmlMessage": {
      "value": "string"
    }
  }
}

],
},
"intentConfirmationSetting": {
  "declinationResponse": {
    "allowInterrupt": boolean,
    "messageGroups": [
      {
        "message": {
          "customPayload": {
            "value": "string"
          },
          "imageResponseCard": {
            "buttons": [
              {
                "text": "string",
                "value": "string"
              }
            ],
            "imageUrl": "string",
            "subtitle": "string",
            "title": "string"
          },
          "plainTextMessage": {
            "value": "string"
          },
          "ssmlMessage": {
            "value": "string"
          }
        },
        "variations": [
          {
            "customPayload": {
              "value": "string"
            },
            "imageResponseCard": {
              "buttons": [
                {
                  "text": "string",
                  "value": "string"
                }
              ],
              "imageUrl": "string",
              "subtitle": "string",
              "title": "string"
            }
          }
        ]
      }
    ]
  }
}

```

```
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
],
},
"promptSpecification": {
    "allowInterrupt": boolean,
    "maxRetries": number,
    "messageGroups": [
        {
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": {
                    "buttons": [
                        {
                            "text": "string",
                            "value": "string"
                        }
                    ],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
                },
                "plainTextMessage": {
                    "value": "string"
                },
                "ssmlMessage": {
                    "value": "string"
                }
            },
            "variations": [
                {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                }
            ]
        }
    ]
}
```

```
        ]
    },
    "intendId": "string",
    "intentName": "string",
    "kendraConfiguration": {
        "kendraIndex": "string",
        "queryFilterString": "string",
        "queryFilterStringEnabled": boolean
    },
    "localeId": "string",
    "outputContexts": [
        {
            "name": "string",
            "timeToLiveInSeconds": number,
            "turnsToLive": number
        }
    ],
    "parentIntentSignature": "string",
    "sampleUtterances": [
        {
            "utterance": "string"
        }
    ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 198)

The identifier of the bot associated with the intent.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 198)

The identifier of the version of the bot associated with the intent.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

creationDateTime (p. 198)

A timestamp of the date and time that the intent was created.

Type: Timestamp

description (p. 198)

The description specified for the intent.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[dialogCodeHook \(p. 198\)](#)

The dialog Lambda function specified for the intent.

Type: [DialogCodeHookSettings \(p. 472\)](#) object

[fulfillmentCodeHook \(p. 198\)](#)

The fulfillment Lambda function specified for the intent.

Type: [FulfillmentCodeHookSettings \(p. 478\)](#) object

[inputContexts \(p. 198\)](#)

The list of input contexts specified for the intent.

Type: Array of [InputContext \(p. 486\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

[intentClosingSetting \(p. 198\)](#)

The closing setting specified for the intent.

Type: [IntentClosingSetting \(p. 487\)](#) object

[intentConfirmationSetting \(p. 198\)](#)

The confirmation setting specified for the intent.

Type: [IntentConfirmationSetting \(p. 488\)](#) object

[intentId \(p. 198\)](#)

A unique identifier for the intent.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[intentName \(p. 198\)](#)

The name specified for the intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[kendraConfiguration \(p. 198\)](#)

Configuration for searching a Amazon Kendra index specified for the intent.

Type: [KendraConfiguration \(p. 493\)](#) object

[localeId \(p. 198\)](#)

The locale that the intent is specified to use.

Type: String

[outputContexts \(p. 198\)](#)

The list of output contexts specified for the intent.

Type: Array of [OutputContext \(p. 499\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

parentIntentSignature (p. 198)

The signature of the parent intent specified for the intent.

Type: String

sampleUtterances (p. 198)

The sample utterances specified for the intent.

Type: Array of [SampleUtterance \(p. 505\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

CreateResourcePolicy

Service: Amazon Lex Model Building V2

Creates a new resource policy with the specified policy statements.

Request Syntax

```
POST /policy/resourceArn HTTP/1.1
Content-type: application/json

{
    "policy": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

resourceArn (p. 205)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request accepts the following data in JSON format.

policy (p. 205)

A resource policy to add to the resource. The policy is a JSON structure that contains one or more statements that define the policy. The policy must follow the IAM syntax. For more information about the contents of a JSON policy document, see [IAM JSON policy reference](#).

If the policy isn't valid, Amazon Lex returns a validation exception.

Type: String

Length Constraints: Minimum length of 2.

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "resourceArn": "string",
    "revisionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[resourceArn \(p. 205\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy was attached to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1011.

[revisionId \(p. 205\)](#)

The current revision of the resource policy. Use the revision ID to make sure that you are updating the most current version of a resource policy when you add a policy statement to a resource, delete a resource, or update a resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateResourcePolicyStatement

Service: Amazon Lex Model Building V2

Adds a new resource policy statement to a bot or bot alias. If a resource policy exists, the statement is added to the current resource policy. If a policy doesn't exist, a new policy is created.

You can't create a resource policy statement that allows cross-account access.

Request Syntax

```
POST /policy/resourceArn/statements/?expectedRevisionId=expectedRevisionId HTTP/1.1
Content-type: application/json

{
  "action": [ "string" ],
  "condition": {
    "string" : {
      "string" : "string"
    }
  },
  "effect": "string",
  "principal": [
    {
      "arn": "string",
      "service": "string"
    }
  ],
  "statementId": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

[expectedRevisionId \(p. 208\)](#)

The identifier of the revision of the policy to edit. If this revision ID doesn't match the current revision ID, Amazon Lex throws an exception.

If you don't specify a revision, Amazon Lex overwrites the contents of the policy with the new values.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

[resourceArn \(p. 208\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request accepts the following data in JSON format.

[action \(p. 208\)](#)

The Amazon Lex action that this policy either allows or denies. The action must apply to the resource type of the specified ARN. For more information, see [Actions, resources, and condition keys for Amazon Lex V2](#).

Type: Array of strings

Length Constraints: Minimum length of 5. Maximum length of 50.

Pattern: `lex:[a-zA-Z*]+$`

Required: Yes

[condition \(p. 208\)](#)

Specifies a condition when the policy is in effect. If the principal of the policy is a service principal, you must provide two condition blocks, one with a `SourceAccount` global condition key and one with a `SourceArn` global condition key.

For more information, see [IAM JSON policy elements: Condition](#).

Type: String to string to string map map

Map Entries: Minimum number of 0 items. Maximum number of 10 items.

Key Length Constraints: Minimum length of 1.

Map Entries: Minimum number of 0 items. Maximum number of 10 items.

Key Length Constraints: Minimum length of 1. Maximum length of 1024.

Value Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

[effect \(p. 208\)](#)

Determines whether the statement allows or denies access to the resource.

Type: String

Valid Values: Allow | Deny

Required: Yes

[principal \(p. 208\)](#)

An IAM principal, such as an IAM users, IAM roles, or AWS services that is allowed or denied access to a resource. For more information, see [AWS JSON policy elements: Principal](#).

Type: Array of [Principal \(p. 501\)](#) objects

Required: Yes

[statementId \(p. 208\)](#)

The name of the statement. The ID is the same as the `Sid` IAM property. The statement name must be unique within the policy. For more information, see [IAM JSON policy elements: Sid](#).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

Required: Yes

[Response Syntax](#)

HTTP/1.1 200

```
Content-type: application/json

{
  "resourceArn": "string",
  "revisionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[resourceArn \(p. 209\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1011.

[revisionId \(p. 209\)](#)

The current revision of the resource policy. Use the revision ID to make sure that you are updating the most current version of a resource policy when you add a policy statement to a resource, delete a resource, or update a resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateSlot

Service: Amazon Lex Model Building V2

Creates a slot in an intent. A slot is a variable needed to fulfill an intent. For example, an `OrderPizza` intent might need slots for size, crust, and number of pizzas. For each slot, you define one or more utterances that Amazon Lex uses to elicit a response from the user.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId/slots/ HTTP/1.1
Content-type: application/json

{
  "description": "string",
  "multipleValuesSetting": {
    "allowMultipleValues": boolean
  },
  "obfuscationSetting": {
    "obfuscationSettingType": "string"
  },
  "slotName": "string",
  "slotTypeId": "string",
  "valueElicitationSetting": {
    "defaultValueSpecification": {
      "defaultValueList": [
        {
          "defaultValue": "string"
        }
      ]
    },
    "promptSpecification": {
      "allowInterrupt": boolean,
      "maxRetries": number,
      "messageGroups": [
        {
          "message": {
            "customPayload": {
              "value": "string"
            },
            "imageResponseCard": {
              "buttons": [
                {
                  "text": "string",
                  "value": "string"
                }
              ],
              "imageUrl": "string",
              "subtitle": "string",
              "title": "string"
            },
            "plainTextMessage": {
              "value": "string"
            },
            "ssmlMessage": {
              "value": "string"
            }
          },
          "variations": [
            {
              "customPayload": {
                "value": "string"
              },
              "imageResponseCard": {
                "buttons": [

```

```
        {
            "text": "string",
            "value": "string"
        }
    ],
    "imageUrl": "string",
    "subtitle": "string",
    "title": "string"
},
"plainTextMessage": {
    "value": "string"
},
"ssmlMessage": {
    "value": "string"
}
}
]
}
],
"sampleUtterances": [
{
    "utterance": "string"
}
],
"slotConstraint": "string",
"waitAndContinueSpecification": {
    "continueResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        }
                    }
                ]
            }
        ]
    }
}
```

```
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
],
},
"stillWaitingResponse": {
    "allowInterrupt": boolean,
    "frequencyInSeconds": number,
    "messageGroups": [
        {
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": {
                    "buttons": [
                        {
                            "text": "string",
                            "value": "string"
                        }
                    ],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
                },
                "plainTextMessage": {
                    "value": "string"
                },
                "ssmlMessage": {
                    "value": "string"
                }
            },
            "variations": [
                {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                }
            ]
        }
    ]
}
```

```
        ],
        "timeoutInSeconds": number
    },
    "waitingResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        }
    }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 212)

The identifier of the bot associated with the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 212)

The version of the bot associated with the slot.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

intentId (p. 212)

The identifier of the intent that contains the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 212)

The identifier of the language and locale that the slot will be used in. The string must match one of the supported locales. All of the bots, intents, slot types used by the slot must have the same locale. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

description (p. 212)

A description of the slot. Use this to help identify the slot in lists.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

multipleValuesSetting (p. 212)

Indicates whether the slot returns multiple values in one response. Multi-value slots are only available in the en-US locale. If you set this value to `true` in any other locale, Amazon Lex throws a `ValidationException`.

If the `multipleValuesSetting` is not set, the default value is `false`.

Type: [MultipleValuesSetting \(p. 497\)](#) object

Required: No

[obfuscationSetting \(p. 212\)](#)

Determines how slot values are used in Amazon CloudWatch logs. If the value of the `obfuscationSetting` parameter is `DefaultObfuscation`, slot values are obfuscated in the log output. If the value is `None`, the actual value is present in the log output.

The default is to obfuscate values in the CloudWatch logs.

Type: [ObfuscationSetting \(p. 498\)](#) object

Required: No

[slotName \(p. 212\)](#)

The name of the slot. Slot names must be unique within the bot that contains the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

Required: Yes

[slotTypeId \(p. 212\)](#)

The unique identifier for the slot type associated with this slot. The slot type determines the values that can be entered into the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: `^(AMAZON\.)[a-zA-Z_]+?|[0-9a-zA-Z]+)$`

Required: Yes

[valueElicitationSetting \(p. 212\)](#)

Specifies prompts that Amazon Lex sends to the user to elicit a response that provides the value for the slot.

Type: [SlotValueElicitationSetting \(p. 520\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botVersion": "string",
  "creationDateTime": number,
  "description": "string",
  "intentId": "string",
  "localeId": "string",
  "multipleValuesSetting": {
    "allowMultipleValues": boolean
  },
  "obfuscationSetting": {
    "obfuscationSettingType": "string"
  },
}
```

```
"slotId": "string",
"slotName": "string",
"slotTypeId": "string",
"valueElicitationSetting": {
    "defaultValueSpecification": {
        "defaultValueList": [
            {
                "defaultValue": "string"
            }
        ]
    },
    "promptSpecification": {
        "allowInterrupt": boolean,
        "maxRetries": number,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        ]
    },
    "sampleUtterances": [
```

```

        {
            "utterance": "string"
        }
    ],
    "slotConstraint": "string",
    "waitAndContinueSpecification": {
        "continueResponse": {
            "allowInterrupt": boolean,
            "messageGroups": [
                {
                    "message": {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    },
                    "variations": [
                        {
                            "customPayload": {
                                "value": "string"
                            },
                            "imageResponseCard": {
                                "buttons": [
                                    {
                                        "text": "string",
                                        "value": "string"
                                    }
                                ],
                                "imageUrl": "string",
                                "subtitle": "string",
                                "title": "string"
                            },
                            "plainTextMessage": {
                                "value": "string"
                            },
                            "ssmlMessage": {
                                "value": "string"
                            }
                        }
                    ]
                }
            ],
            "stillWaitingResponse": {
                "allowInterrupt": boolean,
                "frequencyInSeconds": number,
                "messageGroups": [
                    {
                        "message": {
                            "customPayload": {

```

```
        "value": "string"
    },
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
},
"variations": [
    {
        "customPayload": {
            "value": "string"
        },
        "imageResponseCard": {
            "buttons": [
                {
                    "text": "string",
                    "value": "string"
                }
            ],
            "imageUrl": "string",
            "subtitle": "string",
            "title": "string"
        },
        "plainTextMessage": {
            "value": "string"
        },
        "ssmlMessage": {
            "value": "string"
        }
    }
]
],
"timeoutInSeconds": number
},
"waitingResponse": {
    "allowInterrupt": boolean,
    "messageGroups": [
        {
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": {
                    "buttons": [
                        {
                            "text": "string",
                            "value": "string"
                        }
                    ],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
                }
            }
        }
    ]
}
```

```
        },
        "plainTextMessage": {
            "value": "string"
        },
        "ssmlMessage": {
            "value": "string"
        }
    },
    "variations": [
        {
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            },
            "plainTextMessage": {
                "value": "string"
            },
            "ssmlMessage": {
                "value": "string"
            }
        }
    ]
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 217)

The unique identifier of the bot associated with the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 217)

The version of the bot associated with the slot.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

[creationDateTime \(p. 217\)](#)

The timestamp of the date and time that the slot was created.

Type: Timestamp

[description \(p. 217\)](#)

The description associated with the slot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[intentId \(p. 217\)](#)

The unique identifier of the intent associated with the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[localeId \(p. 217\)](#)

The language and local specified for the slot.

Type: String

[multipleValuesSetting \(p. 217\)](#)

Indicates whether the slot returns multiple values in one response.

Type: [MultipleValuesSetting \(p. 497\)](#) object

[obfuscationSetting \(p. 217\)](#)

Indicates whether the slot is configured to obfuscate values in Amazon CloudWatch logs.

Type: [ObfuscationSetting \(p. 498\)](#) object

[slotId \(p. 217\)](#)

The unique identifier associated with the slot. Use this to identify the slot when you update or delete it.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[slotName \(p. 217\)](#)

The name specified for the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[slotTypeId \(p. 217\)](#)

The unique identifier of the slot type associated with this slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: ^((AMAZON\.)|[a-zA-Z_]+?)|[0-9a-zA-Z]+)\$

[valueElicitationSetting \(p. 217\)](#)

The value elicitation settings specified for the slot.

Type: [SlotValueElicitationSetting \(p. 520\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateSlotType

Service: Amazon Lex Model Building V2

Creates a custom slot type

To create a custom slot type, specify a name for the slot type and a set of enumeration values, the values that a slot of this type can assume.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/slottypes/ HTTP/1.1
Content-type: application/json

{
  "description": "string",
  "parentSlotTypeSignature": "string",
  "slotTypeName": "string",
  "slotTypeValues": [
    {
      "sampleValue": {
        "value": "string"
      },
      "synonyms": [
        {
          "value": "string"
        }
      ]
    }
  ],
  "valueSelectionSetting": {
    "regexFilter": {
      "pattern": "string"
    },
    "resolutionStrategy": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 224)

The identifier of the bot associated with this slot type.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 224)

The identifier of the bot version associated with this slot type.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

[localeId \(p. 224\)](#)

The identifier of the language and locale that the slot type will be used in. The string must match one of the supported locales. All of the bots, intents, and slots used by the slot type must have the same locale. For more information, see [Supported languages](#).

Required: Yes

[Request Body](#)

The request accepts the following data in JSON format.

[description \(p. 224\)](#)

A description of the slot type. Use the description to help identify the slot type in lists.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[parentSlotTypeSignature \(p. 224\)](#)

The built-in slot type used as a parent of this slot type. When you define a parent slot type, the new slot type has the configuration of the parent slot type.

Only `AMAZON.AlphaNumeric` is supported.

Type: String

Required: No

[slotTypeName \(p. 224\)](#)

The name for the slot. A slot type name must be unique within the account.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

Required: Yes

[slotTypeValues \(p. 224\)](#)

A list of `SlotTypeValue` objects that defines the values that the slot type can take. Each value can have a list of synonyms, additional values that help train the machine learning model about the values that it resolves for a slot.

Type: Array of [SlotTypeValue \(p. 519\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10000 items.

Required: No

[valueSelectionSetting \(p. 224\)](#)

Determines the strategy that Amazon Lex uses to select a value from the list of possible values. The field can be set to one of the following values:

- `OriginalValue` - Returns the value entered by the user, if the user value is similar to the slot value.

- **TopResolution** - If there is a resolution list for the slot, return the first value in the resolution list. If there is no resolution list, return null.

If you don't specify the `valueSelectionSetting` parameter, the default is `originalValue`.

Type: [SlotValueSelectionSetting \(p. 523\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botVersion": "string",
  "creationDateTime": number,
  "description": "string",
  "localeId": "string",
  "parentSlotTypeSignature": "string",
  "slotTypeId": "string",
  "slotTypeName": "string",
  "slotTypeValues": [
    {
      "sampleValue": {
        "value": "string"
      },
      "synonyms": [
        {
          "value": "string"
        }
      ]
    }
  ],
  "valueSelectionSetting": {
    "regexFilter": {
      "pattern": "string"
    },
    "resolutionStrategy": "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 226)

The identifier for the bot associated with the slot type.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 226)

The version of the bot associated with the slot type.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

[creationDateTime \(p. 226\)](#)

A timestamp of the date and time that the slot type was created.

Type: Timestamp

[description \(p. 226\)](#)

The description specified for the slot type.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[localeId \(p. 226\)](#)

The specified language and local specified for the slot type.

Type: String

[parentSlotTypeSignature \(p. 226\)](#)

The signature of the base slot type specified for the slot type.

Type: String

[slotTypeId \(p. 226\)](#)

The unique identifier assigned to the slot type. Use this to identify the slot type in the `UpdateSlotType` and `DeleteSlotType` operations.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[slotTypeName \(p. 226\)](#)

The name specified for the slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[slotTypeValues \(p. 226\)](#)

The list of values that the slot type can assume.

Type: Array of [SlotTypeValue \(p. 519\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10000 items.

[valueSelectionSetting \(p. 226\)](#)

The strategy that Amazon Lex uses to select a value from the list of possible values.

Type: [SlotValueSelectionSetting \(p. 523\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateUploadUrl

Service: Amazon Lex Model Building V2

Gets a pre-signed S3 write URL that you use to upload the zip archive when importing a bot or a bot locale.

Request Syntax

```
POST /createuploadurl/ HTTP/1.1
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "importId": "string",
    "uploadUrl": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

importId (p. 229)

An identifier for a unique import job. Use it when you call the [StartImport](#) (p. 343) operation.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

uploadUrl (p. 229)

A pre-signed S3 write URL. Upload the zip archive file that contains the definition of your bot or bot locale.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 567).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteBot

Service: Amazon Lex Model Building V2

Deletes all versions of a bot, including the Draft version. To delete a specific version, use the `DeleteBotVersion` operation.

When you delete a bot, all of the resources contained in the bot are also deleted. Deleting a bot removes all locales, intents, slot, and slot types defined for the bot.

If a bot has an alias, the `DeleteBot` operation returns a `ResourceInUseException` exception. If you want to delete the bot and the alias, set the `skipResourceInUseCheck` parameter to `true`.

Request Syntax

```
DELETE /bots/botId?skipResourceInUseCheck=skipResourceInUseCheck HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 231)

The identifier of the bot to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

skipResourceInUseCheck (p. 231)

When `true`, Amazon Lex doesn't check to see if another resource, such as an alias, is using the bot before it is deleted.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
  "botId": "string",
  "botStatus": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 231)

The unique identifier of the bot that Amazon Lex is deleting.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botStatus \(p. 231\)](#)

The current status of the bot. The status is `Deleting` while the bot and its associated resources are being deleted.

Type: String

Valid Values: `Creating` | `Available` | `Inactive` | `Deleting` | `Failed` | `Versioning` | `Importing`

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

DeleteBotAlias

Service: Amazon Lex Model Building V2

Deletes the specified bot alias.

Request Syntax

```
DELETE /bots/botId/botaliases/botAliasId?skipResourceInUseCheck=skipResourceInUseCheck
HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 234)

The unique identifier of the bot alias to delete.

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

Required: Yes

botId (p. 234)

The unique identifier of the bot associated with the alias to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

skipResourceInUseCheck (p. 234)

When this parameter is true, Amazon Lex doesn't check to see if any other resource is using the alias before it is deleted.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
  "botAliasId": "string",
  "botAliasStatus": "string",
  "botId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botAliasId (p. 234)

The unique identifier of the bot alias to delete.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

botAliasStatus (p. 234)

The current status of the alias. The status is `Deleting` while the alias is in the process of being deleted. Once the alias is deleted, it will no longer appear in the list of aliases returned by the `ListBotAliases` operation.

Type: String

Valid Values: `Creating` | `Available` | `Deleting` | `Failed`

botId (p. 234)

The unique identifier of the bot that contains the alias to delete.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteBotLocale

Service: Amazon Lex Model Building V2

Removes a locale from a bot.

When you delete a locale, all intents, slots, and slot types defined for the locale are also deleted.

Request Syntax

```
DELETE /bots/botId/botversions/botVersion/botlocales/localeId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 237)

The unique identifier of the bot that contains the locale.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 237)

The version of the bot that contains the locale.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId (p. 237)

The identifier of the language and locale that will be deleted. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botId": "string",
    "botLocaleStatus": "string",
    "botVersion": "string",
    "localeId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 237)

The identifier of the bot that contained the deleted locale.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botLocaleStatus (p. 237)

The status of deleting the bot locale. The locale first enters the Deleting status. Once the locale is deleted it no longer appears in the list of locales for the bot.

Type: String

Valid Values: Creating | Building | Built | ReadyExpressTesting | Failed | Deleting | NotBuilt | Importing

botVersion (p. 237)

The version of the bot that contained the deleted locale.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

localeId (p. 237)

The language and locale of the deleted locale.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteBotVersion

Service: Amazon Lex Model Building V2

Deletes a specific version of a bot. To delete all version of a bot, use the [DeleteBot \(p. 231\)](#) operation.

Request Syntax

```
DELETE /bots/botId/botversions/botVersion?skipResourceInUseCheck=skipResourceInUseCheck
HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

[botId \(p. 240\)](#)

The identifier of the bot that contains the version.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

[botVersion \(p. 240\)](#)

The version of the bot to delete.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Required: Yes

[skipResourceInUseCheck \(p. 240\)](#)

By default, the `DeleteBotVersion` operations throws a `ResourceInUseException` exception if you try to delete a bot version that has an alias pointing at it. Set the `skipResourceInUseCheck` parameter to `true` to skip this check and remove the version even if an alias points to it.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
  "botId": "string",
  "botStatus": "string",
  "botVersion": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 240)

The identifier of the bot that is being deleted.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botStatus (p. 240)

The current status of the bot.

Type: String

Valid Values: Creating | Available | Inactive | Deleting | Failed | Versioning | Importing

botVersion (p. 240)

The version of the bot that is being deleted.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteExport

Service: Amazon Lex Model Building V2

Removes a previous export and the associated files stored in an S3 bucket.

Request Syntax

```
DELETE /exports/exportId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

exportId (p. 243)

The unique identifier of the export to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "exportId": "string",
    "exportStatus": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

exportId (p. 243)

The unique identifier of the deleted export.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

exportStatus (p. 243)

The current status of the deletion. When the deletion is complete, the export will no longer be returned by the [ListExports](#) (p. 320) operation and calls to the [DescribeExport](#) (p. 274) with the export identifier will fail.

Type: String

Valid Values: `InProgress` | `Completed` | `Failed` | `Deleting`

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteImport

Service: Amazon Lex Model Building V2

Removes a previous import and the associated file stored in an S3 bucket.

Request Syntax

```
DELETE /imports/importId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

importId (p. 245)

The unique identifier of the import to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "importId": "string",
    "importStatus": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

importId (p. 245)

The unique identifier of the deleted import.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

importStatus (p. 245)

The current status of the deletion. When the deletion is complete, the import will no longer be returned by the [ListImports \(p. 324\)](#) operation and calls to the [DescribeImport \(p. 277\)](#) with the import identifier will fail.

Type: String

Valid Values: `InProgress` | `Completed` | `Failed` | `Deleting`

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteIntent

Service: Amazon Lex Model Building V2

Removes the specified intent.

Deleting an intent also deletes the slots associated with the intent.

Request Syntax

```
DELETE /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 247)

The identifier of the bot associated with the intent.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 247)

The version of the bot associated with the intent.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

intentId (p. 247)

The unique identifier of the intent to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 247)

The identifier of the language and locale where the bot will be deleted. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteResourcePolicy

Service: Amazon Lex Model Building V2

Removes an existing policy from a bot or bot alias. If the resource doesn't have a policy attached, Amazon Lex returns an exception.

Request Syntax

```
DELETE /policy/resourceArn?expectedRevisionId=expectedRevisionId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

[expectedRevisionId \(p. 249\)](#)

The identifier of the revision to edit. If this ID doesn't match the current revision number, Amazon Lex returns an exception

If you don't specify a revision ID, Amazon Lex will delete the current policy.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

[resourceArn \(p. 249\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that has the resource policy attached.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
Content-type: application/json

{
  "resourceArn": "string",
  "revisionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response.

The following data is returned in JSON format by the service.

[resourceArn \(p. 249\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy was deleted from.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1011.

revisionId (p. 249)

The current revision of the resource policy. Use the revision ID to make sure that you are updating the most current version of a resource policy when you add a policy statement to a resource, delete a resource, or update a resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteResourcePolicyStatement

Service: Amazon Lex Model Building V2

Deletes a policy statement from a resource policy. If you delete the last statement from a policy, the policy is deleted. If you specify a statement ID that doesn't exist in the policy, or if the bot or bot alias doesn't have a policy attached, Amazon Lex returns an exception.

Request Syntax

```
DELETE /policy/resourceArn/statements/statementId?expectedRevisionId=expectedRevisionId
HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

[expectedRevisionId \(p. 251\)](#)

The identifier of the revision of the policy to delete the statement from. If this revision ID doesn't match the current revision ID, Amazon Lex throws an exception.

If you don't specify a revision, Amazon Lex removes the current contents of the statement.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

[resourceArn \(p. 251\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

[statementId \(p. 251\)](#)

The name of the statement (SID) to delete from the policy.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
Content-type: application/json

{
    "resourceArn": "string",
    "revisionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response.

The following data is returned in JSON format by the service.

resourceArn ([p. 251](#))

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy statement was removed from.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1011.

revisionId ([p. 251](#))

The current revision of the resource policy. Use the revision ID to make sure that you are updating the most current version of a resource policy when you add a policy statement to a resource, delete a resource, or update a resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteSlot

Service: Amazon Lex Model Building V2

Deletes the specified slot from an intent.

Request Syntax

```
DELETE /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId/  
slots/slotId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 254)

The identifier of the bot associated with the slot to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 254)

The version of the bot associated with the slot to delete.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

intentId (p. 254)

The identifier of the intent associated with the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 254)

The identifier of the language and locale that the slot will be deleted from. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

slotId (p. 254)

The identifier of the slot to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteSlotType

Service: Amazon Lex Model Building V2

Deletes a slot type from a bot locale.

If a slot is using the slot type, Amazon Lex throws a `ResourceInUseException` exception. To avoid the exception, set the `skipResourceInUseCheck` parameter to true.

Request Syntax

```
DELETE /bots/botId/botversions/botVersion/botlocales/localeId/slottypes/slotTypeId?  
skipResourceInUseCheck=skipResourceInUseCheck HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 257)

The identifier of the bot associated with the slot type.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 257)

The version of the bot associated with the slot type.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId (p. 257)

The identifier of the language and locale that the slot type will be deleted from. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

skipResourceInUseCheck (p. 257)

By default, the `DeleteSlotType` operations throws a `ResourceInUseException` exception if you try to delete a slot type used by a slot. Set the `skipResourceInUseCheck` parameter to true to skip this check and remove the slot type even if a slot uses it.

slotTypeId (p. 257)

The identifier of the slot type to delete.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeBot

Service: Amazon Lex Model Building V2

Provides metadata information about a bot.

Request Syntax

```
GET /bots/botId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 259)

The unique identifier of the bot to describe.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botName": "string",
  "botStatus": "string",
  "creationDateTime": number,
  "dataPrivacy": {
    "childDirected": boolean
  },
  "description": "string",
  "idleSessionTTLInSeconds": number,
  "lastUpdatedDateTime": number,
  "roleArn": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 259)

The unique identifier of the bot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botName (p. 259)

The name of the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

botStatus (p. 259)

The current status of the bot. When the status is Available the bot is ready to be used in conversations with users.

Type: String

Valid Values: Creating | Available | Inactive | Deleting | Failed | Versioning | Importing

creationDateTime (p. 259)

A timestamp of the date and time that the bot was created.

Type: Timestamp

dataPrivacy (p. 259)

Settings for managing data privacy of the bot and its conversations with users.

Type: [DataPrivacy \(p. 471\)](#) object

description (p. 259)

The description of the bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

idleSessionTTLInSeconds (p. 259)

The maximum time in seconds that Amazon Lex retains the data gathered in a conversation.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

lastUpdatedDateTime (p. 259)

A timestamp of the date and time that the bot was last updated.

Type: Timestamp

roleArn (p. 259)

The Amazon Resource Name (ARN) of an IAM role that has permission to access the bot.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:iam::[0-9]{12}:role/.*\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeBotAlias

Service: Amazon Lex Model Building V2

Get information about a specific bot alias.

Request Syntax

```
GET /bots/botId/botaliases/botAliasId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 262)

The identifier of the bot alias to describe.

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

Required: Yes

botId (p. 262)

The identifier of the bot associated with the bot alias to describe.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botAliasHistoryEvents": [
        {
            "botVersion": "string",
            "endDate": number,
            "startDate": number
        }
    ],
    "botAliasId": "string",
    "botAliasLocaleSettings": {
        "string": {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "string",
                    "lambdaARN": "string"
                }
            },
            "enabled": boolean
        }
    }
}
```

```
        },
        "botAliasName": "string",
        "botAliasStatus": "string",
        "botId": "string",
        "botVersion": "string",
        "conversationLogSettings": {
            "audioLogSettings": [
                {
                    "destination": {
                        "s3Bucket": {
                            "kmsKeyArn": "string",
                            "logPrefix": "string",
                            "s3BucketArn": "string"
                        }
                    },
                    "enabled": boolean
                }
            ],
            "textLogSettings": [
                {
                    "destination": {
                        "cloudWatch": {
                            "cloudWatchLogGroupArn": "string",
                            "logPrefix": "string"
                        }
                    },
                    "enabled": boolean
                }
            ]
        },
        "creationDateTime": number,
        "description": "string",
        "lastUpdatedDateTime": number,
        "sentimentAnalysisSettings": {
            "detectSentiment": boolean
        }
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botAliasHistoryEvents (p. 262)

A list of events that affect a bot alias. For example, an event is recorded when the version that the alias points to changes.

Type: Array of [BotAliasHistoryEvent \(p. 439\)](#) objects

botAliasId (p. 262)

The identifier of the bot alias.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

botAliasLocaleSettings (p. 262)

The locale settings that are unique to the alias.

Type: String to [BotAliasLocaleSettings \(p. 440\)](#) object map

Map Entries: Maximum number of items.

botAliasName (p. 262)

The name of the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

botAliasStatus (p. 262)

The current status of the alias. When the alias is Available, the alias is ready for use with your bot.

Type: String

Valid Values: Creating | Available | Deleting | Failed

botId (p. 262)

The identifier of the bot associated with the bot alias.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 262)

The version of the bot associated with the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

conversationLogSettings (p. 262)

Specifics of how Amazon Lex logs text and audio conversations with the bot associated with the alias.

Type: [ConversationLogSettings \(p. 469\)](#) object

creationDateTime (p. 262)

A timestamp of the date and time that the alias was created.

Type: Timestamp

description (p. 262)

The description of the bot alias.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

lastUpdatedDateTime (p. 262)

A timestamp of the date and time that the alias was last updated.

Type: Timestamp

[sentimentAnalysisSettings \(p. 262\)](#)

Determines whether Amazon Lex will use Amazon Comprehend to detect the sentiment of user utterances.

Type: [SentimentAnalysisSettings \(p. 507\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeBotLocale

Service: Amazon Lex Model Building V2

Describes the settings that a bot has for a specific locale.

Request Syntax

```
GET /bots/botId/botversions/botVersion/botlocales/localeId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 266)

The identifier of the bot associated with the locale.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 266)

The identifier of the version of the bot associated with the locale.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9])+\$

Required: Yes

localeId (p. 266)

The unique identifier of the locale to describe. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botLocaleHistoryEvents": [
        {
            "event": "string",
            "eventDate": number
        }
    ],
    "botLocaleStatus": "string",
    "botVersion": "string",
    "creationDateTime": number,
```

```
"description": "string",
"failureReasons": [ "string" ],
"intentsCount": number,
"lastBuildSubmittedDateTime": number,
"lastUpdatedDateTime": number,
"localeId": "string",
"localeName": "string",
"nluIntentConfidenceThreshold": number,
"slotTypesCount": number,
"voiceSettings": {
    "voiceId": "string"
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[botId \(p. 266\)](#)

The identifier of the bot associated with the locale.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botLocaleHistoryEvents \(p. 266\)](#)

History of changes, such as when a locale is used in an alias, that have taken place for the locale.

Type: Array of [BotLocaleHistoryEvent \(p. 449\)](#) objects

[botLocaleStatus \(p. 266\)](#)

The status of the bot. If the status is Failed, the reasons for the failure are listed in the failureReasons field.

Type: String

Valid Values: Creating | Building | Built | ReadyExpressTesting | Failed | Deleting | NotBuilt | Importing

[botVersion \(p. 266\)](#)

The identifier of the version of the bot associated with the locale.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

[creationDateTime \(p. 266\)](#)

The date and time that the locale was created.

Type: Timestamp

[description \(p. 266\)](#)

The description of the locale.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[failureReasons \(p. 266\)](#)

If `botLocaleStatus` is Failed, Amazon Lex explains why it failed to build the bot.

Type: Array of strings

[intentsCount \(p. 266\)](#)

The number of intents defined for the locale.

Type: Integer

[lastBuildSubmittedDateTime \(p. 266\)](#)

The date and time that the locale was last submitted for building.

Type: Timestamp

[lastUpdatedDateTime \(p. 266\)](#)

The date and time that the locale was last updated.

Type: Timestamp

[localeId \(p. 266\)](#)

The unique identifier of the described locale.

Type: String

[localeName \(p. 266\)](#)

The name of the locale.

Type: String

[nluIntentConfidenceThreshold \(p. 266\)](#)

The confidence threshold where Amazon Lex inserts the `AMAZON.FallbackIntent` and `AMAZON.KendraSearchIntent` intents in the list of possible intents for an utterance.

Type: Double

Valid Range: Minimum value of 0. Maximum value of 1.

[slotTypesCount \(p. 266\)](#)

The number of slot types defined for the locale.

Type: Integer

[voiceSettings \(p. 266\)](#)

The Amazon Polly voice Amazon Lex uses for voice interaction with the user.

Type: [VoiceSettings \(p. 528\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeBotVersion

Service: Amazon Lex Model Building V2

Provides metadata about a version of a bot.

Request Syntax

```
GET /bots/botId/botversions/botVersion HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 270)

The identifier of the bot containing the version to return metadata for.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 270)

The version of the bot to return metadata for.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botName": "string",
  "botStatus": "string",
  "botVersion": "string",
  "creationDateTime": number,
  "dataPrivacy": {
    "childDirected": boolean
  },
  "description": "string",
  "failureReasons": [ "string" ],
  "idleSessionTTLInSeconds": number,
  "roleArn": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 270)

The identifier of the bot that contains the version.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botName (p. 270)

The name of the bot that contains the version.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

botStatus (p. 270)

The current status of the bot. When the status is Available, the bot version is ready for use.

Type: String

Valid Values: Creating | Available | Inactive | Deleting | Failed | Versioning | Importing

botVersion (p. 270)

The version of the bot to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

creationDateTime (p. 270)

A timestamp of the date and time that the bot version was created.

Type: Timestamp

dataPrivacy (p. 270)

Data privacy settings for the bot version.

Type: [DataPrivacy \(p. 471\)](#) object

description (p. 270)

The description specified for the bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

failureReasons (p. 270)

If the botStatus is Failed, this contains a list of reasons that the version couldn't be built.

Type: Array of strings

[idleSessionTTLInSeconds \(p. 270\)](#)

The number of seconds that a session with the bot remains active before it is discarded by Amazon Lex.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

[roleArn \(p. 270\)](#)

The Amazon Resource Name (ARN) of an IAM role that has permission to access the bot version.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:iam::[0-9]{12}:role/.*\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeExport

Service: Amazon Lex Model Building V2

Gets information about a specific export.

Request Syntax

```
GET /exports/exportId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

exportId (p. 274)

The unique identifier of the export to describe.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "creationDateTime": number,
    "downloadUrl": "string",
    "exportId": "string",
    "exportStatus": "string",
    "failureReasons": [ "string" ],
    "fileFormat": "string",
    "lastUpdatedDateTime": number,
    "resourceSpecification": {
        "botExportSpecification": {
            "botId": "string",
            "botVersion": "string"
        },
        "botLocaleExportSpecification": {
            "botId": "string",
            "botVersion": "string",
            "localeId": "string"
        }
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[creationDateTime \(p. 274\)](#)

The date and time that the export was created.

Type: Timestamp

[downloadUrl \(p. 274\)](#)

A pre-signed S3 URL that points to the bot or bot locale archive. The URL is only available for 5 minutes after calling the `DescribeExport` operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

[exportId \(p. 274\)](#)

The unique identifier of the described export.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[exportStatus \(p. 274\)](#)

The status of the export. When the status is `Complete` the export archive file is available for download.

Type: String

Valid Values: `InProgress` | `Completed` | `Failed` | `Deleting`

[failureReasons \(p. 274\)](#)

If the `exportStatus` is `Failed`, contains one or more reasons why the export could not be completed.

Type: Array of strings

[fileFormat \(p. 274\)](#)

The file format used in the files that describe the bot or bot locale.

Type: String

Valid Values: `LexJson`

[lastUpdatedDateTime \(p. 274\)](#)

The last date and time that the export was updated.

Type: Timestamp

[resourceSpecification \(p. 274\)](#)

The bot, bot ID, and optional locale ID of the exported bot or bot locale.

Type: [ExportResourceSpecification \(p. 474\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeImport

Service: Amazon Lex Model Building V2

Gets information about a specific import.

Request Syntax

```
GET /imports/importId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

importId (p. 277)

The unique identifier of the import to describe.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "creationDateTime": number,
    "failureReasons": [ "string" ],
    "importedResourceId": "string",
    "importedResourceName": "string",
    "importId": "string",
    "importStatus": "string",
    "lastUpdatedDateTime": number,
    "mergeStrategy": "string",
    "resourceSpecification": {
        "botImportSpecification": {
            "botName": "string",
            "botTags": {
                "string": "string"
            },
            "dataPrivacy": {
                "childDirected": boolean
            },
            "idleSessionTTLInSeconds": number,
            "roleArn": "string",
            "testBotAliasTags": {
                "string": "string"
            }
        },
        "botLocaleImportSpecification": {
            "botId": "string",
            "botVersion": "string",
            "localeId": "string",
            "localeName": "string"
        }
    }
}
```

```
        "nluIntentConfidenceThreshold": number,
        "voiceSettings": {
            "voiceId": "string"
        }
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[creationDateTime \(p. 277\)](#)

The date and time that the import was created.

Type: Timestamp

[failureReasons \(p. 277\)](#)

If the `importStatus` field is `Failed`, this provides one or more reasons for the failure.

Type: Array of strings

[importedResourceId \(p. 277\)](#)

The unique identifier that Amazon Lex assigned to the resource created by the import.

Type: String

Length Constraints: Minimum length of 5. Maximum length of 10.

Pattern: `^([0-9a-zA-Z_])+$`

[importedResourceName \(p. 277\)](#)

The name of the imported resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

[importId \(p. 277\)](#)

The unique identifier of the described import.

Type: String

Length Constraints: Fixed length of 10.

Pattern: `^[0-9a-zA-Z]+$`

[importStatus \(p. 277\)](#)

The status of the import process. When the status is `Completed` the resource is imported and ready for use.

Type: String

Valid Values: `InProgress` | `Completed` | `Failed` | `Deleting`

[lastUpdatedDateTime \(p. 277\)](#)

The date and time that the import was last updated.

Type: Timestamp

[mergeStrategy \(p. 277\)](#)

The strategy used when there was a name conflict between the imported resource and an existing resource. When the merge strategy is `FailOnConflict` existing resources are not overwritten and the import fails.

Type: String

Valid Values: `Overwrite` | `FailOnConflict`

[resourceSpecification \(p. 277\)](#)

The specifications of the imported bot or bot locale.

Type: [ImportResourceSpecification \(p. 482\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeIntent

Service: Amazon Lex Model Building V2

Returns metadata about an intent.

Request Syntax

```
GET /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 280)

The identifier of the bot associated with the intent.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 280)

The version of the bot associated with the intent.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

intentId (p. 280)

The identifier of the intent to describe.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 280)

The identifier of the language and locale of the intent to describe. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json
{
}
```

```
"botId": "string",
"botVersion": "string",
"creationDateTime": number,
"description": "string",
"dialogCodeHook": {
    "enabled": boolean
},
"fulfillmentCodeHook": {
    "enabled": boolean
},
"inputContexts": [
    {
        "name": "string"
    }
],
"intentClosingSetting": {
    "closingResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        ]
    }
}
```

```

        }
    ],
},
"intentConfirmationSetting": {
    "declinationResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        ],
        "promptSpecification": {
            "allowInterrupt": boolean,
            "maxRetries": number,
            "messageGroups": [
                {
                    "message": {
                        "customPayload": {
                            "value": "string"
                        }
                    }
                }
            ]
        }
    }
}

```

```

},
"imageResponseCard": {
    "buttons": [
        {
            "text": "string",
            "value": "string"
        }
    ],
    "imageUrl": "string",
    "subtitle": "string",
    "title": "string"
},
"plainTextMessage": {
    "value": "string"
},
"ssmlMessage": {
    "value": "string"
}
},
"variations": [
    {
        "customPayload": {
            "value": "string"
        },
        "imageResponseCard": {
            "buttons": [
                {
                    "text": "string",
                    "value": "string"
                }
            ],
            "imageUrl": "string",
            "subtitle": "string",
            "title": "string"
        },
        "plainTextMessage": {
            "value": "string"
        },
        "ssmlMessage": {
            "value": "string"
        }
    }
]
}
},
"intentId": "string",
"intentName": "string",
"kendraConfiguration": {
    "kendraIndex": "string",
    "queryFilterString": "string",
    "queryFilterStringEnabled": boolean
},
"lastUpdatedDateTime": number,
"localeId": "string",
"outputContexts": [
    {
        "name": "string",
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
],
"parentIntentSignature": "string",
"sampleUtterances": [
    {

```

```
        "utterance": "string"
    }
],
"slotPriorities": [
{
    "priority": number,
    "slotId": "string"
}
]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[botId \(p. 280\)](#)

The identifier of the bot associated with the intent.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botVersion \(p. 280\)](#)

The version of the bot associated with the intent.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

[creationDateTime \(p. 280\)](#)

A timestamp of the date and time that the intent was created.

Type: Timestamp

[description \(p. 280\)](#)

The description of the intent.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[dialogCodeHook \(p. 280\)](#)

The Lambda function called during each turn of a conversation with the intent.

Type: [DialogCodeHookSettings \(p. 472\)](#) object

[fulfillmentCodeHook \(p. 280\)](#)

The Lambda function called when the intent is complete and ready for fulfillment.

Type: [FulfillmentCodeHookSettings \(p. 478\)](#) object

[inputContexts \(p. 280\)](#)

A list of contexts that must be active for the intent to be considered for sending to the user.

Type: Array of [InputContext \(p. 486\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

[intentClosingSetting \(p. 280\)](#)

The response that Amazon Lex sends to when the intent is closed.

Type: [IntentClosingSetting \(p. 487\)](#) object

[intentConfirmationSetting \(p. 280\)](#)

Prompts that Amazon Lex sends to the user to confirm completion of an intent.

Type: [IntentConfirmationSetting \(p. 488\)](#) object

[intentId \(p. 280\)](#)

The unique identifier assigned to the intent when it was created.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[intentName \(p. 280\)](#)

The name specified for the intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[kendraConfiguration \(p. 280\)](#)

Configuration information required to use the AMAZON.KendraSearchIntent intent.

Type: [KendraConfiguration \(p. 493\)](#) object

[lastUpdatedDateTime \(p. 280\)](#)

A timestamp of the date and time that the intent was last updated.

Type: Timestamp

[localeId \(p. 280\)](#)

The language and locale specified for the intent.

Type: String

[outputContexts \(p. 280\)](#)

A list of contexts that are activated when the intent is fulfilled.

Type: Array of [OutputContext \(p. 499\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

[parentIntentSignature \(p. 280\)](#)

The identifier of the built-in intent that this intent is derived from, if any.

Type: String

[sampleUtterances \(p. 280\)](#)

User utterances that trigger this intent.

Type: Array of [SampleUtterance \(p. 505\)](#) objects

[slotPriorities \(p. 280\)](#)

The list that determines the priority that slots should be elicited from the user.

Type: Array of [SlotPriority \(p. 511\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeResourcePolicy

Service: Amazon Lex Model Building V2

Gets the resource policy and policy revision for a bot or bot alias.

Request Syntax

```
GET /policy/resourceArn HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

[resourceArn \(p. 287\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "policy": "string",
  "resourceArn": "string",
  "revisionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[policy \(p. 287\)](#)

The JSON structure that contains the resource policy. For more information about the contents of a JSON policy document, see [IAM JSON policy reference](#).

Type: String

Length Constraints: Minimum length of 2.

[resourceArn \(p. 287\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1011.

[revisionId \(p. 287\)](#)

The current revision of the resource policy. Use the revision ID to make sure that you are updating the most current version of a resource policy when you add a policy statement to a resource, delete a resource, or update a resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeSlot

Service: Amazon Lex Model Building V2

Gets metadata information about a slot.

Request Syntax

```
GET /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId/slots/slotId  
HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 289)

The identifier of the bot associated with the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 289)

The version of the bot associated with the slot.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9])+\$

Required: Yes

intentId (p. 289)

The identifier of the intent that contains the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 289)

The identifier of the language and locale of the slot to describe. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

slotId (p. 289)

The unique identifier for the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "creationDateTime": number,
    "description": "string",
    "intentId": "string",
    "lastUpdatedDateTime": number,
    "localeId": "string",
    "multipleValuesSetting": {
        "allowMultipleValues": boolean
    },
    "obfuscationSetting": {
        "obfuscationSettingType": "string"
    },
    "slotId": "string",
    "slotName": "string",
    "slotTypeId": "string",
    "valueElicitationSetting": {
        "defaultValueSpecification": {
            "defaultValueList": [
                {
                    "defaultValue": "string"
                }
            ]
        },
        "promptSpecification": {
            "allowInterrupt": boolean,
            "maxRetries": number,
            "messageGroups": [
                {
                    "message": {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    },
                    "variations": [
                        {
                            "customPayload": {
                                "value": "string"
                            },
                            "imageResponseCard": {
                                "buttons": [
                                    {
                                        "text": "string",
                                        "value": "string"
                                    }
                                ],
                                "imageUrl": "string",
                                "subtitle": "string",
                                "title": "string"
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```

```
        "text": "string",
        "value": "string"
    }
],
"imageUrl": "string",
"subtitle": "string",
"title": "string"
},
"plainTextMessage": {
    "value": "string"
},
"ssmlMessage": {
    "value": "string"
}
}
]
}
],
"sampleUtterances": [
{
    "utterance": "string"
}
],
"slotConstraint": "string",
"waitAndContinueSpecification": {
    "continueResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
{
        "message": {
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
{
                    "text": "string",
                    "value": "string"
                }
],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            },
            "plainTextMessage": {
                "value": "string"
            },
            "ssmlMessage": {
                "value": "string"
            }
}
],
        "variations": [
{
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
{
                    "text": "string",
                    "value": "string"
                }
],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            }
}
]
}
}
]
```

```
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
],
},
"stillWaitingResponse": {
    "allowInterrupt": boolean,
    "frequencyInSeconds": number,
    "messageGroups": [
        {
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": {
                    "buttons": [
                        {
                            "text": "string",
                            "value": "string"
                        }
                    ],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
                },
                "plainTextMessage": {
                    "value": "string"
                },
                "ssmlMessage": {
                    "value": "string"
                }
            },
            "variations": [
                {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                }
            ]
        }
    ],
}
```

```
        "timeoutInSeconds": number
    },
    "waitingResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        }
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 290)

The identifier of the bot associated with the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 290)

The version of the bot associated with the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9])+\$

creationDateTime (p. 290)

A timestamp of the date and time that the slot was created.

Type: Timestamp

description (p. 290)

The description specified for the slot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

intentId (p. 290)

The identifier of the intent associated with the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

lastUpdatedDateTime (p. 290)

A timestamp of the date and time that the slot was last updated.

Type: Timestamp

localeId (p. 290)

The language and locale specified for the slot.

Type: String

multipleValuesSetting (p. 290)

Indicates whether the slot accepts multiple values in a single utterance.

If the `multipleValuesSetting` is not set, the default value is `false`.

Type: [MultipleValuesSetting \(p. 497\)](#) object

obfuscationSetting (p. 290)

Whether slot values are shown in Amazon CloudWatch logs. If the value is `None`, the actual value of the slot is shown in logs.

Type: [ObfuscationSetting \(p. 498\)](#) object

slotId (p. 290)

The unique identifier generated for the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

slotName (p. 290)

The name specified for the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

slotTypeId (p. 290)

The identifier of the slot type that determines the values entered into the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: ^((AMAZON\\.)[a-zA-Z_]+?|[0-9a-zA-Z]+)\$

valueElicitationSetting (p. 290)

Prompts that Amazon Lex uses to elicit a value for the slot.

Type: [SlotValueElicitationSetting \(p. 520\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeSlotType

Service: Amazon Lex Model Building V2

Gets metadata information about a slot type.

Request Syntax

```
GET /bots/botId/botversions/botVersion/botlocales/localeId/slottypes/slotTypeId/ HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 297)

The identifier of the bot associated with the slot type.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 297)

The version of the bot associated with the slot type.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

localeId (p. 297)

The identifier of the language and locale of the slot type to describe. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

slotTypeId (p. 297)

The identifier of the slot type.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
```

```
"botVersion": "string",
"creationDateTime": number,
"description": "string",
"lastUpdatedDateTime": number,
"localeId": "string",
"parentSlotTypeSignature": "string",
"slotTypeId": "string",
"slotTypeName": "string",
"slotTypeValues": [
  {
    "sampleValue": {
      "value": "string"
    },
    "synonyms": [
      {
        "value": "string"
      }
    ]
  }
],
"valueSelectionSetting": {
  "regexFilter": {
    "pattern": "string"
  },
  "resolutionStrategy": "string"
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[botId \(p. 297\)](#)

The identifier of the bot associated with the slot type.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botVersion \(p. 297\)](#)

The version of the bot associated with the slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

[creationDateTime \(p. 297\)](#)

A timestamp of the date and time that the slot type was created.

Type: Timestamp

[description \(p. 297\)](#)

The description specified for the slot type.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[lastUpdatedDateTime \(p. 297\)](#)

A timestamp of the date and time that the slot type was last updated.

Type: Timestamp

[localeId \(p. 297\)](#)

The language and locale specified for the slot type.

Type: String

[parentSlotTypeSignature \(p. 297\)](#)

The built in slot type used as a parent to this slot type.

Type: String

[slotTypeId \(p. 297\)](#)

The unique identifier for the slot type.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[slotTypeName \(p. 297\)](#)

The name specified for the slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[slotTypeValues \(p. 297\)](#)

The values that the slot type can take. Includes any synonyms for the slot type values.

Type: Array of [SlotTypeValue \(p. 519\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10000 items.

[valueSelectionSetting \(p. 297\)](#)

The strategy that Amazon Lex uses to choose a value from a list of possible values.

Type: [SlotValueSelectionSetting \(p. 523\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListBotAliases

Service: Amazon Lex Model Building V2

Gets a list of aliases for the specified bot.

Request Syntax

```
POST /bots/botId/botaliases/ HTTP/1.1
Content-type: application/json

{
  "maxResults": number,
  "nextToken": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 301)

The identifier of the bot to list aliases for.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

maxResults (p. 301)

The maximum number of aliases to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

nextToken (p. 301)

If the response from the `ListBotAliases` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

Response Syntax

```
HTTP/1.1 200
```

```
Content-type: application/json

{
  "botAliasSummaries": [
    {
      "botAliasId": "string",
      "botAliasName": "string",
      "botAliasStatus": "string",
      "botVersion": "string",
      "creationDateTime": number,
      "description": "string",
      "lastUpdatedDateTime": number
    }
  ],
  "botId": "string",
  "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botAliasSummaries (p. 301)

Summary information for the bot aliases that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more aliases available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [BotAliasSummary](#) (p. 441) objects

botId (p. 301)

The identifier of the bot associated with the aliases.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

nextToken (p. 301)

A token that indicates whether there are more results to return in a response to the `ListBotAliases` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListBotAliases` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 567).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListBotLocales

Service: Amazon Lex Model Building V2

Gets a list of locales for the specified bot.

Request Syntax

```
POST /bots/botId/botversions/botVersion/botlocales/ HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 304)

The identifier of the bot to list locales for.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 304)

The version of the bot to list locales for.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

filters (p. 304)

Provides the specification for a filter used to limit the response to only those locales that match the filter specification. You can only specify one filter and one value to filter on.

Type: Array of [BotLocaleFilter](#) (p. 448) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 304\)](#)

The maximum number of aliases to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 304\)](#)

If the response from the `ListBotLocales` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token as the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 304\)](#)

Specifies sorting parameters for the list of locales. You can sort by locale name in ascending or descending order.

Type: [BotLocaleSortBy \(p. 452\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botLocaleSummaries": [
    {
      "botLocaleStatus": "string",
      "description": "string",
      "lastBuildSubmittedDateTime": number,
      "lastUpdatedDateTime": number,
      "localeId": "string",
      "localeName": "string"
    }
  ],
  "botVersion": "string",
  "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[botId \(p. 305\)](#)

The identifier of the bot to list locales for.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botLocaleSummaries \(p. 305\)](#)

Summary information for the locales that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more locales available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [BotLocaleSummary \(p. 453\)](#) objects

[botVersion \(p. 305\)](#)

The version of the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

[nextToken \(p. 305\)](#)

A token that indicates whether there are more results to return in a response to the `ListBotLocales` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListBotLocales` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListBots

Service: Amazon Lex Model Building V2

Gets a list of available bots.

Request Syntax

```
POST /bots/ HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

[filters \(p. 308\)](#)

Provides the specification of a filter used to limit the bots in the response to only those that match the filter specification. You can only specify one filter and one string to filter on.

Type: Array of [BotFilter \(p. 444\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 308\)](#)

The maximum number of bots to return in each page of results. If there are fewer results than the maximum page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 308\)](#)

If the response from the `ListBots` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 308\)](#)

Specifies sorting parameters for the list of bots. You can specify that the list be sorted by bot name in ascending or descending order.

Type: [BotSortBy \(p. 455\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botSummaries": [
        {
            "botId": "string",
            "botName": "string",
            "botStatus": "string",
            "description": "string",
            "lastUpdatedDateTime": number,
            "latestBotVersion": "string"
        }
    ],
    "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[botSummaries \(p. 309\)](#)

Summary information for the bots that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more bots available, the `nextToken` field contains a token to the next page of results.

Type: Array of [BotSummary \(p. 456\)](#) objects

[nextToken \(p. 309\)](#)

A token that indicates whether there are more results to return in a response to the `ListBots` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListBots` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListBotVersions

Service: Amazon Lex Model Building V2

Gets information about all of the versions of a bot.

The `ListBotVersions` operation returns a summary of each version of a bot. For example, if a bot has three numbered versions, the `ListBotVersions` operation returns four summaries, one for each numbered version and one for the `DRAFT` version.

The `ListBotVersions` operation always returns at least one version, the `DRAFT` version.

Request Syntax

```
POST /bots/botId/botversions/ HTTP/1.1
Content-type: application/json

{
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 311)

The identifier of the bot to list versions for.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

maxResults (p. 311)

The maximum number of versions to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

nextToken (p. 311)

If the response to the `ListBotVersion` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 311\)](#)

Specifies sorting parameters for the list of versions. You can specify that the list be sorted by version name in either ascending or descending order.

Type: [BotVersionSortBy \(p. 459\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "botId": "string",
  "botVersionSummaries": [
    {
      "botName": "string",
      "botStatus": "string",
      "botVersion": "string",
      "creationDateTime": number,
      "description": "string"
    }
  ],
  "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[botId \(p. 312\)](#)

The identifier of the bot to list versions for.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botVersionSummaries \(p. 312\)](#)

Summary information for the bot versions that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more versions available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [BotVersionSummary \(p. 460\)](#) objects

[nextToken \(p. 312\)](#)

A token that indicates whether there are more results to return in a response to the `ListBotVersions` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListBotAliases` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListBuiltInIntents

Service: Amazon Lex Model Building V2

Gets a list of built-in intents provided by Amazon Lex that you can use in your bot.

To use a built-in intent as a the base for your own intent, include the built-in intent signature in the `parentIntentSignature` parameter when you call the `CreateIntent` operation. For more information, see [CreateIntent \(p. 192\)](#).

Request Syntax

```
POST /builtins/locales/localeId/intents/ HTTP/1.1
Content-type: application/json

{
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

`localeId` (p. 314)

The identifier of the language and locale of the intents to list. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

`maxResults` (p. 314)

The maximum number of built-in intents to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Fixed value of 20.

Required: No

`nextToken` (p. 314)

If the response from the `ListBuiltInIntents` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 314\)](#)

Specifies sorting parameters for the list of built-in intents. You can specify that the list be sorted by the built-in intent signature in either ascending or descending order.

Type: [BuiltInIntentSortBy \(p. 462\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "builtInIntentSummaries": [
        {
            "description": "string",
            "intentSignature": "string"
        }
    ],
    "localeId": "string",
    "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[builtInIntentSummaries \(p. 315\)](#)

Summary information for the built-in intents that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more intents available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [BuiltInIntentSummary \(p. 463\)](#) objects

[localeId \(p. 315\)](#)

The language and locale of the intents in the list.

Type: String

[nextToken \(p. 315\)](#)

A token that indicates whether there are more results to return in a response to the `ListBuiltInIntents` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListBotAliases` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListBuiltInSlotTypes

Service: Amazon Lex Model Building V2

Gets a list of built-in slot types that meet the specified criteria.

Request Syntax

```
POST /builtins/locales/localeId/slottypes/ HTTP/1.1
Content-type: application/json

{
    "maxResults": number,
    "nextToken": "string",
    "sortBy": {
        "attribute": "string",
        "order": "string"
    }
}
```

URI Request Parameters

The request uses the following URI parameters.

localeId (p. 317)

The identifier of the language and locale of the slot types to list. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

maxResults (p. 317)

The maximum number of built-in slot types to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Fixed value of 20.

Required: No

nextToken (p. 317)

If the response from the `ListBuiltInSlotTypes` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

sortBy (p. 317)

Determines the sort order for the response from the `ListBuiltInSlotTypes` operation. You can choose to sort by the slot type signature in either ascending or descending order.

Type: [BuiltInSlotTypeSortBy](#) (p. 464) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "builtInSlotTypeSummaries": [
    {
      "description": "string",
      "slotTypeSignature": "string"
    }
  ],
  "localeId": "string",
  "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

builtInSlotTypeSummaries ([p. 318](#))

Summary information for the built-in slot types that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more slot types available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [BuiltInSlotTypeSummary](#) ([p. 465](#)) objects

localeId ([p. 318](#))

The language and locale of the slot types in the list.

Type: String

nextToken ([p. 318](#))

A token that indicates whether there are more results to return in a response to the `ListBuiltInSlotTypes` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListBuiltInSlotTypes` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors](#) ([p. 567](#)).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListExports

Service: Amazon Lex Model Building V2

Lists the exports for a bot or bot locale. Exports are kept in the list for 7 days.

Request Syntax

```
POST /exports/ HTTP/1.1
Content-type: application/json

{
  "botId": "string",
  "botVersion": "string",
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

botId (p. 320)

The unique identifier that Amazon Lex assigned to the bot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

botVersion (p. 320)

The version of the bot to list exports for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: No

filters (p. 320)

Provides the specification of a filter used to limit the exports in the response to only those that match the filter specification. You can only specify one filter and one string to filter on.

Type: Array of [ExportFilter \(p. 473\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 320\)](#)

The maximum number of exports to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 320\)](#)

If the response from the `ListExports` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 320\)](#)

Determines the field that the list of exports is sorted by. You can sort by the `LastUpdatedDateTime` field in ascending or descending order.

Type: [ExportSortBy \(p. 475\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "exportSummaries": [
        {
            "creationDateTime": number,
            "exportId": "string",
            "exportStatus": "string",
            "fileFormat": "string",
            "lastUpdatedDateTime": number,
            "resourceSpecification": {
                "botExportSpecification": {
                    "botId": "string",
                    "botVersion": "string"
                },
                "botLocaleExportSpecification": {
                    "botId": "string",
                    "botVersion": "string",
                    "localeId": "string"
                }
            }
        }
    ],
}
```

```
    "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 321)

The unique identifier assigned to the bot by Amazon Lex.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 321)

The version of the bot that was exported.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

exportSummaries (p. 321)

Summary information for the exports that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter. If there are more exports available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [ExportSummary](#) (p. 476) objects

nextToken (p. 321)

A token that indicates whether there are more results to return in a response to the `ListExports` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListExports` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 567).

InternalServerError

HTTP Status Code: 500

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListImports

Service: Amazon Lex Model Building V2

Lists the imports for a bot or bot locale. Imports are kept in the list for 7 days.

Request Syntax

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
  "botId": "string",
  "botVersion": "string",
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

botId (p. 324)

The unique identifier that Amazon Lex assigned to the bot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

botVersion (p. 324)

The version of the bot to list imports for.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: No

filters (p. 324)

Provides the specification of a filter used to limit the bots in the response to only those that match the filter specification. You can only specify one filter and one string to filter on.

Type: Array of [ImportFilter \(p. 481\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 324\)](#)

The maximum number of imports to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 324\)](#)

If the response from the `ListImports` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 324\)](#)

Determines the field that the list of imports is sorted by. You can sort by the `LastUpdatedDateTime` field in ascending or descending order.

Type: [ImportSortBy \(p. 483\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "importSummaries": [
        {
            "creationDateTime": number,
            "importedResourceId": "string",
            "importedResourceName": "string",
            "importId": "string",
            "importStatus": "string",
            "lastUpdatedDateTime": number,
            "mergeStrategy": "string"
        }
    ],
    "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 325)

The unique identifier assigned by Amazon Lex to the bot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 325)

The version of the bot that was imported. It will always be DRAFT.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

importSummaries (p. 325)

Summary information for the imports that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter. If there are more imports available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [ImportSummary \(p. 484\)](#) objects

nextToken (p. 325)

A token that indicates whether there are more results to return in a response to the `ListImports` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListImports` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListIntents

Service: Amazon Lex Model Building V2

Get a list of intents that meet the specified criteria.

Request Syntax

```
POST /bots/botId/botversions/botVersion/botlocales/localeId/intents/ HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 328)

The unique identifier of the bot that contains the intent.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 328)

The version of the bot that contains the intent.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

localeId (p. 328)

The identifier of the language and locale of the intents to list. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

[filters \(p. 328\)](#)

Provides the specification of a filter used to limit the intents in the response to only those that match the filter specification. You can only specify one filter and only one string to filter on.

Type: Array of [IntentFilter \(p. 489\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 328\)](#)

The maximum number of intents to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 328\)](#)

If the response from the `ListIntents` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 328\)](#)

Determines the sort order for the response from the `ListIntents` operation. You can choose to sort by the intent name or last updated date in either ascending or descending order.

Type: [IntentSortBy \(p. 490\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "intentSummaries": [
        {
            "description": "string",
            "inputContexts": [
                {
                    "name": "string"
                }
            ],
            "intentId": "string",
            "intentName": "string",
            "lastUpdatedDateTime": number,
            "outputContexts": [
                {
                    "name": "string",
                    "recovery": "string"
                }
            ]
        }
    ]
}
```

```
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
],
"parentIntentSignature": "string"
],
"localeId": "string",
"nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId ([p. 329](#))

The identifier of the bot that contains the intent.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion ([p. 329](#))

The version of the bot that contains the intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

intentSummaries ([p. 329](#))

Summary information for the intents that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more intents available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [IntentSummary](#) ([p. 491](#)) objects

localeId ([p. 329](#))

The language and locale of the intents in the list.

Type: String

nextToken ([p. 329](#))

A token that indicates whether there are more results to return in a response to the `ListIntents` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListIntents` operation request to get the next page of results.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors](#) ([p. 567](#)).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListSlots

Service: Amazon Lex Model Building V2

Gets a list of slots that match the specified criteria.

Request Syntax

```
POST /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId/slots/
HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 332)

The identifier of the bot that contains the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 332)

The version of the bot that contains the slot.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

intentId (p. 332)

The unique identifier of the intent that contains the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

[localeId \(p. 332\)](#)

The identifier of the language and locale of the slots to list. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

[filters \(p. 332\)](#)

Provides the specification of a filter used to limit the slots in the response to only those that match the filter specification. You can only specify one filter and only one string to filter on.

Type: Array of [SlotFilter \(p. 510\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 332\)](#)

The maximum number of slots to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 332\)](#)

If the response from the `ListSlots` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 332\)](#)

Determines the sort order for the response from the `ListSlots` operation. You can choose to sort by the slot name or last updated date in either ascending or descending order.

Type: [SlotSortBy \(p. 512\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "intentId": "string",
```

```
"localeId": "string",
"nextToken": "string",
"slotSummaries": [
  {
    "description": "string",
    "lastUpdatedDateTime": number,
    "slotConstraint": "string",
    "slotId": "string",
    "slotName": "string",
    "slotTypeId": "string",
    "valueElicitationPromptSpecification": {
      "allowInterrupt": boolean,
      "maxRetries": number,
      "messageGroups": [
        {
          "message": {
            "customPayload": {
              "value": "string"
            },
            "imageResponseCard": {
              "buttons": [
                {
                  "text": "string",
                  "value": "string"
                }
              ],
              "imageUrl": "string",
              "subtitle": "string",
              "title": "string"
            }
          },
          "plainTextMessage": {
            "value": "string"
          },
          "ssmlMessage": {
            "value": "string"
          }
        },
        "variations": [
          {
            "customPayload": {
              "value": "string"
            },
            "imageResponseCard": {
              "buttons": [
                {
                  "text": "string",
                  "value": "string"
                }
              ],
              "imageUrl": "string",
              "subtitle": "string",
              "title": "string"
            }
          },
          "plainTextMessage": {
            "value": "string"
          },
          "ssmlMessage": {
            "value": "string"
          }
        ]
      }
    }
  }
]
```

}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 333)

The identifier of the bot that contains the slots.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 333)

The version of the bot that contains the slots.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

intentId (p. 333)

The identifier of the intent that contains the slots.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

localeId (p. 333)

The language and locale of the slots in the list.

Type: String

nextToken (p. 333)

A token that indicates whether there are more results to return in a response to the `ListSlots` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListSlots` operation request to get the next page of results.

Type: String

slotSummaries (p. 333)

Summary information for the slots that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more slots available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [SlotSummary](#) (p. 513) objects

Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 567).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListSlotTypes

Service: Amazon Lex Model Building V2

Gets a list of slot types that match the specified criteria.

Request Syntax

```
POST /bots/botId/botversions/botVersion/botlocales/localeId/slottypes/ HTTP/1.1
Content-type: application/json

{
  "filters": [
    {
      "name": "string",
      "operator": "string",
      "values": [ "string" ]
    }
  ],
  "maxResults": number,
  "nextToken": "string",
  "sortBy": {
    "attribute": "string",
    "order": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 337)

The unique identifier of the bot that contains the slot types.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 337)

The version of the bot that contains the slot type.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

localeId (p. 337)

The identifier of the language and locale of the slot types to list. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

[filters \(p. 337\)](#)

Provides the specification of a filter used to limit the slot types in the response to only those that match the filter specification. You can only specify one filter and only one string to filter on.

Type: Array of [SlotTypeFilter \(p. 515\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

[maxResults \(p. 337\)](#)

The maximum number of slot types to return in each page of results. If there are fewer results than the max page size, only the actual number of results are returned.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 1000.

Required: No

[nextToken \(p. 337\)](#)

If the response from the `ListSlotTypes` operation contains more results than specified in the `maxResults` parameter, a token is returned in the response. Use that token in the `nextToken` parameter to return the next page of results.

Type: String

Required: No

[sortBy \(p. 337\)](#)

Determines the sort order for the response from the `ListSlotTypes` operation. You can choose to sort by the slot type name or last updated date in either ascending or descending order.

Type: [SlotTypeSortBy \(p. 516\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "nextToken": "string",
    "slotTypeSummaries": [
        {
            "description": "string",
            "lastUpdatedDateTime": number,
            "parentSlotTypeSignature": "string",
            "slotTypeId": "string",
            "slotTypeName": "string"
        }
    ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 338)

The identifier of the bot that contains the slot types.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 338)

The version of the bot that contains the slot types.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9])+\$

localeId (p. 338)

The language and local of the slot types in the list.

Type: String

nextToken (p. 338)

A token that indicates whether there are more results to return in a response to the `ListSlotTypes` operation. If the `nextToken` field is present, you send the contents as the `nextToken` parameter of a `ListSlotTypes` operation request to get the next page of results.

Type: String

slotTypeSummaries (p. 338)

Summary information for the slot types that meet the filter criteria specified in the request. The length of the list is specified in the `maxResults` parameter of the request. If there are more slot types available, the `nextToken` field contains a token to get the next page of results.

Type: Array of [SlotTypeSummary \(p. 517\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListTagsForResource

Service: Amazon Lex Model Building V2

Gets a list of tags associated with a resource. Only bots, bot aliases, and bot channels can have tags associated with them.

Request Syntax

```
GET /tags/resourceARN HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

resourceARN (p. 341)

The Amazon Resource Name (ARN) of the resource to get a list of tags for.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "tags": [
    {
      "string" : "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

tags (p. 341)

The tags associated with a resource.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartImport

Service: Amazon Lex Model Building V2

Starts importing a bot or bot locale from a zip archive that you uploaded to an S3 bucket.

Request Syntax

```
PUT /imports/ HTTP/1.1
Content-type: application/json

{
    "filePassword": "string",
    "importId": "string",
    "mergeStrategy": "string",
    "resourceSpecification": {
        "botImportSpecification": {
            "botName": "string",
            "botTags": {
                "string" : "string"
            },
            "dataPrivacy": {
                "childDirected": boolean
            },
            "idleSessionTTLInSeconds": number,
            "roleArn": "string",
            "testBotAliasTags": {
                "string" : "string"
            }
        },
        "botLocaleImportSpecification": {
            "botId": "string",
            "botVersion": "string",
            "localeId": "string",
            "nluIntentConfidenceThreshold": number,
            "voiceSettings": {
                "voiceId": "string"
            }
        }
    }
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

filePassword (p. 343)

The password used to encrypt the zip archive that contains the bot or bot locale definition. You should always encrypt the zip archive to protect it during transit between your site and Amazon Lex.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

[importId \(p. 343\)](#)

The unique identifier for the import. It is included in the response from the [CreateUploadUrl \(p. 229\)](#) operation.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

[mergeStrategy \(p. 343\)](#)

The strategy to use when there is a name conflict between the imported resource and an existing resource. When the merge strategy is `FailOnConflict` existing resources are not overwritten and the import fails.

Type: String

Valid Values: `Overwrite` | `FailOnConflict`

Required: Yes

[resourceSpecification \(p. 343\)](#)

Parameters for creating the bot or bot locale.

Type: [ImportResourceSpecification \(p. 482\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "creationDateTime": number,
    "importId": "string",
    "importStatus": "string",
    "mergeStrategy": "string",
    "resourceSpecification": {
        "botImportSpecification": {
            "botName": "string",
            "botTags": {
                "string": "string"
            },
            "dataPrivacy": {
                "childDirected": boolean
            },
            "idleSessionTTLInSeconds": number,
            "roleArn": "string",
            "testBotAliasTags": {
                "string": "string"
            }
        },
        "botLocaleImportSpecification": {
            "botId": "string",
            "botVersion": "string",
            "localeId": "string",
            "nluIntentConfidenceThreshold": number,
            "nluProcessorType": "string",
            "redundantBot": "string"
        }
    }
}
```

```
        "voiceSettings": {  
            "voiceId": "string"  
        }  
    }  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[creationDateTime \(p. 344\)](#)

The date and time that the import request was created.

Type: Timestamp

[importId \(p. 344\)](#)

A unique identifier for the import.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[importStatus \(p. 344\)](#)

The current status of the import. When the status is `Complete` the bot or bot alias is ready to use.

Type: String

Valid Values: `InProgress` | `Completed` | `Failed` | `Deleting`

[mergeStrategy \(p. 344\)](#)

The strategy used when there was a name conflict between the imported resource and an existing resource. When the merge strategy is `FailOnConflict` existing resources are not overwritten and the import fails.

Type: String

Valid Values: `Overwrite` | `FailOnConflict`

[resourceSpecification \(p. 344\)](#)

The parameters used when importing the bot or bot locale.

Type: [ImportResourceSpecification \(p. 482\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

TagResource

Service: Amazon Lex Model Building V2

Adds the specified tags to the specified resource. If a tag key already exists, the existing value is replaced with the new value.

Request Syntax

```
POST /tags/resourceARN HTTP/1.1
Content-type: application/json

{
  "tags": {
    "string" : "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

resourceARN (p. 347)

The Amazon Resource Name (ARN) of the bot, bot alias, or bot channel to tag.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request accepts the following data in JSON format.

tags (p. 347)

A list of tag keys to add to the resource. If a tag key already exists, the existing value is replaced with the new value.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UntagResource

Service: Amazon Lex Model Building V2

Removes tags from a bot, bot alias, or bot channel.

Request Syntax

```
DELETE /tags/resourceARN?tagKeys=tagKeys HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

resourceARN (p. 349)

The Amazon Resource Name (ARN) of the resource to remove the tags from.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

tagKeys (p. 349)

A list of tag keys to remove from the resource. If a tag key does not exist on the resource, it is ignored.

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateBot

Service: Amazon Lex Model Building V2

Updates the configuration of an existing bot.

Request Syntax

```
PUT /bots/botId/ HTTP/1.1
Content-type: application/json

{
    "botName": "string",
    "dataPrivacy": {
        "childDirected": boolean
    },
    "description": "string",
    "idleSessionTTLInSeconds": number,
    "roleArn": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 351)

The unique identifier of the bot to update. This identifier is returned by the [CreateBot \(p. 168\)](#) operation.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

botName (p. 351)

The new name of the bot. The name must be unique in the account that creates the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

dataPrivacy (p. 351)

Provides information on additional privacy protections Amazon Lex should use with the bot's data.

Type: [DataPrivacy \(p. 471\)](#) object

Required: Yes

description (p. 351)

A description of the bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[idleSessionTTLInSeconds \(p. 351\)](#)

The time, in seconds, that Amazon Lex should keep information about a user's conversation with the bot.

A user interaction remains active for the amount of time specified. If no conversation occurs during this time, the session expires and Amazon Lex deletes any data provided before the timeout.

You can specify between 60 (1 minute) and 86,400 (24 hours) seconds.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

Required: Yes

[roleArn \(p. 351\)](#)

The Amazon Resource Name (ARN) of an IAM role that has permissions to access the bot.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:iam::[0-9]{12}:role/.*\$

Required: Yes

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
  "botId": "string",
  "botName": "string",
  "botStatus": "string",
  "creationDateTime": number,
  "dataPrivacy": {
    "childDirected": boolean
  },
  "description": "string",
  "idleSessionTTLInSeconds": number,
  "lastUpdatedDateTime": number,
  "roleArn": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[botId \(p. 352\)](#)

The unique identifier of the bot that was updated.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botName \(p. 352\)](#)

The name of the bot after the update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[botStatus \(p. 352\)](#)

Shows the current status of the bot. The bot is first in the `Creating` status. Once the bot is ready for use, it changes to the `Available` status. After the bot is created, you can use the `DRAFT` version of the bot.

Type: String

Valid Values: `Creating` | `Available` | `Inactive` | `Deleting` | `Failed` | `Versioning` | `Importing`

[creationDateTime \(p. 352\)](#)

A timestamp of the date and time that the bot was created.

Type: Timestamp

[dataPrivacy \(p. 352\)](#)

The data privacy settings for the bot after the update.

Type: [DataPrivacy \(p. 471\)](#) object

[description \(p. 352\)](#)

The description of the bot after the update.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[idleSessionTTLInSeconds \(p. 352\)](#)

The session timeout, in seconds, for the bot after the update.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

[lastUpdatedDateTime \(p. 352\)](#)

A timestamp of the date and time that the bot was last updated.

Type: Timestamp

[roleArn \(p. 352\)](#)

The Amazon Resource Name (ARN) of the IAM role used by the bot after the update.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:iam::[0-9]{12}:role/.*\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateBotAlias

Service: Amazon Lex Model Building V2

Updates the configuration of an existing bot alias.

Request Syntax

```
PUT /bots/botId/botaliases/botAliasId HTTP/1.1
Content-type: application/json

{
    "botAliasLocaleSettings": {
        "string": {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "string",
                    "lambdaARN": "string"
                }
            },
            "enabled": boolean
        }
    },
    "botAliasName": "string",
    "botVersion": "string",
    "conversationLogSettings": {
        "audioLogSettings": [
            {
                "destination": {
                    "s3Bucket": {
                        "kmsKeyArn": "string",
                        "logPrefix": "string",
                        "s3BucketArn": "string"
                    }
                },
                "enabled": boolean
            }
        ],
        "textLogSettings": [
            {
                "destination": {
                    "cloudWatch": {
                        "cloudWatchLogGroupArn": "string",
                        "logPrefix": "string"
                    }
                },
                "enabled": boolean
            }
        ]
    },
    "description": "string",
    "sentimentAnalysisSettings": {
        "detectSentiment": boolean
    }
}
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 355)

The unique identifier of the bot alias.

Length Constraints: Fixed length of 10.

Pattern: `^(\bTSTALIASID\b|[0-9a-zA-Z]+)$`

Required: Yes

botId (p. 355)

The identifier of the bot with the updated alias.

Length Constraints: Fixed length of 10.

Pattern: `^[0-9a-zA-Z]+$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

botAliasLocaleSettings (p. 355)

The new Lambda functions to use in each locale for the bot alias.

Type: String to [BotAliasLocaleSettings \(p. 440\)](#) object map

Map Entries: Maximum number of items.

Required: No

botAliasName (p. 355)

The new name to assign to the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

Required: Yes

botVersion (p. 355)

The new bot version to assign to the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: `^(DRAFT|[0-9]+)$`

Required: No

conversationLogSettings (p. 355)

The new settings for storing conversation logs in Amazon CloudWatch Logs and Amazon S3 buckets.

Type: [ConversationLogSettings \(p. 469\)](#) object

Required: No

description (p. 355)

The new description to assign to the bot alias.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[sentimentAnalysisSettings \(p. 355\)](#)

Determines whether Amazon Lex will use Amazon Comprehend to detect the sentiment of user utterances.

Type: [SentimentAnalysisSettings \(p. 507\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botAliasId": "string",
    "botAliasLocaleSettings": {
        "string" : {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "string",
                    "lambdaARN": "string"
                }
            },
            "enabled": boolean
        }
    },
    "botAliasName": "string",
    "botAliasStatus": "string",
    "botId": "string",
    "botVersion": "string",
    "conversationLogSettings": {
        "audioLogSettings": [
            {
                "destination": {
                    "s3Bucket": {
                        "kmsKeyArn": "string",
                        "logPrefix": "string",
                        "s3BucketArn": "string"
                    }
                },
                "enabled": boolean
            }
        ],
        "textLogSettings": [
            {
                "destination": {
                    "cloudWatch": {
                        "cloudWatchLogGroupArn": "string",
                        "logPrefix": "string"
                    }
                },
                "enabled": boolean
            }
        ]
    },
    "creationDateTime": number,
    "description": "string",
    "lastUpdatedDateTime": number,
    "name": "string",
    "status": "string",
    "version": "string"
}
```

```
"lastUpdatedDateTime": number,  
"sentimentAnalysisSettings": {  
    "detectSentiment": boolean  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[botAliasId \(p. 357\)](#)

The identifier of the updated bot alias.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

[botAliasLocaleSettings \(p. 357\)](#)

The updated Lambda functions to use in each locale for the bot alias.

Type: String to [BotAliasLocaleSettings \(p. 440\)](#) object map

Map Entries: Maximum number of items.

[botAliasName \(p. 357\)](#)

The updated name of the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[botAliasStatus \(p. 357\)](#)

The current status of the bot alias. When the status is Available the alias is ready for use.

Type: String

Valid Values: Creating | Available | Deleting | Failed

[botId \(p. 357\)](#)

The identifier of the bot with the updated alias.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botVersion \(p. 357\)](#)

The updated version of the bot that the alias points to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

[conversationLogSettings \(p. 357\)](#)

The updated settings for storing conversation logs in Amazon CloudWatch Logs and Amazon S3 buckets.

Type: [ConversationLogSettings \(p. 469\)](#) object

[creationDateTime \(p. 357\)](#)

A timestamp of the date and time that the bot was created.

Type: Timestamp

[description \(p. 357\)](#)

The updated description of the bot alias.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[lastUpdatedDateTime \(p. 357\)](#)

A timestamp of the date and time that the bot was last updated.

Type: Timestamp

[sentimentAnalysisSettings \(p. 357\)](#)

Determines whether Amazon Lex will use Amazon Comprehend to detect the sentiment of user utterances.

Type: [SentimentAnalysisSettings \(p. 507\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateBotLocale

Service: Amazon Lex Model Building V2

Updates the settings that a bot has for a specific locale.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/ HTTP/1.1
Content-type: application/json

{
    "description": "string",
    "nluIntentConfidenceThreshold": number,
    "voiceSettings": {
        "voiceId": "string"
    }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 361)

The unique identifier of the bot that contains the locale.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 361)

The version of the bot that contains the locale to be updated. The version can only be the DRAFT version.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId (p. 361)

The identifier of the language and locale to update. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

description (p. 361)

The new description of the locale.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[nluIntentConfidenceThreshold \(p. 361\)](#)

The new confidence threshold where Amazon Lex inserts the AMAZON.FallbackIntent and AMAZON.KendraSearchIntent intents in the list of possible intents for an utterance.

Type: Double

Valid Range: Minimum value of 0. Maximum value of 1.

Required: Yes

[voiceSettings \(p. 361\)](#)

The new Amazon Polly voice Amazon Lex should use for voice interaction with the user.

Type: [VoiceSettings \(p. 528\)](#) object

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botId": "string",
    "botLocaleStatus": "string",
    "botVersion": "string",
    "creationDateTime": number,
    "description": "string",
    "failureReasons": [ "string" ],
    "lastUpdatedDateTime": number,
    "localeId": "string",
    "localeName": "string",
    "nluIntentConfidenceThreshold": number,
    "voiceSettings": {
        "voiceId": "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[botId \(p. 362\)](#)

The identifier of the bot that contains the updated locale.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botLocaleStatus \(p. 362\)](#)

The current status of the locale. When the bot status is Built the locale is ready for use.

Type: String

Valid Values: Creating | Building | Built | ReadyExpressTesting | Failed | Deleting | NotBuilt | Importing

[botVersion \(p. 362\)](#)

The version of the bot that contains the updated locale.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

[creationDateTime \(p. 362\)](#)

A timestamp of the date and time that the locale was created.

Type: Timestamp

[description \(p. 362\)](#)

The updated description of the locale.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[failureReasons \(p. 362\)](#)

If the `botLocaleStatus` is `Failed`, the `failureReasons` field lists the errors that occurred while building the bot.

Type: Array of strings

[lastUpdatedDateTime \(p. 362\)](#)

A timestamp of the date and time that the locale was last updated.

Type: Timestamp

[localeId \(p. 362\)](#)

The language and locale of the updated bot locale.

Type: String

[localeName \(p. 362\)](#)

The updated locale name for the locale.

Type: String

[nluIntentConfidenceThreshold \(p. 362\)](#)

The updated confidence threshold for inserting the `AMAZON.FallbackIntent` and `AMAZON.KendraSearchIntent` intents in the list of possible intents for an utterance.

Type: Double

Valid Range: Minimum value of 0. Maximum value of 1.

[voiceSettings \(p. 362\)](#)

The updated Amazon Polly voice to use for voice interaction with the user.

Type: [VoiceSettings \(p. 528\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateExport

Service: Amazon Lex Model Building V2

Updates the password used to protect an export zip archive.

The password is not required. If you don't supply a password, Amazon Lex generates a zip file that is not protected by a password. This is the archive that is available at the pre-signed S3 URL provided by the [DescribeExport \(p. 274\)](#) operation.

Request Syntax

```
PUT /exports/exportId/ HTTP/1.1
Content-type: application/json

{
    "filePassword": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

[exportId \(p. 365\)](#)

The unique identifier Amazon Lex assigned to the export.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

[filePassword \(p. 365\)](#)

The new password to use to encrypt the export zip archive.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "creationDateTime": number,
    "exportId": "string",
    "exportStatus": "string",
    "fileFormat": "string",
    "lastUpdatedDateTime": number,
    "resourceSpecification": {
        "botExportSpecification": {

```

```
        "botId": "string",
        "botVersion": "string"
    },
    "botLocaleExportSpecification": {
        "botId": "string",
        "botVersion": "string",
        "localeId": "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

[creationDateTime \(p. 365\)](#)

The date and time that the export was created.

Type: Timestamp

[exportId \(p. 365\)](#)

The unique identifier Amazon Lex assigned to the export.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[exportStatus \(p. 365\)](#)

The status of the export. When the status is Completed the export archive is available for download.

Type: String

Valid Values: InProgress | Completed | Failed | Deleting

[fileFormat \(p. 365\)](#)

The file format used for the files that define the resource.

Type: String

Valid Values: LexJson

[lastUpdatedDateTime \(p. 365\)](#)

The date and time that the export was last updated.

Type: Timestamp

[resourceSpecification \(p. 365\)](#)

A description of the type of resource that was exported, either a bot or a bot locale.

Type: [ExportResourceSpecification \(p. 474\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateIntent

Service: Amazon Lex Model Building V2

Updates the settings for an intent.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId HTTP/1.1
Content-type: application/json
```

```
{
    "description": "string",
    "dialogCodeHook": {
        "enabled": boolean
    },
    "fulfillmentCodeHook": {
        "enabled": boolean
    },
    "inputContexts": [
        {
            "name": "string"
        }
    ],
    "intentClosingSetting": {
        "closingResponse": {
            "allowInterrupt": boolean,
            "messageGroups": [
                {
                    "message": {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    },
                    "variations": [
                        {
                            "customPayload": {
                                "value": "string"
                            },
                            "imageResponseCard": {
                                "buttons": [
                                    {
                                        "text": "string",
                                        "value": "string"
                                    }
                                ],
                                "imageUrl": "string",
                                "subtitle": "string",
                                "title": "string"
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```

```
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
}
},
"intentConfirmationSetting": {
    "declinationResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        ]
    }
}
```

```

        ],
    },
    "promptSpecification": {
        "allowInterrupt": boolean,
        "maxRetries": number,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": string
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": string,
                                "value": string
                            }
                        ],
                        "imageUrl": string,
                        "subtitle": string,
                        "title": string
                    }
                },
                "plainTextMessage": {
                    "value": string
                },
                "ssmlMessage": {
                    "value": string
                }
            },
            "variations": [
                {
                    "customPayload": {
                        "value": string
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": string,
                                "value": string
                            }
                        ],
                        "imageUrl": string,
                        "subtitle": string,
                        "title": string
                    }
                },
                "plainTextMessage": {
                    "value": string
                },
                "ssmlMessage": {
                    "value": string
                }
            ]
        }
    },
    "intentName": string,
    "kendraConfiguration": {
        "kendraIndex": string,
        "queryFilterString": string,
        "queryFilterStringEnabled": boolean
    },
    "outputContexts": [
        {
            "name": string,

```

```
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
],
"parentIntentSignature": "string",
"sampleUtterances": [
{
    "utterance": "string"
}
],
"slotPriorities": [
{
    "priority": number,
    "slotId": "string"
}
]
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 368)

The identifier of the bot that contains the intent.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 368)

The version of the bot that contains the intent. Must be DRAFT.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

intentId (p. 368)

The unique identifier of the intent to update.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 368)

The identifier of the language and locale where this intent is used. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

Request Body

The request accepts the following data in JSON format.

[description \(p. 368\)](#)

The new description of the intent.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[dialogCodeHook \(p. 368\)](#)

The new Lambda function to use between each turn of the conversation with the bot.

Type: [DialogCodeHookSettings \(p. 472\)](#) object

Required: No

[fulfillmentCodeHook \(p. 368\)](#)

The new Lambda function to call when all of the intents required slots are provided and the intent is ready for fulfillment.

Type: [FulfillmentCodeHookSettings \(p. 478\)](#) object

Required: No

[inputContexts \(p. 368\)](#)

A new list of contexts that must be active in order for Amazon Lex to consider the intent.

Type: Array of [InputContext \(p. 486\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

Required: No

[intentClosingSetting \(p. 368\)](#)

The new response that Amazon Lex sends the user when the intent is closed.

Type: [IntentClosingSetting \(p. 487\)](#) object

Required: No

[intentConfirmationSetting \(p. 368\)](#)

New prompts that Amazon Lex sends to the user to confirm the completion of an intent.

Type: [IntentConfirmationSetting \(p. 488\)](#) object

Required: No

[intentName \(p. 368\)](#)

The new name for the intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

[kendraConfiguration \(p. 368\)](#)

New configuration settings for connecting to an Amazon Kendra index.

Type: [KendraConfiguration \(p. 493\)](#) object

Required: No

[outputContexts \(p. 368\)](#)

A new list of contexts that Amazon Lex activates when the intent is fulfilled.

Type: Array of [OutputContext \(p. 499\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

[parentIntentSignature \(p. 368\)](#)

The signature of the new built-in intent to use as the parent of this intent.

Type: String

Required: No

[sampleUtterances \(p. 368\)](#)

New utterances used to invoke the intent.

Type: Array of [SampleUtterance \(p. 505\)](#) objects

Required: No

[slotPriorities \(p. 368\)](#)

A new list of slots and their priorities that are contained by the intent.

Type: Array of [SlotPriority \(p. 511\)](#) objects

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "creationDateTime": number,
    "description": "string",
    "dialogCodeHook": {
        "enabled": boolean
    },
    "fulfillmentCodeHook": {
        "enabled": boolean
    },
    "inputContexts": [
        {
            "name": "string"
        }
    ],
    "intentClosingSetting": {
        "closingResponse": {
            "allowInterrupt": boolean,
            "messageGroups": [
                {
                    "message": {
                        "customPayload": {
                            "contentType": "string",
                            "payload": "string"
                        }
                    }
                }
            ]
        }
    }
}
```

```
        "value": "string"
    },
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
},
"variations": [
    {
        "customPayload": {
            "value": "string"
        },
        "imageResponseCard": {
            "buttons": [
                {
                    "text": "string",
                    "value": "string"
                }
            ],
            "imageUrl": "string",
            "subtitle": "string",
            "title": "string"
        },
        "plainTextMessage": {
            "value": "string"
        },
        "ssmlMessage": {
            "value": "string"
        }
    }
]
}
},
"intentConfirmationSetting": {
    "declinationResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    }
                }
            }
        ]
    }
}
```

```
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
},
"variations": [
{
    "customPayload": {
        "value": "string"
    },
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
},
"promptSpecification": {
    "allowInterrupt": boolean,
    "maxRetries": number,
    "messageGroups": [
{
        "message": {
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            },
            "plainTextMessage": {
                "value": "string"
            },
            "ssmlMessage": {
                "value": "string"
            }
        },
        "variations": [
{
        "customPayload": {
```

```

        "value": "string"
    },
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
},
"intentId": "string",
"intentName": "string",
"kendraConfiguration": {
    "kendraIndex": "string",
    "queryFilterString": "string",
    "queryFilterStringEnabled": boolean
},
"lastUpdatedDateTime": number,
"localeId": "string",
"outputContexts": [
    {
        "name": "string",
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
],
"parentIntentSignature": "string",
"sampleUtterances": [
    {
        "utterance": "string"
    }
],
"slotPriorities": [
    {
        "priority": number,
        "slotId": "string"
    }
]
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 373)

The identifier of the bot that contains the intent.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[botVersion \(p. 373\)](#)

The version of the bot that contains the intent. Will always be DRAFT.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

[creationDateTime \(p. 373\)](#)

A timestamp of when the intent was created.

Type: Timestamp

[description \(p. 373\)](#)

The updated description of the intent.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[dialogCodeHook \(p. 373\)](#)

The updated Lambda function called during each turn of the conversation with the user.

Type: [DialogCodeHookSettings \(p. 472\)](#) object

[fulfillmentCodeHook \(p. 373\)](#)

The updated Lambda function called when the intent is ready for fulfillment.

Type: [FulfillmentCodeHookSettings \(p. 478\)](#) object

[inputContexts \(p. 373\)](#)

The updated list of contexts that must be active for the intent to be considered by Amazon Lex.

Type: Array of [InputContext \(p. 486\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

[intentClosingSetting \(p. 373\)](#)

The updated response that Amazon Lex sends the user when the intent is closed.

Type: [IntentClosingSetting \(p. 487\)](#) object

[intentConfirmationSetting \(p. 373\)](#)

The updated prompts that Amazon Lex sends to the user to confirm the completion of an intent.

Type: [IntentConfirmationSetting \(p. 488\)](#) object

[intentId \(p. 373\)](#)

The identifier of the intent that was updated.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[intentName \(p. 373\)](#)

The updated name of the intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[kendraConfiguration \(p. 373\)](#)

The updated configuration for connecting to an Amazon Kendra index with the AMAZON.KendraSearchIntent intent.

Type: [KendraConfiguration \(p. 493\)](#) object

[lastUpdatedDateTime \(p. 373\)](#)

A timestamp of the last time that the intent was modified.

Type: Timestamp

[localeId \(p. 373\)](#)

The updated language and locale of the intent.

Type: String

[outputContexts \(p. 373\)](#)

The updated list of contexts that Amazon Lex activates when the intent is fulfilled.

Type: Array of [OutputContext \(p. 499\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

[parentIntentSignature \(p. 373\)](#)

The updated built-in intent that is the parent of this intent.

Type: String

[sampleUtterances \(p. 373\)](#)

The updated list of sample utterances for the intent.

Type: Array of [SampleUtterance \(p. 505\)](#) objects

[slotPriorities \(p. 373\)](#)

The updated list of slots and their priorities that are elicited from the user for the intent.

Type: Array of [SlotPriority \(p. 511\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateResourcePolicy

Service: Amazon Lex Model Building V2

Replaces the existing resource policy for a bot or bot alias with a new one. If the policy doesn't exist, Amazon Lex returns an exception.

Request Syntax

```
PUT /policy/resourceArn?expectedRevisionId=expectedRevisionId HTTP/1.1
Content-type: application/json

{
    "policy": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

expectedRevisionId (p. 380)

The identifier of the revision of the policy to update. If this revision ID doesn't match the current revision ID, Amazon Lex throws an exception.

If you don't specify a revision, Amazon Lex overwrites the contents of the policy with the new values.

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

resourceArn (p. 380)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Length Constraints: Minimum length of 1. Maximum length of 1011.

Required: Yes

Request Body

The request accepts the following data in JSON format.

policy (p. 380)

A resource policy to add to the resource. The policy is a JSON structure that contains one or more statements that define the policy. The policy must follow the IAM syntax. For more information about the contents of a JSON policy document, see [IAM JSON policy reference](#).

If the policy isn't valid, Amazon Lex returns a validation exception.

Type: String

Length Constraints: Minimum length of 2.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

```
Content-type: application/json

{
  "resourceArn": "string",
  "revisionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[resourceArn \(p. 380\)](#)

The Amazon Resource Name (ARN) of the bot or bot alias that the resource policy is attached to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1011.

[revisionId \(p. 380\)](#)

The current revision of the resource policy. Use the revision ID to make sure that you are updating the most current version of a resource policy when you add a policy statement to a resource, delete a resource, or update a resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ResourceNotFoundException

HTTP Status Code: 404

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateSlot

Service: Amazon Lex Model Building V2

Updates the settings for a slot.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/intents/intentId/slots/slotId/  
HTTP/1.1  
Content-type: application/json  
  
{  
    "description": "string",  
    "multipleValuesSetting": {  
        "allowMultipleValues": boolean  
    },  
    "obfuscationSetting": {  
        "obfuscationSettingType": "string"  
    },  
    "slotName": "string",  
    "slotTypeId": "string",  
    "valueElicitationSetting": {  
        "defaultValueSpecification": {  
            "defaultValueList": [  
                {  
                    "defaultValue": "string"  
                }  
            ]  
        },  
        "promptSpecification": {  
            "allowInterrupt": boolean,  
            "maxRetries": number,  
            "messageGroups": [  
                {  
                    "message": {  
                        "customPayload": {  
                            "value": "string"  
                        },  
                        "imageResponseCard": {  
                            "buttons": [  
                                {  
                                    "text": "string",  
                                    "value": "string"  
                                }  
                            ],  
                            "imageUrl": "string",  
                            "subtitle": "string",  
                            "title": "string"  
                        },  
                        "plainTextMessage": {  
                            "value": "string"  
                        },  
                        "ssmlMessage": {  
                            "value": "string"  
                        }  
                    },  
                    "variations": [  
                        {  
                            "customPayload": {  
                                "value": "string"  
                            },  
                            "imageResponseCard": {  
                                "buttons": [  
                                    {  
                                        "text": "string",  
                                        "value": "string"  
                                    }  
                                ]  
                            }  
                        }  
                    ]  
                }  
            ]  
        }  
    }  
}
```

```
        "text": "string",
        "value": "string"
    }
],
"imageUrl": "string",
"subtitle": "string",
"title": "string"
},
"plainTextMessage": {
    "value": "string"
},
"ssmlMessage": {
    "value": "string"
}
}
]
}
],
"sampleUtterances": [
{
    "utterance": "string"
}
],
"slotConstraint": "string",
"waitAndContinueSpecification": {
    "continueResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
{
        "message": {
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
{
                    "text": "string",
                    "value": "string"
                }
],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            },
            "plainTextMessage": {
                "value": "string"
            },
            "ssmlMessage": {
                "value": "string"
            }
        }
},
        "variations": [
{
            "customPayload": {
                "value": "string"
            },
            "imageResponseCard": {
                "buttons": [
{
                    "text": "string",
                    "value": "string"
                }
],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            }
}
]
}
]
}
]
```

```

        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
],
},
"stillWaitingResponse": {
    "allowInterrupt": boolean,
    "frequencyInSeconds": number,
    "messageGroups": [
        {
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": {
                    "buttons": [
                        {
                            "text": "string",
                            "value": "string"
                        }
                    ],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
                },
                "plainTextMessage": {
                    "value": "string"
                },
                "ssmlMessage": {
                    "value": "string"
                }
            },
            "variations": [
                {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                }
            ]
        }
    ],
}
]
```

```

        "timeoutInSeconds": number
    },
    "waitingResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
            {
                "message": {
                    "customPayload": {
                        "value": "string"
                    },
                    "imageResponseCard": {
                        "buttons": [
                            {
                                "text": "string",
                                "value": "string"
                            }
                        ],
                        "imageUrl": "string",
                        "subtitle": "string",
                        "title": "string"
                    },
                    "plainTextMessage": {
                        "value": "string"
                    },
                    "ssmlMessage": {
                        "value": "string"
                    }
                },
                "variations": [
                    {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    }
                ]
            }
        }
    }
}

```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 383)

The unique identifier of the bot that contains the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 383)

The version of the bot that contains the slot. Must always be DRAFT.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

intentId (p. 383)

The identifier of the intent that contains the slot.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 383)

The identifier of the language and locale that contains the slot. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

slotId (p. 383)

The unique identifier for the slot to update.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

description (p. 383)

The new description for the slot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

multipleValuesSetting (p. 383)

Determines whether the slot accepts multiple values in one response. Multiple value slots are only available in the en-US locale. If you set this value to true in any other locale, Amazon Lex throws a ValidationException.

If the `multipleValuesSetting` is not set, the default value is `false`.

Type: [MultipleValuesSetting \(p. 497\)](#) object

Required: No

[obfuscationSetting \(p. 383\)](#)

New settings that determine how slot values are formatted in Amazon CloudWatch logs.

Type: [ObfuscationSetting \(p. 498\)](#) object

Required: No

[slotName \(p. 383\)](#)

The new name for the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

Required: Yes

[slotTypeId \(p. 383\)](#)

The unique identifier of the new slot type to associate with this slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: `^(AMAZON\.)[a-zA-Z_]+?|[0-9a-zA-Z]+)$`

Required: Yes

[valueElicitationSetting \(p. 383\)](#)

A new set of prompts that Amazon Lex sends to the user to elicit a response the provides a value for the slot.

Type: [SlotValueElicitationSetting \(p. 520\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "creationDateTime": number,
    "description": "string",
    "intentId": "string",
    "lastUpdatedDateTime": number,
    "localeId": "string",
    "multipleValuesSetting": {
        "allowMultipleValues": boolean
    },
    "obfuscationSetting": {

```

```
        "obfuscationSettingType": "string"
    },
    "slotId": "string",
    "slotName": "string",
    "slotTypeId": "string",
    "valueElicitationSetting": {
        "defaultValueSpecification": {
            "defaultValueList": [
                {
                    "defaultValue": "string"
                }
            ]
        },
        "promptSpecification": {
            "allowInterrupt": boolean,
            "maxRetries": number,
            "messageGroups": [
                {
                    "message": {
                        "customPayload": {
                            "value": "string"
                        },
                        "imageResponseCard": {
                            "buttons": [
                                {
                                    "text": "string",
                                    "value": "string"
                                }
                            ],
                            "imageUrl": "string",
                            "subtitle": "string",
                            "title": "string"
                        },
                        "plainTextMessage": {
                            "value": "string"
                        },
                        "ssmlMessage": {
                            "value": "string"
                        }
                    },
                    "variations": [
                        {
                            "customPayload": {
                                "value": "string"
                            },
                            "imageResponseCard": {
                                "buttons": [
                                    {
                                        "text": "string",
                                        "value": "string"
                                    }
                                ],
                                "imageUrl": "string",
                                "subtitle": "string",
                                "title": "string"
                            },
                            "plainTextMessage": {
                                "value": "string"
                            },
                            "ssmlMessage": {
                                "value": "string"
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```

```

},
"sampleUtterances": [
{
    "utterance": "string"
}
],
"slotConstraint": "string",
"waitAndContinueSpecification": {
    "continueResponse": {
        "allowInterrupt": boolean,
        "messageGroups": [
{
            "message": {
                "customPayload": {
                    "value": "string"
                },
                "imageResponseCard": {
                    "buttons": [
{
                        "text": "string",
                        "value": "string"
}
],
                    "imageUrl": "string",
                    "subtitle": "string",
                    "title": "string"
}
},
                    "plainTextMessage": {
                        "value": "string"
}
},
                    "ssmlMessage": {
                        "value": "string"
}
}
},
        "variations": [
{
            "customPayload": {
                "value": "string"
}
},
            "imageResponseCard": {
                "buttons": [
{
                    "text": "string",
                    "value": "string"
}
],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
}
},
            "plainTextMessage": {
                "value": "string"
}
},
            "ssmlMessage": {
                "value": "string"
}
}
]
}
],
"stillWaitingResponse": {
    "allowInterrupt": boolean,
    "frequencyInSeconds": number,
    "messageGroups": [
{

```

```

    "message": {
      "customPayload": {
        "value": "string"
      },
      "imageResponseCard": {
        "buttons": [
          {
            "text": "string",
            "value": "string"
          }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
      },
      "plainTextMessage": {
        "value": "string"
      },
      "ssmlMessage": {
        "value": "string"
      }
    },
    "variations": [
      {
        "customPayload": {
          "value": "string"
        },
        "imageResponseCard": {
          "buttons": [
            {
              "text": "string",
              "value": "string"
            }
          ],
          "imageUrl": "string",
          "subtitle": "string",
          "title": "string"
        },
        "plainTextMessage": {
          "value": "string"
        },
        "ssmlMessage": {
          "value": "string"
        }
      }
    ]
  },
  "timeoutInSeconds": number
},
"waitingResponse": {
  "allowInterrupt": boolean,
  "messageGroups": [
    {
      "message": {
        "customPayload": {
          "value": "string"
        },
        "imageResponseCard": {
          "buttons": [
            {
              "text": "string",
              "value": "string"
            }
          ],
          "imageUrl": "string",
        }
      }
    }
  ]
}

```

```
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
},
"variations": [
{
    "customPayload": {
        "value": "string"
    },
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    },
    "plainTextMessage": {
        "value": "string"
    },
    "ssmlMessage": {
        "value": "string"
    }
}
]
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botId (p. 388)

The identifier of the bot that contains the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 388)

The identifier of the slot version that contains the slot. Will always be DRAFT.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

[creationDateTime \(p. 388\)](#)

The timestamp of the date and time that the slot was created.

Type: Timestamp

[description \(p. 388\)](#)

The updated description of the bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[intentId \(p. 388\)](#)

The intent that contains the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[lastUpdatedDateTime \(p. 388\)](#)

The timestamp of the date and time that the slot was last updated.

Type: Timestamp

[localeId \(p. 388\)](#)

The locale that contains the slot.

Type: String

[multipleValuesSetting \(p. 388\)](#)

Indicates whether the slot accepts multiple values in one response.

Type: [MultipleValuesSetting \(p. 497\)](#) object

[obfuscationSetting \(p. 388\)](#)

The updated setting that determines whether the slot value is obfuscated in the Amazon CloudWatch logs.

Type: [ObfuscationSetting \(p. 498\)](#) object

[slotId \(p. 388\)](#)

The unique identifier of the slot that was updated.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[slotName \(p. 388\)](#)

The updated name of the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)++\$

slotTypeId (p. 388)

The updated identifier of the slot type that provides values for the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: ^((AMAZON\.)[a-zA-Z_]+?|[0-9a-zA-Z]+)\$

valueElicitationSetting (p. 388)

The updated prompts that Amazon Lex sends to the user to elicit a response that provides a value for the slot.

Type: [SlotValueElicitationSetting \(p. 520\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerError

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

UpdateSlotType

Service: Amazon Lex Model Building V2

Updates the configuration of an existing slot type.

Request Syntax

```
PUT /bots/botId/botversions/botVersion/botlocales/localeId/slottypes/slotTypeId HTTP/1.1
Content-type: application/json

{
  "description": "string",
  "parentSlotTypeSignature": "string",
  "slotTypeName": "string",
  "slotTypeValues": [
    {
      "sampleValue": {
        "value": "string"
      },
      "synonyms": [
        {
          "value": "string"
        }
      ]
    }
  ],
  "valueSelectionSetting": {
    "regexFilter": {
      "pattern": "string"
    },
    "resolutionStrategy": "string"
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

botId (p. 396)

The identifier of the bot that contains the slot type.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion (p. 396)

The version of the bot that contains the slot type. Must be DRAFT.

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId (p. 396)

The identifier of the language and locale that contains the slot type. The string must match one of the supported locales. For more information, see [Supported languages](#).

Required: Yes

[slotTypeId \(p. 396\)](#)

The unique identifier of the slot type to update.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

Request Body

The request accepts the following data in JSON format.

[description \(p. 396\)](#)

The new description of the slot type.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

[parentSlotTypeSignature \(p. 396\)](#)

The new built-in slot type that should be used as the parent of this slot type.

Type: String

Required: No

[slotTypeName \(p. 396\)](#)

The new name of the slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

[slotTypeValues \(p. 396\)](#)

A new list of values and their optional synonyms that define the values that the slot type can take.

Type: Array of [SlotTypeValue \(p. 519\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10000 items.

Required: No

[valueSelectionSetting \(p. 396\)](#)

The strategy that Amazon Lex should use when deciding on a value from the list of slot type values.

Type: [SlotValueSelectionSetting \(p. 523\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
    "botId": "string",
    "botVersion": "string",
    "creationDateTime": number,
    "description": "string",
    "lastUpdatedDateTime": number,
    "localeId": "string",
    "parentSlotTypeSignature": "string",
    "slotTypeId": "string",
    "slotTypeName": "string",
    "slotTypeValues": [
        {
            "sampleValue": {
                "value": "string"
            },
            "synonyms": [
                {
                    "value": "string"
                }
            ]
        }
    ],
    "valueSelectionSetting": {
        "regexFilter": {
            "pattern": "string"
        },
        "resolutionStrategy": "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

botId (p. 398)

The identifier of the bot that contains the slot type.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

botVersion (p. 398)

The version of the bot that contains the slot type. This is always DRAFT.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

creationDateTime (p. 398)

The timestamp of the date and time that the slot type was created.

Type: Timestamp

[description \(p. 398\)](#)

The updated description of the slot type.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

[lastUpdatedDateTime \(p. 398\)](#)

A timestamp of the date and time that the slot type was last updated.

Type: Timestamp

[localeId \(p. 398\)](#)

The language and locale of the updated slot type.

Type: String

[parentSlotTypeSignature \(p. 398\)](#)

The updated signature of the built-in slot type that is the parent of this slot type.

Type: String

[slotTypeId \(p. 398\)](#)

The unique identifier of the updated slot type.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

[slotTypeName \(p. 398\)](#)

The updated name of the slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

[slotTypeValues \(p. 398\)](#)

The updated values that the slot type provides.

Type: Array of [SlotTypeValue \(p. 519\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10000 items.

[valueSelectionSetting \(p. 398\)](#)

The updated strategy that Amazon Lex uses to determine which value to select from the slot type.

Type: [SlotValueSelectionSetting \(p. 523\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

PreconditionFailedException

HTTP Status Code: 412

ServiceQuotaExceededException

HTTP Status Code: 402

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Amazon Lex Runtime V2

The following actions are supported by Amazon Lex Runtime V2:

- [DeleteSession \(p. 401\)](#)
- [GetSession \(p. 404\)](#)
- [PutSession \(p. 408\)](#)
- [RecognizeText \(p. 413\)](#)
- [RecognizeUtterance \(p. 419\)](#)
- [StartConversation \(p. 426\)](#)

DeleteSession

Service: Amazon Lex Runtime V2

Removes session information for a specified bot, alias, and user ID.

You can use this operation to restart a conversation with a bot. When you remove a session, the entire history of the session is removed so that you can start again.

You don't need to delete a session. Sessions have a time limit and will expire. Set the session time limit when you create the bot. The default is 5 minutes, but you can specify anything between 1 minute and 24 hours.

If you specify a bot or alias ID that doesn't exist, you receive a `BadRequestException`.

If the locale doesn't exist in the bot, or if the locale hasn't been enables for the alias, you receive a `BadRequestException`.

Request Syntax

```
DELETE /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 401)

The alias identifier in use for the bot that contains the session data.

Required: Yes

botId (p. 401)

The identifier of the bot that contains the session data.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 401)

The locale where the session is in use.

Length Constraints: Minimum length of 1.

Required: Yes

sessionId (p. 401)

The identifier of the session to delete.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "botAliasId": "string",
    "botId": "string",
    "localeId": "string",
    "sessionId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

botAliasId ([p. 402](#))

The alias identifier in use for the bot that contained the session data.

Type: String

botId ([p. 402](#))

The identifier of the bot that contained the session data.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

localeId ([p. 402](#))

The locale where the session was used.

Type: String

Length Constraints: Minimum length of 1.

sessionId ([p. 402](#))

The identifier of the deleted session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

AccessDeniedException

HTTP Status Code: 403

ConflictException

HTTP Status Code: 409

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

GetSession

Service: Amazon Lex Runtime V2

Returns session information for a specified bot, alias, and user.

For example, you can use this operation to retrieve session information for a user that has left a long-running session in use.

If the bot, alias, or session identifier doesn't exist, Amazon Lex V2 returns a `BadRequestException`. If the locale doesn't exist or is not enabled for the alias, you receive a `BadRequestException`.

Request Syntax

```
GET /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 404)

The alias identifier in use for the bot that contains the session data.

Required: Yes

botId (p. 404)

The identifier of the bot that contains the session data.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 404)

The locale where the session is in use.

Length Constraints: Minimum length of 1.

Required: Yes

sessionId (p. 404)

The identifier of the session to return.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
```

```
Content-type: application/json

{
  "interpretations": [
    {
      "intent": {
        "confirmationState": "string",
        "name": "string",
        "slots": {
          "string" : {
            "shape": "string",
            "value": {
              "interpretedValue": "string",
              "originalValue": "string",
              "resolvedValues": [ "string" ]
            },
            "values": [
              "Slot"
            ]
          }
        },
        "state": "string"
      },
      "nluConfidence": {
        "score": number
      },
      "sentimentResponse": {
        "sentiment": "string",
        "sentimentScore": {
          "mixed": number,
          "negative": number,
          "neutral": number,
          "positive": number
        }
      }
    }
  ],
  "messages": [
    {
      "content": "string",
      "contentType": "string",
      "imageResponseCard": {
        "buttons": [
          {
            "text": "string",
            "value": "string"
          }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
      }
    }
  ],
  "sessionId": "string",
  "sessionState": {
    "activeContexts": [
      {
        "contextAttributes": {
          "string" : "string"
        },
        "name": "string",
        "timeToLive": {
          "timeToLiveInSeconds": number,
          "turnsToLive": number
        }
      }
    ]
  }
}
```

```
        }
    ],
    "dialogAction": {
        "slotToElicit": "string",
        "type": "string"
    },
    "intent": {
        "confirmationState": "string",
        "name": "string",
        "slots": {
            "string" : {
                "shape": "string",
                "value": {
                    "interpretedValue": "string",
                    "originalValue": "string",
                    "resolvedValues": [ "string" ]
                },
                "values": [
                    "Slot"
                ]
            }
        },
        "state": "string"
    },
    "originatingRequestId": "string",
    "sessionAttributes": {
        "string" : "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[interpretations \(p. 404\)](#)

A list of intents that Amazon Lex V2 determined might satisfy the user's utterance.

Each interpretation includes the intent, a score that indicates how confident Amazon Lex V2 is that the interpretation is the correct one, and an optional sentiment response that indicates the sentiment expressed in the utterance.

Type: Array of [Interpretation \(p. 550\)](#) objects

Array Members: Maximum number of 5 items.

[messages \(p. 404\)](#)

A list of messages that were last sent to the user. The messages are ordered based on the order that your returned the messages from your Lambda function or the order that messages are defined in the bot.

Type: Array of [Message \(p. 551\)](#) objects

Array Members: Maximum number of 10 items.

[sessionId \(p. 404\)](#)

The identifier of the returned session.

Type: String

Length Constraints: Minimum length of 1.

[sessionState \(p. 404\)](#)

Represents the current state of the dialog between the user and the bot.

You can use this to determine the progress of the conversation and what the next action might be.

Type: [SessionState \(p. 556\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

AccessDeniedException

HTTP Status Code: 403

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

PutSession

Service: Amazon Lex Runtime V2

Creates a new session or modifies an existing session with an Amazon Lex V2 bot. Use this operation to enable your application to set the state of the bot.

Request Syntax

```
POST /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId HTTP/1.1
ResponseContentType: responseContentType
Content-type: application/json

{
  "messages": [
    {
      "content": "string",
      "contentType": "string",
      "imageResponseCard": {
        "buttons": [
          {
            "text": "string",
            "value": "string"
          }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
      }
    }
  ],
  "requestAttributes": {
    "string": "string"
  },
  "sessionState": {
    "activeContexts": [
      {
        "contextAttributes": {
          "string": "string"
        },
        "name": "string",
        "timeToLive": {
          "timeToLiveInSeconds": number,
          "turnsToLive": number
        }
      }
    ],
    "dialogAction": {
      "slotToElicit": "string",
      "type": "string"
    },
    "intent": {
      "confirmationState": "string",
      "name": "string",
      "slots": {
        "string": {
          "shape": "string",
          "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [ "string" ]
          },
          "values": [
            "Slot"
          ]
        }
      }
    }
  }
}
```

```
        }
    },
    "state": "string"
},
"originatingRequestId": "string",
"sessionAttributes": {
    "string" : "string"
}
}
```

URI Request Parameters

The request uses the following URI parameters.

[botAliasId \(p. 408\)](#)

The alias identifier of the bot that receives the session data.

Required: Yes

[botId \(p. 408\)](#)

The identifier of the bot that receives the session data.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

[localeId \(p. 408\)](#)

The locale where the session is in use.

Length Constraints: Minimum length of 1.

Required: Yes

[responseContentType \(p. 408\)](#)

The message that Amazon Lex V2 returns in the response can be either text or speech depending on the value of this parameter.

- If the value is `text/plain; charset=utf-8`, Amazon Lex V2 returns text in the response.

Length Constraints: Minimum length of 1.

[sessionId \(p. 408\)](#)

The identifier of the session that receives the session data.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: Yes

Request Body

The request accepts the following data in JSON format.

[messages \(p. 408\)](#)

A list of messages to send to the user. Messages are sent in the order that they are defined in the list.

Type: Array of [Message \(p. 551\)](#) objects

Array Members: Maximum number of 10 items.

Required: No

[requestAttributes \(p. 408\)](#)

Request-specific information passed between Amazon Lex V2 and the client application.

The namespace `x-amz-lex:` is reserved for special attributes. Don't create any request attributes with the prefix `x-amz-lex:`.

Type: String to string map

Key Length Constraints: Minimum length of 1.

Required: No

[sessionState \(p. 408\)](#)

Sets the state of the session with the user. You can use this to set the current intent, attributes, context, and dialog action. Use the dialog action to determine the next step that Amazon Lex V2 should use in the conversation with the user.

Type: [SessionState \(p. 556\)](#) object

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-Type: contentType
x-amz-lex-messages: messages
x-amz-lex-session-state: sessionState
x-amz-lex-request-attributes: requestAttributes
x-amz-lex-session-id: sessionId

audioStream
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

[contentType \(p. 410\)](#)

The type of response. Same as the type specified in the `responseContentType` field in the request.

Length Constraints: Minimum length of 1.

[messages \(p. 410\)](#)

A list of messages that were last sent to the user. The messages are ordered based on how you return the messages from your Lambda function or the order that the messages are defined in the bot.

Length Constraints: Minimum length of 1.

[requestAttributes \(p. 410\)](#)

Request-specific information passed between the client application and Amazon Lex V2. These are the same as the `requestAttribute` parameter in the call to the `PutSession` operation.

Length Constraints: Minimum length of 1.

sessionId (p. 410)

The identifier of the session that received the data.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

sessionState (p. 410)

Represents the current state of the dialog between the user and the bot.

Use this to determine the progress of the conversation and what the next action may be.

Length Constraints: Minimum length of 1.

The response returns the following as the HTTP body.

audioStream (p. 410)

If the requested content type was audio, the audio version of the message to convey to the user.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

AccessDeniedException

HTTP Status Code: 403

BadGatewayException

HTTP Status Code: 502

ConflictException

HTTP Status Code: 409

DependencyFailedException

HTTP Status Code: 424

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RecognizeText

Service: Amazon Lex Runtime V2

Sends user input to Amazon Lex V2. Client applications use this API to send requests to Amazon Lex V2 at runtime. Amazon Lex V2 then interprets the user input using the machine learning model that it build for the bot.

In response, Amazon Lex V2 returns the next message to convey to the user and an optional response card to display.

Request Syntax

```
POST /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId/text HTTP/1.1
Content-type: application/json

{
    "requestAttributes": {
        "string" : "string"
    },
    "sessionState": {
        "activeContexts": [
            {
                "contextAttributes": {
                    "string" : "string"
                },
                "name": "string",
                "timeToLive": {
                    "timeToLiveInSeconds": number,
                    "turnsToLive": number
                }
            }
        ],
        "dialogAction": {
            "slotToElicit": "string",
            "type": "string"
        },
        "intent": {
            "confirmationState": "string",
            "name": "string",
            "slots": {
                "string" : {
                    "shape": "string",
                    "value": {
                        "interpretedValue": "string",
                        "originalValue": "string",
                        "resolvedValues": [ "string" ]
                    },
                    "values": [
                        "Slot"
                    ]
                }
            },
            "state": "string"
        },
        "originatingRequestId": "string",
        "sessionAttributes": {
            "string" : "string"
        }
    },
    "text": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

[botAliasId \(p. 413\)](#)

The alias identifier in use for the bot that processes the request.

Required: Yes

[botId \(p. 413\)](#)

The identifier of the bot that processes the request.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

[localeId \(p. 413\)](#)

The locale where the session is in use.

Length Constraints: Minimum length of 1.

Required: Yes

[sessionId \(p. 413\)](#)

The identifier of the user session that is having the conversation.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: Yes

Request Body

The request accepts the following data in JSON format.

[requestAttributes \(p. 413\)](#)

Request-specific information passed between the client application and Amazon Lex V2

The namespace `x-amz-lex:` is reserved for special attributes. Don't create any request attributes with the prefix `x-amz-lex:`.

Type: String to string map

Key Length Constraints: Minimum length of 1.

Required: No

[sessionState \(p. 413\)](#)

The current state of the dialog between the user and the bot.

Type: [SessionState \(p. 556\)](#) object

Required: No

text (p. 413)

The text that the user entered. Amazon Lex V2 interprets this text.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "string",
                "name": "string",
                "slots": {
                    "string" : {
                        "shape": "string",
                        "value": {
                            "interpretedValue": "string",
                            "originalValue": "string",
                            "resolvedValues": [ "string" ]
                        },
                        "values": [
                            "Slot"
                        ]
                    }
                },
                "state": "string"
            },
            "nluConfidence": {
                "score": number
            },
            "sentimentResponse": {
                "sentiment": "string",
                "sentimentScore": {
                    "mixed": number,
                    "negative": number,
                    "neutral": number,
                    "positive": number
                }
            }
        }
    ],
    "messages": [
        {
            "content": "string",
            "contentType": "string",
            "imageResponseCard": {
                "buttons": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ],
                "imageUrl": "string",
                "subtitle": "string",
                "title": "string"
            }
        }
    ]
}
```

```
        }
    ],
    "requestAttributes": {
        "string" : "string"
    },
    "sessionId": "string",
    "sessionState": {
        "activeContexts": [
            {
                "contextAttributes": {
                    "string" : "string"
                },
                "name": "string",
                "timeToLive": {
                    "timeToLiveInSeconds": number,
                    "turnsToLive": number
                }
            }
        ],
        "dialogAction": {
            "slotToElicit": "string",
            "type": "string"
        },
        "intent": {
            "confirmationState": "string",
            "name": "string",
            "slots": {
                "string" : {
                    "shape": "string",
                    "value": {
                        "interpretedValue": "string",
                        "originalValue": "string",
                        "resolvedValues": [ "string" ]
                    },
                    "values": [
                        "Slot"
                    ]
                }
            },
            "state": "string"
        },
        "originatingRequestId": "string",
        "sessionAttributes": {
            "string" : "string"
        }
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[interpretations \(p. 415\)](#)

A list of intents that Amazon Lex V2 determined might satisfy the user's utterance.

Each interpretation includes the intent, a score that indicates how confident Amazon Lex V2 is that the interpretation is the correct one, and an optional sentiment response that indicates the sentiment expressed in the utterance.

Type: Array of [Interpretation \(p. 550\)](#) objects

Array Members: Maximum number of 5 items.

messages (p. 415)

A list of messages last sent to the user. The messages are ordered based on the order that you returned the messages from your Lambda function or the order that the messages are defined in the bot.

Type: Array of [Message \(p. 551\)](#) objects

Array Members: Maximum number of 10 items.

requestAttributes (p. 415)

The attributes sent in the request.

Type: String to string map

Key Length Constraints: Minimum length of 1.

sessionId (p. 415)

The identifier of the session in use.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

sessionState (p. 415)

Represents the current state of the dialog between the user and the bot.

Use this to determine the progress of the conversation and what the next action may be.

Type: [SessionState \(p. 556\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

AccessDeniedException

HTTP Status Code: 403

BadGatewayException

HTTP Status Code: 502

ConflictException

HTTP Status Code: 409

DependencyFailedException

HTTP Status Code: 424

InternalServerError

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RecognizeUtterance

Service: Amazon Lex Runtime V2

Sends user input to Amazon Lex V2. You can send text or speech. Clients use this API to send text and audio requests to Amazon Lex V2 at runtime. Amazon Lex V2 interprets the user input using the machine learning model built for the bot.

The following request fields must be compressed with gzip and then base64 encoded before you send them to Amazon Lex V2.

- `requestAttributes`
- `sessionState`

The following response fields are compressed using gzip and then base64 encoded by Amazon Lex V2. Before you can use these fields, you must decode and decompress them.

- `inputTranscript`
- `interpretations`
- `messages`
- `requestAttributes`
- `sessionState`

The example contains a Java application that compresses and encodes a Java object to send to Amazon Lex V2, and a second that decodes and decompresses a response from Amazon Lex V2.

Request Syntax

```
POST /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId/utterance
HTTP/1.1
x-amz-lex-session-state: sessionState
x-amz-lex-request-attributes: requestAttributes
Content-Type: requestContentType
Response-Content-Type: responseContentType

inputStream
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 419)

The alias identifier in use for the bot that should receive the request.

Required: Yes

botId (p. 419)

The identifier of the bot that should receive the request.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

localeId (p. 419)

The locale where the session is in use.

Length Constraints: Minimum length of 1.

Required: Yes

[requestAttributes \(p. 419\)](#)

Request-specific information passed between the client application and Amazon Lex V2

The namespace `x-amz-lex:` is reserved for special attributes. Don't create any request attributes for prefix `x-amz-lex::`.

The `requestAttributes` field must be compressed using gzip and then base64 encoded before sending to Amazon Lex V2.

[requestContentType \(p. 419\)](#)

Indicates the format for audio input or that the content is text. The header must start with one of the following prefixes:

- PCM format, audio data must be in little-endian byte order.
 - `audio/l16; rate=16000; channels=1`
 - `audio/x-l16; sample-rate=16000; channel-count=1`
 - `audio/lpcm; sample-rate=8000; sample-size-bits=16; channel-count=1; is-big-endian=false`
- Opus format
 - `audio/x-cbr-opus-with-preamble;preamble-size=0;bit-rate=256000;frame-size-milliseconds=4`
- Text format
 - `text/plain; charset=utf-8`

Length Constraints: Minimum length of 1.

Required: Yes

[responseContentType \(p. 419\)](#)

The message that Amazon Lex V2 returns in the response can be either text or speech based on the `responseContentType` value.

- If the value is `text/plain; charset=utf-8`, Amazon Lex V2 returns text in the response.
- If the value begins with `audio/`, Amazon Lex V2 returns speech in the response. Amazon Lex V2 uses Amazon Polly to generate the speech using the configuration that you specified in the `requestContentType` parameter. For example, if you specify `audio/mpeg` as the value, Amazon Lex V2 returns speech in the MPEG format.
- If the value is `audio/pcm`, the speech returned is `audio/pcm` at 16 KHz in 16-bit, little-endian format.
- The following are the accepted values:
 - `audio/mpeg`
 - `audio/ogg`
 - `audio/pcm (16 KHz)`
 - `audio/* (defaults to mpeg)`
 - `text/plain; charset=utf-8`

Length Constraints: Minimum length of 1.

[sessionId \(p. 419\)](#)

The identifier of the session in use.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: `[0-9a-zA-Z._:-]+`

Required: Yes

[sessionState \(p. 419\)](#)

Sets the state of the session with the user. You can use this to set the current intent, attributes, context, and dialog action. Use the dialog action to determine the next step that Amazon Lex V2 should use in the conversation with the user.

The `sessionState` field must be compressed using gzip and then base64 encoded before sending to Amazon Lex V2.

Request Body

The request accepts the following binary data.

[inputStream \(p. 419\)](#)

User input in PCM or Opus audio format or text format as described in the `requestContentType` parameter.

Response Syntax

```
HTTP/1.1 200
x-amz-lex-input-mode: inputMode
Content-Type: contentType
x-amz-lex-messages: messages
x-amz-lex-interpretations: interpretations
x-amz-lex-session-state: sessionState
x-amz-lex-request-attributes: requestAttributes
x-amz-lex-session-id: sessionId
x-amz-lex-input-transcript: inputTranscript


```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

[contentType \(p. 421\)](#)

Content type as specified in the `responseContentType` in the request.

Length Constraints: Minimum length of 1.

[inputMode \(p. 421\)](#)

Indicates whether the input mode to the operation was text or speech.

Length Constraints: Minimum length of 1.

[inputTranscript \(p. 421\)](#)

The text used to process the request.

If the input was an audio stream, the `inputTranscript` field contains the text extracted from the audio stream. This is the text that is actually processed to recognize intents and slot values. You can use this information to determine if Amazon Lex V2 is correctly processing the audio that you send.

The `inputTranscript` field is compressed with gzip and then base64 encoded. Before you can use the contents of the field, you must decode and decompress the contents. See the example for a simple function to decode and decompress the contents.

Length Constraints: Minimum length of 1.

[interpretations \(p. 421\)](#)

A list of intents that Amazon Lex V2 determined might satisfy the user's utterance.

Each interpretation includes the intent, a score that indicates how confident Amazon Lex V2 is that the interpretation is the correct one, and an optional sentiment response that indicates the sentiment expressed in the utterance.

The `interpretations` field is compressed with gzip and then base64 encoded. Before you can use the contents of the field, you must decode and decompress the contents. See the example for a simple function to decode and decompress the contents.

Length Constraints: Minimum length of 1.

[messages \(p. 421\)](#)

A list of messages that were last sent to the user. The messages are ordered based on the order that you returned the messages from your Lambda function or the order that the messages are defined in the bot.

The `messages` field is compressed with gzip and then base64 encoded. Before you can use the contents of the field, you must decode and decompress the contents. See the example for a simple function to decode and decompress the contents.

Length Constraints: Minimum length of 1.

[requestAttributes \(p. 421\)](#)

The attributes sent in the request.

The `requestAttributes` field is compressed with gzip and then base64 encoded. Before you can use the contents of the field, you must decode and decompress the contents.

Length Constraints: Minimum length of 1.

[sessionId \(p. 421\)](#)

The identifier of the session in use.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

[sessionState \(p. 421\)](#)

Represents the current state of the dialog between the user and the bot.

Use this to determine the progress of the conversation and what the next action might be.

The `sessionState` field is compressed with gzip and then base64 encoded. Before you can use the contents of the field, you must decode and decompress the contents. See the example for a simple function to decode and decompress the contents.

Length Constraints: Minimum length of 1.

The response returns the following as the HTTP body.

[audioStream \(p. 421\)](#)

The prompt or statement to send to the user. This is based on the bot configuration and context. For example, if Amazon Lex V2 did not understand the user intent, it sends the `clarificationPrompt` configured for the bot. If the intent requires confirmation before taking the fulfillment action, it sends the `confirmationPrompt`. Another example: Suppose that the Lambda function successfully fulfilled the intent, and sent a message to convey to the user. Then Amazon Lex V2 sends that message in the response.

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

AccessDeniedException

HTTP Status Code: 403

BadGatewayException

HTTP Status Code: 502

ConflictException

HTTP Status Code: 409

DependencyFailedException

HTTP Status Code: 424

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 404

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

Examples

[Encode and decode a field](#)

The following example provides two functions, one to compress a string with gzip and then base64 encode it and one to decode and decompress a string.

```
package com.amazonaws.deepsense.util;

import com.amazonaws.deepsense.runtime.conversation.serialize.RuntimeSdkSerializer;
import com.fasterxml.jackson.annotation.JsonInclude;
```

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.google.common.base.Preconditions;

import java.io.IOException;
import java.util.Base64;

public class CompressionAndEncoding {

    private CompressionAndEncoding() {

    }

    /**
     * Given a generic object
     * 1. Serialize the object using jackson
     * 2. Compress the serialized object
     * 3. Base64 encode the compressed data
     */
    public static String compressAndEncodeBase64(Object object) {
        if (object == null) {
            return null;
        }

        String objectAsString = GsonSerializer.serialize(object);
        byte[] compressedString = Compressions.compress(objectAsString);
        return Base64.getEncoder().encodeToString(compressedString);
    }

    /**
     * Given a base64 encoded, compressed, serialized object
     * 1. Base64 decodes the data
     * 2. Decompresses the base64 decoded data
     * 3. Converts the serialized object into a proper POJO
     */
    public static <T> T decodeBase64AndDecompress(String objectAsString, TypeReference<T>
typeRef) {
        if (objectAsString == null) {
            return null;
        }

        Preconditions.checkNotNull(typeRef, "Serialization class can't be null.");
        byte[] decodedBytes = Base64.getDecoder().decode(objectAsString);
        String decompressedObjectAsString = Compressions.uncompress(decodedBytes);
        try {
            return RuntimeSdkSerializer.instance().readValue(decompressedObjectAsString,
typeRef);
        } catch (IOException e) {
            throw new RuntimeException(String.format("Unable to deserialize string %s",
decompressedObjectAsString), e);
        }
    }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartConversation

Service: Amazon Lex Runtime V2

Starts an HTTP/2 bidirectional event stream that enables you to send audio, text, or DTMF input in real time. After your application starts a conversation, users send input to Amazon Lex V2 as a stream of events. Amazon Lex V2 processes the incoming events and responds with streaming text or audio events.

Audio input must be in the following format: `audio/lpcm sample-rate=8000 sample-size-bits=16 channel-count=1; is-big-endian=false`.

The `StartConversation` operation is supported only in the following SDKs:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Request Syntax

```
POST /bots/botId/botAliases/botAliasId/botLocales/localeId/sessions/sessionId/conversation
HTTP/2
x-amz-lex-conversation-mode: conversationMode
Content-type: application/json

{
    "requestEventStream": {
        "AudioInputEvent": {
            "audioChunk": blob,
            "clientTimestampMillis": number,
            "contentType": "string",
            "eventId": "string"
        },
        "ConfigurationEvent": {
            "clientTimestampMillis": number,
            "disablePlayback": boolean,
            "eventId": "string",
            "requestAttributes": {
                "string": "string"
            },
            "responseContentType": "string",
            "sessionState": {
                "activeContexts": [
                    {
                        "contextAttributes": {
                            "string": "string"
                        },
                        "name": "string",
                        "timeToLive": {
                            "timeToLiveInSeconds": number,
                            "turnsToLive": number
                        }
                    }
                ],
                "dialogAction": {
                    "slotToElicit": "string",
                    "type": "string"
                },
                "intent": {
                    "confirmationState": "string",
                    "name": "string",
                    "slots": {
                        "string": {

```

```
        "shape": "string",
        "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [ "string" ]
        },
        "values": [
            "Slot"
        ]
    }
},
"state": "string"
},
"originatingRequestId": "string",
"sessionAttributes": {
    "string" : "string"
}
},
"welcomeMessages": [
{
    "content": "string",
    "contentType": "string",
    "imageResponseCard": {
        "buttons": [
            {
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subtitle": "string",
        "title": "string"
    }
}
],
"DisconnectionEvent": {
    "clientTimestampMillis": number,
    "eventId": "string"
},
"DTMFInputEvent": {
    "clientTimestampMillis": number,
    "eventId": "string",
    "inputCharacter": "string"
},
"PlaybackCompletionEvent": {
    "clientTimestampMillis": number,
    "eventId": "string"
},
"TextInputEvent": {
    "clientTimestampMillis": number,
    "eventId": "string",
    "text": "string"
}
}
}
```

URI Request Parameters

The request uses the following URI parameters.

botAliasId (p. 426)

The alias identifier in use for the bot that processes the request.

Required: Yes

[botId \(p. 426\)](#)

The identifier of the bot to process the request.

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

[conversationMode \(p. 426\)](#)

The conversation type that you are using the Amazon Lex V2. If the conversation mode is `AUDIO` you can send both audio and DTMF information. If the mode is `TEXT` you can only send text.

Valid Values: `AUDIO` | `TEXT`

[localeId \(p. 426\)](#)

The locale where the session is in use.

Length Constraints: Minimum length of 1.

Required: Yes

[sessionId \(p. 426\)](#)

The identifier of the user session that is having the conversation.

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: Yes

Request Body

The request accepts the following data in JSON format.

[requestEventStream \(p. 426\)](#)

Represents the stream of events to Amazon Lex V2 from your application. The events are encoded as HTTP/2 data frames.

Type: [StartConversationRequestEventStream \(p. 559\)](#) object

Required: Yes

Response Syntax

```
HTTP/2 200
Content-type: application/json

{
    "responseEventStream": {
        "AccessDeniedException": {
        },
        "AudioResponseEvent": {
            "audioChunk": blob,
```

```
        "contentType": "string",
        "eventId": "string"
    },
    "BadGatewayException": {
    },
    "ConflictException": {
    },
    "DependencyFailedException": {
    },
    "HeartbeatEvent": {
        "eventId": "string"
    },
    "IntentResultEvent": {
        "eventId": "string",
        "inputMode": "string",
        "interpretations": [
            {
                "intent": {
                    "confirmationState": "string",
                    "name": "string",
                    "slots": {
                        "string": {
                            "shape": "string",
                            "value": {
                                "interpretedValue": "string",
                                "originalValue": "string",
                                "resolvedValues": [ "string" ]
                            },
                            "values": [
                                "Slot"
                            ]
                        }
                    },
                    "state": "string"
                },
                "nluConfidence": {
                    "score": number
                },
                "sentimentResponse": {
                    "sentiment": "string",
                    "sentimentsScore": {
                        "mixed": number,
                        "negative": number,
                        "neutral": number,
                        "positive": number
                    }
                }
            }
        ],
        "requestAttributes": {
            "string" : "string"
        },
        "sessionId": "string",
        "sessionState": {
            "activeContexts": [
                {
                    "contextAttributes": {
                        "string" : "string"
                    },
                    "name": "string",
                    "timeToLive": {
                        "timeToLiveInSeconds": number,
                        "turnsToLive": number
                    }
                }
            ]
        }
    }
},
```

```
"dialogAction": {  
    "slotToElicit": "string",  
    "type": "string"  
},  
"intent": {  
    "confirmationState": "string",  
    "name": "string",  
    "slots": {  
        "string" : {  
            "shape": "string",  
            "value": {  
                "interpretedValue": "string",  
                "originalValue": "string",  
                "resolvedValues": [ "string" ]  
            },  
            "values": [  
                "Slot"  
            ]  
        }  
    },  
    "state": "string"  
},  
"originatingRequestId": "string",  
"sessionAttributes": {  
    "string" : "string"  
}  
}  
}  
},  
"InternalServerException": {  
},  
"PlaybackInterruptionEvent": {  
    "causedByEventId": "string",  
    "eventId": "string",  
    "eventReason": "string"  
},  
"ResourceNotFoundException": {  
},  
"TextResponseEvent": {  
    "eventId": "string",  
    "messages": [  
        {  
            "content": "string",  
            "contentType": "string",  
            "imageResponseCard": {  
                "buttons": [  
                    {  
                        "text": "string",  
                        "value": "string"  
                    }  
                ],  
                "imageUrl": "string",  
                "subtitle": "string",  
                "title": "string"  
            }  
        }  
    ]  
},  
"ThrottlingException": {  
},  
"TranscriptEvent": {  
    "eventId": "string",  
    "transcript": "string"  
},  
"ValidationException": {  
}  
}
```

}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

responseEventStream (p. 428)

Represents the stream of events from Amazon Lex V2 to your application. The events are encoded as HTTP/2 data frames.

Type: [StartConversationResponseEventStream \(p. 561\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 567\)](#).

AccessDeniedException

HTTP Status Code: 403

InternalServerError

HTTP Status Code: 500

ThrottlingException

HTTP Status Code: 429

ValidationException

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Data Types

The following data types are supported by Amazon Lex Model Building V2:

- [AudioLogDestination \(p. 437\)](#)
- [AudioLogSetting \(p. 438\)](#)
- [BotAliasHistoryEvent \(p. 439\)](#)
- [BotAliasLocaleSettings \(p. 440\)](#)
- [BotAliasSummary \(p. 441\)](#)
- [BotExportSpecification \(p. 443\)](#)
- [BotFilter \(p. 444\)](#)
- [BotImportSpecification \(p. 445\)](#)
- [BotLocaleExportSpecification \(p. 447\)](#)
- [BotLocaleFilter \(p. 448\)](#)
- [BotLocaleHistoryEvent \(p. 449\)](#)
- [BotLocaleImportSpecification \(p. 450\)](#)
- [BotLocaleSortBy \(p. 452\)](#)
- [BotLocaleSummary \(p. 453\)](#)
- [BotSortBy \(p. 455\)](#)
- [BotSummary \(p. 456\)](#)
- [BotVersionLocaleDetails \(p. 458\)](#)
- [BotVersionSortBy \(p. 459\)](#)
- [BotVersionSummary \(p. 460\)](#)
- [BuiltInIntentSortBy \(p. 462\)](#)
- [BuiltInIntentSummary \(p. 463\)](#)
- [BuiltInSlotTypeSortBy \(p. 464\)](#)
- [BuiltInSlotTypeSummary \(p. 465\)](#)
- [Button \(p. 466\)](#)
- [CloudWatchLogGroupLogDestination \(p. 467\)](#)
- [CodeHookSpecification \(p. 468\)](#)
- [ConversationLogSettings \(p. 469\)](#)
- [CustomPayload \(p. 470\)](#)
- [DataPrivacy \(p. 471\)](#)
- [DialogCodeHookSettings \(p. 472\)](#)
- [ExportFilter \(p. 473\)](#)
- [ExportResourceSpecification \(p. 474\)](#)
- [ExportSortBy \(p. 475\)](#)
- [ExportSummary \(p. 476\)](#)
- [FulfillmentCodeHookSettings \(p. 478\)](#)
- [ImageResponseCard \(p. 479\)](#)
- [ImportFilter \(p. 481\)](#)
- [ImportResourceSpecification \(p. 482\)](#)
- [ImportSortBy \(p. 483\)](#)
- [ImportSummary \(p. 484\)](#)
- [InputContext \(p. 486\)](#)
- [IntentClosingSetting \(p. 487\)](#)
- [IntentConfirmationSetting \(p. 488\)](#)
- [IntentFilter \(p. 489\)](#)
- [IntentSortBy \(p. 490\)](#)
- [IntentSummary \(p. 491\)](#)

- [KendraConfiguration \(p. 493\)](#)
- [LambdaCodeHook \(p. 494\)](#)
- [Message \(p. 495\)](#)
- [MessageGroup \(p. 496\)](#)
- [MultipleValuesSetting \(p. 497\)](#)
- [ObfuscationSetting \(p. 498\)](#)
- [OutputContext \(p. 499\)](#)
- [PlainTextMessage \(p. 500\)](#)
- [Principal \(p. 501\)](#)
- [PromptSpecification \(p. 502\)](#)
- [ResponseSpecification \(p. 503\)](#)
- [S3BucketLogDestination \(p. 504\)](#)
- [SampleUtterance \(p. 505\)](#)
- [SampleValue \(p. 506\)](#)
- [SentimentAnalysisSettings \(p. 507\)](#)
- [SlotDefaultValue \(p. 508\)](#)
- [SlotDefaultValueSpecification \(p. 509\)](#)
- [SlotFilter \(p. 510\)](#)
- [SlotPriority \(p. 511\)](#)
- [SlotSortBy \(p. 512\)](#)
- [SlotSummary \(p. 513\)](#)
- [SlotTypeFilter \(p. 515\)](#)
- [SlotTypeSortBy \(p. 516\)](#)
- [SlotTypeSummary \(p. 517\)](#)
- [SlotTypeValue \(p. 519\)](#)
- [SlotValueElicitationSetting \(p. 520\)](#)
- [SlotValueRegexFilter \(p. 522\)](#)
- [SlotValueSelectionSetting \(p. 523\)](#)
- [SSMLMessage \(p. 524\)](#)
- [StillWaitingResponseSpecification \(p. 525\)](#)
- [TextLogDestination \(p. 526\)](#)
- [TextLogSetting \(p. 527\)](#)
- [VoiceSettings \(p. 528\)](#)
- [WaitAndContinueSpecification \(p. 529\)](#)

The following data types are supported by Amazon Lex Runtime V2:

- [ActiveContext \(p. 531\)](#)
- [ActiveContextTimeToLive \(p. 533\)](#)
- [AudioInputEvent \(p. 534\)](#)
- [AudioResponseEvent \(p. 536\)](#)
- [Button \(p. 537\)](#)
- [ConfidenceScore \(p. 538\)](#)
- [ConfigurationEvent \(p. 539\)](#)
- [DialogAction \(p. 541\)](#)
- [DisconnectionEvent \(p. 542\)](#)

- [DTMInputEvent \(p. 543\)](#)
- [HeartbeatEvent \(p. 544\)](#)
- [ImageResponseCard \(p. 545\)](#)
- [Intent \(p. 547\)](#)
- [IntentResultEvent \(p. 548\)](#)
- [Interpretation \(p. 550\)](#)
- [Message \(p. 551\)](#)
- [PlaybackCompletionEvent \(p. 552\)](#)
- [PlaybackInterruptionEvent \(p. 553\)](#)
- [SentimentResponse \(p. 554\)](#)
- [SentimentScore \(p. 555\)](#)
- [SessionState \(p. 556\)](#)
- [Slot \(p. 558\)](#)
- [StartConversationRequestEventStream \(p. 559\)](#)
- [StartConversationResponseEventStream \(p. 561\)](#)
- [TextInputEvent \(p. 564\)](#)
- [TextResponseEvent \(p. 565\)](#)
- [TranscriptEvent \(p. 566\)](#)
- [Value \(p. 567\)](#)

Amazon Lex Model Building V2

The following data types are supported by Amazon Lex Model Building V2:

- [AudioLogDestination \(p. 437\)](#)
- [AudioLogSetting \(p. 438\)](#)
- [BotAliasHistoryEvent \(p. 439\)](#)
- [BotAliasLocaleSettings \(p. 440\)](#)
- [BotAliasSummary \(p. 441\)](#)
- [BotExportSpecification \(p. 443\)](#)
- [BotFilter \(p. 444\)](#)
- [BotImportSpecification \(p. 445\)](#)
- [BotLocaleExportSpecification \(p. 447\)](#)
- [BotLocaleFilter \(p. 448\)](#)
- [BotLocaleHistoryEvent \(p. 449\)](#)
- [BotLocaleImportSpecification \(p. 450\)](#)
- [BotLocaleSortBy \(p. 452\)](#)
- [BotLocaleSummary \(p. 453\)](#)
- [BotSortBy \(p. 455\)](#)
- [BotSummary \(p. 456\)](#)
- [BotVersionLocaleDetails \(p. 458\)](#)
- [BotVersionSortBy \(p. 459\)](#)
- [BotVersionSummary \(p. 460\)](#)
- [BuiltInIntentSortBy \(p. 462\)](#)
- [BuiltInIntentSummary \(p. 463\)](#)
- [BuiltInSlotTypeSortBy \(p. 464\)](#)

- [BuiltInSlotTypeSummary \(p. 465\)](#)
- [Button \(p. 466\)](#)
- [CloudWatchLogGroupLogDestination \(p. 467\)](#)
- [CodeHookSpecification \(p. 468\)](#)
- [ConversationLogSettings \(p. 469\)](#)
- [CustomPayload \(p. 470\)](#)
- [DataPrivacy \(p. 471\)](#)
- [DialogCodeHookSettings \(p. 472\)](#)
- [ExportFilter \(p. 473\)](#)
- [ExportResourceSpecification \(p. 474\)](#)
- [ExportSortBy \(p. 475\)](#)
- [ExportSummary \(p. 476\)](#)
- [FulfillmentCodeHookSettings \(p. 478\)](#)
- [ImageResponseCard \(p. 479\)](#)
- [ImportFilter \(p. 481\)](#)
- [ImportResourceSpecification \(p. 482\)](#)
- [ImportSortBy \(p. 483\)](#)
- [ImportSummary \(p. 484\)](#)
- [InputContext \(p. 486\)](#)
- [IntentClosingSetting \(p. 487\)](#)
- [IntentConfirmationSetting \(p. 488\)](#)
- [IntentFilter \(p. 489\)](#)
- [IntentSortBy \(p. 490\)](#)
- [IntentSummary \(p. 491\)](#)
- [KendraConfiguration \(p. 493\)](#)
- [LambdaCodeHook \(p. 494\)](#)
- [Message \(p. 495\)](#)
- [MessageGroup \(p. 496\)](#)
- [MultipleValuesSetting \(p. 497\)](#)
- [ObfuscationSetting \(p. 498\)](#)
- [OutputContext \(p. 499\)](#)
- [PlainTextMessage \(p. 500\)](#)
- [Principal \(p. 501\)](#)
- [PromptSpecification \(p. 502\)](#)
- [ResponseSpecification \(p. 503\)](#)
- [S3BucketLogDestination \(p. 504\)](#)
- [SampleUtterance \(p. 505\)](#)
- [SampleValue \(p. 506\)](#)
- [SentimentAnalysisSettings \(p. 507\)](#)
- [SlotDefaultValue \(p. 508\)](#)
- [SlotDefaultValueSpecification \(p. 509\)](#)
- [SlotFilter \(p. 510\)](#)
- [SlotPriority \(p. 511\)](#)
- [SlotSortBy \(p. 512\)](#)
- [SlotSummary \(p. 513\)](#)
- [SlotTypeFilter \(p. 515\)](#)

- [SlotTypeSortBy \(p. 516\)](#)
- [SlotTypeSummary \(p. 517\)](#)
- [SlotTypeValue \(p. 519\)](#)
- [SlotValueElicitationSetting \(p. 520\)](#)
- [SlotValueRegexFilter \(p. 522\)](#)
- [SlotValueSelectionSetting \(p. 523\)](#)
- [SSMLMessage \(p. 524\)](#)
- [StillWaitingResponseSpecification \(p. 525\)](#)
- [TextLogDestination \(p. 526\)](#)
- [TextLogSetting \(p. 527\)](#)
- [VoiceSettings \(p. 528\)](#)
- [WaitAndContinueSpecification \(p. 529\)](#)

AudioLogDestination

Service: Amazon Lex Model Building V2

The location of audio log files collected when conversation logging is enabled for a bot.

Contents

s3Bucket

The Amazon S3 bucket where the audio log files are stored. The IAM role specified in the `roleArn` parameter of the [CreateBot \(p. 168\)](#) operation must have permission to write to this bucket.

Type: [S3BucketLogDestination \(p. 504\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

AudioLogSetting

Service: Amazon Lex Model Building V2

Settings for logging audio of conversations between Amazon Lex and a user. You specify whether to log audio and the Amazon S3 bucket where the audio file is stored.

Contents

destination

The location of audio log files collected when conversation logging is enabled for a bot.

Type: [AudioLogDestination \(p. 437\)](#) object

Required: Yes

enabled

Determines whether audio logging is enabled for the bot.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotAliasHistoryEvent

Service: Amazon Lex Model Building V2

Provides a record of an event that affects a bot alias. For example, when the version of a bot that the alias points to changes.

Contents

botVersion

The version of the bot that was used in the event.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: No

endDate

The date and time that the event ended.

Type: Timestamp

Required: No

startDate

The date and time that the event started.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotAliasLocaleSettings

Service: Amazon Lex Model Building V2

Specifies settings that are unique to a locale. For example, you can use different Lambda function depending on the bot's locale.

Contents

codeHookSpecification

Specifies the Lambda function that should be used in the locale.

Type: [CodeHookSpecification \(p. 468\)](#) object

Required: No

enabled

Determines whether the locale is enabled for the bot. If the value is `false`, the locale isn't available for use.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotAliasSummary

Service: Amazon Lex Model Building V2

Summary information about bot aliases returned from the [ListBotAliases \(p. 301\)](#) operation.

Contents

botAliasId

The unique identifier assigned to the bot alias. You can use this ID to get detailed information about the alias using the [DescribeBotAlias \(p. 262\)](#) operation.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^(\bTSTALIASID\b|[0-9a-zA-Z]+)\$

Required: No

botAliasName

The name of the bot alias.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

botAliasStatus

The current state of the bot alias. If the status is Available, the alias is ready for use.

Type: String

Valid Values: Creating | Available | Deleting | Failed

Required: No

botVersion

The version of the bot that the bot alias references.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: No

creationDateTime

A timestamp of the date and time that the bot alias was created.

Type: Timestamp

Required: No

description

The description of the bot alias.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

lastUpdatedDateTime

A timestamp of the date and time that the bot alias was last updated.

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotExportSpecification

Service: Amazon Lex Model Building V2

Provided the identity of a the bot that was exported.

Contents

botId

The identifier of the bot assigned by Amazon Lex.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion

The version of the bot that was exported. This will be either DRAFT or the version number.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotFilter

Service: Amazon Lex Model Building V2

Filters the responses returned by the `ListBots` operation.

Contents

name

The name of the field to filter the list of bots.

Type: String

Valid Values: `BotName`

Required: Yes

operator

The operator to use for the filter. Specify `EQ` when the `ListBots` operation should return only aliases that equal the specified value. Specify `CO` when the `ListBots` operation should return aliases that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The value to use for filtering the list of bots.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotImportSpecification

Service: Amazon Lex Model Building V2

Provides the bot parameters required for importing a bot.

Contents

botName

The name that Amazon Lex should use for the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

botTags

A list of tags to add to the bot. You can only add tags when you import a bot. You can't use the `UpdateBot` operation to update tags. To update tags, use the `TagResource` operation.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Required: No

dataPrivacy

By default, data stored by Amazon Lex is encrypted. The `DataPrivacy` structure provides settings that determine how Amazon Lex handles special cases of securing the data for your bot.

Type: [DataPrivacy \(p. 471\)](#) object

Required: Yes

idleSessionTTLInSeconds

The time, in seconds, that Amazon Lex should keep information about a user's conversation with the bot.

A user interaction remains active for the amount of time specified. If no conversation occurs during this time, the session expires and Amazon Lex deletes any data provided before the timeout.

You can specify between 60 (1 minute) and 86,400 (24 hours) seconds.

Type: Integer

Valid Range: Minimum value of 60. Maximum value of 86400.

Required: No

roleArn

The Amazon Resource Name (ARN) of the IAM role used to build and run the bot.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: `^arn:aws:iam::[0-9]{12}:role/.*$`

Required: Yes

testBotAliasTags

A list of tags to add to the test alias for a bot. You can only add tags when you import a bot. You can't use the `UpdateAlias` operation to update tags. To update tags on the test alias, use the `TagResource` operation.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 200 items.

Key Length Constraints: Minimum length of 1. Maximum length of 128.

Value Length Constraints: Minimum length of 0. Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotLocaleExportSpecification

Service: Amazon Lex Model Building V2

Provides the bot locale parameters required for exporting a bot locale.

Contents

botId

The identifier of the bot to create the locale for.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion

The version of the bot to export.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

localeId

The identifier of the language and locale to export. The string must match one of the locales in the bot.

Type: String

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotLocaleFilter

Service: Amazon Lex Model Building V2

Filters responses returned by the `ListBotLocales` operation.

Contents

name

The name of the field to filter the list of bots.

Type: String

Valid Values: `BotLocaleName`

Required: Yes

operator

The operator to use for the filter. Specify `EQ` when the `ListBotLocales` operation should return only aliases that equal the specified value. Specify `CO` when the `ListBotLocales` operation should return aliases that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The value to use for filtering the list of bots.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotLocaleHistoryEvent

Service: Amazon Lex Model Building V2

Provides information about an event that occurred affecting the bot locale.

Contents

event

A description of the event that occurred.

Type: String

Required: Yes

eventDate

A timestamp of the date and time that the event occurred.

Type: Timestamp

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotLocaleImportSpecification

Service: Amazon Lex Model Building V2

Provides the bot locale parameters required for importing a bot locale.

Contents

botId

The identifier of the bot to import the locale to.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

botVersion

The version of the bot to import the locale to. This can only be the DRAFT version of the bot.

Type: String

Length Constraints: Fixed length of 5.

Pattern: ^DRAFT\$

Required: Yes

localeId

The identifier of the language and locale that the bot will be used in. The string must match one of the supported locales. All of the intents, slot types, and slots used in the bot must have the same locale. For more information, see [Supported languages](#).

Type: String

Required: Yes

nluIntentConfidenceThreshold

Determines the threshold where Amazon Lex will insert the AMAZON.FallbackIntent, AMAZON.KendraSearchIntent, or both when returning alternative intents.

AMAZON.FallbackIntent and AMAZON.KendraSearchIntent are only inserted if they are configured for the bot.

For example, suppose a bot is configured with the confidence threshold of 0.80 and the AMAZON.FallbackIntent. Amazon Lex returns three alternative intents with the following confidence scores: IntentA (0.70), IntentB (0.60), IntentC (0.50). The response from the PostText operation would be:

- AMAZON.FallbackIntent
- IntentA
- IntentB
- IntentC

Type: Double

Valid Range: Minimum value of 0. Maximum value of 1.

Required: No

voiceSettings

Defines settings for using an Amazon Polly voice to communicate with a user.

Type: [VoiceSettings \(p. 528\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotLocaleSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of bot locales.

Contents

attribute

The bot locale attribute to sort by.

Type: String

Valid Values: `BotLocaleName`

Required: Yes

order

Specifies whether to sort the bot locales in ascending or descending order.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotLocaleSummary

Service: Amazon Lex Model Building V2

Summary information about bot locales returned by the [ListBotLocales \(p. 304\)](#) operation.

Contents

botLocaleStatus

The current status of the bot locale. When the status is Built the locale is ready for use.

Type: String

Valid Values: Creating | Building | Built | ReadyExpressTesting | Failed | Deleting | NotBuilt | Importing

Required: No

description

The description of the bot locale.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

lastBuildSubmittedDateTime

A timestamp of the date and time that the bot locale was last built.

Type: Timestamp

Required: No

lastUpdatedDateTime

A timestamp of the date and time that the bot locale was last updated.

Type: Timestamp

Required: No

localeId

The language and locale of the bot locale.

Type: String

Required: No

localeName

The name of the bot locale.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of bots.

Contents

attribute

The attribute to use to sort the list of bots.

Type: String

Valid Values: `BotName`

Required: Yes

order

The order to sort the list. You can choose ascending or descending.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotSummary

Service: Amazon Lex Model Building V2

Summary information about a bot returned by the [ListBots \(p. 308\)](#) operation.

Contents

botId

The unique identifier assigned to the bot. Use this ID to get detailed information about the bot with the [DescribeBot \(p. 259\)](#) operation.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

botName

The name of the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

botStatus

The current status of the bot. When the status is Available the bot is ready for use.

Type: String

Valid Values: Creating | Available | Inactive | Deleting | Failed | Versioning | Importing

Required: No

description

The description of the bot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

lastUpdatedDateTime

The date and time that the bot was last updated.

Type: Timestamp

Required: No

latestBotVersion

The latest numerical version in use for the bot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^[0-9]+\$

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotVersionLocaleDetails

Service: Amazon Lex Model Building V2

The version of a bot used for a bot locale.

Contents

sourceBotVersion

The version of a bot used for a bot locale.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotVersionSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of bot versions.

Contents

attribute

The attribute to use to sort the list of versions.

Type: String

Valid Values: `BotVersion`

Required: Yes

order

The order to sort the list. You can specify ascending or descending order.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BotVersionSummary

Service: Amazon Lex Model Building V2

Summary information about a bot version returned by the [ListBotVersions \(p. 311\)](#) operation.

Contents

botName

The name of the bot associated with the version.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

botStatus

The status of the bot. When the status is available, the version of the bot is ready for use.

Type: String

Valid Values: `Creating` | `Available` | `Inactive` | `Deleting` | `Failed` | `Versioning` | `Importing`

Required: No

botVersion

The numeric version of the bot, or `DRAFT` to indicate that this is the version of the bot that can be updated..

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Pattern: ^(DRAFT|[0-9]+)\$

Required: No

creationDateTime

A timestamp of the date and time that the version was created.

Type: Timestamp

Required: No

description

The description of the version.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BuiltInIntentSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of built-in intents.

Contents

attribute

The attribute to use to sort the list of built-in intents.

Type: String

Valid Values: `IntentSignature`

Required: Yes

order

The order to sort the list. You can specify ascending or descending order.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BuiltInIntentSummary

Service: Amazon Lex Model Building V2

Provides summary information about a built-in intent for the [ListBuiltInIntents \(p. 314\)](#) operation.

Contents

description

The description of the intent.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

intentSignature

The signature of the built-in intent. Use this to specify the parent intent of a derived intent.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BuiltInSlotTypeSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of built-in slot types.

Contents

attribute

The attribute to use to sort the list of built-in intents.

Type: String

Valid Values: `SlotTypeSignature`

Required: Yes

order

The order to sort the list. You can choose ascending or descending.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

BuiltInSlotTypeSummary

Service: Amazon Lex Model Building V2

Provides summary information about a built-in slot type for the [ListBuiltInSlotTypes \(p. 317\)](#) operation.

Contents

description

The description of the built-in slot type.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

slotTypeSignature

The signature of the built-in slot type. Use this to specify the parent slot type of a derived slot type.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Button

Service: Amazon Lex Model Building V2

Describes a button to use on a response card used to gather slot values from a user.

Contents

text

The text that appears on the button. Use this to tell the user what value is returned when they choose this button.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

value

The value returned to Amazon Lex when the user chooses this button. This must be one of the slot values configured for the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

CloudWatchLogGroupLogDestination

Service: Amazon Lex Model Building V2

The Amazon CloudWatch Logs log group where the text and metadata logs are delivered. The log group must exist before you enable logging.

Contents

cloudWatchLogGroupArn

The Amazon Resource Name (ARN) of the log group where text and metadata logs are delivered.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: ^arn:[\w\ -]+:logs:[\w\ -]+:[\d]{12}:log-group:[\.\ \-_/#A-Za-z0-9]{1,512}(?:\ *?)\$

Required: Yes

logPrefix

The prefix of the log stream name within the log group that you specified

Type: String

Length Constraints: Maximum length of 1024.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

CodeHookSpecification

Service: Amazon Lex Model Building V2

Contains information about code hooks that Amazon Lex calls during a conversation.

Contents

lambdaCodeHook

Specifies a Lambda function that verifies requests to a bot or fulfills the user's request to a bot.

Type: [LambdaCodeHook \(p. 494\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ConversationLogSettings

Service: Amazon Lex Model Building V2

Configures conversation logging that saves audio, text, and metadata for the conversations with your users.

Contents

audioLogSettings

The Amazon S3 settings for logging audio to an S3 bucket.

Type: Array of [AudioLogSetting \(p. 438\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

textLogSettings

The Amazon CloudWatch Logs settings for logging text and metadata.

Type: Array of [TextLogSetting \(p. 527\)](#) objects

Array Members: Fixed number of 1 item.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

CustomPayload

Service: Amazon Lex Model Building V2

A custom response string that Amazon Lex sends to your application. You define the content and structure the string.

Contents

value

The string that is sent to your application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DataPrivacy

Service: Amazon Lex Model Building V2

By default, data stored by Amazon Lex is encrypted. The `DataPrivacy` structure provides settings that determine how Amazon Lex handles special cases of securing the data for your bot.

Contents

childDirected

For each Amazon Lex bot created with the Amazon Lex Model Building Service, you must specify whether your use of Amazon Lex is related to a website, program, or other application that is directed or targeted, in whole or in part, to children under age 13 and subject to the Children's Online Privacy Protection Act (COPPA) by specifying `true` or `false` in the `childDirected` field. By specifying `true` in the `childDirected` field, you confirm that your use of Amazon Lex **is** related to a website, program, or other application that is directed or targeted, in whole or in part, to children under age 13 and subject to COPPA. By specifying `false` in the `childDirected` field, you confirm that your use of Amazon Lex **is not** related to a website, program, or other application that is directed or targeted, in whole or in part, to children under age 13 and subject to COPPA. You may not specify a default value for the `childDirected` field that does not accurately reflect whether your use of Amazon Lex is related to a website, program, or other application that is directed or targeted, in whole or in part, to children under age 13 and subject to COPPA. If your use of Amazon Lex relates to a website, program, or other application that is directed in whole or in part, to children under age 13, you must obtain any required verifiable parental consent under COPPA. For information regarding the use of Amazon Lex in connection with websites, programs, or other applications that are directed or targeted, in whole or in part, to children under age 13, see the [Amazon Lex FAQ](#).

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DialogCodeHookSettings

Service: Amazon Lex Model Building V2

Settings that determine the Lambda function that Amazon Lex uses for processing user responses.

Contents

enabled

Enables the dialog code hook so that it processes user requests.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ExportFilter

Service: Amazon Lex Model Building V2

Filtes the response form the [ListExports \(p. 320\)](#) operation

Contents

name

The name of the field to use for filtering.

Type: String

Valid Values: `ExportResourceType`

Required: Yes

operator

The operator to use for the filter. Specify EQ when the `ListExports` operation should return only resource types that equal the specified value. Specify CO when the `ListExports` operation should return resource types that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The values to use to fileter the response.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ExportResourceSpecification

Service: Amazon Lex Model Building V2

Provides information about the bot or bot locale that you want to export. You can specify the `botExportSpecification` or the `botLocaleExportSpecification`, but not both.

Contents

botExportSpecification

Parameters for exporting a bot.

Type: [BotExportSpecification \(p. 443\)](#) object

Required: No

botLocaleExportSpecification

Parameters for exporting a bot locale.

Type: [BotLocaleExportSpecification \(p. 447\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ExportSortBy

Service: Amazon Lex Model Building V2

Provides information about sorting a list of exports.

Contents

attribute

The export field to use for sorting.

Type: String

Valid Values: `LastUpdatedDateTime`

Required: Yes

order

The order to sort the list.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ExportSummary

Service: Amazon Lex Model Building V2

Provides summary information about an export in an export list.

Contents

creationDateTime

The date and time that the export was created.

Type: Timestamp

Required: No

exportId

The unique identifier that Amazon Lex assigned to the export.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

exportStatus

The status of the export. When the status is Completed the export is ready to download.

Type: String

Valid Values: InProgress | Completed | Failed | Deleting

Required: No

fileFormat

The file format used in the export files.

Type: String

Valid Values: LexJson

Required: No

lastUpdatedDateTime

The date and time that the export was last updated.

Type: Timestamp

Required: No

resourceSpecification

Information about the bot or bot locale that was exported.

Type: [ExportResourceSpecification \(p. 474\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

FulfillmentCodeHookSettings

Service: Amazon Lex Model Building V2

Determines if a Lambda function should be invoked for a specific intent.

Contents

enabled

Indicates whether a Lambda function should be invoked to fulfill a specific intent.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImageResponseCard

Service: Amazon Lex Model Building V2

A card that is shown to the user by a messaging platform. You define the contents of the card, the card is displayed by the platform.

When you use a response card, the response from the user is constrained to the text associated with a button on the card.

Contents

buttons

A list of buttons that should be displayed on the response card. The arrangement of the buttons is determined by the platform that displays the button.

Type: Array of [Button \(p. 466\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

Required: No

imageUrl

The URL of an image to display on the response card. The image URL must be publicly available so that the platform displaying the response card has access to the image.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 250.

Required: No

subtitle

The subtitle to display on the response card. The format of the subtitle is determined by the platform displaying the response card.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 250.

Required: No

title

The title to display on the response card. The format of the title is determined by the platform displaying the response card.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 250.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImportFilter

Service: Amazon Lex Model Building V2

Filters the response from the [ListImports \(p. 324\)](#) operation.

Contents

name

The name of the field to use for filtering.

Type: String

Valid Values: `ImportResourceType`

Required: Yes

operator

The operator to use for the filter. Specify EQ when the `ListImports` operation should return only resource types that equal the specified value. Specify CO when the `ListImports` operation should return resource types that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The values to use to filter the response.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImportResourceSpecification

Service: Amazon Lex Model Building V2

Provides information about the bot or bot locale that you want to import. You can specify the `botImportSpecification` or the `botLocaleImportSpecification`, but not both.

Contents

botImportSpecification

Parameters for importing a bot.

Type: [BotImportSpecification \(p. 445\)](#) object

Required: No

botLocaleImportSpecification

Parameters for importing a bot locale.

Type: [BotLocaleImportSpecification \(p. 450\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImportSortBy

Service: Amazon Lex Model Building V2

Provides information for sorting a list of imports.

Contents

attribute

The export field to use for sorting.

Type: String

Valid Values: `LastUpdatedDateTime`

Required: Yes

order

The order to sort the list.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImportSummary

Service: Amazon Lex Model Building V2

Provides summary information about an import in an import list.

Contents

creationDateTime

The date and time that the import was created.

Type: Timestamp

Required: No

importedResourceId

The unique identifier that Amazon Lex assigned to the imported resource.

Type: String

Length Constraints: Minimum length of 5. Maximum length of 10.

Pattern: ^([0-9a-zA-Z_])+\$

Required: No

importedResourceName

The name that you gave the imported resource.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

importId

The unique identifier that Amazon Lex assigned to the import.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

importStatus

The status of the resource. When the status is Completed the resource is ready to build.

Type: String

Valid Values: InProgress | Completed | Failed | Deleting

Required: No

lastUpdatedDateTime

The date and time that the import was last updated.

Type: Timestamp

Required: No

mergeStrategy

The strategy used to merge existing bot or bot locale definitions with the imported definition.

Type: String

Valid Values: `Overwrite` | `FailOnConflict`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

InputContext

Service: Amazon Lex Model Building V2

The name of a context that must be active for an intent to be selected by Amazon Lex.

Contents

name

The name of the context.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IntentClosingSetting

Service: Amazon Lex Model Building V2

Provides a statement the Amazon Lex conveys to the user when the intent is successfully fulfilled.

Contents

closingResponse

The response that Amazon Lex sends to the user when the intent is complete.

Type: [ResponseSpecification \(p. 503\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IntentConfirmationSetting

Service: Amazon Lex Model Building V2

Provides a prompt for making sure that the user is ready for the intent to be fulfilled.

Contents

declinationResponse

When the user answers "no" to the question defined in `promptSpecification`, Amazon Lex responds with this response to acknowledge that the intent was canceled.

Type: [ResponseSpecification \(p. 503\)](#) object

Required: Yes

promptSpecification

Prompts the user to confirm the intent. This question should have a yes or no answer.

Amazon Lex uses this prompt to ensure that the user acknowledges that the intent is ready for fulfillment. For example, with the `OrderPizza` intent, you might want to confirm that the order is correct before placing it. For other intents, such as intents that simply respond to user questions, you might not need to ask the user for confirmation before providing the information.

Type: [PromptSpecification \(p. 502\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IntentFilter

Service: Amazon Lex Model Building V2

Filters the response from the `ListIntents` operation.

Contents

name

The name of the field to use for the filter.

Type: String

Valid Values: `IntentName`

Required: Yes

operator

The operator to use for the filter. Specify `EQ` when the `ListIntents` operation should return only aliases that equal the specified value. Specify `CO` when the `ListIntents` operation should return aliases that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The value to use for the filter.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IntentSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of intents.

Contents

attribute

The attribute to use to sort the list of intents.

Type: String

Valid Values: IntentName | LastUpdatedDateTime

Required: Yes

order

The order to sort the list. You can choose ascending or descending.

Type: String

Valid Values: Ascending | Descending

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IntentSummary

Service: Amazon Lex Model Building V2

Summary information about an intent returned by the `ListIntents` operation.

Contents

description

The description of the intent.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

inputContexts

The input contexts that must be active for this intent to be considered for recognition.

Type: Array of [InputContext \(p. 486\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

Required: No

intentId

The unique identifier assigned to the intent. Use this ID to get detailed information about the intent with the `DescribeIntent` operation.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

intentName

The name of the intent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

lastUpdatedDateTime

The timestamp of the date and time that the intent was last updated.

Type: Timestamp

Required: No

outputContexts

The output contexts that are activated when this intent is fulfilled.

Type: Array of [OutputContext \(p. 499\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: No

parentIntentSignature

If this intent is derived from a built-in intent, the name of the parent intent.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

KendraConfiguration

Service: Amazon Lex Model Building V2

Provides configuration information for the AMAZON.KendraSearchIntent intent. When you use this intent, Amazon Lex searches the specified Amazon Kendra index and returns documents from the index that match the user's utterance.

Contents

kendraIndex

The Amazon Resource Name (ARN) of the Amazon Kendra index that you want the AMAZON.KendraSearchIntent intent to search. The index must be in the same account and Region as the Amazon Lex bot.

Type: String

Length Constraints: Minimum length of 32. Maximum length of 2048.

Pattern: ^arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9_-]*\$

Required: Yes

queryFilterString

A query filter that Amazon Lex sends to Amazon Kendra to filter the response from a query. The filter is in the format defined by Amazon Kendra. For more information, see [Filtering queries](#).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5000.

Required: No

queryFilterStringEnabled

Determines whether the AMAZON.KendraSearchIntent intent uses a custom query string to query the Amazon Kendra index.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

LambdaCodeHook

Service: Amazon Lex Model Building V2

Specifies a Lambda function that verifies requests to a bot or fulfills the user's request to a bot.

Contents

codeHookInterfaceVersion

The version of the request-response that you want Amazon Lex to use to invoke your Lambda function.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5.

Required: Yes

lambdaARN

The Amazon Resource Name (ARN) of the Lambda function.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws:lambda:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:function:[a-zA-Z0-9-_]+(/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})?(:[a-zA-Z0-9-_]+)?`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Message

Service: Amazon Lex Model Building V2

The object that provides message text and its type.

Contents

customPayload

A message in a custom format defined by the client application.

Type: [CustomPayload \(p. 470\)](#) object

Required: No

imageResponseCard

A message that defines a response card that the client application can show to the user.

Type: [ImageResponseCard \(p. 479\)](#) object

Required: No

plainTextMessage

A message in plain text format.

Type: [PlainTextMessage \(p. 500\)](#) object

Required: No

ssmlMessage

A message in Speech Synthesis Markup Language (SSML).

Type: [SSMLMessage \(p. 524\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

MessageGroup

Service: Amazon Lex Model Building V2

Provides one or more messages that Amazon Lex should send to the user.

Contents

message

The primary message that Amazon Lex should send to the user.

Type: [Message \(p. 495\)](#) object

Required: Yes

variations

Message variations to send to the user. When variations are defined, Amazon Lex chooses the primary message or one of the variations to send to the user.

Type: Array of [Message \(p. 495\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 2 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

MultipleValuesSetting

Service: Amazon Lex Model Building V2

Indicates whether a slot can return multiple values.

Contents

allowMultipleValues

Indicates whether a slot can return multiple values. When `true`, the slot may return more than one value in a response. When `false`, the slot returns only a single value.

Multi-value slots are only available in the en-US locale. If you set this value to `true` in any other locale, Amazon Lex throws a `ValidationException`.

If the `allowMultipleValues` is not set, the default value is `false`.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ObfuscationSetting

Service: Amazon Lex Model Building V2

Determines whether Amazon Lex obscures slot values in conversation logs.

Contents

obfuscationSettingType

Value that determines whether Amazon Lex obscures slot values in conversation logs. The default is to obscure the values.

Type: String

Valid Values: `None` | `DefaultObfuscation`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

OutputContext

Service: Amazon Lex Model Building V2

Describes a session context that is activated when an intent is fulfilled.

Contents

name

The name of the output context.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: Yes

timeToLiveInSeconds

The amount of time, in seconds, that the output context should remain active. The time is figured from the first time the context is sent to the user.

Type: Integer

Valid Range: Minimum value of 5. Maximum value of 86400.

Required: Yes

turnsToLive

The number of conversation turns that the output context should remain active. The number of turns is counted from the first time that the context is sent to the user.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 20.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PlainTextMessage

Service: Amazon Lex Model Building V2

Defines an ASCII text message to send to the user.

Contents

value

The message to send to the user.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Principal

Service: Amazon Lex Model Building V2

The IAM principal that you allowing or denying access to an Amazon Lex action. You must provide a service or an arn, but not both in the same statement. For more information, see [AWS JSON policy elements: Principal](#).

Contents

arn

The Amazon Resource Name (ARN) of the principal.

Type: String

Length Constraints: Minimum length of 30. Maximum length of 1024.

Pattern: ^arn:aws:iam::[0-9]{12}:(root|(user|role)/.*)\$

Required: No

service

The name of the AWS service that should allowed or denied access to an Amazon Lex action.

Type: String

Length Constraints: Minimum length of 15. Maximum length of 1024.

Pattern: ^[0-9a-zA-Z_.]+\$

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PromptSpecification

Service: Amazon Lex Model Building V2

Specifies a list of message groups that Amazon Lex sends to a user to elicit a response.

Contents

allowInterrupt

Indicates whether the user can interrupt a speech prompt from the bot.

Type: Boolean

Required: No

maxRetries

The maximum number of times the bot tries to elicit a response from the user using this prompt.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 5.

Required: Yes

messageGroups

A collection of messages that Amazon Lex can send to the user. Amazon Lex chooses the actual message to send at runtime.

Type: Array of [MessageGroup \(p. 496\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ResponseSpecification

Service: Amazon Lex Model Building V2

Specifies a list of message groups that Amazon Lex uses to respond the user input.

Contents

allowInterrupt

Indicates whether the user can interrupt a speech response from Amazon Lex.

Type: Boolean

Required: No

messageGroups

A collection of responses that Amazon Lex can send to the user. Amazon Lex chooses the actual response to send at runtime.

Type: Array of [MessageGroup \(p. 496\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

S3BucketLogDestination

Service: Amazon Lex Model Building V2

Specifies an Amazon S3 bucket for logging audio conversations

Contents

kmsKeyArn

The Amazon Resource Name (ARN) of an AWS Key Management Service (KMS) key for encrypting audio log files stored in an S3 bucket.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `^arn:[\w\ -]+:kms:[\w\ -]+\:[\d]{12}:(?:key\/[\w\ -]+\|alias\/[a-zA-Z0-9:\ /_\ -]\{1,256\})$`

Required: No

logPrefix

The S3 prefix to assign to audio log files.

Type: String

Length Constraints: Maximum length of 1024.

Required: Yes

s3BucketArn

The Amazon Resource Name (ARN) of an Amazon S3 bucket where audio log files are stored.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^arn:[\w\ -]+:s3:::[a-zA-Z0-9][\.\-\a-zA-Z0-9]\{1,61\}[a-zA-Z0-9]$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SampleUtterance

Service: Amazon Lex Model Building V2

A sample utterance that invokes an intent or respond to a slot elicitation prompt.

Contents

utterance

The sample utterance that Amazon Lex uses to build its machine-learning model to recognize intents.

Type: String

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SampleValue

Service: Amazon Lex Model Building V2

Defines one of the values for a slot type.

Contents

value

The value that can be used for a slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 140.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SentimentAnalysisSettings

Service: Amazon Lex Model Building V2

Determines whether Amazon Lex will use Amazon Comprehend to detect the sentiment of user utterances.

Contents

detectSentiment

Sets whether Amazon Lex uses Amazon Comprehend to detect the sentiment of user utterances.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotDefaultValue

Service: Amazon Lex Model Building V2

Specifies the default value to use when a user doesn't provide a value for a slot.

Contents

defaultValue

The default value to use when a user doesn't provide a value for a slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 202.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotDefaultValueSpecification

Service: Amazon Lex Model Building V2

Defines a list of values that Amazon Lex should use as the default value for a slot.

Contents

defaultValueList

A list of default values. Amazon Lex chooses the default value to use in the order that they are presented in the list.

Type: Array of [SlotDefaultValue \(p. 508\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 10 items.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotFilter

Service: Amazon Lex Model Building V2

Filters the response from the `ListSlots` operation.

Contents

name

The name of the field to use for filtering.

Type: String

Valid Values: `SlotName`

Required: Yes

operator

The operator to use for the filter. Specify `EQ` when the `ListSlots` operation should return only aliases that equal the specified value. Specify `CO` when the `ListSlots` operation should return aliases that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The value to use to filter the response.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotPriority

Service: Amazon Lex Model Building V2

Sets the priority that Amazon Lex should use when eliciting slot values from a user.

Contents

priority

The priority that a slot should be elicited.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 25.

Required: Yes

slotId

The unique identifier of the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of bots.

Contents

attribute

The attribute to use to sort the list.

Type: String

Valid Values: SlotName | LastUpdatedDateTime

Required: Yes

order

The order to sort the list. You can choose ascending or descending.

Type: String

Valid Values: Ascending | Descending

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotSummary

Service: Amazon Lex Model Building V2

Summary information about a slot, a value that the bot elicits from the user.

Contents

description

The description of the slot.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

lastUpdatedDateTime

The timestamp of the last date and time that the slot was updated.

Type: Timestamp

Required: No

slotConstraint

Whether the slot is required or optional. An intent is complete when all required slots are filled.

Type: String

Valid Values: Required | Optional

Required: No

slotId

The unique identifier of the slot.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

slotName

The name given to the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

slotTypeId

The unique identifier for the slot type that defines the values for the slot.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 25.

Pattern: ^((AMAZON\.)[a-zA-Z_]+?|[0-9a-zA-Z]+)\$

Required: No

valueElicitationPromptSpecification

Prompts that are sent to the user to elicit a value for the slot.

Type: [PromptSpecification \(p. 502\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotTypeFilter

Service: Amazon Lex Model Building V2

Filters the response from the `ListSlotTypes` operation.

Contents

name

The name of the field to use for filtering.

Type: String

Valid Values: `SlotTypeName`

Required: Yes

operator

The operator to use for the filter. Specify `EQ` when the `ListSlotTypes` operation should return only aliases that equal the specified value. Specify `CO` when the `ListSlotTypes` operation should return aliases that contain the specified value.

Type: String

Valid Values: `CO` | `EQ`

Required: Yes

values

The value to use to filter the response.

Type: Array of strings

Array Members: Fixed number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^ [0-9a-zA-Z_()\\s-]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotTypeSortBy

Service: Amazon Lex Model Building V2

Specifies attributes for sorting a list of slot types.

Contents

attribute

The attribute to use to sort the list of slot types.

Type: String

Valid Values: `SlotTypeName` | `LastUpdatedDateTime`

Required: Yes

order

The order to sort the list. You can say ascending or descending.

Type: String

Valid Values: `Ascending` | `Descending`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotTypeSummary

Service: Amazon Lex Model Building V2

Provides summary information about a slot type.

Contents

description

The description of the slot type.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 200.

Required: No

lastUpdatedDateTime

A timestamp of the date and time that the slot type was last updated.

Type: Timestamp

Required: No

parentSlotTypeSignature

If the slot type is derived from a built-on slot type, the name of the parent slot type.

Type: String

Required: No

slotTypeId

The unique identifier assigned to the slot type.

Type: String

Length Constraints: Fixed length of 10.

Pattern: ^[0-9a-zA-Z]+\$

Required: No

slotTypeName

The name of the slot type.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([0-9a-zA-Z][_-]?)+\$

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotTypeValue

Service: Amazon Lex Model Building V2

Each slot type can have a set of values. Each `SlotTypeValue` represents a value that the slot type can take.

Contents

sampleValue

The value of the slot type entry.

Type: [SampleValue \(p. 506\)](#) object

Required: No

synonyms

Additional values related to the slot type entry.

Type: Array of [SampleValue \(p. 506\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10000 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotValueElicitationSetting

Service: Amazon Lex Model Building V2

Settings that you can use for eliciting a slot value.

Contents

defaultValueSpecification

A list of default values for a slot. Default values are used when Amazon Lex hasn't determined a value for a slot. You can specify default values from context variables, session attributes, and defined values.

Type: [SlotDefaultValueSpecification \(p. 509\)](#) object

Required: No

promptSpecification

The prompt that Amazon Lex uses to elicit the slot value from the user.

Type: [PromptSpecification \(p. 502\)](#) object

Required: No

sampleUtterances

If you know a specific pattern that users might respond to an Amazon Lex request for a slot value, you can provide those utterances to improve accuracy. This is optional. In most cases, Amazon Lex is capable of understanding user utterances.

Type: Array of [SampleUtterance \(p. 505\)](#) objects

Required: No

slotConstraint

Specifies whether the slot is required or optional.

Type: String

Valid Values: Required | Optional

Required: Yes

waitAndContinueSpecification

Specifies the prompts that Amazon Lex uses while a bot is waiting for customer input.

Type: [WaitAndContinueSpecification \(p. 529\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotValueRegexFilter

Service: Amazon Lex Model Building V2

Provides a regular expression used to validate the value of a slot.

Contents

pattern

A regular expression used to validate the value of a slot.

Use a standard regular expression. Amazon Lex supports the following characters in the regular expression:

- A-Z, a-z
- 0-9
- Unicode characters ("\\ u<Unicode>")

Represent Unicode characters with four digits, for example "\u0041" or "\u005A".

The following regular expression operators are not supported:

- Infinite repeaters: *, +, or {x,} with no upper bound.
- Wild card (.)

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SlotValueSelectionSetting

Service: Amazon Lex Model Building V2

Contains settings used by Amazon Lex to select a slot value.

Contents

regexFilter

A regular expression used to validate the value of a slot.

Type: [SlotValueRegexFilter \(p. 522\)](#) object

Required: No

resolutionStrategy

Determines the slot resolution strategy that Amazon Lex uses to return slot type values. The field can be set to one of the following values:

- OriginalValue - Returns the value entered by the user, if the user value is similar to the slot value.
- TopResolution - If there is a resolution list for the slot, return the first value in the resolution list as the slot type value. If there is no resolution list, null is returned.

If you don't specify the valueSelectionStrategy, the default is OriginalValue.

Type: String

Valid Values: OriginalValue | TopResolution

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SSMLMessage

Service: Amazon Lex Model Building V2

Defines a Speech Synthesis Markup Language (SSML) prompt.

Contents

value

The SSML text that defines the prompt.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

StillWaitingResponseSpecification

Service: Amazon Lex Model Building V2

Defines the messages that Amazon Lex sends to a user to remind them that the bot is waiting for a response.

Contents

allowInterrupt

Indicates that the user can interrupt the response by speaking while the message is being played.

Type: Boolean

Required: No

frequencyInSeconds

How often a message should be sent to the user. Minimum of 1 second, maximum of 5 minutes.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 300.

Required: Yes

messageGroups

One or more message groups, each containing one or more messages, that define the prompts that Amazon Lex sends to the user.

Type: Array of [MessageGroup \(p. 496\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: Yes

timeoutInSeconds

If Amazon Lex waits longer than this length of time for a response, it will stop sending messages.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 900.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TextLogDestination

Service: Amazon Lex Model Building V2

Defines the Amazon CloudWatch Logs destination log group for conversation text logs.

Contents

cloudWatch

Defines the Amazon CloudWatch Logs log group where text and metadata logs are delivered.

Type: [CloudWatchLogGroupLogDestination \(p. 467\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TextLogSetting

Service: Amazon Lex Model Building V2

Defines settings to enable text conversation logs.

Contents

destination

Defines the Amazon CloudWatch Logs destination log group for conversation text logs.

Type: [TextLogDestination \(p. 526\)](#) object

Required: Yes

enabled

Determines whether conversation logs should be stored for an alias.

Type: Boolean

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

VoiceSettings

Service: Amazon Lex Model Building V2

Defines settings for using an Amazon Polly voice to communicate with a user.

Contents

voiceld

The identifier of the Amazon Polly voice to use.

Type: String

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

WaitAndContinueSpecification

Service: Amazon Lex Model Building V2

Specifies the prompts that Amazon Lex uses while a bot is waiting for customer input.

Contents

continueResponse

The response that Amazon Lex sends to indicate that the bot is ready to continue the conversation.

Type: [ResponseSpecification \(p. 503\)](#) object

Required: Yes

stillWaitingResponse

A response that Amazon Lex sends periodically to the user to indicate that the bot is still waiting for input from the user.

Type: [StillWaitingResponseSpecification \(p. 525\)](#) object

Required: No

waitingResponse

The response that Amazon Lex sends to indicate that the bot is waiting for the conversation to continue.

Type: [ResponseSpecification \(p. 503\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Lex Runtime V2

The following data types are supported by Amazon Lex Runtime V2:

- [ActiveContext \(p. 531\)](#)
- [ActiveContextTimeToLive \(p. 533\)](#)
- [AudioInputEvent \(p. 534\)](#)
- [AudioResponseEvent \(p. 536\)](#)
- [Button \(p. 537\)](#)
- [ConfidenceScore \(p. 538\)](#)
- [ConfigurationEvent \(p. 539\)](#)
- [DialogAction \(p. 541\)](#)
- [DisconnectionEvent \(p. 542\)](#)

- [DTMFInputEvent \(p. 543\)](#)
- [HeartbeatEvent \(p. 544\)](#)
- [ImageResponseCard \(p. 545\)](#)
- [Intent \(p. 547\)](#)
- [IntentResultEvent \(p. 548\)](#)
- [Interpretation \(p. 550\)](#)
- [Message \(p. 551\)](#)
- [PlaybackCompletionEvent \(p. 552\)](#)
- [PlaybackInterruptionEvent \(p. 553\)](#)
- [SentimentResponse \(p. 554\)](#)
- [SentimentScore \(p. 555\)](#)
- [SessionState \(p. 556\)](#)
- [Slot \(p. 558\)](#)
- [StartConversationRequestEventStream \(p. 559\)](#)
- [StartConversationResponseEventStream \(p. 561\)](#)
- [TextInputEvent \(p. 564\)](#)
- [TextResponseEvent \(p. 565\)](#)
- [TranscriptEvent \(p. 566\)](#)
- [Value \(p. 567\)](#)

ActiveContext

Service: Amazon Lex Runtime V2

Contains information about the contexts that a user is using in a session. You can configure Amazon Lex V2 to set a context when an intent is fulfilled, or you can set a context using the [PutSession \(p. 408\)](#), [RecognizeText \(p. 413\)](#), or [RecognizeUtterance \(p. 419\)](#) operations.

Use a context to indicate to Amazon Lex V2 intents that should be used as follow-up intents. For example, if the active context is `order-fulfilled`, only intents that have `order-fulfilled` configured as a trigger are considered for follow up.

Contents

contextAttributes

A list of contexts active for the request. A context can be activated when a previous intent is fulfilled, or by including the context in the request.

If you don't specify a list of contexts, Amazon Lex will use the current list of contexts for the session. If you specify an empty list, all contexts for the session are cleared.

Type: String to string map

Map Entries: Minimum number of 0 items. Maximum number of 10 items.

Key Length Constraints: Minimum length of 1. Maximum length of 100.

Value Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: Yes

name

The name of the context.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: ^([A-Za-z]_?)+\$

Required: Yes

timeToLive

Indicates the number of turns or seconds that the context is active. Once the time to live expires, the context is no longer returned in a response.

Type: [ActiveContextTimeToLive \(p. 533\)](#) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ActiveContextTimeToLive

Service: Amazon Lex Runtime V2

The time that a context is active. You can specify the time to live in seconds or in conversation turns.

Contents

timeToLiveInSeconds

The number of seconds that the context is active. You can specify between 5 and 86400 seconds (24 hours).

Type: Integer

Valid Range: Minimum value of 5. Maximum value of 86400.

Required: Yes

turnsToLive

The number of turns that the context is active. You can specify up to 20 turns. Each request and response from the bot is a turn.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 20.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

AudioInputEvent

Service: Amazon Lex Runtime V2

Represents a chunk of audio sent from the client application to Amazon Lex V2. The audio is all or part of an utterance from the user.

Amazon Lex V2 accumulates audio chunks until it recognizes a natural pause in speech before processing the input.

Contents

audioChunk

An encoded stream of audio.

Type: Base64-encoded binary data object

Required: No

clientTimestampMillis

A timestamp set by the client of the date and time that the event was sent to Amazon Lex V2.

Type: Long

Required: No

contentType

The encoding used for the audio chunk. You must use 8 KHz PCM 16-bit mono-channel little-endian format. The value of the field should be:

```
audio/lpcm; sample-rate=8000; sample-size-bits=16; channel-count=1; is-big-endian=false
```

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

eventId

A unique identifier that your application assigns to the event. You can use this to identify events in logs.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)

- [AWS SDK for Ruby V3](#)

AudioResponseEvent

Service: Amazon Lex Runtime V2

An event sent from Amazon Lex V2 to your client application containing audio to play to the user.

Contents

audioChunk

A chunk of the audio to play.

Type: Base64-encoded binary data object

Required: No

contentType

The encoding of the audio chunk. This is the same as the encoding configure in the `contentType` field of the `ConfigurationEvent`.

Type: String

Length Constraints: Minimum length of 1.

Required: No

eventId

A unique identifier of the event sent by Amazon Lex V2. The identifier is in the form `RESPONSE-N`, where N is a number starting with one and incremented for each event sent by Amazon Lex V2 in the current session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: `[0-9a-zA-Z._:-]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Button

Service: Amazon Lex Runtime V2

A button that appears on a response card show to the user.

Contents

text

The text that is displayed on the button.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

value

The value returned to Amazon Lex V2 when a user chooses the button.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ConfidenceScore

Service: Amazon Lex Runtime V2

Provides a score that indicates the confidence that Amazon Lex V2 has that an intent is the one that satisfies the user's intent.

Contents

score

A score that indicates how confident Amazon Lex V2 is that an intent satisfies the user's intent. Ranges between 0.00 and 1.00. Higher scores indicate higher confidence.

Type: Double

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ConfigurationEvent

Service: Amazon Lex Runtime V2

The initial event sent from the application to Amazon Lex V2 to configure the conversation, including session and request attributes and the response content type.

Contents

clientTimestampMillis

A timestamp set by the client of the date and time that the event was sent to Amazon Lex V2.

Type: Long

Required: No

disablePlayback

Determines whether Amazon Lex V2 should send audio responses to the client application. When this parameter is `false`, the client application needs to create responses for the user.

Type: Boolean

Required: No

eventId

A unique identifier that your application assigns to the event. You can use this to identify events in logs.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

requestAttributes

Request-specific information passed between the client application and Amazon Lex V2.

The namespace `x-amz-lex:` is reserved for special attributes. Don't create any request attributes for prefix `x-amz-lex::`.

Type: String to string map

Key Length Constraints: Minimum length of 1.

Required: No

responseContentType

The message that Amazon Lex V2 returns in the response can be either text or speech based on the `responseContentType` value.

- If the value is `text/plain; charset=utf-8`, Amazon Lex V2 returns text in the response.
- If the value begins with `audio/`, Amazon Lex V2 returns speech in the response. Amazon Lex V2 uses Amazon Polly to generate the speech using the configuration that you specified in the `requestContentType` parameter. For example, if you specify `audio/mpeg` as the value, Amazon Lex V2 returns speech in the MPEG format.
- If the value is `audio pcm`, the speech returned is `audio pcm` in 16-bit, little-endian format.
- The following are the accepted values:

- audio/mpeg
- audio/ogg
- audio/pcm
- audio/* (defaults to mpeg)
- text/plain; charset=utf-8

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

sessionState

The state of the user's session with Amazon Lex V2.

Type: [SessionState \(p. 556\)](#) object

Required: No

welcomeMessages

A list of messages to send to the user.

Type: Array of [Message \(p. 551\)](#) objects

Array Members: Maximum number of 10 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DialogAction

Service: Amazon Lex Runtime V2

The next action that Amazon Lex V2 should take.

Contents

slotToElicit

The name of the slot that should be elicited from the user.

Type: String

Length Constraints: Minimum length of 1.

Required: No

type

The next action that the bot should take in its interaction with the user. The possible values are:

- `Close` - Indicates that there will not be a response from the user. For example, the statement "Your order has been placed" does not require a response.
- `ConfirmIntent` - The next action is asking the user if the intent is complete and ready to be fulfilled. This is a yes/no question such as "Place the order?"
- `Delegate` - The next action is determined by Amazon Lex V2.
- `ElicitSlot` - The next action is to elicit a slot value from the user.

Type: String

Valid Values: `Close` | `ConfirmIntent` | `Delegate` | `ElicitIntent` | `ElicitSlot`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DisconnectionEvent

Service: Amazon Lex Runtime V2

A notification from the client that it is disconnecting from Amazon Lex V2. Sending a `DisconnectionEvent` event is optional, but can help identify a conversation in logs.

Contents

clientTimestampMillis

A timestamp set by the client of the date and time that the event was sent to Amazon Lex V2.

Type: Long

Required: No

eventId

A unique identifier that your application assigns to the event. You can use this to identify events in logs.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DTMFInputEvent

Service: Amazon Lex Runtime V2

A DTMF character sent from the client application. DTMF characters are typically sent from a phone keypad to represent numbers. For example, you can have Amazon Lex V2 process a credit card number input from a phone.

Contents

clientTimestampMillis

A timestamp set by the client of the date and time that the event was sent to Amazon Lex V2.

Type: Long

Required: No

eventId

A unique identifier that your application assigns to the event. You can use this to identify events in logs.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

inputCharacter

The DTMF character that the user pressed. The allowed characters are A - D, 0 - 9, # and *.

Type: String

Length Constraints: Fixed length of 1.

Pattern: ^[A-D0-9#*]{1}\$

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

HeartbeatEvent

Service: Amazon Lex Runtime V2

Event that Amazon Lex V2 sends to indicate that the stream is still open between the client application and Amazon Lex V2

Contents

eventId

A unique identifier of the event sent by Amazon Lex V2. The identifier is in the form `RESPONSE-N`, where N is a number starting with one and incremented for each event sent by Amazon Lex V2 in the current session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: `[0-9a-zA-Z._:-]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ImageResponseCard

Service: Amazon Lex Runtime V2

A card that is shown to the user by a messaging platform. You define the contents of the card, the card is displayed by the platform.

When you use a response card, the response from the user is constrained to the text associated with a button on the card.

Contents

buttons

A list of buttons that should be displayed on the response card. The arrangement of the buttons is determined by the platform that displays the button.

Type: Array of [Button \(p. 537\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 5 items.

Required: No

imageUrl

The URL of an image to display on the response card. The image URL must be publicly available so that the platform displaying the response card has access to the image.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 250.

Required: No

subtitle

The subtitle to display on the response card. The format of the subtitle is determined by the platform displaying the response card.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 250.

Required: No

title

The title to display on the response card. The format of the title is determined by the platform displaying the response card.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 250.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Intent

Service: Amazon Lex Runtime V2

The current intent that Amazon Lex V2 is attempting to fulfill.

Contents

confirmationState

Contains information about whether fulfillment of the intent has been confirmed.

Type: String

Valid Values: Confirmed | Denied | None

Required: No

name

The name of the intent.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

slots

A map of all of the slots for the intent. The name of the slot maps to the value of the slot. If a slot has not been filled, the value is null.

Type: String to [Slot \(p. 558\)](#) object map

Key Length Constraints: Minimum length of 1.

Required: No

state

Contains fulfillment information for the intent.

Type: String

Valid Values: Failed | Fulfilled | InProgress | ReadyForFulfillment | Waiting

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

IntentResultEvent

Service: Amazon Lex Runtime V2

Contains the current state of the conversation between the client application and Amazon Lex V2.

Contents

eventId

A unique identifier of the event sent by Amazon Lex V2. The identifier is in the form `RESPONSE-N`, where N is a number starting with one and incremented for each event sent by Amazon Lex V2 in the current session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

inputMode

Indicates whether the input to the operation was text or speech.

Type: String

Valid Values: Text | Speech | DTMF

Required: No

interpretations

A list of intents that Amazon Lex V2 determined might satisfy the user's utterance.

Each interpretation includes the intent, a score that indicates how confident Amazon Lex V2 is that the interpretation is the correct one, and an optional sentiment response that indicates the sentiment expressed in the utterance.

Type: Array of [Interpretation \(p. 550\)](#) objects

Array Members: Maximum number of 5 items.

Required: No

requestAttributes

The attributes sent in the request.

Type: String to string map

Key Length Constraints: Minimum length of 1.

Required: No

sessionId

The identifier of the session in use.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

sessionState

The state of the user's session with Amazon Lex V2.

Type: [SessionState \(p. 556\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Interpretation

Service: Amazon Lex Runtime V2

An intent that Amazon Lex V2 determined might satisfy the user's utterance. The intents are ordered by the confidence score.

Contents

intent

A list of intents that might satisfy the user's utterance. The intents are ordered by the confidence score.

Type: [Intent \(p. 547\)](#) object

Required: No

nluConfidence

Determines the threshold where Amazon Lex V2 will insert the AMAZON.FallbackIntent, AMAZON.KendraSearchIntent, or both when returning alternative intents in a response. AMAZON.FallbackIntent and AMAZON.KendraSearchIntent are only inserted if they are configured for the bot.

Type: [ConfidenceScore \(p. 538\)](#) object

Required: No

sentimentResponse

The sentiment expressed in an utterance.

When the bot is configured to send utterances to Amazon Comprehend for sentiment analysis, this field contains the result of the analysis.

Type: [SentimentResponse \(p. 554\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Message

Service: Amazon Lex Runtime V2

Container for text that is returned to the customer..

Contents

content

The text of the message.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

contentType

Indicates the type of response.

Type: String

Valid Values: `CustomPayload` | `ImageResponseCard` | `PlainText` | `SSML`

Required: Yes

imageResponseCard

A card that is shown to the user by a messaging platform. You define the contents of the card, the card is displayed by the platform.

When you use a response card, the response from the user is constrained to the text associated with a button on the card.

Type: [ImageResponseCard \(p. 545\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PlaybackCompletionEvent

Service: Amazon Lex Runtime V2

Event sent from the client application to Amazon Lex V2 to indicate that playback of audio is complete and that Amazon Lex V2 should start processing the user's input.

Contents

clientTimestampMillis

A timestamp set by the client of the date and time that the event was sent to Amazon Lex V2.

Type: Long

Required: No

eventId

A unique identifier that your application assigns to the event. You can use this to identify events in logs.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PlaybackInterruptionEvent

Service: Amazon Lex Runtime V2

Event sent from Amazon Lex V2 to indicate to the client application should stop playback of audio. For example, if the client is playing a prompt that asks for the user's telephone number, the user might start to say the phone number before the prompt is complete. Amazon Lex V2 sends this event to the client application to indicate that the user is responding and that Amazon Lex V2 is processing their input.

Contents

causedByEventId

The identifier of the event that contained the audio, DTMF, or text that caused the interruption.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

eventId

A unique identifier of the event sent by Amazon Lex V2. The identifier is in the form RESPONSE-N, where N is a number starting with one and incremented for each event sent by Amazon Lex V2 in the current session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

eventReason

Indicates the type of user input that Amazon Lex V2 detected.

Type: String

Valid Values: DTMF_START_DETECTED | TEXT_DETECTED | VOICE_START_DETECTED

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SentimentResponse

Service: Amazon Lex Runtime V2

Provides information about the sentiment expressed in a user's response in a conversation. Sentiments are determined using Amazon Comprehend. Sentiments are only returned if they are enabled for the bot.

For more information, see [Determine Sentiment](#) in the *Amazon Comprehend developer guide*.

Contents

sentiment

The overall sentiment expressed in the user's response. This is the sentiment most likely expressed by the user based on the analysis by Amazon Comprehend.

Type: String

Valid Values: MIXED | NEGATIVE | NEUTRAL | POSITIVE

Required: No

sentimentScore

The individual sentiment responses for the utterance.

Type: [SentimentScore \(p. 555\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SentimentScore

Service: Amazon Lex Runtime V2

The individual sentiment responses for the utterance.

Contents

mixed

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the **MIXED** sentiment.

Type: Double

Required: No

negative

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the **NEGATIVE** sentiment.

Type: Double

Required: No

neutral

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the **NEUTRAL** sentiment.

Type: Double

Required: No

positive

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the **POSITIVE** sentiment.

Type: Double

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

SessionState

Service: Amazon Lex Runtime V2

The state of the user's session with Amazon Lex V2.

Contents

activeContexts

One or more contexts that indicate to Amazon Lex V2 the context of a request. When a context is active, Amazon Lex V2 considers intents with the matching context as a trigger as the next intent in a session.

Type: Array of [ActiveContext \(p. 531\)](#) objects

Array Members: Minimum number of 0 items. Maximum number of 20 items.

Required: No

dialogAction

The next step that Amazon Lex V2 should take in the conversation with a user.

Type: [DialogAction \(p. 541\)](#) object

Required: No

intent

The active intent that Amazon Lex V2 is processing.

Type: [Intent \(p. 547\)](#) object

Required: No

originatingRequestId

Type: String

Length Constraints: Minimum length of 1.

Required: No

sessionAttributes

Map of key/value pairs representing session-specific context information. It contains application information passed between Amazon Lex V2 and a client application.

Type: String to string map

Key Length Constraints: Minimum length of 1.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)

- [AWS SDK for Ruby V3](#)

Slot

Service: Amazon Lex Runtime V2

A value that Amazon Lex V2 uses to fulfill an intent.

Contents

shape

When the `shape` value is `List`, it indicates that the `values` field contains a list of slot values. When the value is `Scalar`, it indicates that the `value` field contains a single value.

Type: String

Valid Values: `Scalar` | `List`

Required: No

value

The current value of the slot.

Type: [Value \(p. 567\)](#) object

Required: No

values

A list of one or more values that the user provided for the slot. For example, if a for a slot that elicits pizza toppings, the values might be "pepperoni" and "pineapple."

Type: Array of [Slot \(p. 558\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

StartConversationRequestEventStream

Service: Amazon Lex Runtime V2

Represents a stream of events between your application and Amazon Lex V2.

Contents

AudioInputEvent

Speech audio sent from your client application to Amazon Lex V2. Audio starts accumulating when Amazon Lex V2 identifies a voice and continues until a natural pause in the speech is found before processing.

Type: [AudioInputEvent \(p. 534\)](#) object

Required: No

ConfigurationEvent

Configuration information sent from your client application to Amazon Lex V2

Type: [ConfigurationEvent \(p. 539\)](#) object

Required: No

DisconnectionEvent

Event sent from the client application to indicate to Amazon Lex V2 that the conversation is over.

Type: [DisconnectionEvent \(p. 542\)](#) object

Required: No

DTMFInputEvent

DTMF information sent to Amazon Lex V2 by your application. Amazon Lex V2 accumulates the DTMF information from when the user sends the first character and ends

- when there's a pause longer than the value configured for the end timeout.
- when there's a digit that is the configured end character.
- when Amazon Lex V2 accumulates characters equal to the maximum DTMF character configuration.

Type: [DTMFInputEvent \(p. 543\)](#) object

Required: No

PlaybackCompletionEvent

Event sent from the client application to Amazon Lex V2 to indicate that it has finished playing audio and that Amazon Lex V2 should start listening for user input.

Type: [PlaybackCompletionEvent \(p. 552\)](#) object

Required: No

TextInputEvent

Text sent from your client application to Amazon Lex V2. Each `TextInputEvent` is processed individually.

Type: [TextInputEvent \(p. 564\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

StartConversationResponseEventStream

Service: Amazon Lex Runtime V2

Represents a stream of events between Amazon Lex V2 and your application.

Contents

AccessDeniedException

Exception thrown when the credentials passed with the request are invalid or expired. Also thrown when the credentials in the request do not have permission to access the `StartConversation` operation.

Type: Exception

HTTP Status Code: 403

Required: No

AudioResponseEvent

An event sent from Amazon Lex V2 to your client application containing audio to play to the user.

Type: [AudioResponseEvent \(p. 536\)](#) object

Required: No

BadGatewayException

Type: Exception

HTTP Status Code: 502

Required: No

ConflictException

Exception thrown when two clients are using the same AWS account, Amazon Lex V2 bot, and session ID.

Type: Exception

HTTP Status Code: 409

Required: No

DependencyFailedException

Type: Exception

HTTP Status Code: 424

Required: No

HeartbeatEvent

Event that Amazon Lex V2 sends to indicate that the stream is still open between the client application and Amazon Lex V2

Type: [HeartbeatEvent \(p. 544\)](#) object

Required: No

IntentResultEvent

Event sent from Amazon Lex V2 to the client application containing the current state of the conversation between the user and Amazon Lex V2.

Type: [IntentResultEvent \(p. 548\)](#) object

Required: No

InternalServerException

An error occurred with Amazon Lex V2.

Type: Exception

HTTP Status Code: 500

Required: No

PlaybackInterruptionEvent

Event sent from Amazon Lex V2 to indicate to the client application should stop playback of audio. For example, if the client is playing a prompt that asks for the user's telephone number, the user might start to say the phone number before the prompt is complete. Amazon Lex V2 sends this event to the client application to indicate that the user is responding and that Amazon Lex V2 is processing their input.

Type: [PlaybackInterruptionEvent \(p. 553\)](#) object

Required: No

ResourceNotFoundException

Exception thrown if one of the input parameters points to a resource that does not exist. For example, if the bot ID specified does not exist.

Type: Exception

HTTP Status Code: 404

Required: No

TextResponseEvent

The event sent from Amazon Lex V2 to your application with text to present to the user.

Type: [TextResponseEvent \(p. 565\)](#) object

Required: No

ThrottlingException

Exception thrown when your application exceeds the maximum number of concurrent requests.

Type: Exception

HTTP Status Code: 429

Required: No

TranscriptEvent

Event sent from Amazon Lex V2 to your client application that contains a transcript of voice audio.

Type: [TranscriptEvent \(p. 566\)](#) object

Required: No

ValidationException

Exception thrown when one or more parameters could not be validated. The message contains the name of the field that isn't valid.

Type: Exception

HTTP Status Code: 400

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TextInputEvent

Service: Amazon Lex Runtime V2

The event sent from your client application to Amazon Lex V2 with text input from the user.

Contents

clientTimestampMillis

A timestamp set by the client of the date and time that the event was sent to Amazon Lex V2.

Type: Long

Required: No

eventId

A unique identifier that your application assigns to the event. You can use this to identify events in logs.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

text

The text from the user. Amazon Lex V2 processes this as a complete statement.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TextResponseEvent

Service: Amazon Lex Runtime V2

The event sent from Amazon Lex V2 to your application with text to present to the user.

Contents

eventId

A unique identifier of the event sent by Amazon Lex V2. The identifier is in the form RESPONSE-N, where N is a number starting with one and incremented for each event sent by Amazon Lex V2 in the current session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: [0-9a-zA-Z._:-]+

Required: No

messages

A list of messages to send to the user. Messages are ordered based on the order that you returned the messages from your Lambda function or the order that the messages are defined in the bot.

Type: Array of [Message \(p. 551\)](#) objects

Array Members: Maximum number of 10 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

TranscriptEvent

Service: Amazon Lex Runtime V2

Event sent from Amazon Lex V2 to your client application that contains a transcript of voice audio.

Contents

eventId

A unique identifier of the event sent by Amazon Lex V2. The identifier is in the form `RESPONSE-N`, where N is a number starting with one and incremented for each event sent by Amazon Lex V2 in the current session.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 100.

Pattern: `[0-9a-zA-Z._:-]+`

Required: No

transcript

The transcript of the voice audio from the user.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Value

Service: Amazon Lex Runtime V2

The value of a slot.

Contents

interpretedValue

The value that Amazon Lex V2 determines for the slot. The actual value depends on the setting of the value selection strategy for the bot. You can choose to use the value entered by the user, or you can have Amazon Lex V2 choose the first value in the `resolvedValues` list.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

originalValue

The text of the utterance from the user that was entered for the slot.

Type: String

Length Constraints: Minimum length of 1.

Required: No

resolvedValues

A list of additional values that have been recognized for the slot.

Type: Array of strings

Length Constraints: Minimum length of 1.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationException

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see [Signature Version 4 Signing Process](#) in the *Amazon Web Services General Reference*.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: AWS4-HMAC-SHA256

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key/YYYYMMDD/region/service/aws4_request*.

For more information, see [Task 2: Create a String to Sign for Signature Version 4](#) in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see [Handling Dates in Signature Version 4](#) in the *Amazon Web Services General Reference*.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to [AWS Services That Work with IAM](#) in the *IAM User Guide*.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see [Task 1: Create a Canonical Request For Signature Version 4](#) in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.