aws

Entwicklerhandbuch

Amazon Lex V1

Amazon Lex V1: Entwicklerhandbuch

Table of Contents

Wenn Sie Amazon Lex V2 verwenden, lesen Sie stattdessen das Amazon Lex V2-Handbuch.

Wenn Sie Amazon Lex V1 verwenden, empfehlen wir, Ihre Bots auf Amazon Lex V2 zu aktualisieren. Wir fügen V1 keine neuen Funktionen mehr hinzu und empfehlen dringend, V2 für alle neuen Bots zu verwenden.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Was ist Amazon Lex?

Amazon Lex ist ein AWS-Service zum Erstellen umgangssprachlicher Schnittstellen für Anwendungen mit Sprache und Text. Mit Amazon Lex ist jetzt dieselbe gesprächige Engine, mit der Amazon Alexa funktioniert, für alle Entwickler verfügbar. Sie ermöglicht die Entwicklung von anspruchsvollen Chatbots mit natürlicher Sprache in neuen und bestehenden Anwendungen. Amazon Lex bietet die umfassende Funktionalität und Flexibilität von natürlichem Sprachverständnis (Natural Language Understanding, NLU) und automatischer Spracherkennung (Automatic Speech Recognition, ASR) fesselnde Benutzererfahrungen mit lebensechten Gesprächsinteraktionen und neuen Produktkategorien.

Amazon Lex ermöglicht es für alle Developer, gesprächige Chatbots schnell zu entwickeln. Mit Amazon Lex ist keine Deep-Learning-Kompetenz erforderlich. Um einen Bot zu erstellen, geben Sie einfach den grundlegenden Gesprächsfluss in der Amazon Lex -Konsole ein. Amazon Lex verwaltet die Dialoge und stimmt dynamisch die Antworten im Gespräch ab. Mithilfe der Konsole können Sie den Text- oder Stimme-Chatbot erstellen, testen und veröffentlichen. Sie können dann die umgangssprachlichen Schnittstellen zu Bots auf mobilen Geräten, Webanwendungen und Chat-Plattformen (z. B. Facebook Messenger) hinzufügen.

Amazon Lex bietet vorgefertigte Integration mitAWS Lambda, und Sie können einfach Integration mit vielen anderen Diensten auf der AWS-Plattform, einschließlich Amazon Cognito,AWS Mobile Hub, Amazon CloudWatch und Amazon DynamoDB. Durch die Integration mit Lambda erhalten Bots Zugriff auf vordefinierte Serverless Enterprise-Anschlüsse und eine Verknüpfung zu Daten in SaaS-Anwendungen wie Salesforce, HubSpot oder Marketo.

Die Nutzung von Amazon Lex bietet unter anderem folgende Vorteile:

- Einfachheit: Amazon Lex führt Sie durch die Bedienung der Konsole, sodass Sie Ihren eigenen Chatbot in wenigen Minuten erstellen können. Sie geben nur einige Beispiel-Phrasen an, und Amazon Lex erstellt ein Modell mit komplett natürlicher Sprache, über das der Bot in Wechselwirkung stehen kann. Er kann Stimme und Text benutzen, um Fragen zu stellen, Antworten bekommen und fortgeschrittene Aufgaben durchzuführen.

- Demokratisierte Deep Learning-Technologien: Amazon Lex bietet ASR- und NLU-Technologien, um ein System zum Verstehen gesprochener Sprache (Speech Language Understanding, SLU)

zu entwickeln. Durch SLU erhält Amazon Lex natürliche Sprache und Texteingabe, kennt die Absicht hinter der Eingabe und erfüllt die Benutzer-Absicht durch Aufrufen der entsprechenden Geschäftsfunktion.

Spracherkennung und Verstehen von natürlichen Sprachen sind einige von den anspruchsvollsten Probleme zum Lösen in Informatik. Sie erfordern komplexe tiefe Lern-Algorithmen für Training mit großen Datenmengen und Infrastruktur. Amazon Lex stellt tiefe Lern-Technologien, die die gleiche Technologie wie Alexa nutzen, allen Developern zur Verfügung. Amazon Lex Chatbots konvertieren eingehende Sprache in Text und verstehen die Absicht der Benutzer, eine intelligente Antwort zu generieren. So können Sie sich ganz auf die Erstellung der Bots mit differenzierten Mehrwert für Ihre Kunden konzentrieren. Sie definieren ganz neue Produktkategorien, die durch gesprochene Schnittstellen möglich sind.

- Nahtlose Bereitstellung und Skalierung: Mit Amazon Lex können Sie Ihre Chatbots direkt über die Amazon Lex -Konsole erstellen, testen und bereitstellen. Mit Amazon Lex können Sie einfach Ihre Stimme- oder Text-Chatbots für Mobilgeräte, Web-Anwendungen und Chat-Services (z. B. Facebook Messenger) veröffentlichen. Amazon Lex skaliert automatisch, sodass Sie sich keine Gedanken über die Bereitstellung von Hardware und Verwaltung von Infrastruktur zur Unterstützung Ihrer Bot machen.

- Integrierte Integration mit der AWS-Plattform: Amazon Lex verfügt über systemeigene Interoperabilität mit anderen AWS-Services wie Amazon Cognito,AWS Lambda, Amazon CloudWatch undAWS Mobile Hubaus. Sie profitieren von den Stärken der AWS-Plattform in Bezug auf Sicherheit, Überwachung, Benutzerauthentifizierung, Geschäftslogik, Speicher und Entwicklung von mobilen Anwendungen.

- Kostengünstig- Bei Amazon Lex fallen keine Vorabkosten oder Mindestgebühren an. Es werden Ihnen nur die Text- oder Sprachanforderungen berechnet, die getätigt wurden. Aufgrund der nutzungsabhängigen Preisgestaltung und der geringen Kosten pro Anforderung bietet der Service eine kostengünstige Möglichkeit zur Erstellung umgangssprachlicher Schnittstellen. Mit der kostenlose Stufe von Amazon Lex können Sie Amazon Lex einfach ohne Startinvestition testen.

Verwenden Sie zum ersten Mal Amazon Lex?

Wenn Sie Amazon Lex zum ersten Mal verwenden, empfehlen wir, sich die folgenden Abschnitte nacheinander durchzulesen:

1. Erste Schritte mit Amazon Lex: In diesem Abschnitt erstellen Sie Ihr Konto und testen Amazon Lex.
2. API-Referenz : In diesem Abschnitt finden Sie zusätzliche Beispiele, um Amazon Lex zu erkunden.

Amazon Lex — Funktionsweise

Mit Amazon Lex können Sie Anwendungen mithilfe einer Sprach- oder Textschnittstelle erstellen, die auf derselben Technologie basiert, die auch Amazon Alexa unterstützt. Im Folgenden sind die typischen Schritte aufgeführt, die Sie bei der Arbeit mit Amazon Lex ausführen:

1. Erstellen Sie einen Bot und konfigurieren Sie diesen mit einer oder mehreren Absichten, die Sie unterstützen möchten. Konfigurieren Sie den Bot so, dass er das Ziel (die Absicht) des Benutzers versteht, mit dem Benutzer kommuniziert, um Informationen zu erhalten, und die Absicht des Benutzers erfüllt.
2. Testen Sie den Bot. Sie können den von der Amazon Lex Lex-Konsole bereitgestellten Testfensterclient verwenden.
3. Veröffentlichen Sie eine Version und erstellen Sie einen Alias.
4. Stellen Sie den Bot bereit. Sie können den Bot auf Plattformen wie mobilen Anwendungen oder Messaging-Plattformen wie Facebook Messenger bereitstellen.

Bevor Sie beginnen, sollten Sie Sie Sie Sie Sie sich mit folgenden Kernkonzepten und der Terminologie von Amazon Lex vertraut machen:

- Bot — Ein Bot führt automatisierte Aufgaben wie das Bestellen einer Pizza, das Buchen eines Hotels, das Bestellen von Blumen usw. aus. Ein Amazon Lex Lex-Bot basiert auf den Funktionen Automatische Spracherkennung (ASR) und Natural Language Understanding (NLU). Jeder Bot muss einen eindeutigen Namen in Ihrem Konto haben.

  Amazon Lex Lex-Bots können Benutzereingaben, die mit Text oder Sprache bereitgestellt werden, verstehen und sich in natürlicher Sprache unterhalten. Sie können Lambda-Funktionen erstellen und sie als Code-Hooks zu Ihrer Intent-Konfiguration hinzufügen, um Aufgaben zur Validierung und Erfüllung von Benutzerdaten auszuführen.

- Absicht — Eine Absicht ist eine Aktion, die der Benutzer ausführen möchte. Sie erstellen einen Bot, um eine oder mehrere Absichten zu unterstützen. Beispiel: Sie möchten einen Bot erstellen, der Pizza und Getränke bestellt. Für jede Absicht geben Sie die folgenden erforderlichen Informationen ein:

- Name der Absicht — Ein beschreibender Name für die Absicht. Zum Beispiel **`OrderPizza`**. Die Namen der Absicht müssen Sie innerhalb Ihres Kontos eindeutig sein.
- Beispielhafte Äußerungen — Wie ein Benutzer die Absicht vermitteln könnte. Ein Benutzer könnte zum Beispiel sagen: „Kann ich bitte eine Pizza bestellen?“ oder „Ich möchte eine Pizza bestellen.“
- So erfüllen Sie die Absicht — Wie Sie die Absicht erfüllen möchten, nachdem der Benutzer die erforderlichen Informationen bereitgestellt hat (z. B. eine Bestellung bei einer lokalen Pizzeria aufgeben). Wir empfehlen, dass Sie eine Lambda-Funktion erstellen, um die Absicht zu erfüllen.

  Sie können die Absicht optional so konfigurieren, dass Amazon Lex die Informationen einfach an die Kundenanwendung zurücksendet, um den erforderlichen Versand durchzuführen.

Neben benutzerdefinierten Absichten wie der Bestellung einer Pizza bietet Amazon Lex auch integrierte Absichten, mit denen Sie Ihren Bot schnell einrichten können. Weitere Informationen finden Sie unter Integrierte Absichten und Slot-Typen.

- Slot — Eine Absicht kann null oder mehr Slots oder Parameter erfordern. Sie fügen Slots als Teil der Konfiguration einer Absicht hinzu. Zur Laufzeit fordert Amazon Lex den Benutzer zur Eingabe bestimmter Slot-Werte auf. Der Benutzer muss Werte für alle erforderlichen Steckplätze angeben, bevor Amazon Lex die Absicht erfüllen kann.

  So erfordert die Absicht `OrderPizza` Slots wie Größe der Pizza, Art der Kruste und Anzahl der Pizzen. Bei der Konfiguration der Absicht fügen Sie diese Slots hinzu. Für jeden Slot geben Sie den Slot-Typ und eine Aufforderung an, die Amazon Lex an den Client senden soll, um Daten vom Benutzer abzurufen. Ein Benutzer kann mit einem Slot-Wert antworten, der zusätzliche Wörter enthält, z. B. „große Pizza bitte“ oder „Bleiben wir bei kleinen“. Amazon Lex kann den beabsichtigten Slot-Wert immer noch verstehen.

- Slot-Typ — Jeder Slot hat einen Typ. Sie können benutzerdefinierte Slot-Typen erstellen oder integrierte Slot-Typen verwenden. Jeder Slot-Typ muss einen eindeutigen Namen in Ihrem Konto haben. Beispiel: Sie möchten die folgenden Slot-Typen für die Absicht `OrderPizza` erstellen und verwenden:

  - Größe - Mit Aufzählungsswerten `Small`, `Medium`, und `Large`.
  - Kruste - Mit Aufzählungswerten `Thick` und `Thin`.

  Amazon Lex bietet auch integrierte Slot-Typen. `AMAZON.NUMBER` ist beispielsweise ein integrierter Slot-Typ, den Sie für die Anzahl bestellter Pizzas verwenden können. Weitere Informationen finden Sie unter Integrierte Absichten und Slot-Typen.

Eine Liste der Regionen, in denen Amazon Lex verfügbar ist, finden Sie unter Regionen und Endpunkte von AWS in der allgemeinen Referenz zu Amazon Web Services.

In den folgenden Themen finden Sie zusätzliche Informationen. Wir empfehlen, dass Sie diese nacheinander lesen und dann mit den Erste Schritte mit Amazon Lex-Übungen fortfahren.

Themen

- Amazon Lex Lex-Ressourcen kennzeichnen

In Amazon Lex unterstützte Sprachen

Amazon Lex V1 unterstützt eine Vielzahl von Sprachen und Gebietsschemas. Die unterstützten Sprachen und die Funktionen, die sie unterstützen, sind in den folgenden Tabellen aufgeführt.

Amazon Lex V2 unterstützt weitere Sprachen, siehe In Amazon Lex V2 unterstützte Sprachen

Unterstützte Sprachen und Gebietsschemas

Amazon Lex V1 unterstützt die folgenden Sprachen und Gebietsschemas.

Code	Sprache und Gebietsschema
de-DE	Deutsch (Deutsch)
en-AU	Englisch (Australien)
en-GB	Englisch (UK)
en-IN	Englisch (Indien)
en-US	Englisch (USA)
es-419	Spanisch (Lateinamerika)
es-ES	Spanisch (Spanien)
es-US	Spanisch (USA)
fr-CA	Französisch (Kanada)
fr-FR	Französisch (Frankreich)
it-IT	Italienisch (Italien)
ja-JP	Japanisch (Japan)
ko-KR	Koreanisch (Korea)

Sprachen und Gebietsschemas, die von Amazon Lex Lex-Funktionen unterstützt werden

Alle Amazon Lex Lex-Funktionen werden in allen Sprachen und Gebietsschemas unterstützt, sofern sie nicht in dieser Tabelle aufgeführt sind.

Funktionsmerkmal	Unterstützte Sprachen und Gebietsschemas
Absichtskontext festlegen	Englisch (amerikanisch) (en-US)

Programmiermodell

Ein Bot ist der primäre Ressourcentyp in Amazon Lex. Die anderen Ressourcentypen in Amazon Lex sind Absicht, Slot-Typ, Alias und Bot-Kanalzuordnung.

Sie erstellen einen Bot mithilfe der Amazon Lex Lex-Konsole oder der Modelbuilding-API. Die Konsole besitzt eine grafische Benutzeroberfläche, die Sie zum Erstellen eines betriebsfähigen Bots für Ihre Anwendung verwenden können. Wenn Sie dies bevorzugen, können Sie die Modellerstellung-API über die AWS CLI oder ein benutzerdefiniertes Programm verwenden, um einen Bot zu erstellen.

Nach dem Erstellen eines Bots stellen Sie ihn auf einer der unterstützten Plattformen bereit oder integrieren ihn in die eigene Anwendung. Wenn ein Benutzer mit dem Bot interagiert, sendet die Client-Anwendung mithilfe der Amazon Lex-Laufzeit-API Anfragen an den Bot. Wenn ein Benutzer beispielsweise sagt „Ich möchte Pizza bestellen“, sendet Ihr Kunde diese Eingabe mithilfe einer der Runtime-API-Operationen an Amazon Lex. Benutzer können Eingaben in Form von Sprache oder Text bereitstellen.

Sie können auch Lambda-Funktionen erstellen und sie in einer Absicht verwenden. Verwenden Sie diese Code-Hooks für Lambda-Funktionen, um Laufzeitaktivitäten wie Initialisierung, Validierung von Benutzereingaben und Absichtserfüllung durchzuführen. In den folgenden Abschnitten finden Sie zusätzliche Informationen.

Themen

- Operationen der Modellerstellung-API
- Laufzeit-API-Operationen
- Lambda-Funktionen als Code-Hooks

Operationen der Modellerstellung-API

Verwenden Sie die Operationen der Modellerstellung-API, um Bots, Absichten und Slot-Typen zu erstellen. Sie können die Modellerstellung-API auch zum Verwalten, Aktualisieren und Löschen von Ressourcen für den Bot verwenden. Operationen der Modellerstellung-API:

- PutBot, PutBotAlias, PutIntent und PutSlotType zum Erstellen und Aktualisieren von Bots, Bot-Aliassen, Absichten und Slot-Typen.
- CreateBotVersion, CreateIntentVersion und CreateSlotTypeVersion zum Erstellen und Veröffentlichen von Versionen der Bots, Absichten und Slot-Typen.
- GetBot und GetBots, um einen bestimmten Bot oder eine Liste von Bots abzurufen, den bzw. die Sie erstellt haben.
- GetIntent und GetIntents, um eine bestimmte Absicht oder eine Liste von Absichten abzurufen, die Sie erstellt haben.
- GetSlotType und GetSlotTypes, um einen bestimmten Slot-Typ oder eine Liste von Slot-Typen abzurufen, den bzw. die Sie erstellt haben.
- GetBuiltinIntent,GetBuiltinIntents, undGetBuiltinSlotTypes um eine in Amazon Lex integrierte Absicht zu erhalten, eine Liste der integrierten Amazon Lex Lex-Intents bzw. eine Liste der integrierten Slot-Typen, die Sie in Ihrem Bot verwenden können.
- GetBotChannelAssociation und GetBotChannelAssociations zum Abrufen einer Zuordnung zwischen dem Bot und einer Messaging-Plattform oder einer Liste der Zuordnungen zwischen dem Bot und den Messaging-Plattformen.
- DeleteBot, DeleteBotAlias, DeleteBotChannelAssociation, DeleteIntent und DeleteSlotType, um nicht benötigte Ressourcen aus dem Konto zu entfernen.

Sie können die Modellbau-API verwenden, um benutzerdefinierte Tools zur Verwaltung Ihrer Amazon Lex-Ressourcen zu erstellen. Es gilt beispielsweise ein Limit von jeweils 100 Versionen für Bots, Absichten und Slot-Typen. Sie können die Modellerstellung-API verwenden, um ein Tool zu entwickeln, das automatisch alte Versionen löscht, wenn ein Bot sich diesem Limit nähert.

Um sicherzustellen, dass jeweils nur ein Vorgang eine Ressource aktualisiert, verwendet Amazon Lex Prüfsummen. Wenn Sie einePut API-Operation—PutBotPutBotAliasPutIntent, oderPutSlotType —verwenden, um eine Ressource zu aktualisieren, müssen Sie die aktuelle Prüfsumme der Ressource in der Anfrage übergeben. Wenn zwei Tools gleichzeitig versuchen, eine Ressource zu aktualisieren, übergeben beide dieselbe aktuelle Prüfsumme. Die erste Anfrage, Amazon Lex zu erreichen, entspricht der aktuellen Prüfsumme der Ressource. Bis die

zweite Anforderung ankommt, hat sich die Prüfsumme geändert. Das zweite Tool empfängt eine `PreconditionFailedException`-Ausnahme und die Aktualisierung wird beendet.

Die`Get` Operationen —GetBotGetIntent, undGetSlotType — sind letztendlich konsistent. Wenn Sie eine `Get`-Operation unmittelbar nach dem Erstellen oder Ändern einer Ressource mit einer der `Put`-Operationen verwenden, werden die Änderungen möglicherweise nicht zurückgegeben. Nachdem eine `Get`-Operation die letzte Aktualisierung zurückgegeben hat, wird immer die aktualisierte Ressource zurückgegeben, bis diese erneut geändert wird. Sie können bestimmen, ob eine aktualisierte Ressource zurückgegeben wurde, indem Sie die Prüfsumme heranziehen.

Laufzeit-API-Operationen

Client-Anwendungen verwenden die folgenden Runtime-API-Operationen, um mit Amazon Lex zu kommunizieren:

- PostContent— Nimmt Sprach- oder Texteingaben entgegen und gibt Informationen zur Absicht sowie eine Text- oder Sprachnachricht zurück, die dem Benutzer übermittelt werden sollen. Derzeit unterstützt Amazon Lex die folgenden Audioformate:

  Eingabeaudioformate: LPCM und Opus

  Ausgabeaudioformate: MPEG, OGG und PCM

  Die Operation `PostContent` unterstützt Audioeingaben bei 8 kHz und 16 kHz. Anwendungen, bei denen der Endbenutzer am Telefon mit Amazon Lex spricht, wie z. B. ein automatisiertes Callcenter, können 8-kHz-Audio direkt weitergeben.

- PostText: Nimmt Text als Eingabe entgegen und gibt Absichtsdaten und eine Textmitteilung für den Benutzer zurück.

Ihre Client-Anwendung verwendet die Runtime-API, um einen bestimmten Amazon Lex Lex-Bot aufzurufen, um Äußerungen — Benutzertext oder Spracheingabe — zu verarbeiten. Nehmen wir zum Beispiel an, dass ein Benutzer sagt: "Ich möchte Pizza." Der Client sendet diese Benutzereingabe mithilfe einer der Amazon Lex Lex-Runtime-API-Operationen an einen Bot. Anhand der

Benutzereingabe erkennt Amazon Lex, dass die Benutzeranfrage der im Bot definierten`OrderPizza` Absicht entspricht. Amazon Lex bindet den Benutzer in eine Konversation ein, um die erforderlichen Informationen oder Slot-Daten wie Pizzagröße, Beläge und Anzahl der Pizzen zu sammeln. Nachdem der Benutzer alle erforderlichen Slot-Daten bereitgestellt hat, ruft Amazon Lex entweder den Code-Hook der Lambda-Funktion auf, um die Absicht zu erfüllen, oder gibt die Absichtsdaten an den Client zurück, je nachdem, wie die Absicht konfiguriert ist.

Verwenden Sie die Operation PostContent, wenn der Bot Spracheingaben verwendet. Beispielsweise kann eine automatisierte Callcenter-Anwendung Sprache an einen Amazon Lex Lex-Bot statt an einen Agenten senden, um Kundenanfragen zu beantworten. Sie können das 8-kHz-Audioformat verwenden, um Audio direkt vom Telefon an Amazon Lex zu senden.

Das Testfenster in der Amazon Lex-Konsole verwendet diePostContent API, um Text- und Sprachanfragen an Amazon Lex zu senden. Sie verwenden dieses Testfenster in den Erste Schritte mit Amazon Lex-Übungen.

Lambda-Funktionen als Code-Hooks

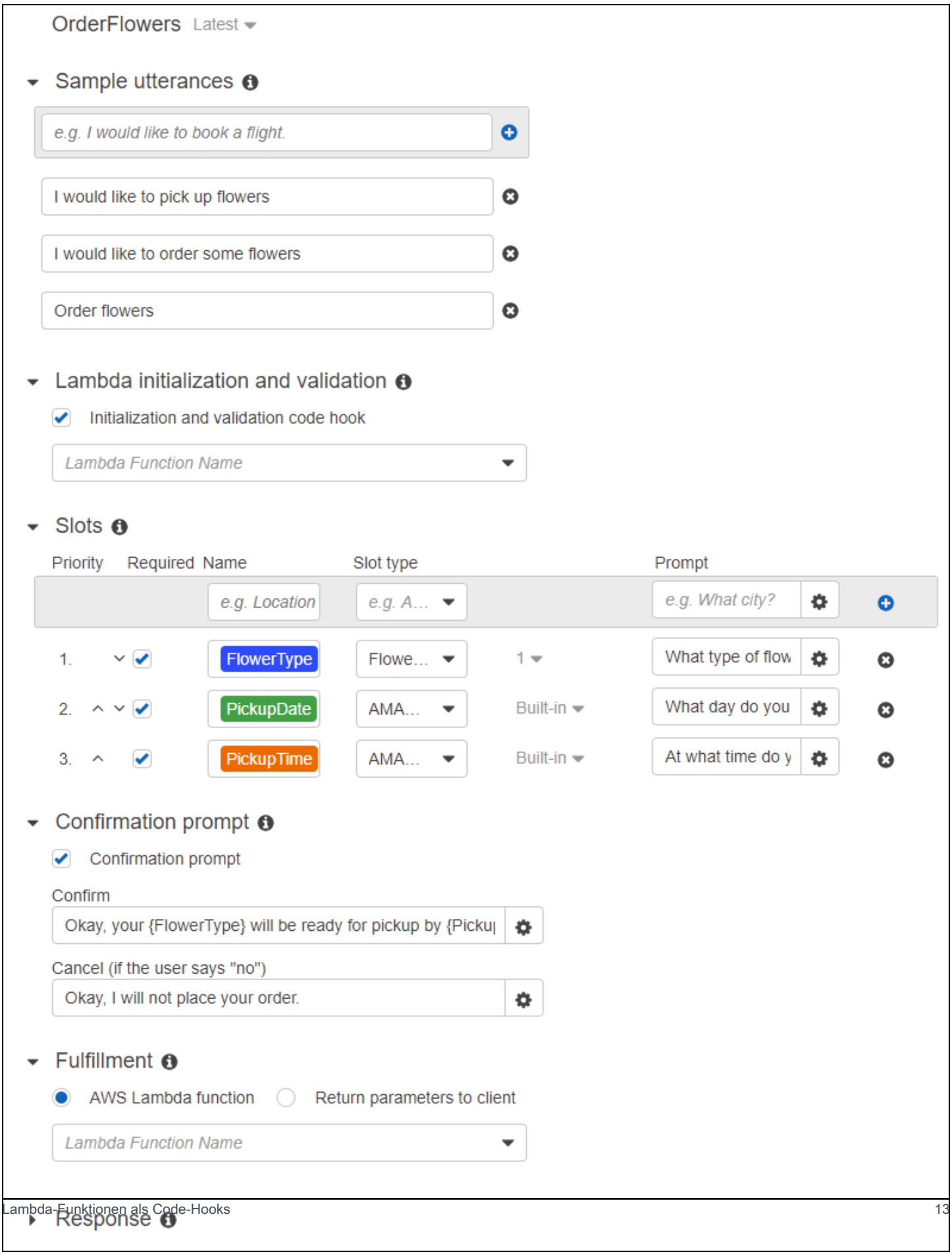
Sie können Ihren Amazon Lex Lex-Bot so konfigurieren, dass er eine Lambda-Funktion als Code-Hook aufruft. Der Code-Haken kann mehrere Aufgaben haben:

- Passt die Benutzerinteraktion an. Wenn Joe beispielsweise nach verfügbaren Pizzabelägen fragt, können Sie Vorkenntnisse über Joes Auswahl nutzen, um eine Teilmenge der Beläge anzuzeigen.
- Überprüft die Benutzereingabe — nehmen wir an, dass Jen nach Feierabend Blumen abholen möchte. Sie können die Zeit überprüfen, die Lena angegeben hat, und eine entsprechende Antwort senden.
- Erfüllt die Absicht des Benutzers — Nachdem Joe alle Informationen für seine Pizzabestellung bereitgestellt hat, kann Amazon Lex eine Lambda-Funktion aufrufen, um die Bestellung bei einer lokalen Pizzeria aufzugeben.

Wenn Sie eine Absicht konfigurieren, geben Sie Lambda-Funktionen als Code-Hooks an den folgenden Stellen an:

- Dialog-Code-Hook für Initialisierung und Validierung — Diese Lambda-Funktion wird bei jeder Benutzereingabe aufgerufen, vorausgesetzt, Amazon Lex hat die Benutzerabsicht verstanden.
- Fulfillment-Code-Hook — Diese Lambda-Funktion wird aufgerufen, nachdem der Benutzer alle Slot-Daten bereitgestellt hat, die zur Erfüllung der Absicht erforderlich sind.

Sie wählen die Absicht und legen die Code-Hooks in der Amazon Lex-Konsole fest, wie im folgenden Screenshot gezeigt:


OrderFlowers Latest
Sample utterances
e.g. I would like to book a flight.
I would like to pick up flowers
I would like to order some flowers
Order flowers
Lambda initialization and validation
Initialization and validation code hook
Lambda Function Name
Slots
Priority Required Name Slot type Prompt
e.g. Location
e.g. A…
e.g. What city?
1. FlowerType Flowe… 1 What type of flow
2. PickupDate AMA… Built-in What day do you
3. PickupTime AMA… Built-in At what time do y
Confirmation prompt
Confirmation prompt
Confirm
Okay, your {FlowerType} will be ready for pickup by {Picku
Cancel (if the user says "no")
Okay, I will not place your order.
Fulfillment
AWS Lambda function
Return parameters to client
Lambda Function Name
Response

Sie können die Code-Haken auch mit den Feldern `dialogCodeHook` und `fulfillmentActivity` in der Operation PutIntent festlegen.

Eine Lambda-Funktion kann Initialisierung, Validierung und Erfüllung durchführen. Die Ereignisdaten, die die Lambda-Funktion empfängt, enthalten ein Feld, das den Aufrufer entweder als Dialog- oder Fulfillment-Code-Hook identifiziert. Sie können diese Informationen verwenden, um den entsprechenden Teil Ihres Codes auszuführen.

Sie können eine Lambda-Funktion verwenden, um einen Bot zu entwickeln, der in komplexen Dialogen navigieren kann. Sie verwenden das`dialogAction` Feld in der Antwort der Lambda-Funktion, um Amazon Lex anzuweisen, bestimmte Aktionen zu ergreifen. Sie können beispielsweise die`ElicitSlot` Dialogaktion verwenden, um Amazon Lex anzuweisen, den Benutzer nach einem Slot-Wert zu fragen, der nicht erforderlich ist. Wenn Sie eine Klärungsaufforderung definiert haben, können Sie die Dialogaktion `ElicitIntent` verwenden, um eine neue Absicht zu erfragen, wenn der Benutzer mit der vorherigen Absicht fertig ist.

Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen.

Verwalten von Mitteilungen

Themen

- Mitteilungstypen
- Kontexte zum Konfigurieren von Mitteilungen
- Unterstützte Mitteilungsformate
- Mitteilungsgruppen
- Antwort-Karten

Wenn Sie einen Bot erstellen, können Sie erklärende oder Informationsmitteilungen konfigurieren, die dieser an den Client senden soll. Betrachten Sie die folgenden Beispiele:

- Sie könnten den Bot mit der folgenden Eingabeaufforderung zur Klärung des Sachverhalts konfigurieren:

  ```
  I don't understand. What would you like to do?
  ```

  Amazon Lex sendet diese Nachricht an den Kunden, wenn dieser die Absicht des Benutzers nicht versteht.

- Nehmen wir an, dass Sie einen Bot erstellen, um die Absicht `OrderPizza` zu unterstützen. Für die Bestellung einer Pizza möchten Sie, dass Benutzer weitere Informationen eingeben, zum Beispiel zur Größe der Pizza, zu Belägen und zum Krustentyp. Sie könnten die folgenden Eingabeaufforderungen konfigurieren:

  ```
  What size pizza do you want?
  What toppings do you want?
  Do you want thick or thin crust?
  ```

  Nachdem Amazon Lex die Absicht des Benutzers, Pizza zu bestellen, festgestellt hat, sendet es diese Nachrichten an den Kunden, um Informationen vom Benutzer zu erhalten.

Dieser Abschnitt erklärt das Entwerfen von Benutzerinteraktionen in Ihrer Bot-Konfiguration.

Mitteilungstypen

Bei einer Mitteilung kann es sich um eine Eingabeaufforderung oder eine Anweisung handeln.

- Eine Eingabeaufforderung ist in der Regel eine Frage und erwartet eine Antwort des Benutzers.
- Eine Anweisung dient zu Informationszwecken. Eine Antwort des Benutzers wird nicht erwartet.

Eine Mitteilung kann Verweise auf Slot, Sitzungsattribute und Anforderungsattribute enthalten. Zur Laufzeit ersetzt Amazon Lex diese Verweise durch tatsächliche Werte.

Verwenden Sie die folgende Syntax, um auf festgelegte Slot-Werte zu verweisen:

```
{SlotName}
```

Verwenden Sie die folgende Syntax, um auf Sitzungsattribute zu verweisen:

```
[SessionAttributeName]
```

Verwenden Sie die folgende Syntax, um auf Anforderungsattribute zu verweisen:

```
((RequestAttributeName))
```

Mitteilungen können Slot-Werte, Sitzungsattribute und Anforderungsattribute enthalten.

Angenommen, Sie konfigurieren die folgende Nachricht in der OrderPizza Intent Ihres Bots:

```
"Hey [FirstName], your {PizzaTopping} pizza will arrive in [DeliveryTime] minutes."
```

Diese Mitteilung bezieht sich sowohl auf Slot-Werte (`PizzaTopping`) als auch auf Sitzungsattribute (`FirstName` und `DeliveryTime`). Zur Laufzeit ersetzt Amazon Lex diese Platzhalter durch Werte und gibt die folgende Meldung an den Client zurück:

```
"Hey John, your cheese pizza will arrive in 30 minutes."
```

Um eckige ([]) oder geschweifte Klammern ({}) in eine Mitteilung einzufügen, verwenden Sie den umgekehrten Schrägstrich (\). Die folgende Mitteilung enthält beispielsweise geschweifte und eckige Klammern:

```
\{Text\} \[Text\]
```

Der an den Client zurückgegebene Text sieht wie folgt aus:

```
{Text} [Text]
```

Weitere Informationen zu Sitzungsattributen enthalten die Beschreibungen der Laufzeit-API-Operationen PostText und PostContent. Ein Beispiel finden Sie unter Reise buchen.

Lambda-Funktionen können auch Nachrichten generieren und sie an Amazon Lex zurückgeben, um sie an den Benutzer zu senden. Wenn Sie Lambda-Funktionen hinzufügen, wenn Sie Ihre Absicht konfigurieren, können Sie Nachrichten dynamisch erstellen. Indem Sie die Nachrichten während der Konfiguration Ihres Bots bereitstellen, können Sie die Erstellung einer Eingabeaufforderung in Ihrer Lambda-Funktion überflüssig machen.

Kontexte zum Konfigurieren von Mitteilungen

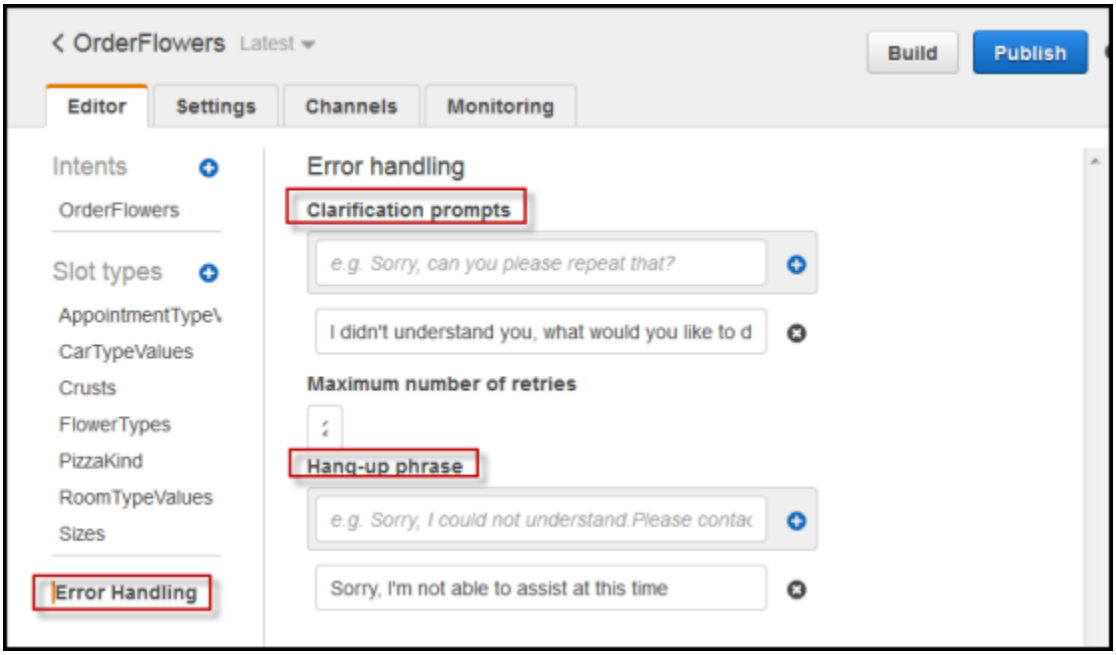
Wenn Sie Ihren Bot erstellen, können Sie Nachrichten in verschiedenen Kontexten erstellen, z. B. Aufforderungen zur Erläuterung im Bot, Aufforderungen zur Eingabe von Slot-Werten und Nachrichten von Intents. Amazon Lex wählt in jedem Kontext eine entsprechende Nachricht aus, um sie an Ihren Benutzer zurückzugeben. Sie können eine Gruppe von Mitteilungen für jeden Kontext

bereitstellen. Wenn Sie dies tun, wählt Amazon Lex nach dem Zufallsprinzip eine Nachricht aus der Gruppe aus. Sie können auch das Format der Mitteilung angeben oder die Mitteilungen zusammen gruppieren. Weitere Informationen finden Sie unter Unterstützte Mitteilungsformate.

Wenn Sie eine Lambda-Funktion mit einer Absicht verknüpft haben, können Sie jede der Nachrichten überschreiben, die Sie bei der Erstellung konfiguriert haben. Eine Lambda-Funktion ist jedoch nicht erforderlich, um eine dieser Nachrichten zu verwenden.

Bot-Mitteilungen

Sie können Ihren Bot mit Erläuterungsaufforderungen und Nachrichten zum Ende der Sitzung konfigurieren. Zur Laufzeit verwendet Amazon Lex die Aufforderung zur Erläuterung, wenn es die Absicht des Benutzers nicht versteht. Sie können konfigurieren, wie oft Amazon Lex eine Klarstellung anfordert, bevor die Nachricht zum Ende der Sitzung gesendet wird. Sie konfigurieren Meldungen auf Bot-Ebene im Abschnitt „Fehlerbehandlung“ der Amazon Lex Lex-Konsole, wie in der folgenden Abbildung dargestellt:



Mit der API konfigurieren Sie Mitteilungen, indem Sie die Felder `clarificationPrompt` und `abortStatement` in der Operation PutBot festlegen.

Wenn Sie eine Lambda-Funktion mit einer Absicht verwenden, gibt die Lambda-Funktion möglicherweise eine Antwort zurück, die Amazon Lex anweist, einen Benutzer nach der Absicht zu fragen. Wenn die Lambda-Funktion keine solche Meldung ausgibt, verwendet Amazon Lex die Aufforderung zur Erläuterung.

Slot-Eingabeaufforderungen

Geben Sie mindestens eine Aufforderungsmitteilung für jeden der erforderlichen Slots in einer Absicht an. Zur Laufzeit verwendet Amazon Lex eine dieser Nachrichten, um den Benutzer aufzufordern, einen Wert für den Slot einzugeben. Beispielsweise für einen `cityName` Slot ist die folgende Aufforderung gültig:

```
Which city would you like to fly to?
```

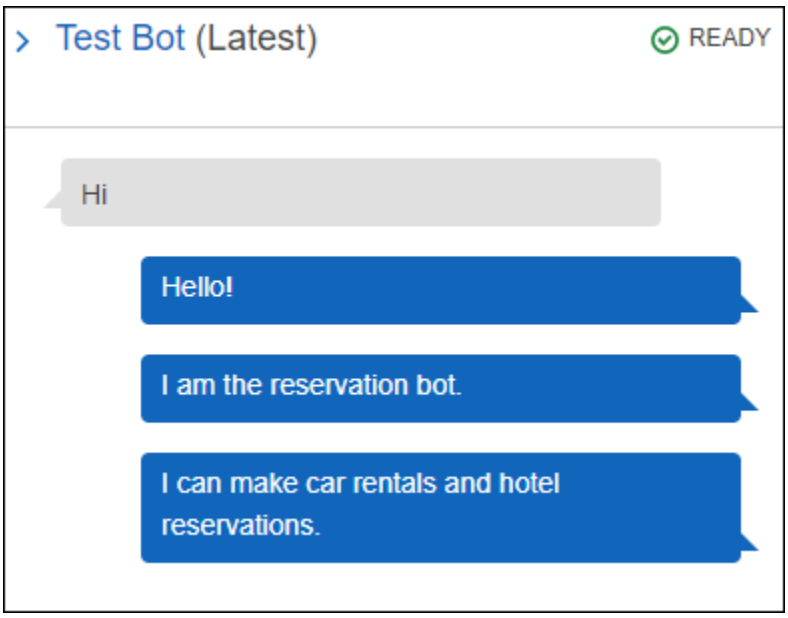
Sie können eine oder mehrere Eingabeaufforderungen für jeden Slot mithilfe der Konsole einrichten. Mit der Operation PutIntent können Sie auch Gruppen von Eingabeaufforderungen erstellen. Weitere Informationen finden Sie unter Mitteilungsgruppen.

Antworten

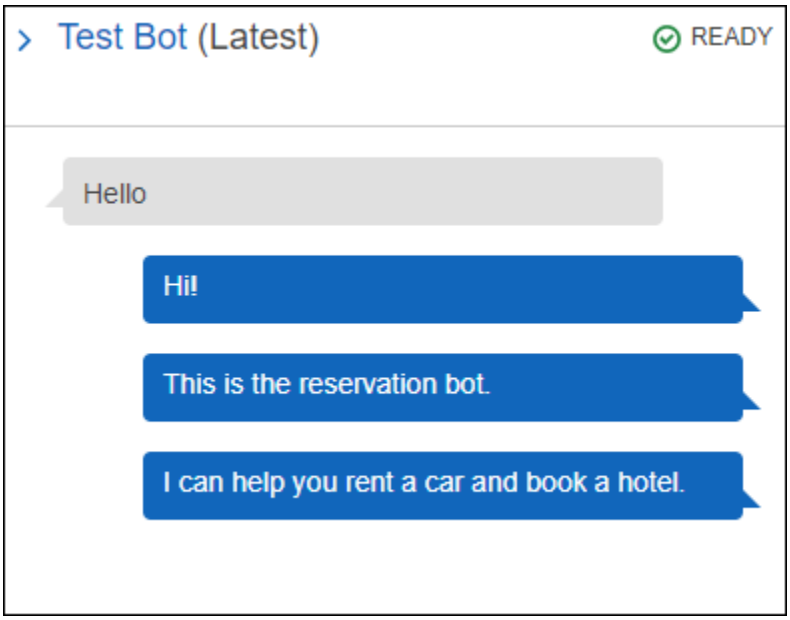
Verwenden Sie den Abschnitt Responses (Antworten) in der Konsole, um dynamische Unterhaltungen für Ihren Bot zu erstellen. Sie können eine oder mehrere Mitteilungsgruppen für eine Antwort erstellen. Zur Laufzeit erstellt Amazon Lex eine Antwort, indem es eine Nachricht aus jeder Nachrichtengruppe auswählt. Weitere Informationen zu Mitteilungsgruppen finden Sie unter Mitteilungsgruppen.

Ihre erste Mitteilungsgruppe könnte beispielsweise unterschiedliche Grußformeln enthalten: „Hallo”, „Guten Tag” und „Grüße”. Die zweite Mitteilungsgruppe könnte verschiedene Einführungen enthalten: „Ich bin der Reservierungs-Bot” und „Dies ist der Reservierungs-Bot.” Eine dritte Mitteilungsgruppe könnte die Fähigkeiten des Bots beschreiben: „Ich helfe Ihnen bei Reservierungen von Mietwagen und Hotelzimmern”, „Sie können Mietwagen reservieren und Hotelbuchungen vornehmen” und „Ich kann Ihnen dabei helfen, ein Auto zu mieten und ein Hotelzimmer zu buchen.”

Lex verwendet eine Mitteilung aus jeder Mitteilungsgruppe, um die Antworten in einem Gespräch dynamisch zu erstellen. Eine Interaktion könnte beispielsweise die folgende sein:



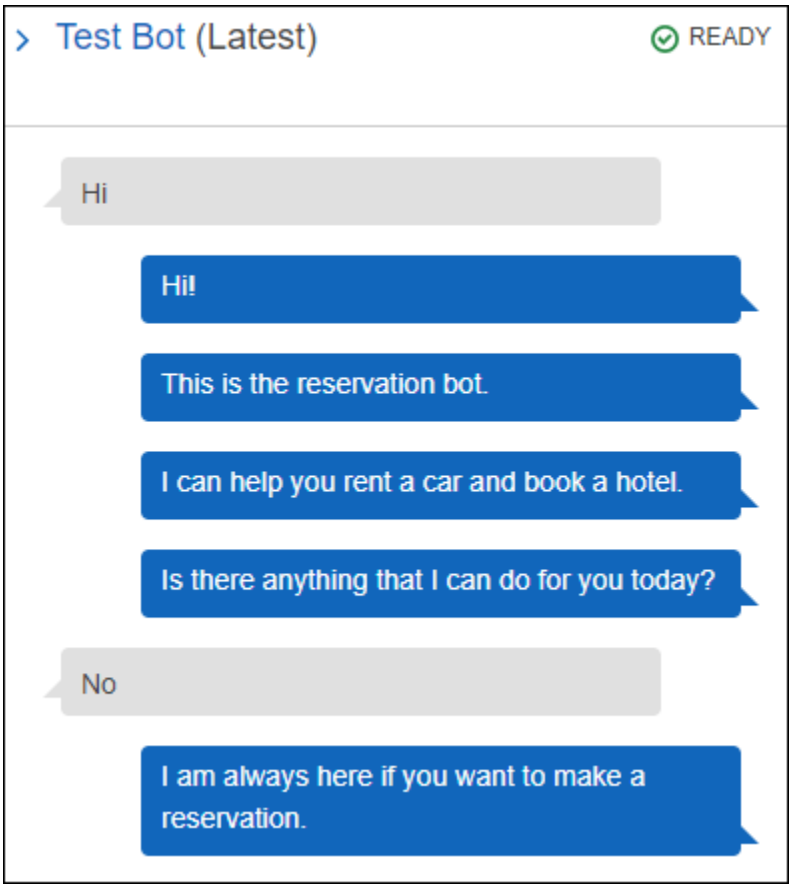
Ein anderes könnte das Folgende sein:



In beiden Fällen kann der Benutzer mit einer neuen Absicht antworten, wie beispielsweise der Absicht `BookCar` oder `BookHotel`.

Sie können den Bot so einrichten, dass er als Antwort eine Folgefrage stellt. Für die vorausgehende Interaktion könnten Sie beispielsweise eine vierte Mitteilungsgruppe mit den folgenden Fragen erstellen: „Kann ich Ihnen helfen, ein Auto oder ein Hotelzimmer zu reservieren?”, „Möchten Sie jetzt eine Reservierung vornehmen?” und „Gibt es etwas, das ich für Sie tun kann?” Für Nachrichten, die

„Nein” als Antwort enthalten, können Sie eine Folge-Eingabeaufforderung erstellen. Folgendes ist ein Beispiel:



Um eine Folge-Eingabeaufforderung zu erstellen, wählen Sie Wait for user reply (Auf Antwort des Benutzers warten). Geben Sie dann die Nachricht oder die Nachrichten ein, die Sie senden möchten, wenn der Benutzer „Nein” sagt. Wenn Sie eine Antwort erstellen, die als Folge-Eingabeaufforderung verwendet werden soll, müssen Sie auch eine geeignete Aussage für den Fall festlegen, dass die Antwort auf die Aussage „Nein” lautet. In der folgenden Abbildung finden Sie ein Beispiel:

Verwenden Sie die Operation `PutIntent`, um mit der API Antworten zu einer Absicht hinzuzufügen. Um eine Antwort festzulegen, richten Sie das Feld `conclusionStatement` in der Anfrage `PutIntent` ein. Um eine Folge-Eingabeaufforderung festzulegen, richten Sie das Feld `followUpPrompt` ein und fügen Sie die Aussage ein, die gesendet werden soll, wenn der Benutzer mit „Nein” antwortet. Sie können nicht das Feld `conclusionStatement` und das Feld `followUpPrompt` gleichzeitig für dieselbe Absicht einrichten.

Unterstützte Mitteilungsformate

Wenn Sie diePostText Operation verwenden oder wenn Sie diePostContent Operation mit demAccept Header auf verwendentext/plain;charset=utf8, unterstützt Amazon Lex Nachrichten in den folgenden Formaten:

- `PlainText`— Die Nachricht enthält einfachen UTF-8-Text.
- `SSML`— Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `CustomPayload`— Die Nachricht enthält ein benutzerdefiniertes Format, das Sie für Ihren Kunden erstellt haben. Sie können die Nutzlast definieren, um den Anforderungen Ihrer Anwendung zu entsprechen.
- `Composite`— Die Nachricht ist eine Sammlung von Nachrichten, eine aus jeder Nachrichtengruppe. Weitere Informationen zu Mitteilungsgruppen finden Sie unter Mitteilungsgruppen.

Standardmäßig gibt Amazon Lex eine der für eine bestimmte Aufforderung definierten Nachrichten zurück. Wenn Sie beispielsweise fünf Nachrichten definieren, um einen Slot-Wert abzurufen, wählt Amazon Lex eine der Nachrichten nach dem Zufallsprinzip aus und gibt sie an den Client zurück.

Wenn Sie möchten, dass Amazon Lex in einer Laufzeitanfrage einen bestimmten Nachrichtentyp an den Client zurückgibt, legen Sie den`x-amzn-lex:accept-content-types` Anforderungsparameter fest. Die Antwort ist auf den angefragten Typ bzw. die angefragten Typen beschränkt. Wenn es mehr als eine Nachricht des angegebenen Typs gibt, gibt Amazon Lex eine nach dem Zufallsprinzip zurück. Weitere Informationen zum Header `x-amz-lex:accept-content-types` finden Sie unter Einrichten des Antworttyps.

Mitteilungsgruppen

Eine Mitteilungsgruppe ist eine Gruppe von geeigneten Antworten auf eine bestimmte Eingabeaufforderung. Verwenden Sie Mitteilungsgruppen, wenn Sie möchten, dass der Bot die Antworten in einem Gespräch dynamisch erstellt. Wenn Amazon Lex eine Antwort an die Client-Anwendung zurückgibt, wählt es nach dem Zufallsprinzip eine Nachricht aus jeder Gruppe aus. Sie können maximal fünf Mitteilungsgruppen für jede Antwort erstellen. Jede Gruppe kann maximal fünf Mitteilungen enthalten. Informationen zum Erstellen von Mitteilungsgruppen in der Konsole finden Sie unter Antworten.

Zum Erstellen einer Mitteilungsgruppe können Sie die Konsole verwenden oder Sie können die Operationen PutBot, PutIntent oder PutSlotType verwenden, um einer Mitteilung eine Gruppennummer zuzuweisen. Wenn Sie keine Nachrichtengruppe oder nur eine Nachrichtengruppe erstellen, sendet Amazon Lex eine einzelne Nachricht in das`Message` Feld. Client-Anwendungen erhalten nur dann mehrere Mitteilungen in einer Antwort, wenn Sie mehr als eine Mitteilungsgruppe in der Konsole erstellt haben oder wenn Sie beim Erstellen oder Aktualisieren einer Absicht mit der Operation PutIntent mehr als eine Mitteilungsgruppe erstellen,

Wenn Amazon Lex eine Nachricht von einer Gruppe sendet, enthält das`Message` Feld der Antwort ein maskiertes JSON-Objekt, das die Nachrichten enthält. Das folgende Beispiel zeigt den Inhalt des Felds `Message`, wenn es mehrere Mitteilungen enthält.

> **Note**
>
> Das Beispiel ist aus Gründen der Lesbarkeit formatiert. Eine Antwort enthält keine Zeilenumbrüche.

```
{\"messages\":[
   {\"type\":\"PlainText\",\"group\":0,\"value\":\"Plain text\"},
   {\"type\":\"SSML\",\"group\":1,\"value\":\"SSML text\"},
   {\"type\":\"CustomPayload\",\"group\":2,\"value\":\"Custom payload\"}
]}
```

Sie können das Format für die Mitteilungen festlegen. Dabei kann es sich um eines der folgenden Formate handeln:

- PlainText— Die Nachricht ist in reinem UTF-8-Text.
- SSML – Die Mitteilung wird in Speech Synthesis Markup Language (SSML) verfasst.
- CustomPayload— Die Nachricht hat ein benutzerdefiniertes Format, das Sie angegeben haben.

Um das Format der Mitteilungen zu kontrollieren, das die Operationen `PostContent` und `PostText` im Feld `Message` zurückgeben, richten Sie das Anfrageattribut `x-amz-lex:accept-content-types` ein. Wenn Sie beispielsweise den Header auf die folgende Option festlegen, erhalten Sie nur Klartext- und SSML-Mitteilungen in der Antwort:

```
x-amz-lex:accept-content-types: PlainText,SSML
```

Wenn Sie ein bestimmtes Mitteilungsformat anfragen und eine Mitteilungsgruppe keine Mitteilung in diesem Format enthält, erhalten Sie die Ausnahme `NoUsableMessageException`. Wenn Sie eine Mitteilungsgruppe verwenden, um Mitteilungen nach Typ zu gruppieren, verwenden Sie nicht den Header `x-amz-lex:accept-content-types`.

Weitere Informationen zum Header `x-amz-lex:accept-content-types` finden Sie unter Einrichten des Antworttyps.

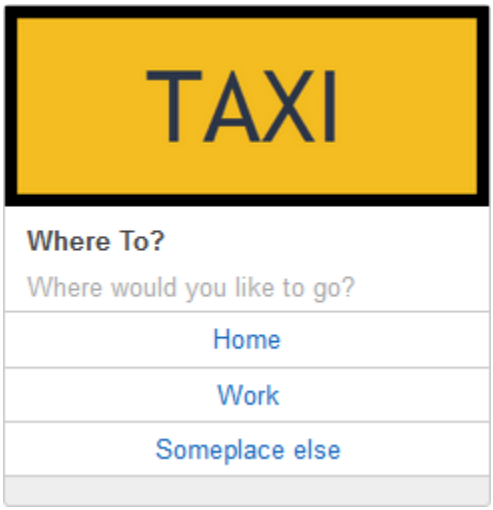
Antwort-Karten

> **Note**
>
> Antwortkarten funktionieren nicht mit dem Amazon Connect Connect-Chat. Ähnliche Funktionen finden Sie jedoch unter Hinzufügen interaktiver Nachrichten zum Chat.

Ein Antwortkarte enthält eine Reihe von geeigneten Reaktionen auf eine Eingabeaufforderung. Verwenden Sie Antwortkarten, um Interaktionen für die Benutzer zu vereinfachen und die

Genauigkeit des Bots zu steigern, indem Rechtschreibfehler in Textinteraktionen reduziert werden. Sie können für jede Aufforderung, die Amazon Lex an Ihre Client-Anwendung sendet, eine Antwortkarte senden. Sie können Antwortkarten mit Facebook Messenger, Slack, Twilio und eigenen Client-Anwendungen verwenden.

In einer Taxianwendung können Sie beispielsweise in der Antwortkarte eine Option für "Home" konfigurieren und als Wert die Anschrift des Benutzers zuweisen. Wenn der Benutzer diese Option auswählt, erhält Amazon Lex die gesamte Adresse als Eingabetext. Sehen Sie sich das folgende Bild an:



Sie können eine Antwortkarte für die folgenden Aufforderungen definieren:

- Abschlussansage
- Bestätigungsaufforderung
- Follow-up-Aufforderung
- Ablehnung
- Slot-Typ-Äußerungen

Sie können für jede Aufforderung nur eine Antwortkarte definieren.

Sie konfigurieren Antwortkarten, wenn Sie eine Absicht erstellen. Sie können mit der Konsole oder der Operation PutIntent zum Zeitpunkt der Erstellung eine statische Antwortkarte definieren. Oder Sie können zur Laufzeit eine dynamische Antwortkarte in einer Lambda-Funktion definieren. Wenn Sie statische und dynamische Antwortkarten definieren, haben die dynamischen Antwortkarten Vorrang.

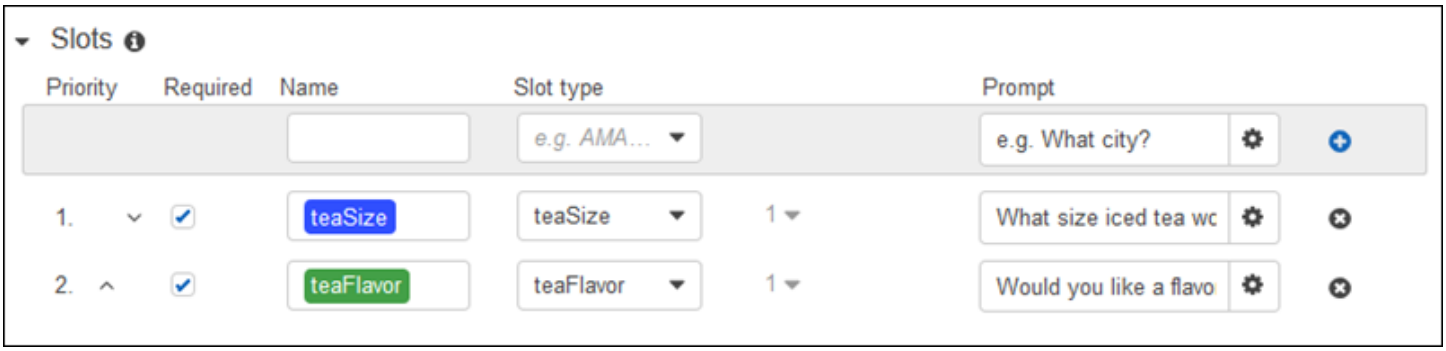
Amazon Lex versendet Antwortkarten in dem Format, das der Kunde versteht. Die Antwortkarten werden für Facebook Messenger, Slack und Twilio konvertiert. Für andere Kunden sendet Amazon

Lex eine JSON-Struktur in derPostText Antwort. Wenn der Kunde beispielsweise Facebook Messenger ist, wandelt Amazon Lex die Antwortkarte in eine generische Vorlage um. Weitere Informationen zu generischen Facebook Messenger-Vorlagen erhalten Sie unter Generic Template auf der Facebook-Website. Ein Beispiel für die JSON-Struktur erhalten Sie unter Dynamisches Generieren von Antwortkarten.

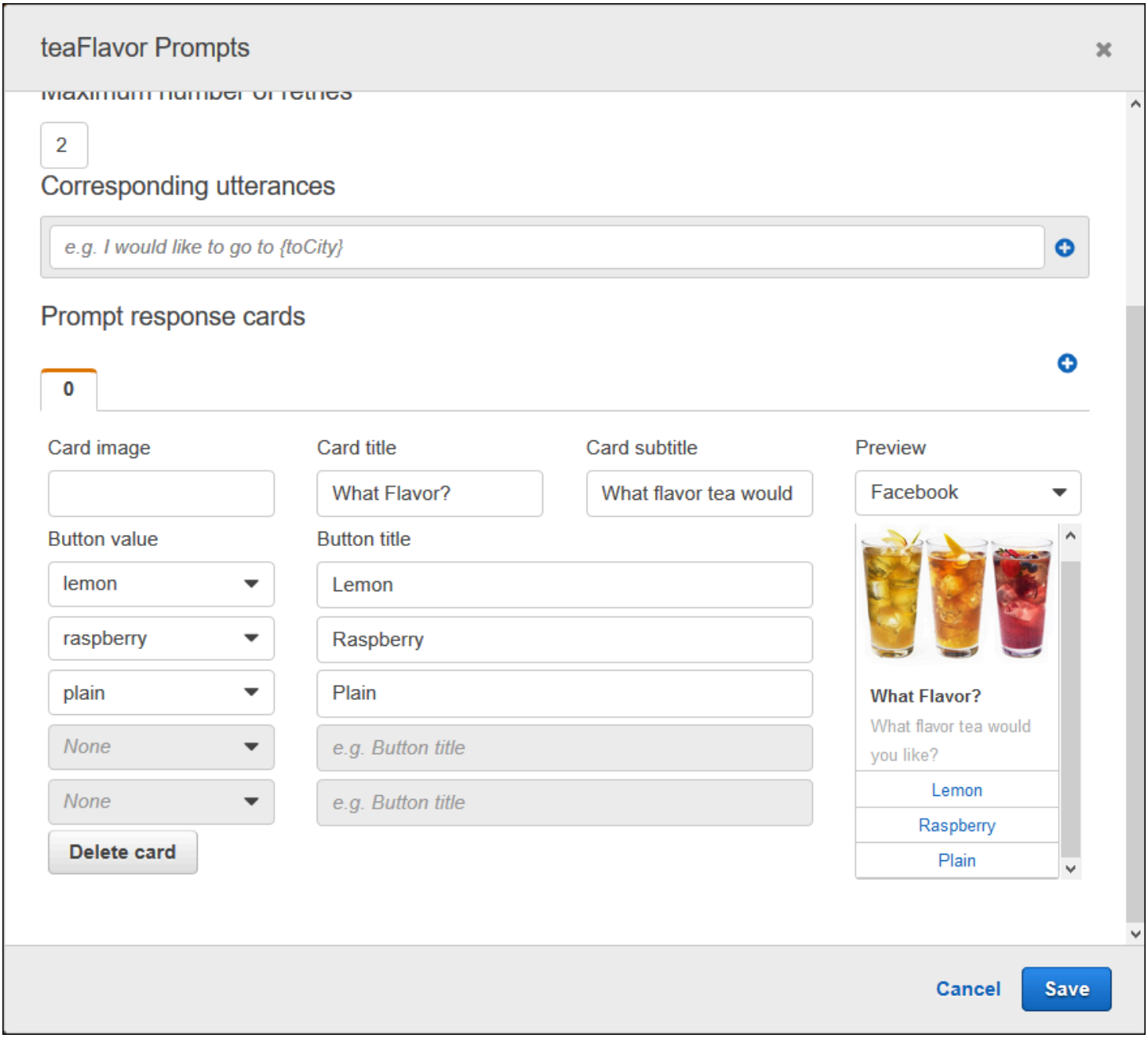
Sie können Antwortkarten nur mit der Operation PostText verwenden. Sie können Antwortkarten nicht mit der Operation PostContent verwenden.

Definieren statischer Antwortkarten

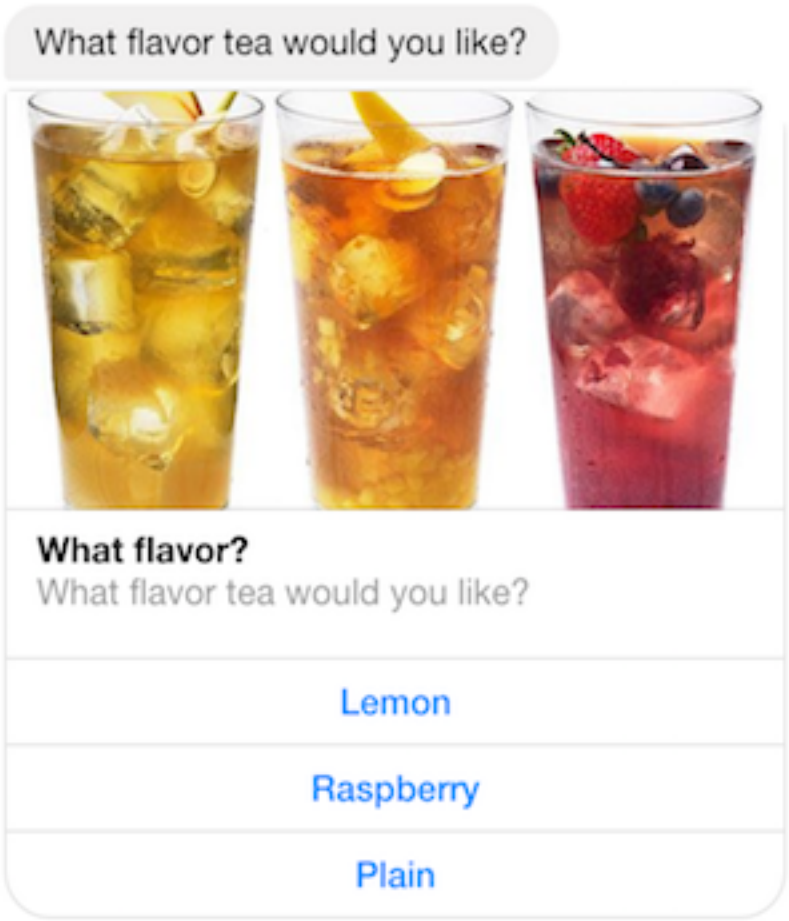
Definieren Sie statische Antwortkarten mit derPutBot Operation oder der Amazon Lex Lex-Konsole, wenn Sie eine Absicht erstellen. Eine statische Antwortkarte wird zusammen mit der Absicht definiert. Verwenden Sie eine statische Antwortkarte, wenn die Antworten feststehen. Nehmen wir an, dass Sie einen Bot mit einer Absicht erstellen, die über einen Slot für das Aroma verfügt. Beim Definieren des Aroma-Slots geben Sie Aufforderungen an, wie im folgenden Konsolen-Screenshot gezeigt:



Bei der Definition von Eingabeaufforderungen können Sie optional eine Antwortkarte zuordnen und Details mit demPutBot Vorgang definieren, oder, wie im folgenden Beispiel gezeigt, in der Amazon Lex Lex-Konsole:



Nehmen wir nun an, dass der Bot in Facebook Messenger integriert wurde. Der Benutzer kann auf die Schaltflächen klicken, um ein Aroma zu wählen, siehe die folgende Abbildung:



Zum Anpassen des Inhalts einer Antwortkarte können Sie auf die Sitzungsattribute verweisen. Zur Laufzeit ersetzt Amazon Lex diese Verweise durch entsprechende Werte aus den Sitzungsattributen. Weitere Informationen finden Sie unter Festlegen von Sitzungsattributen. Ein Beispiel finden Sie unter Eine Antwortkarte verwenden.

Dynamisches Generieren von Antwortkarten

Um Antwortkarten zur Laufzeit dynamisch zu generieren, verwenden Sie die Lambda-Initialisierungs- und Validierungsfunktion für die Absicht. Verwenden Sie eine dynamische Antwortkarte, wenn die Antworten zur Laufzeit in der Lambda-Funktion ermittelt werden. Als Reaktion auf Benutzereingaben generiert die Lambda-Funktion eine Antwortkarte und gibt sie im`dialogAction` Abschnitt der Antwort zurück. Weitere Informationen finden Sie unter Reaktion-Format.

Im Folgenden finden Sie eine Teilantwort einer Lambda-Funktion, die das`responseCard` Element zeigt. Sie generiert eine Benutzererfahrung, die der im vorhergehenden Abschnitt gezeigten ähnelt.

```
responseCard: {
  "version": 1,
  "contentType": "application/vnd.amazonaws.card.generic",
  "genericAttachments": [
    {
      "title": "What Flavor?",
      "subtitle": "What flavor do you want?",
      "imageUrl": "Link to image",
      "attachmentLinkUrl": "Link to attachment",
      "buttons": [
        {
          "text": "Lemon",
          "value": "lemon"
        },
        {
          "text": "Raspberry",
          "value": "raspberry"
        },
        {
          "text": "Plain",
          "value": "plain"
        }
      ]
    }
  ]
}
```

Ein Beispiel finden Sie unter Termin vereinbaren.

Verwaltung von Konversation-Kontext

Der Konversationskontext ist die Information, die ein Benutzer, Ihre Anwendung oder eine Lambda-Funktion einem Amazon Lex Lex-Bot zur Erfüllung einer Absicht zur Verfügung stellt. Der Konversationskontext umfasst Slot-Daten, die der Benutzer bereitstellt, von der Client-Anwendung festgelegte Anforderungsattribute und Sitzungsattribute, die von der Client-Anwendung und den Lambda-Funktionen erstellt werden.

Themen

- Absichtskontext festlegen
- Standard-Slot-Werte verwenden
- Festlegen von Sitzungsattributen

- Festlegen von Anforderungsattributen
- Festlegen der Sitzungs-Zeitüberschreitung
- Freigeben von Informationen zwischen Absichten
- Festlegen von komplexen Attributen

Absichtskontext festlegen

Sie können Amazon Lex kontextabhängige Absichten auslösen lassen. Ein Kontext ist eine Zustandsvariable, die einer Absicht zugeordnet werden kann, wenn Sie einen Bot definieren.

Sie konfigurieren die Kontexte für eine Absicht, wenn Sie die Absicht mithilfe der Konsole oder mithilfe derPutIntent Operation erstellen. Sie können Kontexte nur im englischen (US) (en-US) Gebietsschema verwenden und nur, wenn Sie den`enableModelImprovements` Parameter auf gesetzt haben,`true` als Sie den Bot mit derPutBot Operation erstellt haben.

Es gibt zwei Arten von Beziehungen für Kontexte, Ausgabekontexte und Eingabekontexte. Ein Ausgabekontext wird aktiv, wenn eine zugehörige Absicht erfüllt ist. In der Antwort desPostTextPostContent OR-Vorgangs wird ein Ausgabekontext an Ihre Anwendung zurückgegeben und für die aktuelle Sitzung festgelegt. Nachdem ein Kontext aktiviert wurde, bleibt er für die Anzahl der Runden oder das Zeitlimit aktiv, das bei der Definition des Kontextes konfiguriert wurde.

Ein Eingabekontext spezifiziert Bedingungen, unter denen eine Absicht erkannt werden kann. Eine Absicht kann während einer Konversation nur erkannt werden, wenn alle Eingabekontexte aktiv sind. Eine Absicht ohne Eingabekontexte kann immer anerkannt werden.

Amazon Lex verwaltet automatisch den Lebenszyklus von Kontexten, die durch die Erfüllung von Absichten mit Ausgabekontexten aktiviert werden. Sie können aktive Kontexte auch in einem Aufruf der`PostText` Operation`PostContent` or festlegen.

Sie können außerdem den Kontext einer Konversation festlegen, indem Sie die Lambda-Funktion für die Absicht verwenden. Der Ausgabekontext von Amazon Lex wird an das Eingabeereignis der Lambda-Funktion gesendet. Die Lambda-Funktion kann in ihrer Antwort Kontexte senden. Weitere Informationen finden Sie unter Lambda-Funktions-Eingabe-Ereignis und Antwort-Format.

Angenommen, Sie möchten einen Mietwagen buchen, der so konfiguriert ist, dass er einen Ausgabekontext namens „book_car_filled“ zurückgibt. Wenn die Absicht erfüllt ist, legt Amazon Lex die Ausgabekontextvariable „book_car_filled“ fest. Da es sich bei „book_car_filled“ um einen aktiven

Kontext handelt, wird nun eine Absicht mit dem Kontext „book_car_filled“ als Eingabekontext für die Erkennung berücksichtigt, sofern eine Äußerung des Benutzers als Versuch erkannt wird, diese Absicht hervorzurufen. Sie können dies für Zwecke verwenden, die erst nach der Buchung eines Autos Sinn machen, z. B. um eine Quittung per E-Mail zu versenden oder eine Reservierung zu ändern.

Ausgabekontext

Amazon Lex aktiviert die Ausgabekontexte einer Absicht, wenn die Absicht erfüllt ist. Sie können den Ausgabekontext verwenden, um zu steuern, welche Absichten zur Weiterverfolgung der aktuellen Absicht in Frage kommen.

Jeder Kontext hat eine Liste von Parametern, die in der Sitzung verwaltet werden. Die Parameter sind die Slot-Werte für die erfüllte Absicht. Sie können diese Parameter verwenden, um Slot-Werte für andere Zwecke vorab aufzufüllen. Weitere Informationen finden Sie unter Standard-Slot-Werte verwenden.

Sie konfigurieren den Ausgabekontext, wenn Sie eine Absicht mit der Konsole oder mit derPutIntent Operation erstellen. Sie können eine Absicht mit mehr als einem Ausgabekontext konfigurieren. Wenn die Absicht erfüllt ist, werden alle Ausgabekontexte aktiviert und in derPostTextPostContent OR-Antwort zurückgegeben.

Im Folgenden wird gezeigt, wie Sie einem Intent mithilfe der Konsole einen Ausgabekontext zuweisen.

Wenn Sie einen Ausgabekontext definieren, definieren Sie auch dessen Gültigkeitsdauer, die Dauer oder Anzahl der Runden, in denen der Kontext in den Antworten von Amazon Lex enthalten ist. Eine Runde ist eine Anfrage aus Ihrer Bewerbung an Amazon Lex. Sobald die Anzahl der Runden oder die Zeit abgelaufen ist, ist der Kontext nicht mehr aktiv.

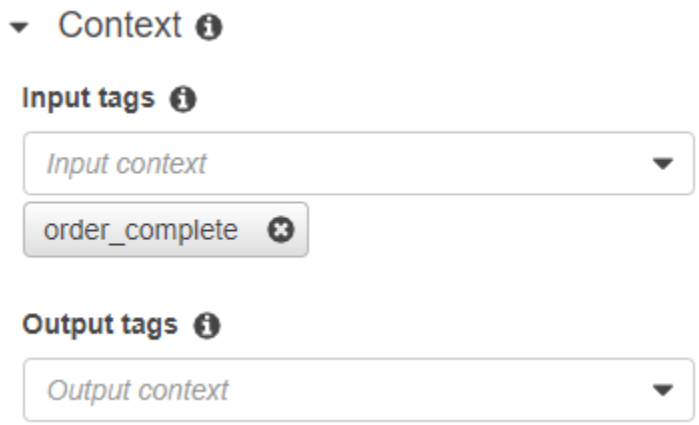
Ihre Anwendung kann den Ausgabekontext nach Bedarf verwenden. Ihre Anwendung kann den Ausgabekontext beispielsweise verwenden, um:

- Ändern Sie das Verhalten der Anwendung je nach Kontext. Beispielsweise könnte ein Reiseantrag eine andere Aktion für den Kontext „book_car_filled“ haben als „rental_hotel_filled“.
- Geben Sie den Ausgabekontext als Eingabekontext für die nächste Äußerung an Amazon Lex zurück. Wenn Amazon Lex die Äußerung als Versuch erkennt, eine Absicht hervorzurufen, verwendet es den Kontext, um die Absichten, die zurückgegeben werden können, auf solche mit dem angegebenen Kontext zu beschränken.

Eingabekontext

Sie legen einen Eingabekontext fest, um die Punkte in der Konversation zu begrenzen, an denen die Absicht erkannt wird. Absichten ohne Eingabekontext können immer erkannt werden.

Sie legen die Eingabekontexte fest, auf die eine Absicht reagiert, indem Sie die Konsole oder den`PutIntent` Vorgang verwenden. Eine Absicht kann mehrere Eingabekontexte haben. Im Folgenden wird gezeigt, wie Sie einer Absicht mithilfe der Konsole einen Eingabekontext zuweisen.



Bei einer Absicht mit mehr als einem Eingabekontext müssen alle Kontexte aktiv sein, um die Absicht auszulösen. Sie können einen Eingabekontext festlegen, wenn Sie diePutSession OperationPostTextPostContent, oder aufrufen.

Sie können die Slots so konfigurieren, dass sie Standardwerte aus dem aktuellen aktiven Kontext übernehmen. Standardwerte werden verwendet, wenn Amazon Lex eine neue Absicht erkennt, aber keinen Slot-Wert erhält. Sie geben den Kontextnamen und den Slot-Namen im Formular an`#context-name.parameter-name`, wenn Sie den Slot definieren. Weitere Informationen finden Sie unter Standard-Slot-Werte verwenden.

Standard-Slot-Werte verwenden

Wenn Sie einen Standardwert verwenden, geben Sie eine Quelle für einen Slot-Wert an, der für neue Zwecke gefüllt werden soll, wenn durch die Benutzereingabe kein Slot bereitgestellt wird. Bei dieser Quelle kann es sich um vorherige Dialog-, Anforderungs- oder Sitzungsattribute oder um einen festen Wert handeln, den Sie bei der Erstellung festgelegt haben.

Sie können Folgendes als Quelle für Ihre Standardwerte verwenden:

- Vorheriger Dialog (Kontexte) — #context -name.parameter-name
- Sitzungsattribute — [Attributname]
- Attribute anfordern — <attribute-name>
- Fester Wert — Jeder Wert, der nicht mit dem vorherigen Wert übereinstimmt

Wenn Sie diePutIntent Operation verwenden, um einem Intent Slots hinzuzufügen, können Sie eine Liste mit Standardwerten hinzufügen. Standardwerte werden in der angegebenen Reihenfolge verwendet. Angenommen, Sie haben eine Absicht mit einem Slot mit der folgenden Definition:

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

Wenn die Absicht erkannt wird, wird der Wert des Slots mit dem Namenreservation-start-date "" auf einen der folgenden Werte gesetzt.

1. Wenn der "book-car-fulfilled" -Kontext aktiv ist, wird der Wert des Parameters „StartDate“ als Standardwert verwendet.

2. Wenn der "book-car-fulfilled" -Kontext nicht aktiv ist oder wenn der Parameter „StartDate“ nicht gesetzt ist, wird der Wert des SitzungsattributsreservationStartDate "" als Standardwert verwendet.
3. Wenn keiner der ersten beiden Standardwerte verwendet wird, hat der Slot keinen Standardwert und Amazon Lex ermittelt wie gewohnt einen Wert.

Wenn ein Standardwert für den Slot verwendet wird, wird der Slot nicht abgerufen, auch wenn er erforderlich ist.

Festlegen von Sitzungsattributen

Sitzungsattribute enthalten anwendungsspezifische Informationen, die während einer Sitzung zwischen einem Bot und einer Client-Anwendung übertragen werden. Amazon Lex übergibt Sitzungsattribute an alle Lambda-Funktionen, die für einen Bot konfiguriert sind. Wenn eine Lambda-Funktion Sitzungsattribute hinzufügt oder aktualisiert, leitet Amazon Lex die neuen Informationen an die Client-Anwendung zurück. Beispiel:

- In Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole) nutzt der Beispiel-Bot das Sitzungsattribut `price` für die Verwaltung des Blumenpreises. Die Lambda-Funktion legt dieses Attribut basierend auf der Art der Blumen fest, die bestellt wurden. Weitere Informationen finden Sie unter Schritt 5 (optional): Prüfen der Details des Informationsflusses (Konsole).
- In Reise buchen verwendet der Beispiel-Bot das Sitzungsattribut `currentReservation` zur Verwaltung einer Kopie der Slot-Typ-Daten während der Konversation, um ein Hotel oder einen Mietwagen zu buchen. Weitere Informationen finden Sie unter Informationsfluss im Detail.

Verwenden Sie Sitzungsattribute in Ihren Lambda-Funktionen, um einen Bot zu initialisieren und um Eingabeaufforderungen und Antwortkarten anzupassen. Beispiel:

- Initialisierung — In einem Bot zur Pizzabestellung übergibt die Client-Anwendung den Standort des Benutzers als Sitzungsattribut beim ersten Aufruf an denPostText VorgangPostContent oder. Zum Beispiel `"Location": "111 Maple Street"`. Die Lambda-Funktion verwendet diese Informationen, um die nächstgelegene Pizzeria für die Bestellung zu finden.
- Personalize Sie Eingabeaufforderungen — Konfigurieren Sie Eingabeaufforderungen und Antwortkarten, um auf Sitzungsattribute zu verweisen. Zum Beispiel: „Hey [FirstName], welche Toppings möchtest du?“ Wenn Sie den Vornamen des Benutzers als Sitzungsattribut (`{"FirstName": "Jo"}`) übergeben, ersetzt Amazon Lex den Platzhalter durch den Namen.

Anschließend sendet die Anwendung eine personalisierte Aufforderung an den Benutzer "Hallo Jo, welchen Belag wünschen Sie?"

Sitzungsattribute werden für die Dauer der Sitzung beibehalten. Amazon Lex speichert sie in einem verschlüsselten Datenspeicher, bis die Sitzung endet. Der Client kann Sitzungsattribute in einer Anforderung erstellen, indem er entweder die Operation PostContent oder die Operation PostText aufruft, wobei das Feld `sessionAttributes` auf einen Wert gesetzt ist. Eine Lambda-Funktion kann ein Sitzungsattribut in einer Antwort erstellen. Nachdem der Client oder eine Lambda-Funktion ein Sitzungsattribut erstellt hat, wird der gespeicherte Attributwert immer dann verwendet, wenn die Client-Anwendung kein`sessionAttribute` Feld in einer Anfrage an Amazon Lex enthält.

Angenommen, Sie haben zwei Sitzungsattribute `{"x": "1", "y": "2"}`. Wenn der Client den`PostText` Vorgang`PostContent` oder aufruft, ohne das`sessionAttributes` Feld anzugeben, ruft Amazon Lex die Lambda-Funktion mit den gespeicherten Sitzungsattributen (`{"x": 1, "y": 2}`) auf. Wenn die Lambda-Funktion keine Sitzungsattribute zurückgibt, gibt Amazon Lex die gespeicherten Sitzungsattribute an die Client-Anwendung zurück.

Wenn entweder die Client-Anwendung oder eine Lambda-Funktion Sitzungsattribute übergibt, aktualisiert Amazon Lex die gespeicherten Sitzungsattribute. Wird ein bestehender Wert wie `{"x": 2}` übergeben, wird der gespeicherte Wert aktualisiert. Wenn Sie einen neuen Satz Sitzungsattribute übergeben, wie z. B. `{"z": 3}`, werden die vorhandenen Werte entfernt, und nur der neue Wert wird beibehalten. Wird eine leere Zuordnung, `{}`, übergeben, werden die gespeicherten Werte gelöscht.

Um Sitzungsattribute an Amazon Lex zu senden, erstellen Sie eine string-to-string Zuordnung der Attribute. Das folgende Beispiel zeigt, wie Sitzungsattribute zugeordnet werden:

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

Bei der `PostText`-Operation fügen Sie die Zuordnung wie folgt mittels des Felds `sessionAttributes` in den Text der Anforderung ein:

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

Bei der `PostContent`-Operation kodieren Sie die Zuordnung mit base64 und senden diese als `x-amz-lex-session-attributes`-Header.

Wenn Sie binäre oder strukturierte Daten in einem Sitzungsattribut übermitteln, müssen Sie die Daten zunächst in eine einfache Zeichenfolge transformieren. Weitere Informationen finden Sie unter Festlegen von komplexen Attributen.

Festlegen von Anforderungsattributen

Anforderungsattribute enthalten anforderungsspezifische Informationen und gelten nur für die aktuelle Anforderung. Eine Client-Anwendung sendet diese Informationen an Amazon Lex. Verwenden Sie Anforderungsattribute zur Weitergabe von Informationen, die nicht während der ganzen Sitzung erhalten bleiben müssen. Sie können eigene Anforderungsattribute erstellen oder vordefinierte verwenden. Zum Senden von Anforderungsattributen nutzen Sie den `x-amz-lex-request-attributes`-Header in einem the section called “PostContent” oder das `requestAttributes`-Feld in einer the section called “PostText”-Anforderung. Da Anforderungsattribute nicht wie Sitzungsattribute anforderungsübergreifend erhalten bleiben, werden sie nicht in `PostContent`- oder `PostText`-Antworten zurückgegeben.

> **Note**
>
> Nutzen Sie Sitzungsattribute, wenn Sie möchten, dass Informationen anforderungsübergreifend erhalten bleiben.

Der Namespace `x-amz-lex:` ist für die vordefinierten Anforderungsattribute reserviert. Erstellen Sie keine Anforderungsattribute mit dem Präfix `x-amz-lex:`.

Festlegen von vordefinierten Anforderungsattributen

Amazon Lex stellt vordefinierte Anforderungsattribute bereit, mit denen Sie die an Ihren Bot gesendeten Informationen verwalten können. Die Attribute bleiben nicht für die gesamte Sitzung erhalten, daher müssen Sie die vordefinierten Attribute in jede Anforderung mit einschließen. Alle vordefinierten Attribute befinden sich im Namespace `x-amz-lex:`.

Zusätzlich zu den folgenden vordefinierten Attributen bietet Amazon Lex vordefinierte Attribute für Messaging-Plattformen. Eine Liste dieser Attribute finden Sie unter Das Installieren eines Amazon Lex Bots auf einer Messaging-Plattform.

Einrichten des Antworttyps

Wenn Sie über zwei Client-Anwendungen mit unterschiedlichen Funktionen verfügen, müssen Sie das Format der Mitteilungen in einer Antwort möglicherweise einschränken. Beispielsweise möchten Sie die an einen Web-Client gesendeten Mitteilungen auf Klartext beschränken, einen mobilen Client jedoch befähigen, sowohl Klartext als auch Speech Synthesis Markup Language (SSML) zu verwenden. Um das Format der von den Operationen PostContent und PostText zurückgegebenen Mitteilungen einzurichten, verwenden Sie das Anforderungsattribut `x-amz-lex:accept-content-types"`.

Sie können das Attribut auf eine beliebige Kombination der folgenden Mitteilungstypen festlegen:

- `PlainText`— Die Nachricht enthält einfachen UTF-8-Text.
- `SSML`— Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `CustomPayload`— Die Nachricht enthält ein benutzerdefiniertes Format, das Sie für Ihren Kunden erstellt haben. Sie können die Nutzlast definieren, um den Anforderungen Ihrer Anwendung zu entsprechen.

Amazon Lex gibt nur Nachrichten mit dem angegebenen Typ im`Message` Feld der Antwort zurück. Sie können mehr als einen Wert festlegen, indem Sie die einzelnen Werte durch Komma trennen. Wenn Sie Mitteilungsgruppen verwenden, muss jede Mitteilungsgruppe mindestens eine Mitteilung des angegebenen Typs enthalten. Andernfalls erhalten Sie den Fehler `NoUsableMessageException`. Weitere Informationen finden Sie unter Mitteilungsgruppen.

> **Note**
>
> Das Anforderungsattribut `x-amz-lex:accept-content-types` hat keine Auswirkungen auf den Inhalt des HTML-Texts. Der Inhalt einer `PostText`-Operation ist immer UTF-8-Klartext. Der Text einer Antwort der `PostContent`-Operation enthält Daten in dem im Header `Accept` der Anforderung festgelegten Format.

Einstellen der bevorzugten Zeitzone

Nutzen Sie das Anforderungsattribut `x-amz-lex:time-zone`, um die Zeitzone festzulegen, die zum Auflösen von Datumsangaben verwendet wird, sodass diese relativ zur Zeitzone des Benutzers ist. Wenn Sie in dem Attribut `x-amz-lex:time-zone` keine Zeitzone angeben, ist der Standardwert von der Region abhängig, die Sie für Ihren Bot verwenden.

Region	Standardzeitzone
USA Ost (Nord-Virginia)	`America/New_York`
USA West (Oregon)	`America/Los_Angeles`
Asien-Pazifik (Singapur)	`Asia/Singapore`
Asien-Pazifik (Sydney)	`Australia/Sydney`
Asien-Pazifik (Tokio)	`Asia/Tokyo`
Europe (Frankfurt)	`Europe/Berlin`
Europa (Irland)	`Europe/Dublin`
Europe (London)	`Europe/London`

Zum Beispiel, wenn der Benutzer als Antwort`tomorrow` auf die Frage „An welchem Tag soll Ihr Paket geliefert werden?“ antwortet. Das tatsächliche Datum, an dem das Paket zugestellt wird, hängt von der Zeitzone des Benutzers ab. Beispiel: Wenn es in New York am 16. September 01:00 Uhr ist, ist in Los Angeles erst der 15. September und die dortige Uhrzeit ist 22:00 Uhr. Wenn Ihr Service in der Region USA Ost (Nord-Virginia) läuft und eine Person in Los Angeles ein Paket bestellt, das „morgen“ in der Standardzeitzone zugestellt werden soll, wird das Paket am 17. und nicht am 16. zugestellt. Wenn Sie jedoch das Anforderungsattribut `x-amz-lex:time-zone` auf `America/Los_Angeles` setzen, wird das Paket am 16. ausgeliefert.

Sie können das Attribut auf einen beliebigen IANA (Internet Assigned Number Authority)-Zeitzonennamen setzen. Eine Liste der Zeitzonennamen finden Sie unter dem Artikel zu der Liste der TZ-Datenbankzeitzonen auf Wikipedia.

Festlegen von benutzerdefinierten Anforderungsattributen

Ein benutzerdefiniertes Anforderungsattribut besteht aus Daten, die Sie bei jeder Anforderung an Ihren Bot senden. Sie senden die Informationen im `amz-lex-request-attributes`-Header einer `PostContent`-Anforderung oder im Feld `requestAttributes` einer `PostText`-Anforderung.

Um Anforderungsattribute an Amazon Lex zu senden, erstellen Sie eine string-to-string Zuordnung der Attribute. Das folgende Beispiel zeigt, wie Anforderungsattribute zugewiesen werden:

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

Bei der `PostText`-Operation fügen Sie die Zuordnung wie folgt mittels des Felds `requestAttributes` in den Text der Anforderung ein:

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

Bei der `PostContent`-Operation kodieren Sie die Zuordnung mit base64 und senden diese als `x-amz-lex-request-attributes`-Header.

Wenn Sie binäre oder strukturierte Daten in einem Anforderungsattribut übermitteln, müssen Sie die Daten zunächst in eine einfache Zeichenfolge transformieren. Weitere Informationen finden Sie unter Festlegen von komplexen Attributen.

Festlegen der Sitzungs-Zeitüberschreitung

Amazon Lex speichert Kontextinformationen — Slot-Daten und Sitzungsattribute —, bis eine Konversationssitzung endet. Um zu steuern, wie lange eine Sitzung für einen Bot dauert, legen Sie einen Zeitüberschreitungswert für die Sitzung fest. Standardmäßig dauert eine Sitzung 5 Minuten, Sie können aber auch eine Dauer zwischen 0 und 1 440 Minuten (24 Stunden) angeben.

Angenommen, Sie erstellen einen `ShoeOrdering`-Bot zur Unterstützung von Absichten wie `OrderShoes` und `GetOrderStatus`. Wenn Amazon Lex feststellt, dass der Benutzer Schuhe bestellen möchte, fragt es nach Slot-Daten. Sie fragt beispielsweise nach Schuhgröße, Farbe, Marke etc. Wenn der Benutzer einige der Slot-Daten angibt, den Schuhkauf aber nicht abschließt, merkt sich Amazon Lex alle Slot-Daten und Sitzungsattribute für die gesamte Sitzung. Wenn der Benutzer vor Ablauf der Sitzung zu dieser zurückkehrt, kann er die fehlenden Slot-Daten bereitstellen und den Kauf abschließen.

In der Amazon Lex Lex-Konsole legen Sie das Sitzungs-Timeout fest, wenn Sie einen Bot erstellen. Mit der AWS-Befehlszeilenschnittstelle (AWS CLI) oder API legen Sie das Timeout fest, wenn Sie einen Bot mit derPutBot Operation erstellen oder aktualisieren, indem Sie dasInSeconds Feld idleSessionTTL setzen.

Freigeben von Informationen zwischen Absichten

Amazon Lex unterstützt den Informationsaustausch zwischen verschiedenen Absichten. Teilen Sie mit Sitzungsattributen Informationen zwischen Absichten.

Angenommen, ein Benutzer des Bots `ShoeOrdering` bestellt Schuhe. Der Bot schaltet sich in die Konversation mit dem Benutzer ein und erfasst Slot-Daten wie Schuhgröße, Farbe und Marke. Wenn der Benutzer eine Bestellung aufgibt, legt die Lambda-Funktion, die die Bestellung erfüllt, das`orderNumber` Sitzungsattribut fest, das die Bestellnummer enthält. Der Benutzer verwendet die Absicht `GetOrderStatus`, um den Status der Bestellung zu erhalten. Der Bot kann den Benutzer nach Slot-Daten fragen, wie beispielsweise Bestellnummer oder -datum. Wenn der Bot die notwendigen Informationen hat, gibt er den Status der Bestellung zurück.

Wenn Sie der Meinung sind, dass Ihre Benutzer möglicherweise Absichten während der Sitzung ändern, können Sie Ihren Bot so konfigurieren, dass der Status der letzten Bestellung wiedergegeben wird. Anstatt den Benutzer erneut nach Bestellinformationen zu fragen, nutzen Sie das Sitzungsattribut `orderNumber`, um Informationen absichtsübergreifend zu teilen und die Absicht `GetOrderStatus` zu erfüllen. Der Bot führt dies durch, indem er den Status der letzten vom Benutzer aufgegebenen Bestellung zurückgibt.

Ein Beispiel für gemeinsame Nutzung von Informationen für mehrere Absichten finden Sie unter Reise buchen.

Festlegen von komplexen Attributen

Sitzungs- und Anforderungsattribute sind string-to-string Zuordnungen von Attributen und Werten. In vielen Fällen können Sie mit der Zeichenfolgen-Zuordnung Attributwerte zwischen Ihrer Clientanwendung und einem Bot übertragen. In einigen Fällen müssen Sie jedoch möglicherweise binäre Daten oder eine komplexe Struktur übertragen, die schwer in eine Zeichenfolgen-Zuordnung konvertiert werden kann. Das folgende JSON-Objekt stellt beispielsweise ein Array der drei beliebtesten Städte in den USA dar:

```
{
   "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"
         }
      }
   ]
}
```

Dieses Datenarray lässt sich nicht gut auf eine string-to-string Karte übertragen. In diesem Fall können Sie ein Objekt in eine einfache Zeichenfolge transformieren, sodass Sie sie mit den Operationen PostContent und PostText an Ihren Bot senden können.

Wenn Sie beispielsweise verwenden, können Sie die`JSON.stringify` Operation verwenden JavaScript, um ein Objekt in JSON zu konvertieren, und die`JSON.parse` Operation, um JSON-Text in ein JavaScript Objekt zu konvertieren:

```
// 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);
```

Um Sitzungsattribute mit dem`PostContent` Vorgang zu senden, müssen Sie die Attribute base64-codieren, bevor Sie sie dem Anforderungsheader hinzufügen, wie im folgenden JavaScript Code gezeigt:

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

Sie können binäre Daten an die Operationen `PostContent` und `PostText` senden, indem Sie die Daten zunächst in eine mit base64 kodierte Zeichenfolge konvertieren und diese Zeichenfolge dann als Wert in den Sitzungsattributen übermitteln:

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

Verwenden von Zuverlässigkeitswert

Wenn ein Benutzer eine Äußerung abgibt, verwendet Amazon Lex das natürliche Sprachverständnis (NLU), um die Anfrage des Benutzers zu verstehen und die richtige Absicht zurückzugeben. Standardmäßig gibt Amazon Lex die wahrscheinlichste Absicht zurück, die von Ihrem Bot definiert wurde.

In einigen Fällen kann es für Amazon Lex schwierig sein, die wahrscheinlichste Absicht zu bestimmen. Zum Beispiel könnte der Benutzer eine mehrdeutige Äußerung abgeben, oder es gibt zwei Absichten, die ähnlich sind. Um die richtige Absicht zu ermitteln, können Sie Ihr Domainwissen mit derZuverlässigkeitswerteiner Liste alternativer Absichten. Ein Konfidenzwert ist eine Bewertung von Amazon Lex, die zeigt, wie zuversichtlich es ist, dass eine Absicht die richtige Absicht ist.

Um den Unterschied zwischen zwei alternativen Absichten zu ermitteln, können Sie deren Konfidenzwerte vergleichen. Wenn beispielsweise eine Absicht einen Konfidenzwert von 0,95 hat und eine andere eine Punktzahl von 0,65 hat, ist die erste Absicht wahrscheinlich korrekt. Wenn jedoch eine Absicht eine Punktzahl von 0,75 hat und eine andere eine Punktzahl von 0,72 hat, besteht eine Unklarheit zwischen den beiden Absichten, die Sie möglicherweise mit Domainwissen in Ihrer Anwendung diskriminieren können.

Sie können auch Konfidenzwerte verwenden, um Testanwendungen zu erstellen, die bestimmen, ob Änderungen an den Äußerungen einer Absicht einen Unterschied im Verhalten des Bots machen. Beispielsweise können Sie die Konfidenzwerte für die Absichten eines Bots mithilfe einer Reihe von Äußerungen abrufen und dann die Absichten mit neuen Äußerungen aktualisieren. Sie können dann die Konfidenzwerte überprüfen, um festzustellen, ob es eine Verbesserung gab.

Die Konfidenzwerte, die Amazon Lex zurückgibt, sind Vergleichswerte. Sie sollten sich nicht auf sie als absolute Punktzahl verlassen. Die Werte können sich aufgrund von Verbesserungen an Amazon Lex ändern.

Wenn Sie Konfidenzwerte verwenden, gibt Amazon Lex die wahrscheinlichste Absicht und bis zu 4 alternative Absichten mit den zugehörigen Ergebnissen in jeder Antwort zurück. Wenn alle Konfidenzwerte unter einem Schwellenwert liegen, enthält Amazon Lex`AMAZON.FallbackIntent`, der`AMAZON.KendraSearchIntent`oder beides, wenn Sie sie konfiguriert haben. Sie können den Standardschwellenwert verwenden oder Ihren eigenen Schwellenwert festlegen.

Der folgende JSON-Code zeigt`alternativeIntents`Feld in der Antwort vonPostTextverwenden.

```
"alternativeIntents": [
   {
      "intentName": "string",
      "nluIntentConfidence": {
         "score": number
      },
      "slots": {
         "string" : "string"
      }
   }
],
```

Legen Sie den Schwellenwert fest, wenn Sie einen Bot erstellen oder aktualisieren. Sie können entweder die -API oder die Amazon Lex Lex-Konsole verwenden. Für die unten aufgeführten Regionen müssen Sie sich anmelden, um Genauigkeitsverbesserungen und Konfidenzwerte zu ermöglichen. Wählen Sie in der Konsole Konfidenzwerte imErweiterte OptionenAbschnitts erstellt. Legen Sie mit der -API die`enableModelImprovements`-Parameter beim Aufrufen desPutBotverwenden. :

- USA Ost (Nord-Virginia): (us-east-1)
- USA West (Oregon): (us-west-2)
- Asien-Pazifik (Sydney): (ap-southeast-2)
- Europa (Irland) (eu-west-1)

In allen anderen Regionen sind Standardgenauigkeitsverbesserungen und Konfidenz-Score-Unterstützung verfügbar.

Um den Konfidenzschwellenwert zu ändern, legen Sie ihn in der Konsole fest oder verwenden Sie diePutBotverwenden. Der Schwellenwert muss eine Zahl zwischen 1,00 und 0,00 liegen.

Um die Konsole zu verwenden, legen Sie den Konfidenzschwellenwert fest, wenn Sie Ihren Bot erstellen oder aktualisieren.

So legen Sie den Konfidenzschwellenwert beim Erstellen eines Bots fest (Konsole)

- InErstellen eines BotsGeben Sie im FeldSchwellenwertfield.

So aktualisieren Sie den Konfidenzschwellenwert (Konsole)

1. Wählen Sie in der Liste Ihrer Bots den Bot, der aktualisiert werden soll.
2. Wählen Sie die Registerkarte Settings.
3. Wählen Sie im linken NavigationsbereichAllgemeinesaus.
4. Aktualisieren Sie den Wert imSchwellenwertfield.

So setzen oder aktualisieren Sie den Konfidenzschwellenwert (SDK)

- Legen Sie den Wert für`nluIntentConfidenceThreshold`Parameter desPutBotverwenden. Der folgende JSON-Code zeigt den festzulegenden Parameter.

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

Verwaltung von Sitzungen

Um die Absicht zu ändern, die Amazon Lex in einem Gespräch mit dem Benutzer verwendet, können Sie die Antwort von Ihrem Dialogcode-Hook Lambda-Funktion verwenden oder die Sitzungsverwaltungs-APIs in Ihrer benutzerdefinierten Anwendung verwenden.

Verwenden einer Lambda-Funktion

Wenn Sie eine Lambda-Funktion verwenden, ruft Amazon Lex sie mit einer JSON-Struktur auf, die die Eingabe für die Funktion enthält. Die JSON-Struktur enthält ein Feld namens`currentIntent`das enthält die Absicht, die Amazon Lex als wahrscheinlichste Absicht für die Äußerung des Benutzers identifiziert hat. Die JSON-Struktur enthält auch eine`alternativeIntents`-Feld, das bis zu vier zusätzliche Absichten enthält, die die Absicht des Benutzers erfüllen können. Jede Absicht enthält ein Feld namens`nluIntentConfidenceScore`das enthält den Konfidenzwert, den Amazon Lex der Absicht zugewiesen hat.

Um eine alternative Absicht zu verwenden, geben Sie diese im`ConfirmIntent`oder das`ElicitSlot`Dialogaktion in Ihrer Lambda-Funktion.

Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen .

Verwenden der Session -Management-API

Um eine andere Absicht als die aktuelle Absicht zu verwenden, verwenden Sie diePutSessionverwenden. Wenn Sie beispielsweise entscheiden, dass die erste Alternative der von Amazon Lex gewählten Absicht vorzuziehen ist, können Sie die`PutSession`-Operation, um Absichten zu ändern, damit die nächste Absicht, mit der der Benutzer interagiert, diejenige ist, die Sie ausgewählt haben.

Weitere Informationen finden Sie unter Verwalten von Sitzungen mit der Amazon Lex API.

Konversationsprotokolle

Sie aktivieren Konversationsprotokolle zum Speichern von Bot-Interaktionen. Sie können diese Protokolle verwenden, um die Leistung Ihres Bots zu überprüfen und Probleme mit Konversationen zu beheben. Sie können Text für die PostText-Operation protokollieren. Sie können sowohl Text als auch Audio für die PostContent-Operation protokollieren. Wenn Sie Konversationsprotokolle aktivieren, erhalten Sie eine detaillierte Ansicht der Konversationen, die Benutzer mit Ihrem Bot haben.

Beispielsweise hat eine Sitzung mit Ihrem Bot eine Sitzungs-ID. Sie können diese ID verwenden, um das Transkript der Konversation zu erhalten, einschließlich der Äußerungen des Benutzers und der entsprechenden Bot-Antworten. Sie erhalten auch Metadaten wie den Namen der Absicht und Slot-Werte für eine Äußerung.

> **Note**
>
> Sie können keine Konversationsprotokolle mit einem Bot verwenden, der dem Children's Online Privacy Protection Act (COPPA) unterliegt.

Konversationsprotokolle werden für einen Alias konfiguriert. Jeder Alias kann unterschiedliche Einstellungen für seine Text- und Audioprotokolle haben. Sie können Textprotokolle, Audioprotokolle oder beides für jeden Alias aktivieren. Textprotokolle, Transkripte von Audioeingaben und zugehörige

Metadaten in CloudWatch Protokollen in Protokollen in Protokollen. Audioprotokolle speichern Audioeingaben in Amazon S3. Sie können die Verschlüsselung von Text- und Audioprotokollen mithilfe von kundenverwalteten AWS KMS-CMKs aktivieren.

Verwenden Sie die Konsole oder die PutBotAlias-Operation, um die Protokollierung zu konfigurieren. Sie können keine Konversationen für den`$LATEST` Alias Ihres Bots oder für den Test-Bot protokollieren, der in der Amazon Lex Lex-Konsole verfügbar ist. Nach der Aktivierung von Konversationsprotokollen für einen AliasPostContent oder einemPostText Vorgang für diesen Alias werden die Text- oder Audioäußerungen in der konfigurierten Protokollgruppe Logs oder im S3-Bucket CloudWatch protokolliert.

Themen

- IAM-Richtlinien für Konversationsprotokolle
- Konfigurieren von Konversationsprotokollen
- Verschlüsseln von Konversationsprotokollen
- Textprotokolle in Amazon CloudWatch Logs anzeigen
- Zugreifen auf Audioprotokolle in Amazon S3
- Überwachung des Konversationsprotokollstatus mit CloudWatch Metriken

IAM-Richtlinien für Konversationsprotokolle

Abhängig von der Art der Protokollierung, die Sie auswählen, benötigt Amazon Lex die Erlaubnis, Amazon CloudWatch Logs- und Amazon Simple Storage Service (S3) -Buckets zum Speichern Ihrer Protokolle zu verwenden. Sie müssenAWS Identity and Access Management Rollen und Berechtigungen erstellen, damit Amazon Lex auf diese Ressourcen zugreifen kann.

Erstellen einer IAM-Rolle und von Richtlinien für Konversationsprotokolle

Um Konversationsprotokolle zu aktivieren, müssen Sie Schreibrechte für CloudWatch Logs und Amazon S3 erteilen. Wenn Sie die Objektverschlüsselung für Ihre S3-Objekte aktivieren, müssen Sie den AWS KMS-Schlüsseln, die zum Verschlüsseln der Objekte verwendet werden, Zugriffsberechtigung erteilen.

Sie können den IAMAWS Management Console, die IAM-API oder den verwenden,AWS Command Line Interface um die Rolle und die Richtlinien zu erstellen. Diese Anweisungen verwenden die AWS CLI, um die Rolle und die Richtlinien zu erstellen. Informationen zum Erstellen AWS Identity and Access Management Richtlinien

> **Note**
>
> Der folgende Code ist für Linux und MacOS formatiert. Ersetzen Sie unter Windows das Linux-Zeilenfortsetzungszeichen (\) durch ein Caret-Zeichen (^).

So erstellen Sie eine IAM-Rolle für Konversationsprotokolle

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis mit dem Namen **`LexConversationLogsAssumeRolePolicyDocument.json`**, fügen Sie den folgenden Code hinzu und speichern Sie es. In diesem Richtliniendokument wird Amazon Lex der Rolle als vertrauenswürdige Entität hinzugefügt. Auf diese Weise kann Lex die Rolle übernehmen, um Protokolle für die Ressourcen bereitzustellen, die für Konversationsprotokolle konfiguriert sind.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "lex.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

2. Führen Sie in den den folgenden Befehl ausAWS CLI, um die IAM-Rolle für Konversationsprotokolle für Konversationsprotokolle zu erstellen.

   ```
   aws iam create-role \
       --role-name role-name \
       --assume-role-policy-document file://
   LexConversationLogsAssumeRolePolicyDocument.json
   ```

Erstellen Sie als Nächstes eine Richtlinie und weisen Sie sie der Rolle an, die Rolle an, die Amazon Lex das Schreiben CloudWatch in Die

So können Sie eine IAM-Richtlinie für die Protokollierung von Konversationstexten in CloudWatch Protokollen

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis namens**`LexConversationLogsCloudWatchLogsPolicy.json`**, fügen Sie die folgende IAM-Richtlinie hinzu und speichern Sie es.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "logs:CreateLogStream",
              "logs:PutLogEvents"
          ],
          "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:*"
      }
  ]
}
```

2. Erstellen Sie in derAWS CLI die IAM-Richtlinie, die der Protokollgruppe CloudWatch Logs Schreibrechte gewährt.

```
aws iam create-policy \
    --policy-name cloudwatch-policy-name \
    --policy-document file://LexConversationLogsCloudWatchLogsPolicy.json
```

3. Fügen Sie die Richtlinie an die IAM-Rolle an, die Sie für Konversationsprotokolle für Konversationsprotokolle an.

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

Wenn Sie Audio in einem S3-Bucket protokollieren, erstellen Sie eine Richtlinie, die es Amazon Lex ermöglicht, in den Bucket zu schreiben.

So erstellen Sie eine IAM-Richtlinie für die Audioprotokollierung in einem S3-Bucket

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis mit dem Namen **`LexConversationLogsS3Policy.json`**, fügen Sie die folgende Richtlinie hinzu und speichern Sie es.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:PutObject"
          ],
          "Resource": "arn:aws:s3:::bucket-name/*"
      }
  ]
}
```

2. Erstellen Sie in derAWS CLI die IAM-Richtlinie, die Ihrem S3-Bucket Schreibrechte gewährt.

```
aws iam create-policy \
    --policy-name s3-policy-name \
    --policy-document file://LexConversationLogsS3Policy.json
```

3. Fügen Sie die Richtlinie der Rolle an, die Sie für Konversationsprotokolle erstellt haben.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::account-id:policy/s3-policy-name \
    --role-name role-name
```

Erteilung der Erlaubnis zum Übergeben einer IAM-Rolle

Wenn Sie die Konsole, das oder einAWS SDK verwendenAWS Command Line Interface, um eine IAM-Rolle für Konversationsprotokolle anzugeben, muss der Benutzer, der die IAM-Rolle für Konversationsprotokolle angibt, über die Berechtigung verfügen, die Rolle an Amazon Lex zu übergeben. Damit der Benutzer die Rolle an Amazon Lex übergeben kann, müssen Sie dem Benutzer, der Rolle oder der Gruppe die`PassRole` Berechtigung verleihen.

Die folgende Richtlinie definiert die Berechtigung, die dem Benutzer, der Rolle oder der Gruppe erteilt werden soll. Sie können die `iam:AssociatedResourceArn` und `iam:PassedToService`-Bedingungsschlüssel verwenden, um den Umfang der Berechtigung einzuschränken. Weitere Informationen finden Sie im Benutzerhandbuch unter Gewähren von Benutzerberechtigungen zum Übergeben einer Rolle an einenAWS Dienst sowie unter IAM- undAWS STS Bedingungskontextschlüsseln.AWS Identity and Access Management

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lex.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": "arn:aws:lex:region:account-
id:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

Konfigurieren von Konversationsprotokollen

Sie aktivieren und deaktivieren Konversationsprotokolle über die Konsole oder das `conversationLogs`-Feld der `PutBotAlias`-Operation. Sie können Audioprotokolle, Textprotokolle oder beides aktivieren oder deaktivieren. Die Protokollierung beginnt bei neuen Botsitzungen. Änderungen an den Protokolleinstellungen werden für aktive Sitzungen nicht berücksichtigt.

Verwenden Sie eine Amazon Logs-Protokollgruppe in IhremAWS Konto, um CloudWatch Textprotokolle zu speichern. Sie können jede gültige Protokollgruppe verwenden. Die Protokollgruppe muss sich in derselben Region befinden wie der Amazon-Lex-Bot in derselben Region befinden wie der Amazon-Lex-Bot. Weitere Informationen zum Erstellen einer CloudWatch Protokollgruppe finden Sie unter Arbeiten mit Protokollgruppen und Protokollstreams im Amazon CloudWatch Logs-Benutzerhandbuch.

Verwenden Sie einen Amazon S3 S3-Bucket in IhremAWS Konto, um Audioprotokolle zu speichern. Sie können jeden gültigen S3-Bucket verwenden. Der Bucket muss sich in derselben Region befinden wie der Amazon-Lex-Bot. Weitere Informationen zum Erstellen eines S3-Buckets finden Sie unter Erstellen eines Buckets im Amazon Simple Storage Storage Service Getting Started Guide für Amazon Simple Storage Storage Service Getting Started Guide für Amazon Simple Storage Storage Service

Sie müssen eine IAM-Rolle mit Richtlinien bereitstellen, die es Amazon Lex ermöglichen, in die konfigurierte Protokollgruppe oder den konfigurierten Bucket zu schreiben. Weitere Informationen finden Sie unter Erstellen einer IAM-Rolle und von Richtlinien für Konversationsprotokolle.

Wenn Sie eine serviceverknüpfte Rolle mithilfe von erstellenAWS Command Line Interface, müssen Sie der Rolle mithilfe der folgenden`custom-suffix` Option ein benutzerdefiniertes Suffix hinzufügen:

```
aws iam create-service-linked-role \
    --aws-service-name lex.amazon.aws.com \
    --custom-suffix suffix
```

Die IAM-Rolle, die Sie zum Aktivieren von Konversationsprotokollen verwenden, muss über die`iam:PassRole` entsprechende Berechtigung verfügen. Die folgende Richtlinie sollte an die Rolle angefügt werden.

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

Aktivieren von Konversationsprotokollen

So aktivieren Sie Protokolle über die Konsole

1. Öffnen Sie die Amazon Lex Lex-Konsole https://console.aws.amazon.com/lex.
2. Wählen Sie aus der Liste einen Bot aus.

3. Wählen Sie die Registerkarte Settings (Einstellungen) und dann im linken Menü Conversation logs (Konversationsprotokolle) aus.
4. Wählen Sie in der Liste der Aliasse das Einstellungssymbol für den Alias aus, für den Sie Konversationsprotokolle konfigurieren möchten.
5. Wählen Sie aus, ob Text, Audio oder beides protokolliert werden soll.
6. Geben Sie für die Textprotokollierung den Namen der Amazon CloudWatch Logs-Protokollgruppe ein.
7. Geben Sie für die Audioprotokollierung die S3-Bucket-Informationen ein.
8. Optional. Um Audioprotokolle zu verschlüsseln, wählen Sie den AWS KMS-Schlüssel für die Verschlüsselung aus.
9. Wählen Sie eine IAM-Rolle mit den erforderlichen Berechtigungen aus.
10. Wählen Sie Save (Speichern), um Konversationen zu protokollieren.

So aktivieren Sie Textprotokolle mit der API

1. Rufen Sie die PutBotAlias-Operation mit einem Eintrag im `logSettings`-Element des `conversationLogs`-Feldes auf.
   - Setzen Sie das `destination`-Element auf `CLOUDWATCH_LOGS`.
   - Setzen Sie das `logType`-Element auf `TEXT`.
   - Geben Sie das`resourceArn` Mitglied auf den Amazon-Ressourcennamen (ARN) der CloudWatch Protokollgruppe für die -Konsole als Ziel für die Die
2. Geben Sie dem`iamRoleArn` Element des`conversationLogs` Felds auf den Amazon-Ressourcennamen (ARN) einer IAM-Rolle an, die Berechtigung zum Aktivieren von Konversationsprotokollen für die angegebenen Ressourcen an.

So aktivieren Sie Audioprotokolle mit der API

1. Rufen Sie die PutBotAlias-Operation mit einem Eintrag im `logSettings`-Element des `conversationLogs`-Feldes auf.
   - Setzen Sie das `destination`-Element auf `S3`.
   - Setzen Sie das `logType`-Element auf `AUDIO`.
   - Setzen Sie das `resourceArn`-Element auf den ARN des Amazon S3-Buckets, in dem die Audioprotokolle gespeichert sind.

- Optional. Um Audioprotokolle mit einem bestimmten AWS KMS-Schlüssel zu verschlüsseln, setzen Sie das `kmsKeyArn`-Element auf den ARN des Schlüssels, der für die Verschlüsselung verwendet wird.

2. Geben Sie dem`iamRoleArn` Element des`conversationLogs` Felds auf den Amazon-Ressourcennamen (ARN) einer IAM-Rolle an, die Berechtigung zum Aktivieren von Konversationsprotokollen für die angegebenen Ressourcen an.

Deaktivieren von Konversationsprotokollen

So deaktivieren Sie Protokolle über die Konsole

1. Öffnen Sie die Amazon Lex Lex-Konsole https://console.aws.amazon.com/lex.
2. Wählen Sie aus der Liste einen Bot aus.
3. Wählen Sie die Registerkarte Settings (Einstellungen) und dann im linken Menü Conversation logs (Konversationsprotokolle) aus.
4. Wählen Sie in der Liste der Aliasse das Einstellungssymbol für den Alias aus, für den Sie Konversationsprotokolle konfigurieren möchten.
5. Entfernen Sie die Markierung bei Text, Audio oder beidem, um die Protokollierung zu deaktivieren.
6. Wählen Sie Save (Speichern), um die Protokollierung von Konversationen zu beenden.

So deaktivieren Sie Protokolle mit der API

- Rufen Sie die `PutBotAlias`-Operation ohne das `conversationLogs`-Feld auf.

So deaktivieren Sie Textprotokolle mit der API

- • Wenn Sie Audio protokollieren:
    - Rufen Sie die PutBotAlias-Operation mit einem `logSettings`-Eintrag nur für `AUDIO` auf.
    - Der Aufruf der `PutBotAlias`-Operation darf keinen `logSettings`-Eintrag für `TEXT` haben.
  - Wenn Sie keine Audiodaten protokollieren
    - Rufen Sie die PutBotAlias-Operation ohne das `conversationLogs`-Feld auf.

So deaktivieren Sie Audioprotokolle mit der API

- • Wenn Sie Text protokollieren
  - Rufen Sie die PutBotAlias-Operation mit einem `logSettings`-Eintrag nur für `TEXT` auf.
  - Der Aufruf der `PutBotAlias`-Operation darf keinen `logSettings`-Eintrag für `AUDIO` haben.
- Wenn Sie keinen Text protokollieren
  - Rufen Sie die PutBotAlias-Operation ohne das `conversationLogs`-Feld auf.

Verschlüsseln von Konversationsprotokollen

Sie können eine Verschlüsselung verwenden, um den Inhalt Ihrer Konversationsprotokolle zu schützen. Für Text- und Audioprotokolle können Sie vomAWS KMS Kunden verwaltete CMKs verwenden, um Daten in Ihrer CloudWatch Logs-Protokollgruppe und Ihrem S3-Bucket zu verschlüsseln.

> **Note**
>
> Amazon Lex unterstützt nur symmetrische CMKs. Sie können kein asymmetrisches CMK verwenden, um Ihre Daten zu verschlüsseln.

Sie aktivieren die Verschlüsselung mithilfe einesAWS KMS Schlüssels in der Protokollgruppe CloudWatch Logs, die Amazon Lex für Textprotokolle verwendet. Sie können keinen AWS KMS-Schlüssel in den Protokolleinstellungen angeben, um die AWS KMS-Verschlüsselung Ihrer Protokollgruppe zu aktivieren. Weitere Informationen finden Sie unter Verschlüsseln von ProtokolldatenAWS KMS in CloudWatch CloudWatch Protokollen mithilfe

Für Audioprotokolle verwenden Sie die Standardverschlüsselung auf Ihrem S3-Bucket oder geben einen AWS KMS-Schlüssel zum Verschlüsseln Ihrer Audioobjekte an. Selbst wenn Ihr S3-Bucket Standardverschlüsselung verwendet, können Sie dennoch einen anderen AWS KMS-Schlüssel zum Verschlüsseln Ihrer Audioobjekte angeben. Weitere Informationen finden Sie unter Amazon S3 S3-Standardverschlüsselung für S3-Buckets im Amazon Simple Storage Storage Service-Entwicklerhandbuch.

Amazon Lex benötigtAWS KMS Berechtigungen, wenn Sie Ihre Audioprotokolle verschlüsseln möchten. Sie müssen der IAM-Rolle für Konversationsprotokolle hinzufügen. Wenn Sie die

Standardverschlüsselung für Ihren S3-Bucket verwenden, muss Ihre Richtlinie Zugriff auf den für diesen Bucket konfigurierten AWS KMS-Schlüssel gewähren. Wenn Sie einen AWS KMS-Schlüssel in den Audioprotokolleinstellungen angeben, müssen Sie den Zugriff auf diesen Schlüssel gewähren.

Wenn Sie keine Rolle für Konversationsprotokolle erstellt haben, finden Sie weitere Informationen unter IAM-Richtlinien für Konversationsprotokolle.

So erstellen Sie eine IAM-Richtlinie für die Verwendung einesAWS KMS Schlüssels zum Verschlüsseln von Audioprotokollen

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis mit dem Namen **`LexConversationLogsKMSPolicy.json`**, fügen Sie die folgende Richtlinie hinzu und speichern Sie es.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "kms:GenerateDataKey"
          ],
          "Resource": "kms-key-arn"
      }
  ]
}
```

2. Erstellen Sie in der die IAM-RichtlinieAWS CLI, die die Erlaubnis erteilt, denAWS KMS Schlüssel für die Verschlüsselung von Audioprotokollen zu verwenden.

```
aws iam create-policy \
    --policy-name kms-policy-name \
    --policy-document file://LexConversationLogsKMSPolicy.json
```

3. Fügen Sie die Richtlinie der Rolle an, die Sie für Konversationsprotokolle erstellt haben.

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

Textprotokolle in Amazon CloudWatch Logs anzeigen

Amazon Lex speichert Textprotokolle für Ihre Konversationen in Amazon CloudWatch Logs. Um die Protokolle anzuzeigen, können Sie die CloudWatch Logs-Konsole oder die API verwenden. Weitere Informationen finden Sie unter Durchsuchen von Protokolldaten mithilfe von Filtermustern und CloudWatch Logs Insights-Abfragesyntax im Amazon CloudWatch Logs-Benutzerhandbuch.

Amazon Lex

1. Öffnen Sie die Amazon Lex Lex-Konsole https://console.aws.amazon.com/lex.
2. Wählen Sie aus der Liste einen Bot aus.
3. Wählen Sie die Registerkarte Settings (Einstellungen) und dann im linken Menü Conversation logs (Konversationsprotokolle).
4. Wählen Sie den Link unter Textprotokolle, um die Protokolle für den Alias in der CloudWatch Konsole anzuzeigen.

Sie können auch die CloudWatch Konsole oder die API verwenden, um Ihre Protokolleinträge einzusehen. Um die Protokolleinträge zu suchen, navigieren Sie zu der Protokollgruppe, die Sie für den Alias konfiguriert haben. Sie finden das Log-Stream-Präfix für Ihre Protokolle in der Amazon Lex Lex-Konsole oder mithilfe desGetBotAlias Vorgangs.

Protokolleinträge für eine Benutzeräußerungen befinden sich in mehreren Protokolldatenstreams. Eine Äußerung in der Konversation hat einen Eintrag in einem der Protokolldatenstreams mit dem angegebenen Präfix. Ein Eintrag im Protokolldatenstream enthält die folgenden Informationen.

```
{
   "messageVersion": "1.0",
   "botName": "bot name",
   "botAlias": "bot alias",
   "botVersion": "bot version",
   "inputTranscript": "text used to process the request",
   "botResponse": "response from the bot",
   "intent": "matched intent",
   "nluIntentConfidence": "number",
   "slots": {
       "slot name": "slot value",
       "slot name": null,
       "slot name": "slot value"
       ...
```

```
  },
  "alternativeIntents": [
      {
          "name": "intent name",
          "nluIntentConfidence": "number",
          "slots": {
              "slot name": slot value,
              "slot name": null,
              "slot name": slot value
              ...
          }
      },
      {
          "name": "intent name",
          "nluIntentConfidence": number,
          "slots": {}
      }
  ],
  "developerOverride": "true" | "false",
  "missedUtterance": true | false,
  "inputDialogMode": "Text" | "Speech",
  "requestId": "request ID",
  "s3PathForAudio": "S3 path to audio file",
  "userId": "user ID",
  "sessionId": "session ID",
  "sentimentResponse": {
      "sentimentScore": "{Positive: number, Negative: number, Neutral: number,
Mixed: number}",
      "sentimentLabel": "Positive" | "Negative" | "Neutral" | "Mixed"
  },
  "slotToElicit": "slot name",
  "dialogState": "ElicitIntent" | "ConfirmIntent" | "ElicitSlot" | "Fulfilled" |
"ReadyForFulfillment" | "Failed",
  "responseCard": {
      "genericAttachments": [
          ...
      ],
      "contentType": "application/vnd.amazonaws.card.generic",
      "version": 1
   },
  "locale": "locale",
  "timestamp": "ISO 8601 UTC timestamp",
  "kendraResponse": {
     "totalNumberOfResults": number,
```

```
    "resultItems": [
        {
            "id": "query ID",
            "type": "DOCUMENT" | "QUESTION_ANSWER" | "ANSWER",
            "additionalAttributes": [
                {
                   ...
                }
            ],
            "documentId": "document ID",
            "documentTitle": {
                "text": "title",
                "highlights": null
            },
            "documentExcerpt": {
                "text": "text",
                "highlights": [
                    {
                        "beginOffset": number,
                        "endOffset": number,
                        "topAnswer": true | false
                    }
                ]
            },
            "documentURI": "URI",
            "documentAttributes": []
        }
    ],
    "facetResults": [],
    "sdkResponseMetadata": {
        "requestId": "request ID"
    },
    "sdkHttpMetadata": {
        "httpHeaders": {
            "Content-Length": "number",
            "Content-Type": "application/x-amz-json-1.1",
            "Date": "date and time",
            "x-amzn-RequestId": "request ID"
        },
        "httpStatusCode": 200
    },
    "queryId": "query ID"
},
"sessionAttributes": {
```

```
        "attribute name": "attribute value"
        ...
    },
   "requestAttributes": {
        "attribute name": "attribute value"
        ...
    }
}
```

Der Inhalt des Protokolleintrags hängt vom Ergebnis einer Transaktion und der Konfiguration des Bots und der Anfrage ab.

- Die Felder `intent`, `slots` und `slotToElicit` werden nicht in einem Eintrag angezeigt, wenn das `missedUtterance`-Feld den Wert `true` hat.
- Das `s3PathForAudio`-Feld wird nicht angezeigt, wenn Audioprotokolle deaktiviert sind oder wenn das `inputDialogMode`-Feld `Text` ist.
- Das `responseCard`-Feld wird nur angezeigt, wenn Sie eine Antwortkarte für den Bot definiert haben.
- Die `requestAttributes`-Karte wird nur angezeigt, wenn Sie in der Anforderung Anforderungsattribute angegeben haben.
- Das`kendraResponse` Feld ist nur vorhanden, wenn er eine Anfrage zur Suche in einem Amazon Kendra Kendra-Index`AMAZON.KendraSearchIntent` stellt.
- Das`developerOverride` Feld ist wahr, wenn in der Lambda-Funktion des Bots eine alternative Absicht angegeben wurde.
- Die `sessionAttributes`-Karte wird nur angezeigt, wenn Sie Sitzungsattribute in der Anforderung angegeben haben.
- Die `sentimentResponse`-Karte wird nur angezeigt, wenn Sie den Bot so konfigurieren, dass er Stimmungswerte zurückgibt.

> **Note**
>
> Das Eingabeformat kann sich auch ohne entsprechende Änderung der `messageVersion` ändern. Ihr Code sollte keinen Fehler ausgeben, wenn neue Felder vorhanden sind.

Sie müssen über eine Rolle und eine Richtlinie verfügen, damit Amazon Lex in CloudWatch Logs schreiben kann. Weitere Informationen finden Sie unter IAM-Richtlinien für Konversationsprotokolle.

Zugreifen auf Audioprotokolle in Amazon S3

Amazon Lex speichert Audioprotokolle für Ihre Konversationen in einem S3-Bucket in einem S3-Bucket.

So greifen Sie über die Konsole auf Audioprotokolle zu

1. Öffnen Sie die Amazon Lex Lex-Konsole https://console.aws.amazon.com/lex.
2. Wählen Sie aus der Liste einen Bot aus.
3. Wählen Sie die Registerkarte Settings (Einstellungen) und dann im linken Menü Conversation logs (Konversationsprotokolle).
4. Wählen Sie den Link unter Audioprotokolle, um auf die Protokolle für den Alias in der Amazon S3 S3-Konsole zuzugreifen.

Sie können auch die Amazon S3 S3-Konsole oder die API verwenden, um auf Audioprotokolle zuzugreifen. Sie können das S3-Objektschlüsselpräfix der Audiodateien in der Amazon Lex-Konsole oder in dem`resourcePrefix` Feld in der`GetBotAlias` Operationsantwort sehen.

Überwachung des Konversationsprotokollstatus mit CloudWatch Metriken

Verwenden Sie Amazon CloudWatch , um die Lieferkennzahlen Ihrer Konversationsprotokolle zu überwachen. Sie können Alarme für Metriken festlegen, damit Sie Probleme mit der Protokollierung erkennen, wenn sie auftreten sollten.

Amazon Lex bietet vier Metriken im`AWS/Lex` Namespace für Konversationsprotokolle:

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

Weitere Informationen finden Sie unter CloudWatch Metriken für Konversationsprotokolle.

Die Erfolgskennzahlen zeigen, dass Amazon Lex Ihre Audio- oder Textprotokolle erfolgreich an ihre Ziele geschrieben hat.

Die Fehlerkennzahlen zeigen, dass Amazon Lex keine Audio- oder Textprotokolle an das angegebene Ziel senden konnte. In der Regel ist dies ein Konfigurationsfehler. Wenn Ihre Fehlermetriken über Null liegen, überprüfen Sie Folgendes:

- Stellen Sie sicher, dass Amazon Lex eine vertrauenswürdige Entität für die IAM-Rolle ist.
- Stellen Sie für die Textprotokollierung sicher, dass die Protokollgruppe CloudWatch Logs existiert. Stellen Sie für die Audioprotokollierung sicher, dass der S3-Bucket vorhanden ist.
- Stellen Sie sicher, dass die IAM-Rolle, die Amazon Lex für den Zugriff auf die CloudWatch Logs-Protokollgruppe oder den S3-Bucket verwendet, über Schreibberechtigungen für die Protokollgruppe oder den Bucket verfügt.
- Stellen Sie sicher, dass der S3-Bucket in derselben Region wie der Amazon Lex Lex-Bot existiert und zu Ihrem Konto gehört.
- Wenn Sie einenAWS KMS Schlüssel für die S3-Verschlüsselung verwenden, stellen Sie sicher, dass es keine Richtlinien gibt, die Amazon Lex daran hindern, Ihren Schlüssel zu verwenden, und stellen Sie sicher, dass die von Ihnen angegebene IAM-Rolle über die erforderlichenAWS KMS Berechtigungen verfügt. Weitere Informationen finden Sie unter IAM-Richtlinien für Konversationsprotokolle.

Verwalten von Sitzungen mit der Amazon Lex API

Wenn ein Benutzer eine Konversation mit Ihrem Bot startet, erstellt Amazon Lex eineSitzungaus. Die zwischen Ihrer Anwendung und Amazon Lex ausgetauschten Informationen ergeben den Sitzungsstatus für die Konversation. Wenn Sie eine Anforderung absenden, wird die Sitzung anhand einer Kombination aus dem Bot-Namen und einer von Ihnen angegebenen Benutzer-ID identifiziert. Weitere Informationen über die Benutzer-ID finden Sie im Feld `userId` in der Operation PostContent oder PostText.

Die Antwort aus einer Sitzungsoperation enthält eine eindeutige Sitzungs-ID, anhand der eine bestimmte Sitzung mit einem Benutzer identifiziert wird. Sie können diese Kennung beim Testen oder zum Beheben von Problemen mit Ihrem Bot verwenden.

Sie können den zwischen Ihrer Anwendung und Ihrem Bot gesendeten Sitzungsstatus ändern. Beispielsweise können Sie Sitzungsattribute erstellen und ändern, die benutzerdefinierte Informationen zur Session enthalten. Außerdem können Sie den Gesprächsablauf ändern, indem Sie den Dialogkontext für die Interpretation der nächste Äußerung festlegen.

Es gibt zwei Möglichkeiten, wie Sie den Sitzungsstatus aktualisieren können. Die erste besteht darin, eine Lambda-Funktion mit dem`PostContent`oder`PostText`operation, die nach jeder Konversationsschritt aufgerufen wird. Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen . Die andere besteht in der Verwendung der Amazon Lex -Laufzeit-API in Ihrer Anwendung, um Änderungen am Sitzungsstatus vorzunehmen.

Die Amazon Lex -Laufzeit-API bietet Operationen, mit denen Sie die Sitzungsinformationen für eine Konversation mit Ihrem Bot verwalten können. Hierzu gehören die Operation PutSession, die Operation GetSession und die Operation DeleteSession. Sie können mit diesen Operationen Informationen über den Status der Sitzung Ihres Benutzers mit Ihrem Bot anfordern und eine differenzierte Kontrolle über den Status erlangen.

Verwenden Sie die Operation `GetSession`, wenn Sie den aktuellen Status der Sitzung anfordern möchten. Die Operation gibt den aktuellen Status der Sitzung, einschließlich des Status des Dialogs mit Ihrem Benutzer und aller festgelegten Sitzungsattribute und Slot-Werte für die letzten drei Absichten zurück, mit denen der Benutzer interagiert hat.

Die Operation `PutSession` ermöglicht es Ihnen, den aktuellen Sitzungsstatus direkt zu bearbeiten. Sie können die Art von Dialogaktion festlegen, die der Bot als Nächstes ausführen wird. Dadurch haben Sie Kontrolle über den Gesprächsablauf mit dem Bot. Festlegen der Dialogaktion`type`field zu`Delegated`damit Amazon Lex die nächste Aktion für den Bot bestimmt.

Sie können mit der Operation `PutSession` eine neue Sitzung mit einem Bot erstellen und die Absicht festlegen, mit der der Bot beginnen soll. Sie können mit der Operation `PutSession` auch von einer Absicht zu einer anderen wechseln. Wenn Sie eine Sitzung erstellen oder die Absicht ändern, können Sie auch den Sitzungsstatus, wie z. B. Slot-Werte und Sitzungsattribute, festlegen. Wenn die neue Absicht abgeschlossen ist, haben Sie die Möglichkeit, die vorherige Absicht neu zu starten. Sie können das`GetSession`-Operation, um den Dialogstatus der vorherigen Absicht von Amazon Lex zu erhalten. Anhand dieser Informationen können Sie dann den Dialogstatus der Absicht festlegen.

Die Antwort von der Operation `PutSession` enthält die gleichen Informationen wie die von der Operation `PostContent`. Sie können diese Informationen, genauso wie die Antwort von der Operation `PostContent`, verwenden, um vom Benutzer die nächste Teilinformation anzufordern.

Sie können mit der Operation `DeleteSession` eine vorhandene Sitzung entfernen und mit einer neuen Sitzung ganz von vorne beginnen. Wenn Sie beispielsweise Ihren Bot testen, können Sie mit der Operation `DeleteSession` Testsitzungen von Ihrem Bot entfernen.

Die Sitzungsoperationen funktionieren mit Ihren -Fulfillment-Lambda-Funktionen. Beispiel: Ihre Lambda-Funktion kehrt zurück`Failed`als Erfüllungsstatus können Sie die`PutSession`-Operation, um den Dialogaktionsstatus auf`close`und`fulfillmentState`zu`ReadyForFulfillment`um den Erfüllungsschritt erneut zu versuchen.

Es folgen einige Aufgaben, die Sie mit den Sitzungsoperationen ausführen können:

- Veranlassen des Bots zum Starten einer Konversation, anstatt auf den Benutzer zu warten.
- Wechseln von Absichten während einer Konversation.
- Zurückkehren zur einer vorherigen Absicht.
- Starten oder Neustarten einer Konversation während der Interaktion.
- Validieren von Slot-Werten und Veranlassen des Bots, für ungültige Werte neue Werte anzufordern.

Jede dieser Aufgaben wird im Folgenden beschrieben.

Wechseln von Absichten

Sie können mit der Operation `PutSession` von einer Absicht zu einer anderen wechseln. Sie können mit ihr auch zu einer vorherigen Absicht zurückwechseln. Sie können mit der Operation `PutSession` Sitzungsattribute oder Slot-Werte für die neue Absicht einstellen.

- Aufrufen der `PutSession`-Operation. Legen Sie als Absichtsnamen den Namen der neuen Absicht und als Dialogaktion `Delegate` fest. Sie können auch alle für die neue Absicht erforderlichen Slot-Werte oder Sitzungsattribute festlegen.
- Amazon Lex beginnt eine Konversation mit dem Benutzer unter Verwendung der neuen Absicht.

Fortsetzen einer vorherigen Absicht

Um eine vorherige Absicht fortzusetzen, fordern Sie mit der Operation `GetSession` eine Zusammenfassung der Absicht an und legen Sie mit der Operation `PutSession` dann wieder den vorherigen Dialogstatus der Absicht fest.

- Aufrufen der `GetSession`-Operation. Die Antwort von der Operation enthält eine Zusammenfassung des Dialogstatus der letzten drei Absichten, mit denen der Benutzer interagiert hat.

- Rufen Sie mittels der Informationen aus der Absichtszusammenfassung die Operation `PutSession` auf. Dadurch gelangt der Benutzer wieder zur vorherigen Absicht an der gleichen Stelle im Gespräch.

In einigen Fällen kann es erforderlich sein, dass die Konversation Ihres Benutzers mit Ihrem Bot fortgesetzt wird. Angenommen, Sie haben einen Kundenservice-Bot erstellt. Ihre Anwendung stellt fest, dass der Benutzer mit einem Kundendienstmitarbeiter sprechen muss. Nach der Unterhaltung mit dem Benutzer kann der Mitarbeiter das Gespräch mit den vom ihm erfassten Informationen wieder an den Bot weiterleiten.

Um eine Sitzung fortzusetzen, verwenden Sie Schritte ähnlich wie diese:

- Ihre Anwendung stellt fest, dass der Benutzer mit einem Kundendienstmitarbeiter sprechen muss.
- Fordern Sie mit der Operation `GetSession` den aktuellen Dialogstatus der Absicht an.
- Der Kundendienstmitarbeiter spricht mit dem Benutzer und löst das Problem.
- Legen Sie mit der Operation `PutSession` den Dialogstatus der Absicht fest. Hierzu gehören möglicherweise Festlegen von Slot-Werten, Einstellen von Sitzungsattributen oder Ändern der Absicht.
- Der Bot setzt die Konversation mit dem Benutzer fort.

Sie können den `checkpointLabel`-Parameter der `PutSession`-Operation verwenden, um eine Absicht zu beschriften, damit Sie sie später finden können. Beispielsweise kann ein Bot, der einen Kunden um Informationen bittet, eine `Waiting`-Absicht eingehen, während der Kunde die Informationen zusammenträgt. Der Bot erstellt eine Prüfpunktbeschriftung für die aktuelle Absicht und startet die `Waiting`-Absicht dann. Wenn der Kunde zurückkehrt, kann der Bot die vorherige Absicht mithilfe der Prüfpunktbeschriftung finden und zurückschalten.

Die Absicht muss in der `recentIntentSummaryView`-Struktur vorhanden sein, die von der `GetSession`-Operation zurückgegeben wird. Wenn Sie in der `GetSession`-Vorgangsanforderung eine Prüfpunktbeschriftung angeben, werden maximal drei Absichten mit dieser Prüfpunktbeschriftung zurückgegeben.

- Verwenden Sie den `GetSession`-Vorgang, um den aktuellen Status der Sitzung abzurufen.
- Verwenden Sie den `PutSession`-Vorgang, um der letzten Absicht eine Prüfpunktbeschriftung hinzuzufügen. Bei Bedarf können Sie diesen `PutSession`-Aufruf verwenden, um zu einer anderen Absicht zu wechseln.

- Wenn es an der Zeit ist, zur beschrifteten Absicht zurückzukehren, rufen Sie den `GetSession`-Vorgang auf, um eine aktuelle Absichtsliste zurückzugeben. Sie können das`checkpointLabelFilter`-Parameter, damit Amazon Lex nur Absichten mit der angegebenen Prüfpunktbeschriftung zurück gibt.

Starten einer neuen Sitzung

Wenn Sie möchten, dass der Bot die Konversation mit Ihrem Benutzer startet, können Sie dazu Operation `PutSession` verwenden.

- Erstellen Sie eine Begrüßungsabsicht ohne Slots und eine abschließende Nachricht, die den Benutzer auffordert, eine Absicht zu nennen. Beispiel: „Was möchten Sie bestellen? Sie können Folgendes sagen: 'Ein Getränk bestellen' oder 'Eine Pizza bestellen'.“
- Aufrufen der `PutSession`-Operation. Legen Sie als Absichtsnamen den Namen Ihrer Begrüßungsabsicht und als Dialogaktion `Delegate` fest.
- Amazon Lex antwortet mit der Aufforderung Ihrer Begrüßungsabsicht, um die Konversation mit Ihrem Benutzer zu starten.

Validieren der Slot-Werte

Sie können Antworten an Ihren Bot mit Ihrer Client-Anwendung validieren. Wenn die Antwort nicht gültig ist, können Sie mit der Operation `PutSession` eine neue Antwort von Ihrem Benutzer anfordern. Angenommen, Ihr Bot zur Aufnahme von Blumenbestellungen kann nur Rosen, Tulpen und Lilien verkaufen. Wenn der Benutzer Nelken bestellt, kann Ihre Anwendung wie folgt vorgehen:

- Untersuchen des Slot-Wertes, der von der Antwort `PostText` oder `PostContent` zurückgegeben wird.
- Wenn der Slot-Wert nicht gültig ist, Aufrufen der Operation `PutSession`. Ihre Anwendung sollte den Slot-Wert löschen, das Feld `slotToElicit` festlegen und den Wert `dialogAction.type` auf `elicitSlot` einstellen. Optional können Sie die`message`und`messageFormat`-Felder, wenn Sie die Nachricht ändern möchten, mit der Amazon Lex den Slot-Wert anfordert.

Bot-Bereitstellungsoptionen

Derzeit bietet Amazon Lex die folgenden Bot-Bereitstellungsoptionen:

- AWS Mobile SDK — Mithilfe der AWS Mobile SDKs können Sie mobile Anwendungen erstellen, die mit Amazon Lex kommunizieren.
- Facebook Messenger — Sie können Ihre Facebook Messenger-Seite in Ihren Amazon Lex Lex-Bot integrieren, sodass Endbenutzer auf Facebook mit dem Bot kommunizieren können. In der aktuellen Implementierung unterstützt diese Integration nur eingegebene Text-Mitteilungen.
- Slack — Sie können Ihren Amazon Lex Lex-Bot in eine Slack-Messaging-Anwendung integrieren.
- Twilio — Sie können Ihren Amazon Lex Lex-Bot in den Twilio Simple Messaging Service (SMS) integrieren.

Beispiele finden Sie unter Die Bereitstellung von Amazon Lex Bots.

Integrierte Absichten und Slot-Typen

Um die Erstellung von Bots zu vereinfachen, können Sie in Amazon Lex standardmäßig integrierte Intents und Slot-Typen verwenden.

Themen

- Integrierte Absichten
- Integrierte Slot-Typen

Integrierte Absichten

Für allgemeine Aktionen können Sie die standardmäßige integrierte Intents-Bibliothek verwenden. Um eine Absicht aus einer integrierten Absicht zu erstellen, wählen Sie eine integrierte Absicht in der Konsole und weisen ihr einen neuen Namen zu. Die neue Absicht hat die Konfiguration der Basisabsicht, wie z. B. die Beispieläußerungen.

In der aktuellen Implementierung ist Folgendes nicht möglich:

- Fügen Sie der Basisabsicht Beispieläußerungen hinzu oder entfernen Sie sie
- Konfigurieren von Slots für integrierte Absichten

Um einem Bot eine integrierte Absicht hinzuzufügen

1. Melden Sie sich bei der Amazon Lex Lex-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/lex/.

2. Wählen Sie den Bot aus, dem Sie die integrierte Absicht hinzufügen möchten.
3. Wählen Sie im Navigationsbereich das Pluszeichen (+) neben Intents (Absichten).
4. Wählen Sie unter Add intent (Absicht hinzufügen) die Option Search existing intents (Vorhandene Absichten durchsuchen).
5. Geben Sie im Feld Suchabsichten den Namen der integrierten Absicht ein, die Sie Ihrem Bot hinzufügen möchten.
6. Geben Sie unter Integrierte Absicht kopieren einen Namen für die Absicht ein und wählen Sie dann Hinzufügen aus.
7. Konfigurieren Sie die Absicht nach Bedarf für Ihren Bot.

Themen

- AMAZON.CancelIntent
- AMAZON.FallbackIntent
- AMAZON.HelpIntent
- AMAZON.KendraSearchIntent
- AMAZON.PauseIntent
- AMAZON.RepeatIntent
- AMAZON.ResumeIntent
- AMAZON.StartOverIntent
- AMAZON.StopIntent

> **Note**
>
> Für das Gebietsschema Englisch (USA) (en-US) unterstützt Amazon Lex Absichten aus den integrierten Alexa-Standardabsichten. Die Liste der integrierten Absichten finden Sie unter Standard Built-in Intents (Integrierte Standardabsichten) im Alexa Skills Kit.
> Amazon Lex unterstützt die folgenden Absichten nicht:
>
> - `AMAZON.YesIntent`
> - `AMAZON.NoIntent`
> - Die Absichten in der Built-in Intent Library (Bibliothek integrierter Absichten) im Alexa Skills Kit

AMAZON.CancelIntent

Reagiert auf Wörter und Ausdrücke, die darauf hinweisen, dass der Benutzer die aktuelle Interaktion abbrechen möchte. Ihre Anwendung kann diese Absicht nutzen, um Slot-Typ-Werte und andere Attribute zu entfernen, bevor die Interaktion mit dem Benutzer beendet wird.

Allgemeine Äußerungen:

- abbrechen
- macht nichts
- vergiss es

AMAZON.FallbackIntent

Wenn die Eingabe eines Benutzers zu einer Absicht nicht den Erwartungen eines Bot entspricht, können Sie Amazon Lex so konfigurieren, dass eine Fallback-Absicht aufgerufen wird. Wenn beispielsweise die Benutzereingabe „Ich möchte Süßigkeiten bestellen“ nicht mit einer Absicht in Ihrem `OrderFlowers` Bot übereinstimmt, ruft Amazon Lex die Fallback-Absicht auf, um die Antwort zu bearbeiten.

Sie fügen eine Fallback-Absicht hinzu, indem Sie Ihrem Bot den integrierten Absichtstyp `AMAZON.FallbackIntent` hinzufügen. Sie können die Absicht mithilfe der PutBot-Operation oder durch Auswahl der Absicht aus der Liste der integrierten Absichten in der Konsole angeben.

Das Aufrufen einer Fallback-Absicht verwendet zwei Schritte. Im ersten Schritt wird die Fallback-Absicht basierend auf der Eingabe des Benutzers abgeglichen. Wenn die Fallback-Absicht übereinstimmt, hängt das Verhalten des Bots von der Anzahl der Wiederholungen ab, die für eine Eingabeaufforderung konfiguriert wurden. Wenn beispielsweise die maximale Anzahl von Versuchen, eine Absicht zu bestimmen, 2 beträgt, gibt der Bot die Klärungsaufforderung des Bots zweimal zurück, bevor er die Fallback-Absicht aufruft.

Amazon Lex entspricht in den folgenden Situationen der Fallback-Absicht:

- Die Eingabe des Benutzers für eine Absicht stimmt nicht mit der Eingabe überein, die der Bot erwartet
- Audioeingabe ist Rauschen, oder Texteingaben werden nicht als Wörter erkannt.
- Die Benutzereingabe ist mehrdeutig und Amazon Lex kann nicht bestimmen, welche Absicht aufgerufen werden soll.

Die Fallback-Absicht wird aufgerufen, wenn:

- Der Bot erkennt die Benutzereingabe nach der konfigurierten Anzahl von Versuchen zur Klärung, wenn die Konversation gestartet wird, nicht als Absicht.
- Eine Absicht erkennt die Benutzereingabe nach der konfigurierten Anzahl von Versuchen nicht als Slot-Wert.
- Eine Absicht erkennt die Benutzereingabe nicht als Antwort auf eine Bestätigungsaufforderung nach der konfigurierten Anzahl von Versuchen.

Sie können Folgendes mit einer Fallback-Absicht verwenden:

- Eine Fulfillment-Lambda-Funktion
- Eine Schlussfolgerung
- Eine Follow-up-Eingabeaufforderung

Es ist nicht möglich, Folgendes zu einer Fallback-Absicht hinzuzufügen:

- Äußerungen
- Slots
- Eine Lambda-Funktion zur Initialisierung und Validierung
- Eine Bestätigungsaufforderung

Wenn Sie sowohl eine Stornierungsanweisung als auch eine Fallback-Absicht für einen Bot konfiguriert haben, verwendet Amazon Lex die Fallback-Absicht. Wenn Sie möchten, dass Ihr Bot über eine Stornoerklärung verfügt, können Sie die Fulfillment-Funktion für die alternative Absicht verwenden, um dasselbe Verhalten wie eine Stornoerklärung bereitzustellen. Weitere Informationen finden Sie im Parameter `abortStatement` der PutBot-Operation.

Verwenden von Klärungsaufforderungen

Wenn Sie Ihrem Bot eine Klärungsaufforderung bereitstellen, wird über die Eingabeaufforderung eine gültige Absicht vom Benutzer angefordert. Die Klärungsaufforderung wird so oft, wie von Ihnen konfiguriert, wiederholt. Danach wird die Fallback-Absicht aufgerufen.

Wenn Sie bei der Erstellung eines Bots keine Klarstellungsaufforderung einrichten und der Benutzer die Konversation nicht mit einer gültigen Absicht beginnt, ruft Amazon Lex sofort Ihre Fallback-Absicht auf.

Wenn Sie eine Fallback-Absicht ohne Aufforderung zur Klärung verwenden, ruft Amazon Lex den Fallback unter den folgenden Umständen nicht auf:

- Wenn der Benutzer auf eine Follow-up-Aufforderung antwortet, aber keine Absicht bereitstellt. Zum Beispiel als Antwort auf eine Folgeaufforderung mit der Aufschrift „Möchten Sie heute noch etwas anderes?“ , sagt der Benutzer „Ja“. Amazon Lex gibt die Ausnahme 400 Bad Request zurück, da es keine Klarstellungsaufforderung gibt, die an den Benutzer gesendet werden muss, um eine Absicht zu erhalten.
- Wenn Sie eine AWS Lambda-Funktion verwenden, geben Sie einen `ElicitIntent`-Dialogtyp zurück. Da Amazon Lex keine Klarstellungsaufforderung hat, um eine Absicht des Benutzers zu erhalten, gibt es die Ausnahme 400 Bad Request zurück.
- Wenn Sie die `PutSession`-Operation verwenden, senden Sie einen `ElicitIntent`-Dialogtyp. Da Amazon Lex keine Klarstellungsaufforderung hat, um eine Absicht des Benutzers zu erhalten, gibt es die Ausnahme 400 Bad Request zurück.

Verwenden einer Lambda-Funktion mit einer Fallback-Absicht

Wenn eine Fallback-Absicht aufgerufen wird, hängt die Antwort von der Einstellung des Parameters `fulfillmentActivity` für die Operation PutIntent ab. Der Bot führt einen der folgenden Schritte aus:

- Gibt die Absicht-Informationen an die Client-Anwendung zurück.
- Ruft die Fulfillment-Lambda-Funktion auf. Sie ruft die Funktion mit den Sitzungsvariablen auf, die für die Sitzung festgelegt sind.

Weitere Informationen zum Festlegen der Antwort, wenn eine Fallback-Absicht aufgerufen wird, finden Sie im Parameter `fulfillmentActivity` der PutIntent-Operation.

Wenn Sie die Fulfillment-Lambda-Funktion in Ihrer Fallback-Absicht verwenden, können Sie diese Funktion verwenden, um eine andere Absicht aufzurufen oder um irgendeine Form der Kommunikation mit dem Benutzer durchzuführen, z. B. eine Rückrufnummer zu sammeln oder eine Sitzung mit einem Kundendienstmitarbeiter zu eröffnen.

Sie können in einer Fallback-Intent-Lambda-Funktion jede Aktion ausführen, die Sie in der Fulfillment-Funktion für jede andere Absicht ausführen können. Weitere Informationen zum Erstellen einer Erfüllungsfunktion mit AWS Lambda finden Sie unter Verwenden von Lambda-Funktionen.

Eine Fallback-Absicht kann mehrmals in derselben Sitzung aufgerufen werden. Nehmen wir beispielsweise an, dass Ihre Lambda-Funktion die `ElicitIntent` Dialogaktion verwendet, um den Benutzer zu einer anderen Absicht aufzufordern. Wenn Amazon Lex nach der konfigurierten Anzahl von Versuchen nicht auf die Absicht des Benutzers schließen kann, ruft es die Fallback-Absicht erneut auf. Außerdem wird die Fallback-Absicht aufgerufen, wenn der Benutzer nach der konfigurierten Anzahl von Versuchen nicht mit einem gültigen Slot-Wert antwortet.

Sie können eine Lambda-Funktion so konfigurieren, dass sie mithilfe einer Sitzungsvariablen nachverfolgt, wie oft die Fallback-Absicht aufgerufen wird. Ihre Lambda-Funktion kann eine andere Aktion ausführen, wenn sie öfter aufgerufen wird als der Schwellenwert, den Sie in Ihrer Lambda-Funktion festgelegt haben. Weitere Informationen zu Sitzungsvariablen finden Sie unter Festlegen von Sitzungsattributen.

AMAZON.HelpIntent

Reagiert auf Wörter oder Ausdrücke, die darauf hinweisen, dass der Benutzer bei der Interaktion mit Ihrem Bot Hilfe benötigt. Wenn diese Absicht aufgerufen wird, können Sie Ihre Lambda-Funktion oder -Anwendung so konfigurieren, dass sie Informationen über die Fähigkeiten Ihres Bots bereitstellt, Folgefragen zu Hilfebereichen stellt oder die Interaktion einem menschlichen Agenten übergibt.

Allgemeine Äußerungen:

- help
- hilf mir
- kannst du mir helfen

AMAZON.KendraSearchIntent

Verwenden Sie die Absicht, um nach Dokumenten zu suchen, die Sie mit Amazon Kendra indexiert haben. `AMAZON.KendraSearchIntent` Wenn Amazon Lex die nächste Aktion in einer Konversation mit dem Benutzer nicht bestimmen kann, löst es die Suchabsicht aus.

Das `AMAZON.KendraSearchIntent` ist nur im Gebietsschema Englisch (USA) (en-US) und in den Regionen USA Ost (Nord-Virginia), USA West (Oregon) und Europa (Irland) verfügbar.

Amazon Kendra ist ein machine-learning-based Suchdienst, der Dokumente in natürlicher Sprache wie PDF-Dokumente oder Microsoft Word-Dateien indexiert. Es kann indizierte Dokumente durchsuchen und die folgenden Arten von Antworten auf Fragen zurückgeben:

- Antworten
- Einträge aus häufig gestellten Fragen, die die Fragen möglicherweise beantworten
- Dokumente, die sich auf die Fragen beziehen

Ein Beispiel für die Verwendung von `AMAZON.KendraSearchIntent` finden Sie unter Beispiel: Einen FAQ-Bot für einen Amazon Kendra Kendra-Index erstellen.

Wenn Sie eine `AMAZON.KendraSearchIntent` Absicht für Ihren Bot konfigurieren, ruft Amazon Lex die Absicht immer dann auf, wenn es die Benutzeräußerung für einen Slot oder eine Absicht nicht ermitteln kann. Wenn Ihr Bot beispielsweise eine Antwort für einen Slot-Typ namens „Pizzabelag“ auslöst und der Benutzer sagt: „Was ist eine Pizza? „, ruft Amazon Lex `AMAZON.KendraSearchIntent` an, um die Frage zu beantworten. Wenn keine Antwort von Amazon Kendra eingeht, wird die Konversation wie im Bot konfiguriert fortgesetzt.

Wenn Sie `AMAZON.KendraSearchIntent` sowohl das als auch das `AMAZON.FallbackIntent` im selben Bot verwenden, verwendet Amazon Lex die Absichten wie folgt:

1. Amazon Lex nennt das`AMAZON.KendraSearchIntent`. Die Absicht nennt Amazon Kendra `Query` Operation.
2. Wenn Amazon Kendra eine Antwort zurückgibt, zeigt Amazon Lex dem Benutzer das Ergebnis an.
3. Wenn Amazon Kendra keine Antwort erhält, fordert Amazon Lex den Benutzer erneut auf. Die nächste Aktion hängt von der Antwort des Benutzers ab.
   - Wenn die Antwort des Benutzers eine Äußerung enthält, die Amazon Lex erkennt, z. B. das Ausfüllen eines Slot-Werts oder die Bestätigung einer Absicht, wird die Konversation mit dem Benutzer wie für den Bot konfiguriert fortgesetzt.
   - Wenn die Antwort des Benutzers keine Äußerung enthält, die Amazon Lex erkennt, ruft Amazon Lex den `Query` Vorgang erneut auf.
4. Wenn nach der konfigurierten Anzahl von Wiederholungen keine Antwort erfolgt, ruft Amazon Lex den an `AMAZON.FallbackIntent` und beendet die Konversation mit dem Benutzer.

Es gibt drei Möglichkeiten, mit dem eine Anfrage `AMAZON.KendraSearchIntent` an Amazon Kendra zu stellen:

- Lassen Sie die Suchabsicht die Anfrage für Sie stellen. Amazon Lex ruft Amazon Kendra mit der Äußerung des Benutzers als Suchzeichenfolge auf. Wenn Sie die Absicht erstellen, können Sie eine Abfragefilterzeichenfolge definieren, die die Anzahl der Antworten begrenzt, die Amazon Kendra zurückgibt. Amazon Lex verwendet den Filter in der Abfrageanforderung.
- Fügen Sie der Anfrage zusätzliche Abfrageparameter hinzu, um die Suchergebnisse mithilfe Ihrer Dialog-Lambda-Funktion einzugrenzen. Sie fügen der `delegate` Dialogaktion ein `kendraQueryFilterString` Feld hinzu, Amazon Kendra Kendra-Abfrageparameter enthält. Wenn Sie der Anfrage mit der Lambda-Funktion Abfrageparameter hinzufügen, haben diese Vorrang vor dem Abfragefilter, den Sie bei der Erstellung der Absicht definiert haben.
- Erstellen Sie eine neue Abfrage mit der Dialog-Lambda-Funktion. Sie können eine vollständige Amazon Kendra Kendra-Abfrageanforderung erstellen, die Amazon Lex sendet. Sie legen die Abfrage im Feld `kendraQueryRequestPayload` in der Dialogaktion `delegate` fest. Das Feld `kendraQueryRequestPayload` hat Vorrang vor dem Feld `kendraQueryFilterString`.

Um den `queryFilterString` Parameter anzugeben, wenn Sie einen Bot erstellen, oder um das `kendraQueryFilterString` Feld anzugeben, wenn Sie die `delegate` Aktion in einer Dialog-Lambda-Funktion aufrufen, geben Sie eine Zeichenfolge an, die als Attributfilter für die Amazon Kendra Kendra-Abfrage verwendet wird. Wenn die Zeichenfolge kein gültiger Attributfilter ist, wird zur Laufzeit die Ausnahme `InvalidBotConfigException` zurückgegeben. Weitere Informationen zu Attributfiltern finden Sie unter Verwenden von Dokumentattributen zum Filtern von Abfragen im Amazon Kendra Developer Guide.

Um die Abfrage zu kontrollieren, die Amazon Lex an Amazon Kendra sendet, können Sie in dem `kendraQueryRequestPayload` Feld Ihrer Dialog-Lambda-Funktion eine Abfrage angeben. Wenn die Abfrage nicht gültig ist, gibt Amazon Lex eine `InvalidLambdaResponseException` Ausnahme zurück. Weitere Informationen finden Sie unter Abfragevorgang im Amazon Kendra Developer Guide.

Ein Beispiel für die Verwendung von `AMAZON.KendraSearchIntent` finden Sie unter Beispiel: Einen FAQ-Bot für einen Amazon Kendra Kendra-Index erstellen.

IAM-Richtlinie für Amazon Kendra Search

Um den `AMAZON.KendraSearchIntent` Intent zu verwenden, müssen Sie eine Rolle verwenden, die AWS Identity and Access Management (IAM) -Richtlinien bereitstellt, die es Amazon Lex ermöglichen, eine Runtime-Rolle anzunehmen, die berechtigt ist, den Amazon Kendra `Query` Kendra-Intent aufzurufen. Die IAM-Einstellungen, die Sie verwenden, hängen davon ab, ob Sie die `AMAZON.KendraSearchIntent` mit der Amazon Lex Lex-Konsole oder mit einem AWS-SDK oder

dem AWS Command Line Interface (AWS CLI) erstellen. Wenn Sie die Konsole verwenden, können Sie wählen, ob Sie der mit dem Amazon Lex-Service verbundenen Rolle die Berechtigung zum Aufrufen von Amazon Kendra hinzufügen oder eine Rolle speziell für den Aufruf des Amazon `Query` Kendra Kendra-Vorgangs verwenden möchten. Wenn Sie die AWS CLI oder ein SDK verwenden, um die Absicht zu erstellen, müssen Sie eine Rolle speziell zum Aufrufen der `Query`-Operation verwenden.

Anfügen von Berechtigungen

Sie können die Konsole verwenden, um der standardmäßigen serviceverknüpften Rolle von Amazon Lex Berechtigungen für den Zugriff auf den Amazon `Query` Kendra-Vorgang zuzuweisen. Wenn Sie der serviceverknüpften Rolle Berechtigungen zuordnen, müssen Sie keine spezielle Runtime-Rolle erstellen und verwalten, um eine Verbindung mit dem Amazon Kendra Kendra-Index herzustellen.

Der Benutzer, die Rolle oder die Gruppe, die Sie für den Zugriff auf die Amazon Lex-Konsole verwenden, muss über Berechtigungen zur Verwaltung von Rollenrichtlinien verfügen. Fügen Sie der Konsolenzugriffsrolle die folgende IAM-Richtlinie hinzu. Wenn Sie diese Berechtigungen erteilen, verfügt die Rolle über Berechtigungen zum Ändern der vorhandenen Richtlinie für die serviceverknüpfte Rolle.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:GetRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "*"
        }
    ]
}
```

Angeben einer Rolle

Sie können die Konsole, die oder die API verwendenAWS CLI, um eine Laufzeitrolle anzugeben, die beim Aufrufen des Amazon Kendra `Query` Kendra-Vorgangs verwendet werden soll.

Der Benutzer, die Rolle oder die Gruppe, die Sie zur Angabe der Runtime-Rolle verwenden, muss über die `iam:PassRole` entsprechende Berechtigung verfügen. Die folgende Richtlinie definiert die Berechtigung. Sie können die Bedingungskontextschlüssel `iam:AssociatedResourceArn` und `iam:PassedToService` verwenden, um den Umfang der Berechtigungen weiter einzuschränken. Weitere Informationen finden Sie unter IAM und AWS STS Condition Context Keys im AWS Identity and Access ManagementBenutzerhandbuch.

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

Die Runtime-Rolle, die Amazon Lex zum Aufrufen von Amazon Kendra verwenden muss, muss über die `kendra:Query` entsprechenden Berechtigungen verfügen. Wenn Sie eine bestehende IAM-Rolle für die Erlaubnis zum Aufrufen des Amazon Kendra `Query` Kendra-Vorgangs verwenden, muss der Rolle die folgende Richtlinie beigefügt sein.

Sie können die IAM-Konsole, die IAM-API oder die verwenden, um eine Richtlinie AWS CLI zu erstellen und sie einer Rolle zuzuordnen. In diesen Anweisungen wird die AWS CLI zum Erstellen der Rolle und Richtlinien verwendet.

> **Note**
>
> Der folgende Code ist für Linux und MacOS formatiert. Ersetzen Sie unter Windows das Linux-Zeilenfortsetzungszeichen (\) durch ein Caret-Zeichen (^).

So fügen Sie einer Rolle die Berechtigung für die Query-Operation hinzu

1. Erstellen Sie im aktuellen Verzeichnis ein Dokument mit dem Namen **`KendraQueryPolicy.json`**, fügen Sie ihm folgenden Code hinzu und speichern Sie es.

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

2. Führen Sie in der den folgenden Befehl ausAWS CLI, um die IAM-Richtlinie für die Ausführung des Amazon Kendra `Query` Kendra-Vorgangs zu erstellen.

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

3. Hängen Sie die Richtlinie an die IAM-Rolle an, mit der Sie den Vorgang aufrufen. `Query`

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

Sie können wählen, ob Sie die mit dem Service verknüpfte Amazon Lex Lex-Rolle aktualisieren oder eine Rolle verwenden möchten, die Sie bei der Erstellung der `AMAZON.KendraSearchIntent` für Ihren Bot erstellt haben. Das folgende Verfahren zeigt, wie Sie die zu verwendende IAM-Rolle auswählen.

Um die Runtime-Rolle für AMAZON anzugeben. KendraSearchIntent

1. Melden Sie sich bei der Amazon Lex Lex-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/lex/.
2. Wählen Sie den Bot, dem Sie `AMAZON.KendraSearchIntent` hinzufügen möchten.
3. Wählen Sie das Pluszeichen (+) neben Intents (Absichten).
4. Wählen Sie unter Add intent (Absicht hinzufügen) die Option Search existing intents (Vorhandene Absichten durchsuchen).
5. Geben Sie unter Search intents (Absichten suchen) **`AMAZON.KendraSearchIntent`** ein und wählen Sie dann Add (Hinzufügen).
6. Geben Sie unter Copy built-in intent (Integrierte Absicht kopieren) einen Namen für die Absicht ein, z. B. **`KendraSearchIntent`**, und wählen Sie dann Add (Hinzufügen).
7. Öffnen Sie den Abschnitt Amazon Kendra query (Amazon Kendra-Abfrage).
8. Wählen Sie unter IAM role (IAM-Rolle) eine der folgenden Optionen:

   - Um die mit dem Service verknüpfte Amazon Lex Lex-Rolle zu aktualisieren, sodass Ihr Bot Amazon Kendra Kendra-Indizes abfragen kann, wählen Sie Amazon Kendra Kendra-Berechtigungen hinzufügen.
   - Um eine Rolle zu verwenden, die berechtigt ist, den Amazon Kendra `Query` Kendra-Vorgang aufzurufen, wählen Sie Bestehende Rolle verwenden.

Verwenden von Anforderungs- und Sitzungsattributen als Filter

Um die Antwort von Amazon Kendra auf Elemente zu filtern, die sich auf die aktuelle Konversation beziehen, verwenden Sie Sitzungs- und Anforderungsattribute als Filter, indem Sie den `queryFilterString` Parameter hinzufügen, wenn Sie Ihren Bot erstellen. Sie geben einen Platzhalter für das Attribut an, wenn Sie die Absicht erstellen, und dann ersetzt Amazon Lex V2 einen Wert, bevor Amazon Kendra aufgerufen wird. Weitere Informationen zu Anforderungsattributen finden Sie unter Festlegen von Anforderungsattributen. Weitere Informationen über Sitzungsattribute finden Sie unter Festlegen von Sitzungsattributen.

Im Folgenden finden Sie ein Beispiel für einen `queryFilterString` Parameter, der eine Zeichenfolge verwendet, um die Amazon Kendra Kendra-Abfrage zu filtern.

```
"{"equalsTo": {"key": "City", "value": {"stringValue": "Seattle"}}}"
```

Im Folgenden finden Sie ein Beispiel für einen `queryFilterString` Parameter, der ein Sitzungsattribut verwendet, das aufgerufen wird`"SourceURI"`, um die Amazon Kendra Kendra-Abfrage zu filtern.

```
"{"equalsTo": {"key": "SourceURI","value": {"stringValue": "[FileURL]"}}}"
```

Im Folgenden finden Sie ein Beispiel für einen `queryFilterString` Parameter, der ein Anforderungsattribut verwendet, das aufgerufen wird`"DepartmentName"`, um die Amazon Kendra Kendra-Abfrage zu filtern.

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

Die `AMAZON.KendraSearchInteng` Filter verwenden dasselbe Format wie die Amazon Kendra Kendra-Suchfilter. Weitere Informationen finden Sie unter Verwenden von Dokumentattributen zum Filtern von Suchergebnissen im Amazon Kendra Developer Guide.

Die mit dem verwendete Abfragefilterzeichenfolge `AMAZON.KendraSearchIntent` muss Kleinbuchstaben für den ersten Buchstaben jedes Filters verwenden. Der folgende ist beispielsweise ein gültiger Abfragefilter für. `AMAZON.KendraSearchIntent`

```
{
    "andAllFilters": [
        {
            "equalsTo": {
                "key": "City",
                "value": {
                    "stringValue": "Seattle"
                }
            }
        },
        {
            "equalsTo": {
                "key": "State",
                "value": {
                    "stringValue": "Washington"
                }
            }
        }
    ]
}
```

Verwenden der Suchantwort

Amazon Kendra gibt die Antwort auf eine Suche in der Absichtserklärung zurück. `conclusion` Die Absicht muss eine `conclusion` Aussage enthalten, es sei denn, eine Fulfillment-Lambda-Funktion erzeugt eine Abschlussnachricht.

Amazon Kendra bietet vier Arten von Antworten.

- `x-amz-lex:kendra-search-response-question_answer-question-<N>`— Die Frage aus einer häufig gestellten Frage, die der Suche entspricht.
- `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— Die Antwort aus einer häufig gestellten Frage, die der Suche entspricht.
- `x-amz-lex:kendra-search-response-document-<N>`— Ein Auszug aus einem Dokument im Index, der sich auf den Text der Äußerung bezieht.
- `x-amz-lex:kendra-search-response-document-link-<N>`— Die URL eines Dokuments im Index, das sich auf den Text der Äußerung bezieht.
- `x-amz-lex:kendra-search-response-answer-<N>`— Ein Auszug aus einem Dokument im Index, das die Frage beantwortet.

Die Antworten werden in `request`-Attributen zurückgegeben. Für jedes Attribut kann es bis zu fünf Antworten geben, nummeriert von 1 bis 5. Weitere Informationen zu Antworten finden Sie unter Antworttypen im Amazon Kendra Developer Guide.

Die Anweisung `conclusion` muss eine oder mehrere Nachrichtengruppen aufweisen. Jede Nachrichtengruppe enthält eine oder mehrere Nachrichten. Jede Nachricht kann eine oder mehrere Platzhaltervariablen enthalten, die in der Antwort von Amazon Kendra durch Anforderungsattribute ersetzt werden. In der Nachrichtengruppe muss mindestens eine Nachricht vorhanden sein, in der alle Variablen in der Nachricht durch Anforderungsattributwerte in der Laufzeitantwort ersetzt werden, oder in der Gruppe muss eine Nachricht ohne Platzhaltervariablen vorhanden sein. Die Anforderungsattribute werden durch doppelte Klammern ("((" "))") hervorgehoben. Die folgenden Nachrichtengruppennachrichten stimmen mit allen Antworten von Amazon Kendra überein:

- „Ich habe eine häufig gestellte Frage für Sie gefunden: ((x-amz-lex: kendra-search-response-question _answer-question-1)), und die Antwort lautet ((: _answer-answer-1))“ x-amz-lex kendra-search-response-question
- „Ich habe einen Auszug aus einem hilfreichen Dokument gefunden: ((: -1))“ x-amz-lex kendra-search-response-document

- „Ich denke, die Antwort auf Ihre Fragen lautet ((x-amz-lex: kendra-search-response-answer -1))“

Verwenden einer Lambda-Funktion zur Verwaltung der Anfrage und Antwort

Die `AMAZON.KendraSearchIntent` Absicht kann Ihren Dialog-Code-Hook und Ihren Fulfillment-Code-Hook verwenden, um die Anfrage an Amazon Kendra und die Antwort zu verwalten. Verwenden Sie die Dialogcode-Hook-Lambda-Funktion, wenn Sie die Anfrage ändern möchten, die Sie an Amazon Kendra senden, und die Lambda-Funktion für den Fulfillment-Code-Hook, wenn Sie die Antwort ändern möchten.

Erstellen einer Abfrage mit dem Dialogcode-Hook

Sie können den Dialog-Code-Hook verwenden, um eine Abfrage zu erstellen, die an Amazon Kendra gesendet werden soll. Die Verwendung des Dialogcode-Hooks ist optional. Wenn Sie keinen Dialog-Code-Hook angeben, erstellt Amazon Lex eine Abfrage aus der Benutzeräußerung und verwendet die`queryFilterString`, die Sie bei der Konfiguration der Absicht angegeben haben, sofern Sie eine angegeben haben.

Sie können zwei Felder in der Dialog-Code-Hook-Antwort verwenden, um die Anfrage an Amazon Kendra zu ändern:

- `kendraQueryFilterString`— Verwenden Sie diese Zeichenfolge, um Attributfilter für die Amazon Kendra Kendra-Anforderung anzugeben. Sie können die Abfrage mithilfe eines beliebigen in Ihrem Index definierten Indexfelds filtern. Informationen zur Struktur der Filterzeichenfolge finden Sie unter Verwenden von Dokumentattributen zum Filtern von Abfragen im Amazon Kendra Developer Guide. Wenn die angegebene Filterzeichenfolge ungültig ist, erhalten Sie die Ausnahme `InvalidLambdaResponseException`. Die `kendraQueryFilterString`-Zeichenfolge überschreibt alle Abfragezeichenfolgen, die im für diese Absicht konfigurierten `queryFilterString` angegeben sind.
- `kendraQueryRequestPayload`— Verwenden Sie diese Zeichenfolge, um eine Amazon Kendra Kendra-Abfrage anzugeben. Ihre Anfrage kann alle Funktionen von Amazon Kendra verwenden. Wenn Sie keine gültige Abfrage angeben, erhalten Sie die Ausnahme `InvalidLambdaResponseException`. Weitere Informationen finden Sie unter Query im Amazon Kendra Developer Guide.

Nachdem Sie den Filter oder die Abfragezeichenfolge erstellt haben, senden Sie die Antwort an Amazon Lex, wobei das `dialogAction` Feld der Antwort auf gesetzt ist`delegate`. Amazon Lex

sendet die Anfrage an Amazon Kendra und sendet dann die Abfrageantwort an den Fulfillment-Code-Hook zurück.

Verwenden des Erfüllungscode-Hooks für die Antwort

Nachdem Amazon Lex eine Anfrage an Amazon Kendra gesendet hat, wird die Abfrageantwort an die `AMAZON.KendraSearchIntent` Fulfillment-Lambda-Funktion zurückgegeben. Das Eingabeereignis für den Code-Hook enthält die vollständige Antwort von Amazon Kendra. Die Abfragedaten haben dieselbe Struktur wie die, die von der Amazon Kendra `Query` Kendra-Operation zurückgegeben wurde. Weitere Informationen finden Sie unter Syntax der Abfrageantwort im Amazon Kendra Developer Guide.

Der Erfüllungscode-Hook ist optional. Wenn keine vorhanden ist oder wenn der Code-Hook keine Nachricht in der Antwort zurückgibt, verwendet Amazon Lex die `conclusion` Anweisung für Antworten.

Beispiel: Einen FAQ-Bot für einen Amazon Kendra Kendra-Index erstellen

In diesem Beispiel wird ein Amazon Lex Lex-Bot erstellt, der einen Amazon Kendra Kendra-Index verwendet, um Antworten auf Benutzerfragen zu geben. Der Bot zu häufig gestellten Fragen verwaltet den Dialog für den Benutzer. Er verwendet die Absicht `AMAZON.KendraSearchIntent`, um den Index abzufragen und die Antwort für den Benutzer bereitzustellen. Bei der Erstellung des Bots gehen Sie folgendermaßen vor:

1. Erstellen Sie einen Bot, mit dem Ihre Kunden interagieren werden, um Antworten von diesem Bot zu erhalten.
2. Erstellen Sie eine benutzerdefinierte Absicht. Ihr Bot muss mindestens eine Absicht mit mindestens einer Äußerung aufweisen. Diese Absicht ermöglicht die Entwicklung des Bots, wird anderweitig jedoch nicht verwendet.
3. Fügen Sie die `KendraSearchIntent` Absicht zu Ihrem Bot hinzu und konfigurieren Sie ihn so, dass er mit Ihrem Amazon Kendra Kendra-Index funktioniert.
4. Testen Sie den Bot, indem Sie Fragen stellen, die durch Dokumente beantwortet werden, die in Ihrem Amazon Kendra Kendra-Index gespeichert sind.

Bevor Sie dieses Beispiel verwenden können, müssen Sie einen Amazon Kendra Kendra-Index erstellen. Weitere Informationen finden Sie unter Erste Schritte mit einem S3-Bucket (Konsole) im Amazon Kendra Developer Guide.

So erstellen Sie einen Bot für häufig gestellte Fragen

1. Melden Sie sich bei der Amazon Lex Lex-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/lex/.
2. Wählen Sie im Navigationsbereich Bots.
3. Wählen Sie Erstellen aus.
4. Wählen Sie Custom bot (Benutzerdefinierter Bot) aus. Konfigurieren Sie den Bot wie folgt:
   - Bot-Name — Geben Sie dem Bot einen Namen, der seinen Zweck angibt, z. **`KendraTestBot`** B.
   - Stimme ausgeben — Wählen Sie „Keine“.
   - Sitzungs-Timeout — Geben Sie ein**`5`**.
   - Stimmungsanalyse — Wählen Sie Nein.
   - COPPA — Wählen Sie Nein.
   - Speicherung von Benutzeräußerungen — Wählen Sie Nicht speichern aus.
5. Wählen Sie Erstellen aus.

Um einen Bot erfolgreich erstellen zu können, müssen Sie mindestens eine Absicht mit mindestens einer Beispieläußerung erstellen. Diese Absicht ist erforderlich, um Ihren Amazon Lex Lex-Bot zu erstellen, wird jedoch nicht für die Antwort auf häufig gestellte Fragen verwendet. Die Äußerung für die Absicht darf nicht auf Fragen zutreffen, die Ihr Kunde stellt.

So erstellen Sie die erforderliche Absicht

1. Wählen Sie auf der Seite Getting started with your bot (Erste Schritte mit Ihrem Bot) die Option Create intent (Absicht erstellen).
2. Wählen Sie unter Add intent (Absicht hinzufügen) die Option Create intent (Absicht erstellen).
3. Geben Sie im Dialogfeld Create intent (Absicht erstellen) einen Namen für die Absicht ein, z. B. **`RequiredIntent`**.
4. Geben Sie unter Sample utterances (Beispieläußerungen) eine Äußerung ein, z. B. **`Required utterance`**.
5. Wählen Sie Save intent (Absicht speichern).

Erstellen Sie nun die Absicht, einen Amazon Kendra Kendra-Index und die Antwortnachrichten, die er zurückgeben soll, zu durchsuchen.

Um ein AMAZON zu erstellen. KendraSearchIntent Absicht und Antwortnachricht

1. Wählen Sie im Navigationsbereich das Pluszeichen (+) neben Intents (Absichten).
2. Wählen Sie unter Add intent (Absicht hinzufügen) die Option Search existing intents (Vorhandene Absichten durchsuchen).
3. Geben Sie in das Feld Suchabsichten den Text ein **`AMAZON.KendraSearchIntent`** und wählen Sie ihn dann aus der Liste aus.
4. Geben Sie der Absicht unter Copy built-in intent (Integrierte Absicht kopieren) einen Namen, z. B. **`KendraSearchIntent`**, und wählen Sie dann Add (Hinzufügen).
5. Wählen Sie im Absichtseditor Amazon Kendra query (Amazon Kendra-Abfrage), um die Abfrageoptionen zu öffnen.
6. Wählen Sie im Menü Amazon Kendra index (Amazon Kendra-Index) den Index, den die Absicht durchsuchen soll.
7. Fügen Sie im Abschnitt Response (Antwort) die folgenden drei Meldungen hinzu:

   ```
   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. Wählen Sie Save intent (Absicht speichern) und anschließend Build (Erstellen), um den Bot zu erstellen.

Verwenden Sie schließlich das Konsolentestfenster, um Antworten von Ihrem Bot zu testen. Ihre Fragen sollten sich in der Domäne befinden, die Ihr Index unterstützt.

So testen Sie Ihren Bot für häufig gestellte Fragen

1. Geben Sie im Konsolentestfenster eine Frage für Ihren Index ein.
2. Überprüfen Sie die Antwort im Antwortbereich des Testfensters.

3. Wählen Sie Clear chat history (Chatverlauf löschen), um das Testfenster für eine andere Frage zurückzusetzen.

AMAZON.PauseIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, eine Interaktion mit einem Bot zu unterbrechen, um später darauf zurückzukommen. Ihre Lambda-Funktion oder -Anwendung muss Absichtsdaten in Sitzungsvariablen speichern, oder Sie müssen die GetSession Operation verwenden, um Absichtsdaten abzurufen, wenn Sie die aktuelle Absicht fortsetzen.

Allgemeine Äußerungen:

- Pause
- pausiere das

AMAZON.RepeatIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, die vorherige Nachricht zu wiederholen. Ihre Anwendung muss eine Lambda-Funktion verwenden, um die vorherigen Absichtsinformationen in Sitzungsvariablen zu speichern, oder Sie müssen die GetSession Operation verwenden, um die vorherigen Absichtsinformationen abzurufen.

Allgemeine Äußerungen:

- wiederholen
- sag das nochmal
- wiederhole das

AMAZON.ResumeIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, eine zuvor unterbrochene Absicht wieder aufzunehmen. Ihre Lambda-Funktion oder -Anwendung muss die Informationen verwalten, die erforderlich sind, um die vorherige Absicht wieder aufzunehmen.

Allgemeine Äußerungen:

- fortsetzen

- fortsetzen
- mach weiter

AMAZON.StartOverIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, die Verarbeitung der aktuellen Absicht zu beenden und von vorne zu beginnen. Sie können Ihre Lambda-Funktion oder die `PutSession` Operation verwenden, um den ersten Slot-Wert erneut abzurufen.

Allgemeine Äußerungen:

- fange von vorne an
- Neustart
- wieder anfangen

AMAZON.StopIntent

Reagiert auf Wörter und Ausdrücke, die darauf hinweisen, dass der Benutzer die Verarbeitung der aktuellen Absicht beenden und die Interaktion mit einem Bot beenden möchte. Ihre Lambda-Funktion oder -Anwendung sollte alle vorhandenen Attribute und Slot-Typwerte löschen und dann die Interaktion beenden.

Allgemeine Äußerungen:

- stop
- aus
- Halt die Klappe

Integrierte Slot-Typen

Amazon Lex unterstützt integrierte Steckplatztypen, die definieren, wie Daten im Steckplatz erkannt und verarbeitet werden. Sie können Slots dieser Art in Ihren Absichten erstellen. Es müssen dann keine Enumerationswerte für häufig verwendete Slot-Daten wie Datum, Uhrzeit und Ort mehr erstellt werden. Integrierte Slot-Typen haben keine Versionen.

Slot-Typ	Kurzbeschreibung	Unterstützte Gebietsschemata
Amazon.airport	Erkennt Wörter, die für einen Flughafen stehen.	Alle Gebietsschemas
AMAZON. AlphaNumeric	Erkennt aus Buchstaben und Zahlen bestehende Wörter.	Alle Gebietsschemas außer Koreanisch (ko-KR)
Amazon.city	Erkennt Wörter, die für eine Stadt stehen.	Alle Gebietsschemas
Amazon.Land	Erkennt Wörter, die für ein Land stehen.	Alle Gebietsschemas
AMAZON.DATE	Erkennt Wörter, die für ein Datum stehen, und konvertiert sie in ein Standardformat.	Alle Gebietsschemas
AMAZON.DURATION	Erkennt Wörter, die für eine Dauer stehen, und konvertiert sie in ein Standardformat.	Alle Gebietsschemas
AMAZON. EmailAddress	Erkennt Wörter, die für eine E-Mail-Adresse stehen, und wandelt sie in eine Standard-E-Mail-Adresse um.	Alle Gebietsschemas
AMAZON. FirstName	Erkennt Wörter, die für einen Vornamen stehen.	Alle Gebietsschemas

Slot-Typ	Kurzbeschreibung	Unterstützte Gebietsschemata	
AMAZONAS.LastName	Erkennt Wörter, die für einen Nachnamen stehen.	Alle Gebietsschemas	
AMAZON.NUMBER	Erkennt numerische Wörter und wandelt sie in Ziffern um.	Alle Gebietsschemas	
AMAZON.Percentage	Erkennt Wörter, die eine Prozentzahl darstellen, und wandelt sie in eine Zahl und ein Prozentzeichen (%) um.	Alle Gebietsschemas	
AMAZON.PhoneNumber	Erkennt Wörter, die für eine Telefonnummer stehen, und wandelt sie in eine numerische Zeichenfolge um.	Alle Gebietsschemas	
AMAZONAS.SpeedUnit	Erkennt Wörter, die eine Geschwindigkeitseinheit darstellen, und wandelt sie in eine Standardabkürzung um.	Englisch (amerikanisch) (en-US)	
Amazon.state	Erkennt Wörter, die für einen Staat stehen.	Alle Gebietsschemas	

Slot-Typ	Kurzbeschreibung	Unterstützte Gebietsschemata	
AMAZONAS. StreetName	Erkennt Wörter, die für einen Straßenna men stehen.	Alle Sprachen außer Englisch (US) (en-US)	
AMAZON.TIME	Erkennt Wörter, die auf Zeiten hinweisen, und konvertiert sie in ein Zeitformat.	Alle Gebietsschemas	
AMAZON. WeightUnit	Erkennt Wörter, die eine Gewichtse inheit darstellen, und wandelt sie in eine Standardabkürzung um	Englisch (amerikan isch) (en-US)	

> **Note**
>
> Für das Gebietsschema Englisch (USA) (en-US) unterstützt Amazon Lex Steckplatztypen aus dem Alexa Skill Kit. Eine Liste der verfügbaren integrierten Slot-Typen finden Sie unter Slot Type Reference in der Dokumentation zum Alexa Skills Kit.
>
> - Amazon Lex unterstützt die `AMAZON.LITERAL` oder die `AMAZON.SearchQuery` integrierten Steckplatztypen nicht.

Amazon.airport

Bietet eine Liste von Flughäfen. Beispiele sind unter anderem:

- Internationaler Flughafen John F. Kennedy
- Flughafen Melbourne

AMAZON. AlphaNumeric

Erkennt aus Buchstaben und Zahlen bestehende Zeichenfolgen, z. B. **APQ123**.

Dieser Slot-Typ ist in der koreanischen Sprache (ko-KR) nicht verfügbar.

Sie können den Slot-Typ `AMAZON.AlphaNumeric` für Zeichenfolgen verwenden, die Folgendes enthalten:

- Alphabetische Zeichen, z. B. **ABC**
- Numerische Zeichen, z. B. **123**
- Eine Kombination aus alphanumerischen Zeichen, z. B. **ABC123**

Sie können dem Slot-Typ `AMAZON.AlphaNumeric` einen regulären Ausdruck hinzufügen, um die für den Slot eingegebenen Werte zu validieren. Sie können beispielsweise einen regulären Ausdruck verwenden, um Folgendes zu validieren:

- Postleitzahlen im Vereinigten Königreich oder Kanada
- Führerscheinnummern
- Fahrgestellnummern

Verwenden Sie einen regulären Standardausdruck. Amazon Lex unterstützt die folgenden Zeichen im regulären Ausdruck:

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

Amazon Lex unterstützt auch Unicode-Zeichen in regulären Ausdrücken. Die Form lautet `\u`*`Unicode`*. Verwenden Sie vier Ziffern, um Unicode-Zeichen darzustellen. Beispiel: `[\u0041-\u005A]` ist gleichbedeutend mit [A–Z].

Die folgenden Operatoren für reguläre Ausdrücke werden nicht unterstützt:

- Endlose Wiederholer: *, +, oder {x,} ohne Obergrenze.
- Platzhalter (.)

Die maximale Länge des regulären Ausdrucks beträgt 300 Zeichen. Die maximale Länge einer Zeichenfolge, die in einem AMAZON-Verzeichnis gespeichert ist. AlphaNumeric Der Slot-Typ, der einen regulären Ausdruck verwendet, ist 30 Zeichen lang.

Im Folgenden finden Sie einige Beispiele für reguläre Ausdrücke.

- Alphanumerische Zeichenfolgen, z. B. **APQ123** oder **APQ1**: `[A-Z]{3}[0-9]{1,3}` oder eine stärker eingeschränkte `[A-DP-T]{3} [1-5]{1,3}`
- US Postal Service Priority Mail im internationalen Format, z. B: **CP123456789US**: `CP[0-9]{9}US`
- Bankleitzahlen, z. B. **123456789**: `[0-9]{9}`

Um den regulären Ausdruck für einen Slot-Typ festzulegen, verwenden Sie die Konsole oder die Operation PutSlotType. Der reguläre Ausdruck wird beim Speichern des Slot-Typs validiert. Wenn der Ausdruck nicht gültig ist, gibt Amazon Lex eine Fehlermeldung zurück.

Wenn Sie einen regulären Ausdruck in einem Slot-Typ verwenden, vergleicht Amazon Lex die Eingabe in Slots dieses Typs mit dem regulären Ausdruck. Wenn die Eingabe mit dem Ausdruck übereinstimmt, wird der Wert für den Slot akzeptiert. Wenn die Eingabe nicht übereinstimmt, fordert Amazon Lex den Benutzer auf, die Eingabe zu wiederholen.

Amazon.city

Bietet eine Liste lokaler und weltweiter Städte. Der Slot-Typ erkennt gängige Varianten von Städtenamen. Amazon Lex konvertiert nicht von einer Variante zu einem offiziellen Namen.

Beispiele:

- New York
- Reykjavik
- Tokio
- Versailles

Amazon.Land

Die Namen von Ländern auf der ganzen Welt. Beispiele:

- Australien
- Deutschland

- Japan
- Vereinigte Staaten
- Uruguay

AMAZON.DATE

Konvertiert Wörter, die für Datumsangaben stehen, in ein Datumsformat.

Das Datum wird nach Ihren Wünschen im ISO-8601-Datumsformat bereitgestellt. Das Datum, an dem Ihre Absicht im Slot eingeht, kann je nach dem vom Benutzer geäußerten Satz variieren.

- Äußerungen, die einem bestimmten Datum zugeordnet sind, wie „heute“, „jetzt“ oder „fünfundzwanzigster November“, werden in ein vollständiges Datum umgewandelt:. `2020-11-25` Standardmäßig werden Datumsangaben verwendet, die am oder nach dem aktuellen Datum liegen.
- Äußerungen, die einer bestimmten Woche zugeordnet sind, z. B. „diese Woche“ oder „nächste Woche“, werden in das Datum des ersten Wochentags umgewandelt. Im ISO-8601-Format beginnt die Woche am Montag und endet am Sonntag. Wenn heute beispielsweise der 25.11.2020 ist, wird „nächste Woche“ in umgewandelt. `2020-11-30`
- Äußerungen, die einem Monat, aber keinem bestimmten Tag zugeordnet sind, wie z. B. „nächster Monat“, werden in den letzten Tag des Monats umgewandelt. Wenn heute beispielsweise der 25.11.2020 ist, wird „nächster Monat“ in umgewandelt. `2020-12-31`
- Äußerungen, die einem Jahr, aber keinem bestimmten Monat oder Tag zugeordnet sind, wie z. B. „nächstes Jahr“, werden in den letzten Tag des folgenden Jahres umgewandelt. Wenn heute beispielsweise der 25.11.2020 ist, wird „nächstes Jahr“ in umgewandelt. `2021-12-31`

AMAZON.DURATION

Konvertiert Wörter, die eine Dauer angeben, in eine numerische Dauer.

Die Dauer wird in ein Format aufgelöst, das auf dem ISO-8601-Dauerformat basiert. `PnYnMnWnDTnHnMnS` Das `P` gibt an, dass es sich um eine Dauer handelt, `n` es handelt sich um einen numerischen Wert und der darauf folgende Großbuchstabe `n` ist das spezifische Datums- oder Zeitelement. `P3D`Bedeutet zum Beispiel 3 Tage. A `T` wird verwendet, um anzuzeigen, dass die verbleibenden Werte eher Zeitelemente als Datumselemente darstellen.

Beispiele:

- „zehn Minuten“: `PT10M`
- „fünf Stunden“: `PT5H`
- „drei Tage“: `P3D`
- „fünfundvierzig Sekunden“: `PT45S`
- „acht Wochen“: `P8W`
- „sieben Jahre“: `P7Y`
- „fünf Stunden zehn Minuten“: `PT5H10M`
- „zwei Jahre drei Stunden zehn Minuten“: `P2YT3H10M`

AMAZON. EmailAddress

Erkennt Wörter, die eine E-Mail-Adresse in der Form „Benutzername@Domäne“ darstellen. Adressen können die folgenden Sonderzeichen im Benutzernamen enthalten: Unterstrich (_), Bindestrich (-), Punkt (.) und Pluszeichen (+).

AMAZON. FirstName

Häufig verwendete Vornamen. Dieser Slot-Typ erkennt sowohl formelle Namen als auch informelle Spitznamen. Der Name, der zu Ihrer Absicht gesendet wurde, ist der vom Benutzer gesendete Wert. Amazon Lex konvertiert nicht vom Spitznamen zum offiziellen Namen.

Für Vornamen, die ähnlich klingen, aber unterschiedlich geschrieben sind, sendet Amazon Lex Ihrer Absicht ein einheitliches Formular.

Verwenden Sie im Gebietsschema Englisch (US) (en-US) den Slot-Namen Amazon.US_First_Name.

Beispiele:

- Emily
- John
- Sophie

AMAZONAS. LastName

Häufig verwendete Nachnamen. Für Namen, die ähnlich klingen, aber unterschiedlich geschrieben sind, sendet Amazon Lex Ihrer Absicht ein einheitliches Formular.

Verwenden Sie im Gebietsschema Englisch (US) (en-US) den Slot-Namen Amazon.US_LAST_NAME.

Beispiele:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON.NUMBER

Konvertiert Wörter oder Zahlen, die eine Zahl ausdrücken, in Ziffern, einschließlich Dezimalzahlen. Die folgende Tabelle zeigt, wie der Slot-Typ `AMAZON.NUMBER` numerische Wörter erfasst.

Eingabe	Antwort
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

AMAZON.Percentage

Wandelt Wörter und Symbole, die einen Prozentwert repräsentieren, in einen numerischen Wert mit einem Prozentzeichen (%) um.

Wenn der Benutzer eine Zahl ohne Prozentzeichen oder das Wort "percent" eingibt, wird dem Slot-Wert diese Zahl zugewiesen. Die folgende Tabelle zeigt, wie der Slot-Typ `AMAZON.Percentage` Prozentwerte erfasst.

Eingabe	Antwort
50 percent	50%
0,4 Prozent	0.4%
23.5%	23.5%
fünfundzwanzig Prozent	25 %

AMAZON. PhoneNumber

Wandelt die Zahlen oder Wörter, die eine Telefonnummer repräsentieren, in ein Zeichenfolgenformat ohne Satzzeichen um, wie nachfolgend dargestellt.

Typ	Beschreibung	Eingabe	Ergebnis
Auslandsrufnummer mit vorangestelltem Pluszeichen (+)	11-stellige Rufnummer mit vorangestelltem Pluszeichen.	+61 7 4445 1061 +1 (509) 555-1212	`+61744431061` `+15095551212`
Auslandsrufnummer ohne vorangestelltes Pluszeichen (+)	11-stellige Rufnummer ohne vorangestelltes Pluszeichen	1 (509) 555-1212 61 7 4445 1061	`15095551212` `61744451061`
Inlandsrufnummer	10-stellige Zahl ohne Ländervorwahl	(03) 5115 4444 (509) 555-1212	`0351154444` `5095551212`
Ortsrufnummer	7-stellige Rufnummer ohne Ländervorwahl und Ortsnetzkennzahl	555-1212	`5551212`

AMAZONAS. SpeedUnit

Wandelt Wörter, die eine Geschwindigkeitseinheit repräsentieren, in die entsprechende Abkürzung um. Beispiel: "Meilen pro Stunde" wird umgewandelt in `mph`.

Dieser Slot-Typ ist nur im Gebietsschema Englisch (US) (en-US) verfügbar.

Die folgenden Beispiele zeigen, wie der Slot-Typ `AMAZON.SpeedUnit` Geschwindigkeitseinheiten erfasst.

Geschwindigkeitseinheit	Abkürzung
miles per hour, mph, MPH, m/h	mph
Kilometer pro Stunde, km pro Stunde, kmph, KMPH, km/h	kmph
Meter pro Sekunde, mps, MPS, m/s	mps
Seemeilen pro Stunde, Knoten, Knoten	knot

Amazon.state

Die Namen der geografischen und politischen Regionen innerhalb der Länder.

Beispiele:

- Bayern
- Präfektur Fukushima
- Pazifischer Nordwesten
- Queensland
- Wales

AMAZONAS. StreetName

Die Straßennamen innerhalb einer typischen Straßenadresse. Dazu gehört nur der Straßenname, nicht die Hausnummer.

Dieser Slot-Typ ist in der Ländereinstellung Englisch (US) (en-US) nicht verfügbar.

Beispiele:

- Canberra Avenue
- Vordere Straße
- Marktstraße

AMAZON.TIME

Wandelt Wörter, die Zeiten repräsentieren, in Zeitwerte um. Beinhaltet Resolutionen für unklare Zeiten. Wenn ein Benutzer eine mehrdeutige Uhrzeit eingibt, verwendet Amazon Lex das `slotDetails` Attribut eines Lambda-Ereignisses, um Auflösungen für die mehrdeutigen Zeiten an Ihre Lambda-Funktion weiterzuleiten. Beispiel: Wenn der Bot den Benutzer zur Angabe einer Lieferzeit auffordert, kann der Benutzer mit "10 o'clock" antworten. Diese Zeitangabe ist zweideutig. Sie kann 10:00 Uhr (10:00 AM) oder 22:00 Uhr (10:00 PM) bedeuten. In diesem Fall ist der Wert in der `slots` Map`null`, und die `slotDetails` Entität enthält die beiden möglichen Auflösungen der Zeit. Amazon Lex gibt Folgendes in die Lambda-Funktion ein:

```
"slots": {
   "deliveryTime": null
},
"slotDetails": {
   "deliveryTime": {
      "resolutions": [
         {
            "value": "10:00"
         },
         {
            "value": "22:00"
         }
      ]
   }
}
```

Wenn der Benutzer mit einer eindeutigen Uhrzeit antwortet, sendet Amazon Lex die Uhrzeit an Ihre Lambda-Funktion im `slots` Attribut des Lambda-Ereignisses und das `slotDetails` Attribut ist leer. Wenn Ihr Benutzer beispielsweise auf die Aufforderung zur Angabe einer Lieferzeit mit „22:00 Uhr“ antwortet, gibt Amazon Lex Folgendes in die Lambda-Funktion ein:

```
"slots": {
   "deliveryTime": "22:00"
}
```

Weitere Informationen zu den Daten, die von Amazon Lex an eine Lambda-Funktion gesendet werden, finden Sie unterEingabe-Ereignis-Format.

AMAZON. WeightUnit

Wandelt Wörter, die eine Gewichtseinheit repräsentieren, in die entsprechende Abkürzung um. Beispiel: "kilogram" wird umgewandelt in `kg`.

Dieser Slot-Typ ist nur im Gebietsschema Englisch (US) (en-US) verfügbar.

Die folgenden Beispiele zeigen, wie der Slot-Typ `AMAZON.WeightUnit` Gewichtseinheiten erfasst:

Gewichtseinheit	Abkürzung
Kilogramm, Kilos, kgs, KGS	kg
Gramm, gms, gm, GMS, g	g
Milligramm, mg, mgs	mg
Pfund, lbs, LBS	lbs
Unzen, oz, OZ	oz
Tonne, ton, t	t
Kilotonne, kt	kt

Benutzerdefinierte Slot-Typen

Für jede Absicht können Sie Parameter mit den Daten angeben, die von der Absicht benötigt werden, um die Benutzeranforderung zu erfüllen. Diese Parameter (Slots) sind von einem bestimmten Typ. Ein Slot-Typ ist eine Liste von Werten, die Amazon Lex verwendet, um das Machine-Learning-Modell so zu trainieren, dass es Werte für einen Slot erkennt. Sie können beispielsweise einen Slot namens "`Genres.`" definieren. Jeder Wert im Slot-Typ ist der Name eines Genres: "comedy",

"adventure", "documentary" usw. Sie können ein Synonym für einen Slot-Typ-Wert definieren. Sie können beispielsweise die Synonyme "funny" und "humorous" für den Wert "comedy" definieren.

Sie können den Slot-Typ so konfigurieren, dass die Auflösung auf die Slot-Werte beschränkt wird. Die Slot-Werte werden als Enumeration verwendet und der vom Benutzer eingegebene Wert wird nur in den Slot-Wert aufgelöst, wenn es sich um einen der Slot-Werte oder ein Synonym handelt. Ein Synonym wird in den entsprechenden Slot-Wert aufgelöst. Wenn der Benutzer beispielsweise "funny" eingibt, wird dies als Slot-Wert "comedy" aufgelöst.

Alternativ können Sie den Slot-Typ so konfigurieren, dass die Werte erweitert werden. Slot-Werte werden als Schulungsdaten verwendet und der Slot wird in den vom Benutzer bereitgestellten Wert aufgelöst, wenn dieser den Slot-Werten und Synonymen entspricht. Dies ist das Standardverhalten.

Amazon Lex führt eine Liste möglicher Auflösungen für einen Slot. Jeder Eintrag in der Liste enthält einen Auflösungswert, den Amazon Lex als zusätzliche Möglichkeiten für den Slot erkannt hat. Ein Auflösungswert stellt die beste Methode für die Zuordnung des Slot-Werts dar. Die Liste enthält bis zu fünf Werte.

Wenn der vom Benutzer eingegebene Wert ein Synonym ist, ist der erste Eintrag in der Liste der Auflösungswerte der Slot-Typ-Wert. Wenn der Benutzer beispielsweise "funny" eingibt, enthält das Feld `slots` "funny" und der erste Eintrag im Feld `slotDetails` lautet "comedy". Sie können `valueSelectionStrategy` beim Erstellen oder Aktualisieren eines Slot-Typs mit der Operation PutSlotType konfigurieren, damit der Slot-Wert mit dem ersten Wert in der Auflösungsliste gefüllt wird.

Wenn Sie eine Lambda-Funktion verwenden, enthält das Eingabeereignis für die Funktion eine Auflösungsliste namens`slotDetails`. Das folgende Beispiel zeigt den Abschnitt mit den Steckplätzen und den Steckplatzdetails der Eingabe für eine Lambda-Funktion:

```
"slots": {
   "MovieGenre": "funny";
},
"slotDetails": {
   "Movie": {
      "resolutions": [
         "value": "comedy"
      ]
   }
}
```

Für jeden Slot-Typ können maximal 10 000 Werte und Synonyme definiert werden. Jeder Bot kann maximal 50 000 Slot-Typenwerte und Synonyme aufweisen. Sie können beispielsweise über 5 Slot-Typen mit jeweils 5 000 Werten und 5 000 Synonymen oder über 10 Slot-Typen mit jeweils 2 500 Werten und 2 500 Synonymen verfügen. Wenn Sie diese Limits überschreiten, erhalten Sie beim Aufruf der Operation PutBot die Ausnahme `LimitExceededException`.

Slot-Verschleierung

Mit Amazon Lex können Sie den Inhalt von Slots verschleiern oder ausblenden, sodass der Inhalt nicht sichtbar ist. Um vertrauliche Daten zu schützen, die als Slot-Werte erfasst wurden, können Sie die Slot-Verschleierung aktivieren, um diese Werte in Gesprächsprotokollen zu maskieren.

Wenn Sie die Slot-Werte verschleiern möchten, wird Amazon Lex den Wert des Steckplatzes durch den Namen des Steckplatzes in Gesprächsprotokollen ersetzt. Bei einem aufgerufenen `full_name`, Slot würde der Wert des Slots wie folgt verschleiert:

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

Wenn eine Äußerung Klammerzeichen ({}) enthält, wird Amazon Lex die Klammerzeichen mit zwei umgekehrten Schrägstrichen () maskiert. Beispielsweise `{John Stiles}` wird der Text wie folgt verschleiert:

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

Slot-Werte werden in Konversationsprotokollen verschleiert. Die Slot-Werte sind weiterhin in der Antwort von der`PostContent`und`PostText`Operationen und die Slot-Werte stehen für Ihre Validierungs- und Fulfillment-Lambda-Funktionen zur Verfügung. Wenn Sie Slot-Werte in Ihren Eingabeaufforderungen oder Antworten verwenden, werden diese Slot-Werte nicht in Gesprächsprotokollen verschleiert.

In der ersten Runde einer Konversation verschleiert Amazon Lex Slot-Werte, wenn es einen Steckplatz und einen Slot-Wert in der Äußerung erkennt. Wenn kein Slot-Wert erkannt wird, verschleiert Amazon Lex die Äußerung nicht.

Im zweiten und späteren Wendungen kennt Amazon Lex den Schlitz zu entlocken und ob der Slot-Wert verschleiert werden sollte. Wenn Amazon Lex den Slot-Wert erkannt wird, wird der Wert verschleiert. Wenn Amazon Lex keinen Wert erkannt wird, wird die gesamte Äußerung verschleiert. Alle Slot-Werte in verpassten Äußerungen werden nicht verschleiert.

Amazon Lex verschleiert Slot-Werte, die Sie in Anforderungs- oder Sitzungsattributen speichern. Wenn Sie Slot-Werte speichern, die als Attribut verschleiert werden sollen, müssen Sie den Wert verschlüsseln oder anderweitig verschleiern.

Amazon Lex verschleiert den Slot-Wert im Audio nicht. Es verschleiert den Slot-Wert in der Audiotranskription.

Sie müssen nicht alle Slots in einem Bot verschleiern. Sie können wählen, welche Slots verschleiern, indem Sie die Konsole oder die Amazon Lex API verwenden. Wählen Sie in der Konsole in den Einstellungen für einen Steckplatz die Option Slot-Verschleierung aus. Wenn Sie die API verwenden, setzen Sie das `obfuscationSetting` Feld des Steckplatzes auf, `DEFAULT_OBFUSCATION` wenn Sie den PutIntent Vorgang aufrufen.

Stimmungsanalyse

Sie können die Stimmungsanalyse verwenden, um die Stimmungen einer Benutzeräußerung zu ermitteln. Mit den Stimmungsinformationen können Sie den Konversationsfluss verwalten oder eine Analyse nach dem Anruf durchführen. Wenn die Benutzerstimmung beispielsweise negativ ist, können Sie einen Workflow erstellen, um eine Konversation an einen menschlichen Agenten zu übergeben.

Amazon Lex lässt sich in Amazon Comprehend integrieren, um die Stimmung der Nutzer zu erkennen. Die Antwort von Amazon Comprehend gibt an, ob die allgemeine Stimmung des Textes positiv, neutral, negativ oder gemischt ist. Die Antwort enthält die wahrscheinlichste Stimmung für die Äußerung des Benutzers und die Punktzahl für jede der Stimmungskategorien. Die Punktzahl stellt die Wahrscheinlichkeit dar, dass die Stimmung korrekt erkannt wurde.

Sie aktivieren die Stimmungsanalyse für einen Bot mithilfe der Konsole oder mithilfe der Amazon Lex Lex-API. Wählen Sie in der Amazon Lex-Konsole den Tab Einstellungen für Ihren Bot und setzen Sie dann die Option Stimmungsanalyse auf Ja. Wenn Sie die API verwenden, rufen Sie den PutBot-Vorgang mit auf `true` eingestelltem `detectSentiment`-Feld auf.

Wenn die Stimmungsanalyse aktiviert ist, gibt die Antwort aus den Vorgängen PostContent und PostText ein Feld mit der Bezeichnung `sentimentResponse` in der Bot-Antwort mit anderen

Metadaten zurück. Das Feld `sentimentResponse` verfügt über die zwei Felder `SentimentLabel` und `SentimentScore`, die das Ergebnis der Stimmungsanalyse enthalten. Wenn Sie eine Lambda-Funktion verwenden, ist das`sentimentResponse` Feld in den an Ihre Funktion gesendeten Ereignisdaten enthalten.

Der folgende Code ist ein Beispiel für das Feld `sentimentResponse`, das als Teil der Antwort `PostContent` `PostText` oder zurückgegeben wird. Das Feld `SentimentScore` beinhaltet eine Zeichenfolge, die die Ergebnisse für die Antwort enthält.

```
{
    "SentimentScore":
        "{
        Mixed: 0.030585512690246105,
        Positive: 0.94992071056365967,
        Neutral: 0.0141543131828308,
        Negative: 0.00893945890665054
        }",
    "SentimentLabel": "POSITIVE"
}
```

Amazon Lex ruft Amazon Comprehend in Ihrem Namen an, um die Stimmung in jeder vom Bot verarbeiteten Äußerung zu ermitteln. Durch die Aktivierung der Stimmungsanalyse stimmen Sie den Servicebedingungen und -vereinbarungen für Amazon Comprehend zu. Weitere Informationen zu Preisen für Amazon Comprehend finden Sie unter Amazon Comprehend Comprehend-Preise.

Weitere Informationen zur Funktionsweise der Stimmungsanalyse von Amazon Comprehend finden Sie im Amazon Comprehend Developer Guide im Amazon Comprehend Developer Guide unter Determinate the Sentiment.

Amazon Lex Lex-Ressourcen kennzeichnen

Um Ihnen bei der Verwaltung Ihrer Amazon Lex Lex-Bots, Bot-Aliasse und Bot-Channels zu helfen, können Sie jeder Ressource Metadaten zuweisenStichworte. Ein Tag ist eine Markierung, die Sie einer AWS-Ressource zuordnen. Jedes Tag besteht aus einem Schlüssel und einem Wert.

Mit Tags können Sie AWS-Ressourcen auf unterschiedliche Weise kategorisieren (z. B. nach Zweck, Eigentümer oder Anwendung). Tags helfen Ihnen bei Folgendem:

- Identifizieren und Organisieren Ihrer AWS-Ressourcen. Viele AWS-Ressourcen unterstützen das Markieren mit Tags (kurz: Tagging). So können Ressourcen aus verschiedenen Services dasselbe

Tag zuweisen, um anzugeben, dass die Ressourcen verbunden sind. Beispielsweise können Sie einen Bot und die Lambda-Funktionen, die er verwendet, mit demselben Tag kennzeichnen.

- Zuordnen von Kosten. Sie aktivieren diese Tags im AWS Billing and Cost Management-Dashboard. AWS verwendet die Tags zur Kategorisierung Ihrer Kosten und zur Bereitstellung eines monatlichen Kostenzuordnungsberichts. Für Amazon Lex Lex-Daten können Sie Kosten für jeden Alias mit Tags zuordnen, die für den Alias spezifisch sind, mit Ausnahme des`$LATEST`Alias. Sie weisen Kosten für die`$LATEST`Alias mit Tags für Ihren Amazon Lex Lex-Bot. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags im AWS Billing and Cost Management-Benutzerhandbuch.
- Kontrollieren Sie den Zugriff auf Ihre -Ressourcen. Sie können Tags für Amazon Lex Lex-Ressourcen zum Steuern des Zugriffs auf Amazon Lex Lex-Ressourcen verwenden. Diese Richtlinien können einer IAM-Rolle oder einem Benutzer angefügt werden, um die Tag-basierte Zugriffssteuerung zu aktivieren. Weitere Informationen finden Sie unter ABAC mit Amazon Lex . Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Verwenden Sie ein Tag, um auf eine Ressource zuzugreifen.

Sie können mit Tags arbeiten, indem Sie dieAWS Management Console, derAWS Command Line Interfaceoder die Amazon Lex API.

Markieren Ihrer Ressourcen

Wenn Sie die Amazon Lex Lex-Konsole verwenden, können Sie Ressourcen beim Erstellen kennzeichnen, oder Sie können die Tags später hinzufügen. Sie können die Konsole auch verwenden, um vorhandene Tags zu aktualisieren oder zu entfernen.

Wenn Sie dieAWS CLIoder die Amazon Lex Lex-API verwenden Sie die folgenden Operationen, um Tags für Ihre Ressourcen zu verwalten:

- ListTagsForResource— Anzeigen der mit einer Ressource verbundenen Tags.
- PutBotundPutBotAlias— Wenden Sie Tags an, wenn Sie einen Bot oder einen Bot-Alias erstellen.
- TagResource— fügen Tags auf einer bestehenden Ressource hinzu und ändern sie.
- UntagResource— entfernen Tags aus einer Ressource.

Die folgenden Ressourcen in Amazon Lex Lex-Unterstützung beim Tagging:

- Bots - verwenden Sie einen Amazon-Ressourcennamen (ARN) wie folgt:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}`
- Bot-Aliasse - verwenden Sie einen ARN wie folgt:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}:${bot-alias}`
- Bot-Channels - verwenden Sie einen ARN wie folgt:
  - `arn:${partition}:lex:${region}:${account}:bot-channel:${bot-name}:${bot-alias}:${channel-name}`

Tag-Einschränkungen

Die folgenden grundlegenden Einschränkungen gelten für Tags auf Amazon Lex Lex-Ressourcen:

- Maximale Anzahl an Tags - 50
- Maximale Schlüssellänge — 128 Zeichen
- Maximale Länge des Werts — 256 Zeichen
- Gültige Zeichen für Schlüssel und Wert — a—z, A-Z, 0-9, Leerzeichen und die folgenden Zeichen: _.:/= + - und @
- Bei Schlüsseln und Werten wird die Groß-/Kleinschreibung berücksichtigt.
- Verwenden Sie nicht `aws:` als Präfix für den Schlüssel. Dieses Präfix ist für AWS reserviert.

Tagging von Ressourcen (Konsole)

Sie können die Konsole verwenden, um Tags auf einem Bot, einem Bot-Alias oder einer Bot-Channel-Ressource zu verwalten. Sie können Tags hinzufügen, wenn Sie eine Ressource erstellen, oder Sie können Tags hinzufügen, ändern oder von vorhandenen Ressourcen entfernen.

So fügen Sie ein Tag hinzu, wenn Sie einen Bot erstellen:

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie Create (Erstellen), um einen neuen Bot zu erstellen.
3. Wählen Sie unten auf der Seite Create your bot (Bot erstellen) die Option Tags aus.
4. Wählen Sie Add tag (Tag hinzufügen), und fügen Sie dem Bot ein oder mehrere Tags hinzu. Sie können bis zu 50 Tags hinzufügen.

So fügen Sie ein Tag hinzu, wenn Sie einen Bot-Alias erstellen:

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie den Bot aus, dem Sie den Bot-Alias hinzufügen möchten.
3. Wählen Sie Settings aus.
4. Fügen Sie den Aliasnamen hinzu, wählen Sie die Bot-Version und dann Add tags (Tags hinzufügen)aus.
5. Wählen Sie Add tag (Tag hinzufügen), und fügen Sie dem Bot-Alias ein oder mehrere Tags hinzu. Sie können bis zu 50 Tags hinzufügen.

So fügen Sie ein Tag hinzu, wenn Sie einen Bot-Channel erstellen

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie den Bot aus, dem Sie den Bot-Channel hinzufügen möchten.
3. Wählen Sie Channels und dann den Channel aus, den Sie hinzufügen möchten.
4. Fügen Sie die Details für den Bot-Channel hinzu, und wählen Sie dann Tagsaus.
5. Wählen Sie Add tag (Tag hinzufügen), und fügen Sie dem Bot-Channel ein oder mehrere Tags hinzu. Sie können bis zu 50 Tags hinzufügen.

So fügen Sie beim Importieren eines Bots ein Tag hinzu

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.
3. Wählen Sie die ZIP-Datei für den Import des Bots.
4. Wählen Sie Tags und dann Add tag (Tag hinzufügen), um dem Bot ein oder mehrere Tags hinzuzufügen. Sie können bis zu 50 Tags hinzufügen.

So fügen Sie einem vorhandenen Bot ein Tag hinzu, oder entfernen bzw. ändern es:

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.

2. Wählen Sie im linken Menü Bots und dann den Bot aus, den Sie ändern möchten.
3. Wählen Sie Settings (Einstellungen) und dann im linken Menü die Option General (Allgemein).
4. Wählen Sie Tags, und fügen Sie Tags für den Bot hinzu, ändern oder entfernen Sie diese.

So fügen Sie einem Bot-Alias ein Tag hinzu, oder entfernen bzw. ändern es:

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie im linken Menü Bots und dann den Bot aus, den Sie ändern möchten.
3. Wählen Sie Settings (Einstellungen) und dann im linken Menü Aliases (Aliasse).
4. Wählen Sie Manage tags (Tags verwalten) für den Alias, den Sie ändern möchten, und fügen Sie dann Tags für den Bot-Alias hinzu, oder ändern bzw. entfernen Sie sie.

So fügen Sie einem vorhandenen Bot-Channel ein Tag hinzu, oder entfernen bzw. ändern es:

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie im linken Menü Bots und dann den Bot aus, den Sie ändern möchten.
3. Wählen Sie Channels aus.
4. Wählen Sie Tags, und fügen Sie Tags für den Bot-Channel hinzu, ändern oder entfernen Sie diese.

Tagging von Ressourcen (AWS CLI)

Sie können die AWS CLI verwenden um Tags auf einer Bot-, einer Bot-Alias- oder einer Bot-Channel-Ressource zu verwalten. Sie können Tags hinzufügen, wenn Sie einen Bot oder einen Bot-Alias erstellen, oder Sie können Tags aus einem Bot, einem Bot-Alias oder einem Bot-Channel hinzufügen, ändern oder entfernen.

Alle Beispiele sind für Linux und macOS formatiert. Um den Befehl in Windows zu verwenden, ersetzen Sie das Linux-Fortsetzungszeichen (\) durch ein Caret (^).

So fügen Sie ein Tag hinzu, wenn Sie einen Bot erstellen:

- Der folgende abgekürzte `put-bot` AWS CLI-Befehl zeigt die Parameter, die Sie verwenden müssen, um ein Tag hinzuzufügen, wenn Sie einen Bot erstellen. Um tatsächlich einen Bot

zu erstellen, müssen Sie andere Parameter angeben. Weitere Informationen finden Sie unter Schritt 4: Erste Schritte (AWS CLI) .

```
aws lex-models put-bot \
    --tags '[{"key": "key1", "value": "value1"}, \
             {"key": "key2", "value": "value2"}]'
```

So fügen Sie ein Tag hinzu, wenn Sie einen Bot-Alias erstellen:

- Der folgende abgekürzte `put-bot-alias` AWS CLI-Befehl zeigt die Parameter, die Sie verwenden müssen, um ein Tag hinzuzufügen, wenn Sie einen Bot-Alias erstellen. Um tatsächlich einen Bot-Alias zu erstellen, müssen Sie andere Parameter angeben. Weitere Informationen finden Sie unter Übung 5: Erstellen eines Alias (AWS CLI) .

```
aws lex-models put-bot \
    --tags '[{"key": "key1", "value": "value1"}, \
             {"key": "key2", "value": "value2"}]"
```

So listen Sie Tags auf einer Ressource auf:

- Verwenden Sie den `list-tags-for-resource` AWS CLI-Befehl, um die Ressourcen anzuzeigen, die mit einem Bot, Bot-Alias oder Bot-Channel verknüpft sind.

```
aws lex-models list-tags-for-resource \
    --resource-arn bot, bot alias, or bot channel ARN
```

So fügen Sie Tags zu einer Ressource hinzu oder ändern sie:

- Verwenden Sie den `tag-resource` AWS CLI-Befehl, um einen Bot, einen Bot-Alias oder einen Bot-Channel hinzuzufügen oder zu ändern.

```
aws lex-models tag-resource \
    --resource-arn bot, bot alias, or bot channel ARN \
    --tags '[{"key": "key1", "value": "value1"}, \
             {"key": "key2", "value": "value2"}]'
```

So entfernen Sie Tags von einer Ressource:

- Verwenden Sie den `untag-resource` AWS CLI-Befehl, um Tags aus einem Bot, Bot-Alias oder Bot-Channel zu entfernen.

```
aws lex-models untag-resource \
    --resource-arn bot, bot alias, or bot channel ARN \
    --tag-keys '["key1", "key2"]'
```

Erste Schritte mit Amazon Lex

Amazon Lex bietet API-Operationen, die Sie in Ihre vorhandenen Anwendungen integrieren können. Eine Liste der unterstützten Operationen finden Sie unter API-Referenz. Sie können alle der folgenden Optionen verwenden:

- AWS-SDK — Wenn Sie die SDKs verwenden, werden Ihre Anfragen an Amazon Lex automatisch mit den von Ihnen angegebenen Anmeldeinformationen signiert und authentifiziert. Diese Variante ist die empfohlene Option für die Anwendungsprogrammierung.
- AWS CLI — Sie können das verwenden AWS CLI , um auf jede Amazon Lex Lex-Funktion zuzugreifen, ohne Code schreiben zu müssen.
- AWS-Konsole — Die Konsole ist der einfachste Weg, um mit dem Testen und Verwenden von Amazon Lex zu beginnen

Wenn Sie Amazon Lex noch nicht kennen, empfehlen wir Ihnen, zuerst zu lesenAmazon Lex — Funktionsweise.

Themen

Schritt 1: Richten Sie ein AWS Konto ein und erstellen Sie einen Administratorbenutzer

Bevor Sie Amazon Lex zum ersten Mal verwenden, führen Sie die folgenden Aufgaben aus:

1. Melden Sie sich an für AWS
2. Erstellen eines Benutzers

Melden Sie sich an für AWS

Wenn Sie bereits ein AWS Konto haben, überspringen Sie diese Aufgabe.

Wenn Sie sich für Amazon Web Services (AWS) registrieren, wird Ihr AWS Konto automatisch für alle Dienste in registriert AWS, einschließlich Amazon Lex. Berechnet werden Ihnen aber nur die Services, die Sie nutzen.

Mit Amazon Lex zahlen Sie nur für die Ressourcen, die Sie nutzen. Wenn Sie ein neuer AWS Kunde sind, können Sie kostenlos mit Amazon Lex beginnen. Weitere Informationen finden Sie unter AWS – kostenloses Nutzungskontingent.

Wenn Sie bereits ein AWS Konto haben, fahren Sie mit der nächsten Aufgabe fort. Wenn Sie kein AWS -Konto haben, führen Sie die folgenden Schritte zum Erstellen eines Kontos aus.

Um ein AWS Konto zu erstellen

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für einen anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird ein erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

Notieren Sie sich Ihre AWS Konto-ID, da Sie sie für die nächste Aufgabe benötigen.

Erstellen eines Benutzers

Dienste in AWS, wie Amazon Lex, erfordern, dass Sie beim Zugriff auf sie Anmeldeinformationen angeben, damit der Service feststellen kann, ob Sie über Berechtigungen für den Zugriff auf die Ressourcen verfügen, die diesem Service gehören. Für die Konsole müssen Sie Ihr Passwort eingeben. Wir empfehlen jedoch nicht, dass Sie die AWS Anmeldeinformationen für Ihr AWS Konto verwenden. Stattdessen empfehlen wir Folgendes:

- Verwenden Sie AWS Identity and Access Management (IAM), um einen Benutzer zu erstellen
- Fügen Sie den Benutzer einer IAM-Gruppe mit Administratorberechtigungen hinzu
- Gewähren Sie dem erstellten -Benutzer Administratorberechtigungen.

Sie können dann AWS mit einer speziellen URL und den Anmeldeinformationen des Benutzers darauf zugreifen.

Für die Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie einen Benutzer namens (`adminuser`) mit Administratorrechten haben. Befolgen Sie die Schritte zum Einrichten des `adminuser` in Ihrem Konto.

Erstellen eines Administrator-Benutzers und Anmelden in der Konsole

1. Erstellen Sie einen Administratorbenutzer namens `adminuser` in Ihrem AWS -Konto. Anweisungen finden Sie im IAM-Benutzerhandbuch unter Erstellen Ihrer ersten Benutzer - und Administratorgruppe.
2. Als Benutzer können Sie sich AWS Management Console mit einer speziellen URL bei der anmelden. Weitere Informationen finden Sie unter Wie sich Benutzer in Ihrem Konto anmelden im IAM-Benutzerhandbuch.

Weitere Informationen zu IAM finden Sie unter:

- AWS Identity and Access Management (ICH BIN)
- Erste Schritte
- IAM Benutzerhandbuch

Nächster Schritt

Schritt 2: Richten Sie das ein AWS Command Line Interface

Schritt 2: Richten Sie das ein AWS Command Line Interface

Wenn Sie Amazon Lex lieber mit AWS Command Line Interface (AWS CLI) verwenden möchten, laden Sie es herunter und konfigurieren Sie es.

> **Important**
>
> Sie benötigen das nicht AWS CLI , um die Schritte in den Übungen Erste Schritte auszuführen. Für einige spätere Übungen in diesem Handbuch wird die AWS CLI jedoch benötigt. Wenn Sie lieber mit der Konsole beginnen möchten, überspringen Sie diesen Schritt

und fahren mit Schritt 3: Einstiegsübung (Konsole) fort. Wenn Sie das später benötigen AWS CLI, kehren Sie hierher zurück, um es einzurichten.

Um das einzurichten AWS CLI

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface -Benutzerhandbuch:
   - Erste Schritte mit dem AWS Command Line Interface
   - Konfigurieren von AWS Command Line Interface
2. Fügen Sie am Ende der AWS CLI Konfigurationsdatei ein benanntes Profil für den Administratorbenutzer hinzu. Sie verwenden dieses Profil bei der Ausführung von AWS CLI Befehlen. Weitere Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   Eine Liste der verfügbaren AWS Regionen finden Sie unter Regionen und Endpunkte in der Allgemeine Amazon Web Services-Referenz.
3. Geben Sie den Hilfebefehl in die Eingabeaufforderung ein, um die Einrichtung zu überprüfen:

   ```
   aws help
   ```

Schritt 3: Einstiegsübung (Konsole)

Schritt 3: Einstiegsübung (Konsole)

Der einfachste Weg, die Verwendung von Amazon Lex zu erlernen, ist die Verwendung der Konsole. Damit Sie starten können, haben wir die folgenden Übungen erstellt. Für alle wird die Konsole benötigt:

- Übung 1 — Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints, eines vordefinierten Bots, der die gesamte erforderliche Bot-Konfiguration bereitstellt. Sie müssen nur ein Minimum an Arbeit aufwenden, um das end-to-end Setup zu testen.

  Darüber hinaus verwenden Sie den Lambda-Funktions-Blueprint, bereitgestellt von AWS Lambda, um eine Lambda-Funktion zu erstellen. Bei der Funktion handelt es sich um einen Code-Haken, der vordefinierten Code verwendet, der mit Ihrem Bot kompatibel ist.
- Übung 2 — Erstellen Sie einen benutzerdefinierten Bot, indem Sie einen Bot manuell erstellen und konfigurieren. Sie erstellen auch eine Lambda-Funktion als Code-Hook. Beispiel-Code wird bereitgestellt.
- Übung 3 — Veröffentlichen Sie einen Bot und erstellen Sie dann eine neue Version davon. Als Teil dieser Übung erstellen Sie einen Alias, der auf die Bot-Version zeigt.

Themen

Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole)

In dieser Übung führen Sie folgende Aufgaben aus:

- Erstellen Sie Ihren ersten Amazon Lex Lex-Bot und testen Sie ihn in der Amazon Lex Lex-Konsole.

  Für diese Übung verwenden Sie den OrderFlowersBauplan. Weitere Informationen über Pläne finden Sie unter Amazon Lex undAWS LambdaBlueprints.

- Erstellen Sie eineAWS Lambda Funktion und testen Sie sie in der Lambda-Konsole. Während der Bearbeitung einer Anfrage ruft Ihr Bot diese Lambda-Funktion auf. In dieser Übung verwenden Sie einen Lambda-Blueprint (lex-order-flowers-python), der in derAWS Lambda Konsole bereitgestellt wird, um Ihre Lambda-Funktion zu erstellen. Der Blueprint-Code veranschaulicht, wie Sie dieselbe Lambda-Funktion verwenden können, um Initialisierung und Validierung durchzuführen und die`OrderFlowers` Absicht zu erfüllen.

- Aktualisieren Sie den Bot, um die Lambda-Funktion als Code-Hook hinzuzufügen, um die Absicht zu erfüllen. Testen end-to-end Sie die Erfahrung.

Die folgenden Abschnitte erläutern, was Pläne tun.

Amazon Lex Bot: Überblick über den Bauplan

Sie verwenden den OrderFlowersBlueprint, um einen Amazon Lex Lex-Bot zu erstellen. Weitere Informationen zur Struktur eines Bots finden Sie unterAmazon Lex — Funktionsweise. Der Bot ist vorkonfiguriert wie folgt:

- Absicht — OrderFlowers
- Slot-Typen - Ein benutzerdefinierter Slot-Typ namens `FlowerTypes` mit Aufzählungswerten: `roses`, `lilies`, und `tulips`.
- Slots: Die Absicht erfordert die folgenden Informationen (d. h. Slots), bevor der Bot die Absicht erfüllen kann.
  - `PickupTime`(AMAZON.TIME integrierter Typ)
  - `FlowerType`(FlowerTypes benutzerdefinierter Typ)
  - `PickupDate` (AMAZON.DATE integrierter Typ)
- Äußerung: Die folgenden Beispieläußerungen zeigen die Absicht des Benutzers an:
  - "Ich möchte Blumen abholen."
  - "Ich möchte einige Blumen bestellen."
- fordert - Nachdem der Bot die Absicht identifiziert, verwendet er die folgenden Anweisungen zum Ausfüllen der Slots:
  - Anforderung für den `FlowerType` Slot - "Was für Blumen möchten Sie bestellen?"
  - Aufforderung zur Eingabe`PickupDate` des Termins — „An welchem Tag soll der {FlowerType} abgeholt werden?“
  - Aufforderung zur Eingabe`PickupTime` des Zeitpunkts — „Um wie viel Uhr soll der {FlowerType} abgeholt werden?“
  - Bestätigungserklärung — „Okay, Ihr {FlowerType} wird bis {PickupTime} am {PickupDate} zur Abholung bereit sein. Ist das OK?"

AWS Lambda-Funktion: Plan-Zusammenfassung

Die Lambda-Funktion in dieser Übung führt sowohl Initialisierungs- als auch Validierungs- und Erfüllungsaufgaben aus. Daher aktualisieren Sie nach dem Erstellen der Lambda-Funktion die Intent-Konfiguration, indem Sie dieselbe Lambda-Funktion als Code-Hook angeben, um sowohl die Initialisierungs- als auch die Validierungs- und Erfüllungsaufgaben zu erledigen.

- Als Initialisierungs- und Validierungscodehook führt die Lambda-Funktion eine grundlegende Validierung durch. Wenn der Benutzer beispielsweise eine Uhrzeit für die Abholung angibt, die außerhalb der normalen Geschäftszeiten liegt, weist die Lambda-Funktion Amazon Lex an, den Benutzer erneut nach der Uhrzeit zu fragen.
- Als Teil des Fulfillment-Code-Hooks gibt die Lambda-Funktion eine zusammenfassende Meldung zurück, die angibt, dass die Blumenbestellung aufgegeben wurde (das heißt, die Absicht ist erfüllt).

Nächster Schritt

Schritt 1: Erstellen eines Amazon-Lex-Botts (Amazon-Lex-Basisberechtigungen)

Schritt 1: Erstellen eines Amazon-Lex-Botts (Amazon-Lex-Basisberechtigungen)

Erstellen Sie für diese Übung einen Bot zum Bestellen von Blumen, genannt OrderFlowersBot.

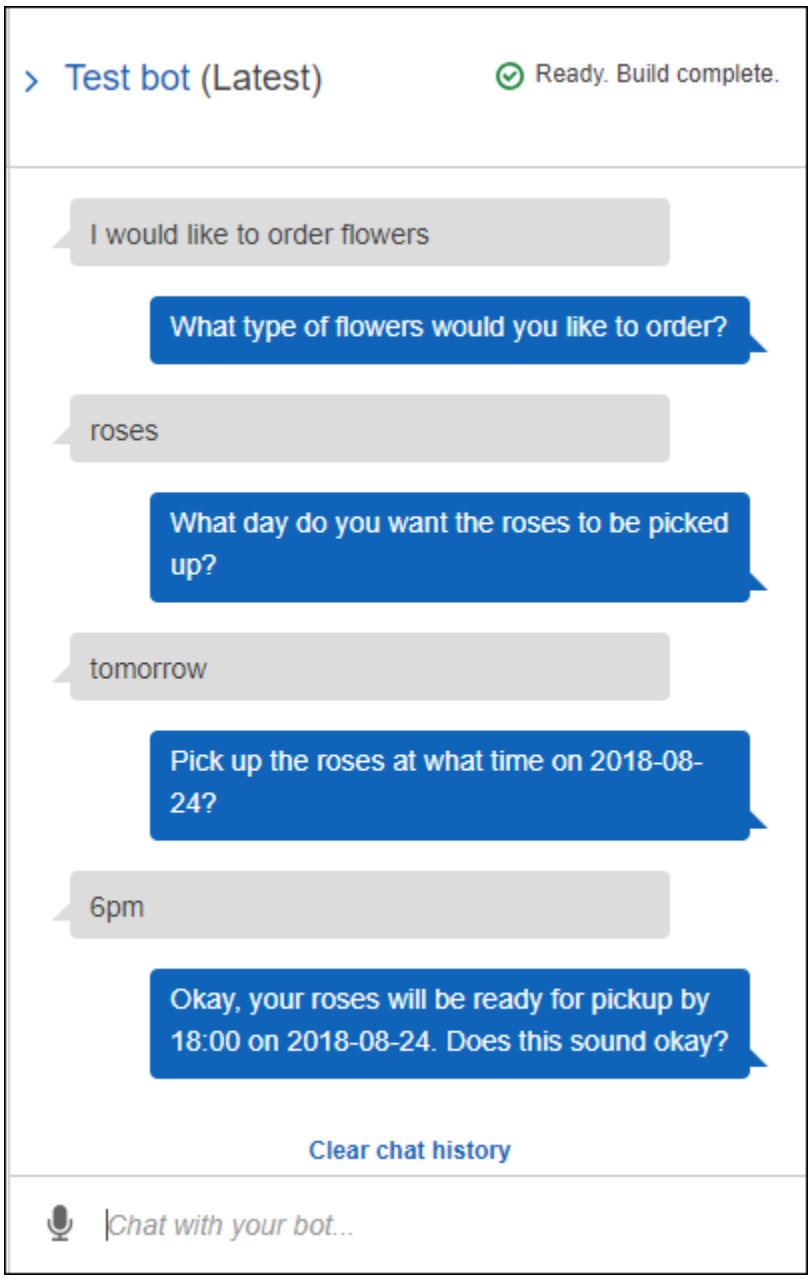
So erstellen Sie einen Amazon Lex Lex-Bot (Konsole)

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie Sie Sie Sie Sie Sie Amazon Lex Sie Sie Sie unter https://console.aws.amazon.com/lex/.
2. Wenn dies Ihr erster Bot ist, wählen Sie Get Started (Erste Schritte); wählen Sie andernfalls auf der Bots-Seite Create (Erstellen).
3. Geben Sie auf der Seite Create your Lex bot die folgenden Informationen an und wählen Sie dann Create.
   - Wählen Sie den Plan OrderFlowers aus.
   - Belassen Sie den Standard-Bot-Namen (OrderFlowers).
   - Wählen Sie für COPPA **No** aus.
   - Wählen Sie für das Speichern von Benutzeräußerungen die entsprechende Antwort aus.
4. Wählen Sie Create (Erstellen) aus. Die Konsole stellt die erforderlichen Anfragen an Amazon Lex, um die Konfiguration zu speichern. Die Konsole zeigt dann das Editorfenster für Bots an.

5. Warten Sie auf die Bestätigung, dass Ihr Bot erstellt wurde.
6. Testen Sie den Bot.

> **Note**
>
> Sie können den Bot durch die Eingabe von Text in das Testfenster oder durch Auswahl der Mikrofonschaltfläche im Testfenster und folgende Spracheingabe testen.

Geben Sie den folgenden Beispieltext ein, um mit dem Bot ein Gespräch über die Bestellung von Blumen zu führen:



Aus dieser Eingabe leitet der Bot die Absicht `OrderFlowers` ab und fordert Slot-Daten an. Wenn Sie alle erforderlichen Slot-Daten angegeben haben, erfüllt der Bot die Absicht (`OrderFlowers`), indem er alle Daten an die Client-Anwendung (in diesem Fall die Konsole) zurückgibt. Die Konsole zeigt die Daten im Testfenster an.

Das heißt:

- Die Aussage "An welchem Tag möchten Sie die Rosen abholen?" enthält den Ausdruck "Rosen", weil die Aufforderung für den Slot `pickupDate` unter Verwendung von Ersetzungen (`{FlowerType}`) konfiguriert wurde. Verifizieren Sie dies in der Konsole.

- Die "Okay, die Rosen werden bereit sein..."-Aussage ist die von Ihnen konfigurierte Bestätigungsaufforderung.
- Die letzte Anweisung ("`FlowerType:roses...`") bezeichnet einfach die Slot-Daten, die an den Client zurückgegeben werden, in diesem Fall das Testfenster. In der nächsten Übung verwenden Sie eine Lambda-Funktion, um die Absicht zu erfüllen. In diesem Fall erhalten Sie eine Meldung, dass die Bestellung erfüllt ist.

Nächster Schritt

Schritt 2 (optional): Überprüfen der Details des Informationsflusses (Konsole)

Schritt 2 (optional): Überprüfen der Details des Informationsflusses (Konsole)

In diesem Abschnitt wird der Informationsfluss zwischen einem Kunden und Amazon Lex für jede Benutzereingabe in unserer Beispielkonversation erläutert.

Das Beispiel verwendet das Konsolentestfenster für die Konversation mit dem Bot.

Um das Amazon Lex Lex-Testfenster zu öffnen

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie Sie Sie Sie Sie Sie Amazon Lex Sie Sie Sie unter https://console.aws.amazon.com/lex/.
2. Wählen Sie den zu testenden Bot aus.
3. Wählen Sie auf der rechten Seite der Konsole Chatbda-Basisberechtigungen.

Wählen Sie das einschlägige Thema, um den Informationsfluss für gesprochene oder eingetippte Inhalte darzustellen.

Themen

- Schritt 2a (optional): Prüfen der Details des Informationsflusses gesprochener Inhalte (Konsole)
- Schritt 2b (optional): Prüfen der Details des Informationsflusses eingetippter Inhalte (Konsole)

Schritt 2a (optional): Prüfen der Details des Informationsflusses gesprochener Inhalte (Konsole)

In diesem Abschnitt wird der Informationsfluss zwischen dem Kunden und Amazon Lex erläutert, wenn der Client Sprache zum Senden von Anfragen verwendet. Weitere Informationen finden Sie unter PostContent.

1. Der Benutzer sagt: „Ich möchte Blumen bestellen.“

   a. Der Client (Konsole) sendet die folgende PostContent-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/
content HTTP/1.1
x-amz-lex-session-attributes: "e30="
Content-Type: "audio/x-l16; sample-rate=16000; channel-count=1"
Accept: "audio/mpeg"


Request body
input stream
```

   Sowohl die Anfrage-URI als auch der Text liefern Informationen an Amazon Lex:

   - Request-URI — Stellt den Bot-Namen (`OrderFlowers`), den Bot-Alias (`$LATEST`) und den Benutzernamen (eine zufällige Zeichenfolge, die den Benutzer identifiziert) bereit. `content`gibt an, dass dies ein`PostContent` API-Anfrage ist (kein`PostText` Anfrage).
   - Anfordern von Headern
     - `x-amz-lex-session-attributes`— Der base64-kodierte Wert steht für „{}“. Wenn der Client die erste Anforderung stellt, gibt es keine Sitzungsattribute.
     - `Content-Type`: Gibt das Audioformat wieder.
   - Anforderungstext: Der Audiostream der Benutzereingabe („Ich möchte Blumen bestellen.“)

   > **Note**
   >
   > Wenn der Benutzer sich entscheidet, den Text („Ich möchte Blumen bestellen“) an die `PostContent`-API zu senden, anstatt ihn zu sprechen, ist der Anforderungstext die Benutzereingabe. Die `Content-Type`-Kopfzeile wird entsprechend eingestellt:
   >
   > ```
   > POST /bot/OrderFlowers/alias/$LATEST/
   > user/4o9wwdhx6nlheferh6a73fujd3118f5w/content HTTP/1.1
   > x-amz-lex-session-attributes: "e30="
   > Content-Type: "text/plain; charset=utf-8"
   > Accept: accept
   >
   > Request body
   > ```

```
input stream
```

b. Aus dem Eingabe-Stream erkennt Amazon Lex die Absicht (`OrderFlowers`). Es wählt dann einen der Slots der Absicht (in diesem Fall den `FlowerType`) und eine der Aufforderungen zur Abfrage von Werten. Dann sendet es eine Antwort mit den folgenden Kopfzeilen:

```
x-amz-lex-dialog-state:ElicitSlot
x-amz-lex-input-transcript:I would like to order some flowers.
x-amz-lex-intent-name:OrderFlowers
x-amz-lex-message:What type of flowers would you like to order?
x-amz-lex-session-attributes:e30=
x-amz-lex-slot-to-elicit:FlowerType
x-amz-lex-
slots:eyJQaWNrdXBUaW1lIjpudWxsLCJGbG93ZXJUeXBlIjpudWxsLCJQaWNrdXBEYXRlIjpudWxsfQ==
```

Die Kopfzeilenwerte liefern die folgenden Informationen:

- `x-amz-lex-input-transcript`: Liefert das Transkript des Audios (Benutzereingabe) der Anforderung.
- `x-amz-lex-message`— Stellt das Transkript des Audios bereit, das Amazon Lex in der Antwort zurückgegeben hat
- `x-amz-lex-slots`: Die base64-kodierte Version der Slots und Werte:

  ```
  {"PickupTime":null,"FlowerType":null,"PickupDate":null}
  ```

- `x-amz-lex-session-attributes`: Die base64-kodierte Version der Sitzungsattribute ({}):

Der Client gibt das Audio im Antworttext wieder.

2. Der Benutzer sagt: Rosen

a. Der Client (Konsole) sendet die folgende PostContent-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/
content HTTP/1.1
x-amz-lex-session-attributes: "e30="
Content-Type: "audio/x-l16; sample-rate=16000; channel-count=1"
```

```
Accept: "audio/mpeg"


Request body
input stream ("roses")
```

Der Anforderungstext ist der Audiostream der Benutzereingabe (Rosen). Die `sessionAttributes` bleiben leer.

b. Amazon Lex interpretiert den Eingabestream im Kontext der aktuellen Absicht (es erinnert sich, dass es diesen Benutzer um Informationen zum`FlowerType` Slot gebeten hat). Amazon Lex aktualisiert zunächst den Slot-Wert für die aktuelle Absicht. Er wählt einen anderen Slot (`PickupDate`) zusammen mit einer der Aufforderungen ("Wann möchten Sie die Rosen abholen?") und gibt eine Antwort mit folgendem Header zurück:

```
x-amz-lex-dialog-state:ElicitSlot
x-amz-lex-input-transcript:roses
x-amz-lex-intent-name:OrderFlowers
x-amz-lex-message:When do you want to pick up the roses?
x-amz-lex-session-attributes:e30=
x-amz-lex-slot-to-elicit:PickupDate
x-amz-lex-
slots:eyJQaWNrdXBUaW1lIjpudWxsLCJGbG93ZXJUeXBlIjoicm9zaSdzIiwiUGlja3VwRGF0ZSI6bnVsbH0=
```

Die Kopfzeilenwerte liefern die folgenden Informationen:

- `x-amz-lex-slots`: Die base64-kodierte Version der Slots und Werte:

  ```
  {"PickupTime":null,"FlowerType":"roses","PickupDate":null}
  ```

- `x-amz-lex-session-attributes`: Die base64-kodierte Version der Sitzungsattribute ({}):

Der Client gibt das Audio im Antworttext wieder.

3. Der Benutzer sagt: morgen

   a. Der Client (Konsole) sendet die folgende PostContent-Anforderung an Amazon Lex:

   ```
   POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/
   content HTTP/1.1
   ```

```
x-amz-lex-session-attributes: "e30="
Content-Type: "audio/x-l16; sample-rate=16000; channel-count=1"
Accept: "audio/mpeg"


Request body
input stream ("tomorrow")
```

Der Anforderungstext ist der Audiostream der Benutzereingabe („morgen“). Die `sessionAttributes` bleiben leer.

b. Amazon Lex interpretiert den Eingabestream im Kontext der aktuellen Absicht (es erinnert sich, dass es diesen Benutzer um Informationen zum`PickupDate` Slot gebeten hat). Amazon Lex aktualisiert den Wert slot (`PickupDate`) für den aktuellen Intent. Dann wählt es einen anderen Slot, um einen Wert zu erfragen (`PickupTime`) und eine der Aufforderungen zur Eingabe eines Werts (Wann möchten Sie die Rosen am 18.03.2017 abholen?) und gibt eine Antwort mit den folgenden Kopfzeilen zurück:

```
x-amz-lex-dialog-state:ElicitSlot
x-amz-lex-input-transcript:tomorrow
x-amz-lex-intent-name:OrderFlowers
x-amz-lex-message:When do you want to pick up the roses on 2017-03-18?
x-amz-lex-session-attributes:e30=
x-amz-lex-slot-to-elicit:PickupTime
x-amz-lex-
slots:eyJQaWNrdXBUaW1lIjpudWxsLCJGbG93ZXJUeXBlIjoicm9zaSdzIiwiUGlja3VwRGF0ZSI6IjIwMTctM
x-amzn-RequestId:3a205b70-0b69-11e7-b447-eb69face3e6f
```

Die Kopfzeilenwerte liefern die folgenden Informationen:

- `x-amz-lex-slots`: Die base64-kodierte Version der Slots und Werte:

  ```
  {"PickupTime":null,"FlowerType":"roses","PickupDate":"2017-03-18"}
  ```

- `x-amz-lex-session-attributes`: Die base64-kodierte Version der Sitzungsattribute ({}):

Der Client gibt das Audio im Antworttext wieder.

4. Der Benutzer sagt: 18 Uhr

a. Der Client (Konsole) sendet die folgende PostContent-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/
content HTTP/1.1
x-amz-lex-session-attributes: "e30="
Content-Type: "text/plain; charset=utf-8"
Accept: "audio/mpeg"


Request body
input stream ("6 pm")
```

Der Anforderungstext ist der Audiostream der Benutzereingabe („18 Uhr“). Die `sessionAttributes` bleiben leer.

b. Amazon Lex interpretiert den Eingabestream im Kontext der aktuellen Absicht (es erinnert sich, dass es diesen Benutzer um Informationen zum`PickupTime` Slot gebeten hat). Es aktualisiert zuerst den Slot-Wert für die aktuelle Absicht.

Jetzt stellt Amazon Lex fest, dass es Informationen für alle Steckplätze hat. Die Absicht `OrderFlowers` ist jedoch mit einer Bestätigungsmitteilung konfiguriert. Daher benötigt Amazon Lex eine ausdrückliche Bestätigung des Benutzers, bevor es mit der Erfüllung der Absicht fortfahren kann. Es sendet eine Antwort mit den folgenden Kopfzeilen, die eine Bestätigung anfordert, bevor die Blumen bestellt werden:

```
x-amz-lex-dialog-state:ConfirmIntent
x-amz-lex-input-transcript:six p. m.
x-amz-lex-intent-name:OrderFlowers
x-amz-lex-message:Okay, your roses will be ready for pickup by 18:00 on
 2017-03-18.  Does this sound okay?
x-amz-lex-session-attributes:e30=
x-amz-lex-
slots:eyJQaWNrdXBUaW1lIjoiMTg6MDAiLCJGbG93ZXJUeXBlIjoicm9zaSdzIiwiUGlja3VwRGF0ZSI6IjIwM
x-amzn-RequestId:083ca360-0b6a-11e7-b447-eb69face3e6f
```

Die Kopfzeilenwerte liefern die folgenden Informationen:

- `x-amz-lex-slots`: Die base64-kodierte Version der Slots und Werte:

```
{"PickupTime":"18:00","FlowerType":"roses","PickupDate":"2017-03-18"}
```

- `x-amz-lex-session-attributes`: Die base64-kodierte Version der Sitzungsattribute ({}):

Der Client gibt das Audio im Antworttext wieder.

5. Der Benutzer sagt: Ja

   a. Der Client (Konsole) sendet die folgende PostContent-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/
content HTTP/1.1
x-amz-lex-session-attributes: "e30="
Content-Type: "audio/x-l16; sample-rate=16000; channel-count=1"
Accept: "audio/mpeg"


Request body
input stream ("Yes")
```

   Der Anforderungstext ist der Audiostream der Benutzereingabe („Ja“). Die `sessionAttributes` bleiben leer.

   b. Amazon Lex interpretiert den Eingabestream und versteht, dass der Benutzer mit der Bestellung fortfahren möchte. Die Absicht `OrderFlowers` wird mit `ReturnIntent` als Erfüllungsaktivität konfiguriert. Dadurch wird Amazon Lex angewiesen, alle Absichtsdaten an den Kunden zurückzugeben. Amazon Lex gibt eine Antwort mit der folgenden Antwort zurück:

```
x-amz-lex-dialog-state:ReadyForFulfillment
x-amz-lex-input-transcript:yes
x-amz-lex-intent-name:OrderFlowers
x-amz-lex-session-attributes:e30=
x-amz-lex-
slots:eyJQaWNrdXBUaW1lIjoiMTg6MDAiLCJGbG93ZXJUeXBlIjoicm9zaSdzIiwiUGlja3VwRGF0ZSI6IjIwM
```

   Die Antwortkopfzeile `x-amz-lex-dialog-state` ist eingestellt auf `ReadyForFulfillment`. Der Client kann die Absicht dann erfüllen.

6. Testen Sie den Bot jetzt erneut. Um einen neuen Benutzerkontext einzurichten, wählen Sie den Link Clear in der Konsole. Geben Sie Daten für die Absicht `OrderFlowers` an und fügen Sie dabei einige ungültige Daten ein. Beispiel:

   - Jasmin als Blumenart (zählt nicht zu den unterstützten Blumenarten)
   - Gestern als den Tag, an dem Sie die Blumen abholen möchten

   Beachten Sie, dass der Bot diese Werte akzeptiert, weil Sie keinen Code haben, um die Benutzerdaten zu initialisieren und zu validieren. Im nächsten Abschnitt fügen Sie dazu eine Lambda-Funktion hinzu. Beachten Sie Folgendes im Zusammenhang mit Lambda-Funktion:

   - Sie validiert Slot-Daten nach jeder Benutzereingabe. Sie erfüllt schließlich die Absicht. Der Bot verarbeitet also die Blumenbestellung und gibt eine Mitteilung an den Benutzer zurück, statt einfach Slot-Daten an den Client zurückzugeben. Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen.
   - Sie stellt auch die Sitzungsattribute ein. Weitere Informationen über Sitzungsattribute finden Sie unter PostText.

     Nachdem Sie den Abschnitt "Erste Schritte" abgeschlossen haben, können Sie die zusätzlichen Übungen ausführen (Weitere Beispiele: Amazon Lex-Bots erstellen). Reise buchen verwendet Sitzungsattribute, um Informationen über Absichten hinweg gemeinsam zu nutzen und so eine dynamische Unterhaltung mit dem Benutzer zu führen.

Nächster Schritt

Schritt 3: Erstellen einer Lambda-Funktion (Lambda-Funktion)

Schritt 2b (optional): Prüfen der Details des Informationsflusses eingetippter Inhalte (Konsole)

In diesem Abschnitt wird der Informationsfluss zwischen dem Client und Amazon Lex erklärt, in dem der Client die `PostText`-API zur Übermittlung von Anforderungen verwendet. Weitere Informationen finden Sie unter PostText.

1. Der Benutzer gibt ein: Ich möchte Blumen bestellen.

   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
```

```
"Content-Encoding":"amz-1.0"

{
    "inputText": "I would like to order some flowers",
    "sessionAttributes": {}
}
```

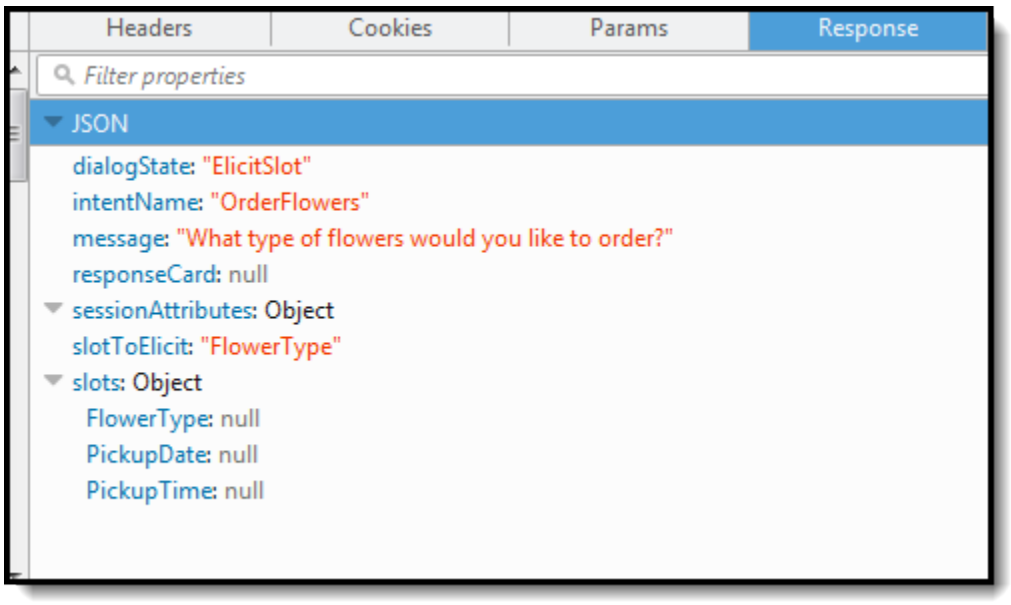
Sowohl die Anfrage-URI als auch der Text liefern Informationen an Amazon Lex:

- Anforderungs-URI — Stellt den Botnamen (`OrderFlowers`), den Bot-Alias (`$LATEST`) und den Benutzernamen (eine zufällige Zeichenfolge zur Identifizierung des Benutzers) bereit. Der abschließende `text` zeigt an, dass es sich um eine `PostText`-API-Anforderung handelt (und nicht um `PostContent`).

- Anforderungsinhalt: Enthält die Benutzereingabe (`inputText`) und leere `sessionAttributes`. Wenn der Client die erste Anforderung stellt, gibt es keine Sitzungsattribute. Die Lambda-Funktion initiiert sie später.

b. Anhand des`inputText` erkennt Amazon Lex die Absicht (`OrderFlowers`). Diese Absicht enthält keine Code-Hooks (d. h. die Lambda-Funktionen) für die Initialisierung und Validierung von Benutzereingaben oder deren Erfüllung.

   Amazon Lex wählt eines der Felder der Absicht (`FlowerType`) aus, um den Wert zu ermitteln. Es wählt auch eine der Aufforderungen zur Angabe von Werten für den Slot aus (alle Teil der Konfiguration der Absicht) und sendet dann die folgende Antwort an den Client zurück. Die Konsole zeigt die Mitteilung in der Antwort an den Benutzer an.



Der Client zeigt die Mitteilung in der Antwort an.

2. Benutzer gibt ein: Rosen

   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

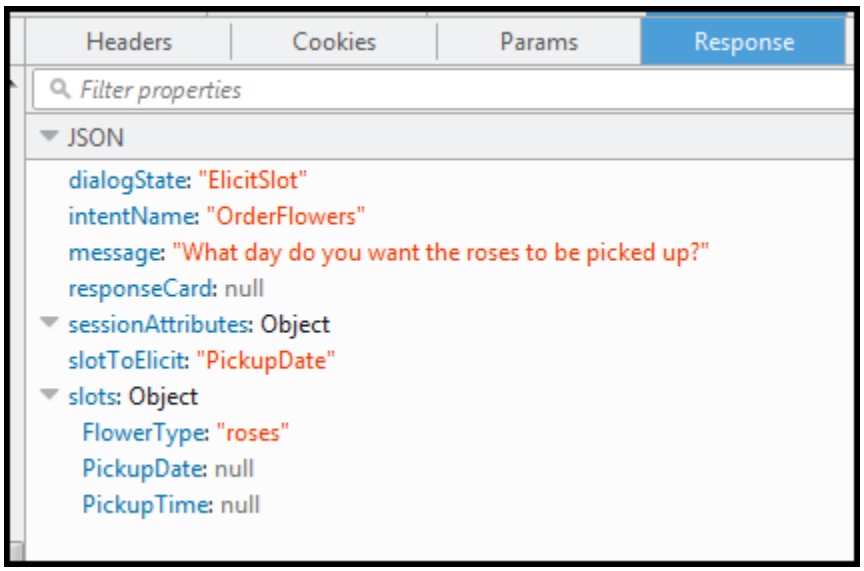
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "roses",
    "sessionAttributes": {}
}
```

   `inputText` im Anforderungstext stellt die Benutzereingabe bereit. Die `sessionAttributes` bleiben leer.

   b. Amazon Lex interpretiert das zuerst`inputText` im Kontext der aktuellen Absicht — der Service erinnert sich, dass er den bestimmten Benutzer um Informationen über den`FlowerType` Slot gebeten hat. Amazon Lex aktualisiert zuerst den Slot-Wert für die aktuelle Absicht und wählt einen anderen Slot (`PickupDate`) zusammen mit einer der zugehörigen Eingabeaufforderungen aus. An welchem Tag sollen die Rosen abgeholt werden? — für den Slot.

   Anschließend gibt Amazon Lex die folgende Antwort zurück:



Der Client zeigt die Mitteilung in der Antwort an.

3. Benutzer gibt ein: morgen

   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

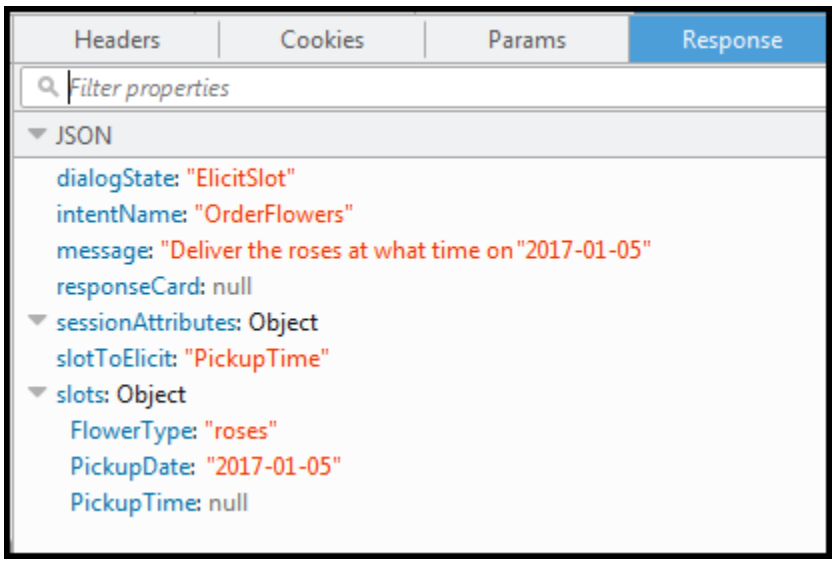
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "tomorrow",
    "sessionAttributes": {}
}
```

   `inputText` im Anforderungstext stellt die Benutzereingabe bereit. Die `sessionAttributes` bleiben leer.

   b. Amazon Lex interpretiert das zuerst`inputText` im Kontext der aktuellen Absicht — der Service erinnert sich, dass er den bestimmten Benutzer um Informationen über den`PickupDate` Slot gebeten hat. Amazon Lex aktualisiert den Wert slot (`PickupDate`) für den aktuellen Intent. Es wählt einen anderen Slot aus, um den Wert zu erfragen (`PickupTime`). Es gibt eine der Eingabeaufforderungen zur Wertermittlung zurück. Zu welcher Uhrzeit sollen die Rosen am 05.01.2017 geliefert werden? — an den Kunden.

   Amazon Lex gibt dann die folgende Antwort zurück:



Der Client zeigt die Mitteilung in der Antwort an.

4. Benutzer gibt ein: 18 Uhr

   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

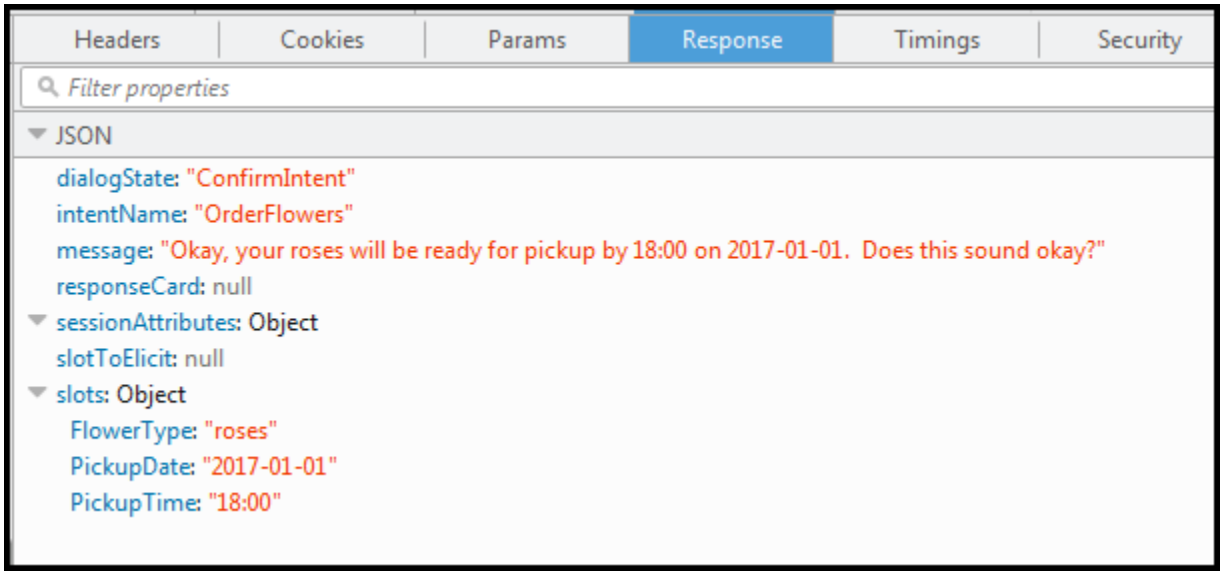
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "6 pm",
    "sessionAttributes": {}
}
```

   `inputText` im Anforderungstext stellt die Benutzereingabe bereit. Die `sessionAttributes` bleiben leer.

   b. Amazon Lex interpretiert das zuerst`inputText` im Kontext der aktuellen Absicht — der Service erinnert sich, dass er den bestimmten Benutzer um Informationen über den`PickupTime` Slot gebeten hat. Amazon Lex aktualisiert zunächst den Slot-Wert für die aktuelle Absicht. Jetzt stellt Amazon Lex fest, dass es Informationen für alle Steckplätze hat.

   Die Absicht `OrderFlowers` ist mit einer Bestätigungsmitteilung konfiguriert. Daher benötigt Amazon Lex eine ausdrückliche Bestätigung des Benutzers, bevor es mit der Erfüllung der Absicht fortfahren kann. Amazon Lex sendet die folgende Nachricht an den Kunden und bittet um Bestätigung, bevor die Blumen bestellt werden:



Der Client zeigt die Mitteilung in der Antwort an.

5. Benutzer gibt ein: Ja

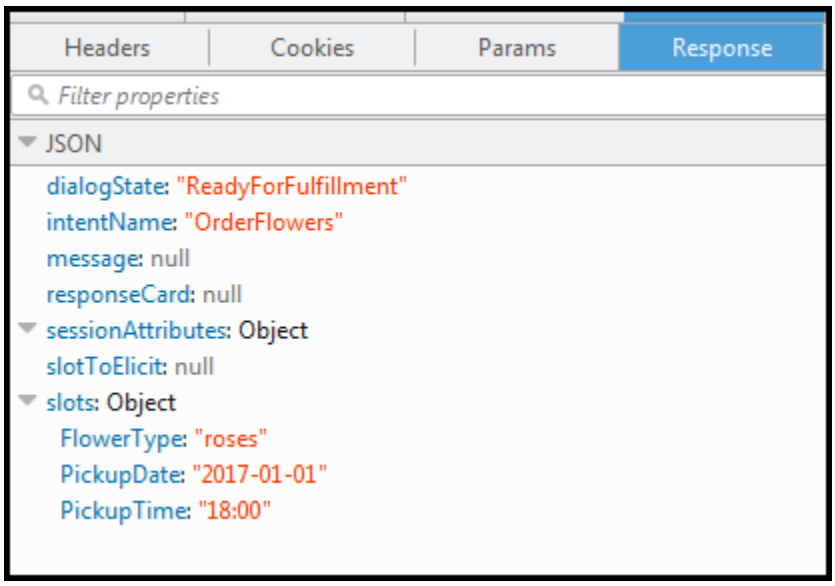
   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "Yes",
    "sessionAttributes": {}
}
```

   `inputText` im Anforderungstext stellt die Benutzereingabe bereit. Die `sessionAttributes` bleiben leer.

   b. Amazon Lex interpretiert das`inputText` im Kontext der Bestätigung der aktuellen Absicht. Es versteht, dass der Benutzer mit der Bestellung fortfahren möchte. Die`OrderFlowers` Absicht wird`ReturnIntent` als Erfüllungsaktivität konfiguriert (es gibt keine Lambda-Funktion, um die Absicht zu erfüllen). Daher gibt Amazon Lex die folgenden Slot-Daten an den Client zurück.



Amazon Lex hat`dialogState` den Ton angegeben`ReadyForFulfillment`. Der Client kann die Absicht dann erfüllen.

6. Testen Sie nun den Bot erneut. Dafür müssen Sie den Link Clear in der Konsole wählen, um einen neuen (Benutzer-)Kontext zu etablieren. Jetzt, da Sie die Daten für die Absicht Blumenbestellung angegeben haben, versuchen Sie, ungültige Daten anzugeben. Beispiel:

   - Jasmin als Blumenart (zählt nicht zu den unterstützten Blumenarten)
   - Gestern als den Tag, an dem Sie die Blumen abholen möchten

   Beachten Sie, dass der Bot diese Werte akzeptiert, weil Sie keinen Code haben, um Benutzerdaten zu initialisieren/zu bestätigen. Im nächsten Abschnitt fügen Sie dazu eine Lambda-Funktion hinzu. Beachten Sie Folgendes im Zusammenhang mit Lambda-Funktion:

   - Die Lambda-Funktion validiert die Slot-Daten nach jeder Benutzereingabe. Sie erfüllt schließlich die Absicht. Der Bot verarbeitet also die Blumenbestellung und gibt eine Mitteilung an den Benutzer zurück, statt einfach Slot-Daten an den Client zurückzugeben. Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen.
   - Die Lambda-Funktion legt auch die Sitzungsattribute fest. Weitere Informationen über Sitzungsattribute finden Sie unter PostText.

     Nachdem Sie den Abschnitt "Erste Schritte" abgeschlossen haben, können Sie die zusätzlichen Übungen ausführen (Weitere Beispiele: Amazon Lex-Bots erstellen). Reise

buchen verwendet Sitzungsattribute, um Informationen über Absichten hinweg gemeinsam zu nutzen und so eine dynamische Unterhaltung mit dem Benutzer zu führen.

Nächster Schritt

Schritt 3: Erstellen einer Lambda-Funktion (Lambda-Funktion)

Schritt 3: Erstellen einer Lambda-Funktion (Lambda-Funktion)

Erstellen Sie eine Lambda-Funktion (mithilfe des lex-order-flowers-pythonBlueprints) und führen Sie einen Testaufruf mithilfe von Beispielereignisdaten in derAWS Lambda Konsole durch.

Sie kehren zur Amazon Lex Lex-Konsole zurück und fügen die Lambda-Funktion als Code-Hook hinzu, um die`OrderFlowers` Absicht in der Datei zu erfüllen`OrderFlowersBot`, die Sie im vorherigen Abschnitt erstellt haben.

So erstellen Sie eine Lambda-Funktion (Konsole)

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie Create function (Funktion erstellen).
3. Wählen Sie auf der Seite Create function (Funktion erstellen) die Option Use a blueprint (Blueprint verwenden) aus. Geben Sie **`lex-`** in das Textfeld für den Filter ein. Drücken Sie anschließend `Enter`, um die Vorlage zu suchen, und wählen Sie die Vorlage `lex-order-flowers-python` aus.

   Lambda-Funktions-Blueprints sind sowohl in Node.js als auch in Python verfügbar. Verwenden Sie für diese Übung den auf Python basierenden Plan.
4. Führen Sie auf der Seite Basic information (Basisinformationen) folgende Schritte aus:

   - Geben Sie einen Lambda-Funktionsnamen (`OrderFlowersCodeHook`) ein.
   - Wählen Sie für Execution role (Ausführungsrolle) die Option Create a new role with basic Lambda permissions (Neue Rolle mit Lambda-Berechtigungen erstellen)
   - Übernehmen Sie im Übrigen die Standardwerte.
5. Wählen Sie Create function (Funktion erstellen).
6. Wenn Sie ein anderes Gebietsschema als Englisch (US) (en-US) verwenden, aktualisieren Sie die Absichtsnamen wie unter beschriebenAktualisieren eines Blueprints für ein bestimmtes Gebietsschema.

7. Lambda-Funktion testen

   a. Wählen Sie Select a test event (Testereignis auswählen), Configure test events (Testereignisse konfigurieren) aus.

   b. Wählen Sie Amazon Lex Order Flowers aus der Liste der Event-Vorlagen aus. Dieses Beispielereignis entspricht dem Anforderungs-/Antwortmodell von Amazon Lex (siehe Verwenden von Lambda-Funktionen). Geben Sie dem Testereignis einen Namen (`LexOrderFlowersTest`).

   c. Wählen Sie Create (Erstellen) aus.

   d. Wählen Sie Test (Testen) aus, um den Code-Haken zu testen.

   e. Stellen Sie sicher, dass die Lambda-Funktion erfolgreich ausgeführt wurde. Die Antwort entspricht in diesem Fall dem Amazon Lex Lex-Antwortmodell.

Nächster Schritt

Schritt 4: Fügen Sie die Lambda-Funktion als Code-Hook hinzu (Konsole)

Schritt 4: Fügen Sie die Lambda-Funktion als Code-Hook hinzu (Konsole)

In diesem Abschnitt aktualisieren Sie die Konfiguration der OrderFlowers Absicht, die Lambda-Funktion zu verwenden, wie folgt:

- Verwenden Sie zunächst die Lambda-Funktion als Code-Hook, um die`OrderFlowers` Absicht zu erfüllen. Sie testen den Bot und stellen sicher, dass Sie eine Erfüllungsnachricht von der Lambda-Funktion erhalten haben. Amazon Lex ruft die Lambda-Funktion erst auf, nachdem Sie Daten für alle erforderlichen Felder für die Bestellung von Blumen angegeben haben.
- Konfigurieren Sie dieselbe Lambda-Funktion als Code-Hook, um die Initialisierung und Validierung durchzuführen. Sie testen und überprüfen, ob die Lambda-Funktion eine Validierung durchführt (indem Sie Slot-Daten angeben).

Um eine Lambda-Funktion als Code-Hook hinzuzufügen (Konsole)

1. Wählen Sie in der Amazon Lex-Konsole den OrderFlowersBot aus. Die Konsole zeigt die OrderFlowersAbsicht. Stellen Sie sicher, dass die Absichtsversion auf `$LATEST` eingestellt ist, da dies die einzige Version ist, die wir ändern können.
2. Fügen Sie die Lambda-Funktion als Fulfillment-Code-Hook hinzu und testen Sie sie.

a. Wählen Sie im Editor AWS LambdaFunktion als Fulfilment und wählen Sie die Lambda-Funktion aus, die Sie im vorherigen Schritt erstellt haben (`OrderFlowersCodeHook`). Wählen Sie OK, um Amazon Lex die Erlaubnis zum Aufrufen der Lambda-Funktion.

   Sie konfigurieren diese Lambda-Funktion als Code-Hook, um die Absicht zu erfüllen. Amazon Lex ruft diese Funktion erst auf, wenn alle zur Erfüllung der Absicht erforderlichen Slot-Daten des Benutzers vorliegen.

b. Geben Sie eine Goodbye message an.

c. Wählen Sie Build aus.

d. Testen Sie den Bot mit der vorherigen Konversation.

Die letzte Aussage „Danke, deine Bestellung für Rosen...“ ist eine Antwort der Lambda-Funktion, die du als Code-Hook konfiguriert hast. Im vorherigen Abschnitt gab es keine Lambda-Funktion. Jetzt verwenden Sie eine Lambda-Funktion, um die`OrderFlowers` Absicht tatsächlich zu erfüllen.

3. Fügen Sie die Lambda-Funktion als Initialisierungs- und Validierungscode-Hook hinzu und testen Sie.

   Der von Ihnen verwendete Lambda-Beispielfunktionscode kann sowohl die Validierung als auch die Erfüllung von Benutzereingaben durchführen. Das Eingabeereignis, das die Lambda-Funktion empfängt, hat ein Feld (`invocationSource`), anhand dessen der Code bestimmt, welcher Teil des Codes ausgeführt werden soll. Weitere Informationen finden Sie unter Lambda-Funktions-Eingabe-Ereignis und Antwort-Format.

   a. Wählen Sie die "$LATEST"-Version der `OrderFlowers`-Absicht aus. Das ist die einzige Version, die Sie aktualisieren können.

   b. Wählen Sie im Editor unter Options den Wert Initialization and validation aus.

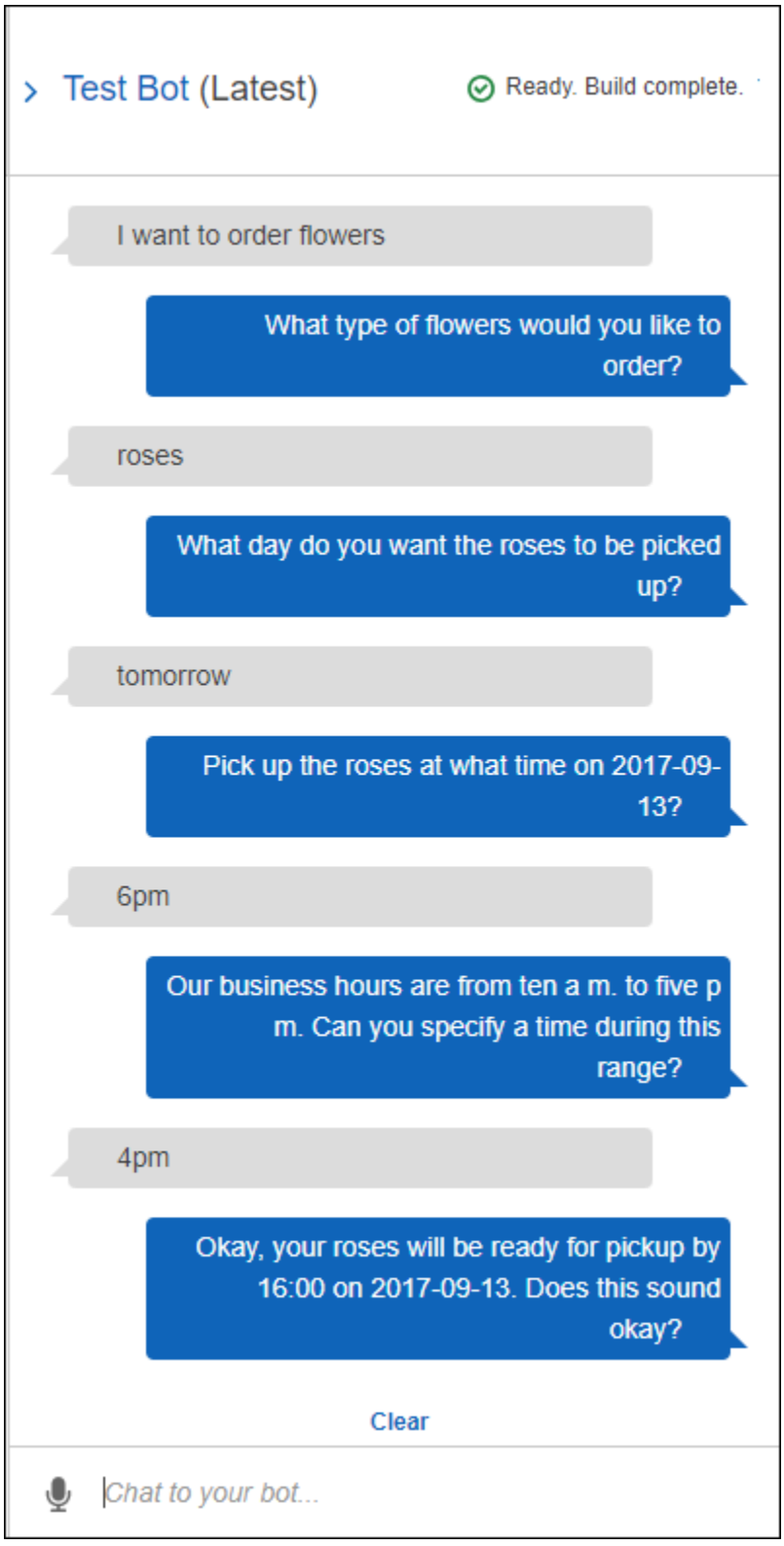
   c. Wählen Sie erneut dieselbe Lambda-Funktion aus.

   d. Wählen Sie Build aus.

   e. Testen Sie den Bot.

      Sie sind jetzt bereit, wie in der folgenden Abbildung gezeigt, mit Amazon Lex zu sprechen. Um den Validierungsteil zu testen, wählen Sie die Uhrzeit 18 Uhr. Ihre Lambda-Funktion gibt eine Antwort zurück („Unsere Geschäftszeiten sind von 10 bis 17 Uhr.“) und fordert Sie

erneut auf. Nachdem Sie alle gültigen Slot-Daten angegeben haben, erfüllt die Lambda-Funktion die Bestellung.



Nächster Schritt

[Schritt 5 (optional): Prüfen der Details des Informationsflusses (Konsole)]

Schritt 5 (optional): Prüfen der Details des Informationsflusses (Konsole)

In diesem Abschnitt wird der Informationsfluss zwischen dem Client und Amazon Lex für jede Benutzereingabe erläutert, einschließlich der Integration der Lambda-Funktion.

> **Note**
>
> In diesem Abschnitt wird davon ausgegangen, dass der Client mithilfe der`PostText` Runtime-API Anfragen an Amazon Lex sendet, und die Anfrage- und Antwortdetails werden entsprechend angezeigt. Ein Beispiel für den Informationsfluss zwischen dem Client und Amazon Lex, bei dem der Kunde die`PostContent` API verwendet, finden Sie unterSchritt 2a (optional): Prüfen der Details des Informationsflusses gesprochener Inhalte (Konsole) .

Weitere Informationen über die `PostText`-Laufzeit-API und weitere Details zu den Anforderungen und Antworten, die in den folgenden Schritten gezeigt werden, finden Sie unter PostText.

1. Benutzer: Ich möchte einige Blumen bestellen.

   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/ignw84y6seypre4xly5rimopuri2xwnd/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "I would like to order some flowers",
    "sessionAttributes": {}
}
```

   Sowohl die Anfrage-URI als auch der Text liefern Informationen an Amazon Lex:

   - Anforderungs-URI — Stellt den Botnamen (`OrderFlowers`), den Bot-Alias (`$LATEST`) und den Benutzernamen (eine zufällige Zeichenfolge zur Identifizierung des Benutzers) bereit. Der abschließende `text` zeigt an, dass es sich um eine `PostText`-API-Anforderung handelt (und nicht um `PostContent`).

- Anforderungsinhalt: Enthält die Benutzereingabe (`inputText`) und leere `sessionAttributes`. Wenn der Client die erste Anforderung stellt, gibt es keine Sitzungsattribute. Die Lambda-Funktion initiiert sie später.

b. Anhand des`inputText` erkennt Amazon Lex die Absicht (`OrderFlowers`). Diese Absicht wird mit einer Lambda-Funktion als Code-Hook für die Initialisierung und Validierung von Benutzerdaten konfiguriert. Daher ruft Amazon Lex diese Lambda-Funktion auf, indem es die folgenden Informationen als Ereignisdaten übergibt:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {},
    "bot": {
        "name": "OrderFlowers",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": null,
            "FlowerType": null,
            "PickupDate": null
        },
        "confirmationStatus": "None"
    }
}
```

Weitere Informationen finden Sie unter Eingabe-Ereignis-Format.

Zusätzlich zu den Informationen, die der Kunde gesendet hat, enthält Amazon Lex auch die folgenden zusätzlichen Daten:

- `messageVersion`— Derzeit unterstützt Amazon Lex nur die Version 1.0.
- `invocationSource`— Gibt den Zweck des Aufrufs einer Lambda-Funktion an. In diesem Fall ist es die Durchführung der Initialisierung und Validierung von Benutzerdaten. Derzeit weiß Amazon Lex, dass der Benutzer nicht alle Slot-Daten angegeben hat, um die Absicht zu erfüllen.

- `currentIntent`-Daten, wobei alle Slot-Werte auf Null gesetzt sind.

c. Zu diesem Zeitpunkt sind alle Slot-Werte auf Null gesetzt. Die Lambda-Funktion muss nichts validieren. Die Lambda-Funktion gibt die folgende Antwort an Amazon Lex zurück:

```
{
    "sessionAttributes": {},
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": null,
            "PickupDate": null
        }
    }
}
```

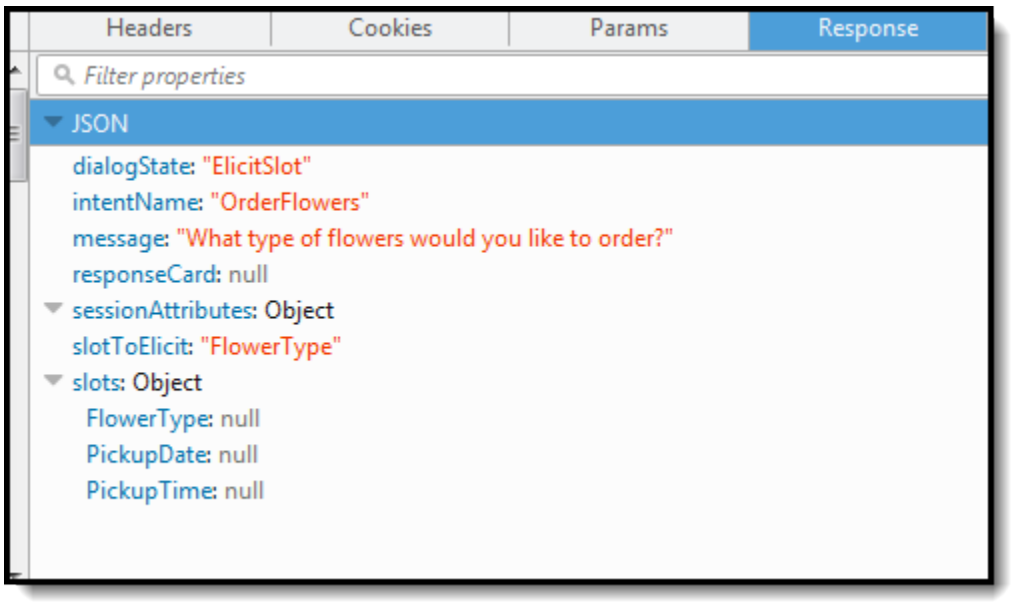
Weitere Informationen über das Antwortformat finden Sie unter Reaktion-Format.

Beachten Sie Folgendes:

- `dialogAction.type`— Wenn Sie diesen Wert auf festlegen`Delegate`, delegiert die Lambda-Funktion die Verantwortung für die Entscheidung über die nächste Vorgehensweise an Amazon Lex.

> **Note**
>
> Wenn die Lambda-Funktion bei der Benutzerdatenüberprüfung etwas feststellt, weist sie Amazon Lex an, was als Nächstes zu tun ist, wie in den nächsten Schritten gezeigt.

d. Laut dem`dialogAction.type` entscheidet Amazon Lex über die nächste Vorgehensweise. Da keiner der Slots ausgefüllt ist, entscheidet es, den Wert für den Slot `FlowerType` zu erfragen. Es wählt auch eine der Aufforderungen zur Angabe von Werten für diesen Slot aus ("Welche Art Blumen möchten Sie bestellen?") und sendet dann die folgende Antwort an den Client zurück:



Der Client zeigt die Mitteilung in der Antwort an.

2. Benutzer: Rosen

   a. Der Kunde sendet die folgendePostText Anfrage an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/ignw84y6seypre4xly5rimopuri2xwnd/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "roses",
    "sessionAttributes": {}
}
```

   Im Anforderungstext liefert der `inputText` die Benutzereingabe. Die `sessionAttributes` bleiben leer.

   b. Amazon Lex interpretiert das zunächst`inputText` im Kontext der aktuellen Absicht. Der Service "erinnert sich" daran, den spezifischen Benutzer nach Informationen über den Slot `FlowerType` gefragt zu haben. Es aktualisiert den Slot-Wert in der aktuellen Absicht und ruft die Lambda-Funktion mit den folgenden Ereignisdaten auf:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
```

```
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {},
    "bot": {
        "name": "OrderFlowers",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": null
        },
        "confirmationStatus": "None"
    }
}
```

Beachten Sie Folgendes:

- `invocationSource`: Ist immer noch `DialogCodeHook` (wir validieren einfach Benutzerdaten).
- `currentIntent.slots`— Amazon Lex hat den`FlowerType` Slot auf Rosen aktualisiert.

c. Entsprechend dem`invocationSource` Wert von`DialogCodeHook` führt die Lambda-Funktion eine Benutzerdatenvalidierung durch. Es wird`roses` als gültiger Slot-Wert erkannt (und`Price` als Sitzungsattribut festgelegt) und gibt die folgende Antwort an Amazon Lex zurück.

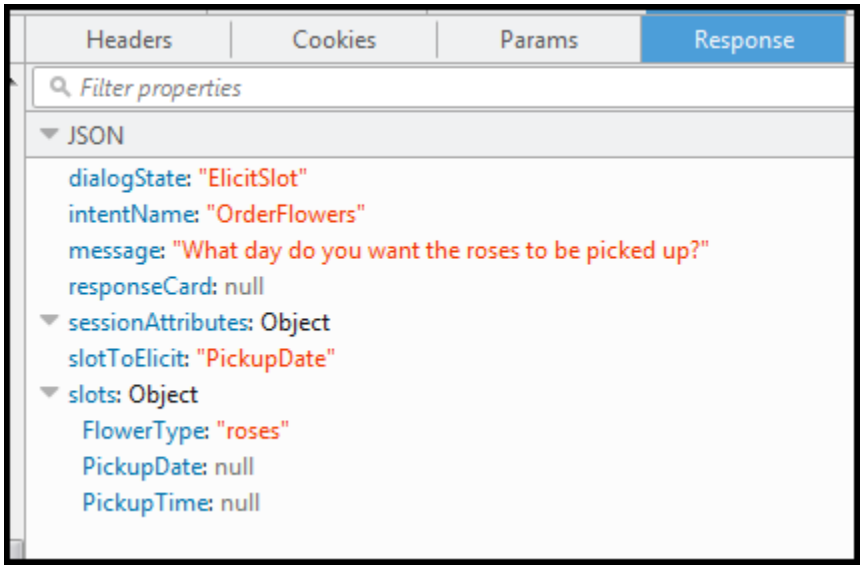
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": null
        }
```

```
    }
}
```

Beachten Sie Folgendes:

- `sessionAttributes`— Die Lambda-Funktion hat`Price` (von den Rosen) als Sitzungsattribut hinzugefügt.
- `dialogAction.type`: Wird auf `Delegate` gesetzt. Die Benutzerdaten waren gültig, sodass die Lambda-Funktion Amazon Lex anweist, die nächste Vorgehensweise zu wählen.

d. Laut dem`dialogAction.type` wählt Amazon Lex die nächste Vorgehensweise. Amazon Lex weiß, dass es mehr Steckplatzdaten benötigt, und wählt daher den nächsten unbesetzten Steckplatz (`PickupDate`) mit der höchsten Priorität entsprechend der gewünschten Konfiguration aus. Amazon Lex wählt eine der Aufforderungen zur Wertermittlung aus: „An welchem Tag sollen die Rosen abgeholt werden?“ —für diesen Slot gemäß der Intent-Konfiguration und sendet dann die folgende Antwort zurück an den Client:



Der Client zeigt einfach die Mitteilung in der Antwort an: "An welchem Tag möchten Sie die Rosen abholen?"

3. Benutzer: morgen

a. Der Kunde sendet die folgendePostText Anfrage an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/ignw84y6seypre4xly5rimopuri2xwnd/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "tomorrow",
    "sessionAttributes": {
        "Price": "25"
    }
}
```

Im Anforderungstext liefert `inputText` die Benutzereingabe und der Client gibt die Sitzungsattribute an den Service zurück.

b. Amazon Lex erinnert sich an den Kontext — dass es Daten für den`PickupDate` Slot abgerufen hat. In diesem Kontext weiß es, dass der Wert `inputText` für den Slot `PickupDate` ist. Amazon Lex ruft dann die Lambda-Funktion auf, indem es das folgende Ereignis sendet:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {
        "Price": "25"
    },
    "bot": {
        "name": "OrderFlowersCustomWithRespCard",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        },
        "confirmationStatus": "None"
    }
```

```
}
```

Amazon Lex hat das aktualisiert,`currentIntent.slots` indem es den`PickupDate` Wert festgelegt hat. Beachten Sie auch, dass der Dienst das`sessionAttributes` unverändert an die Lambda-Funktion übergibt.

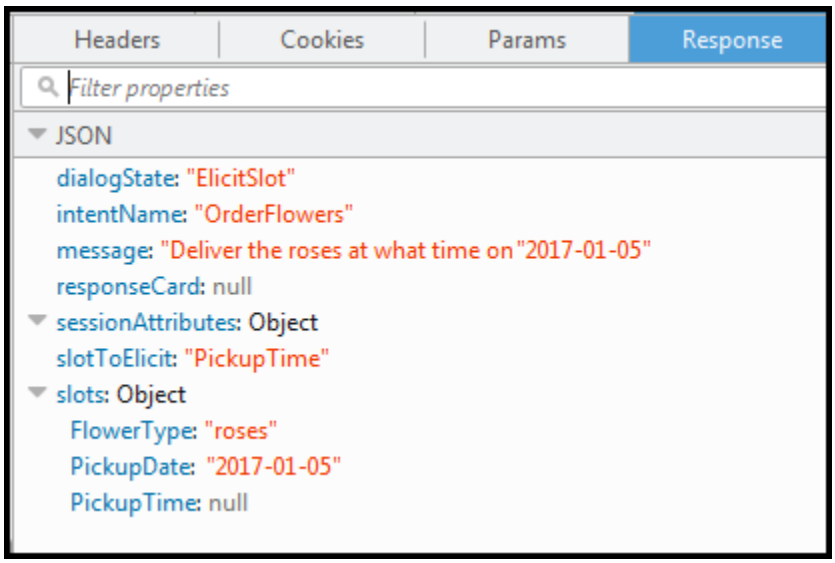
c. Gemäß dem`invocationSource` Wert von`DialogCodeHook` führt die Lambda-Funktion eine Benutzerdatenvalidierung durch. Es erkennt, dass der`PickupDate` Slot-Wert gültig ist, und gibt die folgende Antwort an Amazon Lex zurück:

```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        }
    }
}
```

Beachten Sie Folgendes:

- `sessionAttributes`: Keine Änderung.
- `dialogAction.type`: Wird auf `Delegate` gesetzt. Die Benutzerdaten waren gültig, und die Lambda-Funktion weist Amazon Lex an, die nächste Vorgehensweise zu wählen.

d. Laut dem`dialogAction.type` wählt Amazon Lex die nächste Vorgehensweise. Amazon Lex weiß, dass es mehr Steckplatzdaten benötigt, und wählt daher den nächsten unbesetzten Steckplatz (`PickupTime`) mit der höchsten Priorität entsprechend der gewünschten Konfiguration aus. Amazon Lex wählt eine der Eingabeaufforderungen aus („Liefern Sie die Rosen zu welcher Uhrzeit am 05.01.2017?“) für diesen Slot gemäß der Intent-Konfiguration und sendet die folgende Antwort zurück an den Client:



Der Kunde zeigt die Nachricht in der Antwort an: „Liefern Sie die Rosen zu welcher Uhrzeit am 05.01.2017?“

4. Benutzer: 16 Uhr

   a. Der Kunde sendet die folgendePostText Anfrage an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/ignw84y6seypre4xly5rimopuri2xwnd/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "4 pm",
    "sessionAttributes": {
        "Price": "25"
    }
}
```

   Im Anforderungstext liefert der `inputText` die Benutzereingabe. Der Client übergibt die `sessionAttributes` in der Anforderung.

   b. Amazon Lex versteht den Kontext. Es versteht, dass es Daten für den Slot `PickupTime` erfragt hat. In diesem Zusammenhang weiß es, dass der`inputText` Wert für den`PickupTime` Slot gilt. Amazon Lex ruft dann die Lambda-Funktion auf, indem es das folgende Ereignis sendet:

```
{
    "messageVersion": "1.0",
```

```
    "invocationSource": "DialogCodeHook",
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {
        "Price": "25"
    },
    "bot": {
        "name": "OrderFlowersCustomWithRespCard",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        },
        "confirmationStatus": "None"
    }
}
```

Amazon Lex hat das aktualisiert,`currentIntent.slots` indem es den`PickupTime` Wert festgelegt hat.

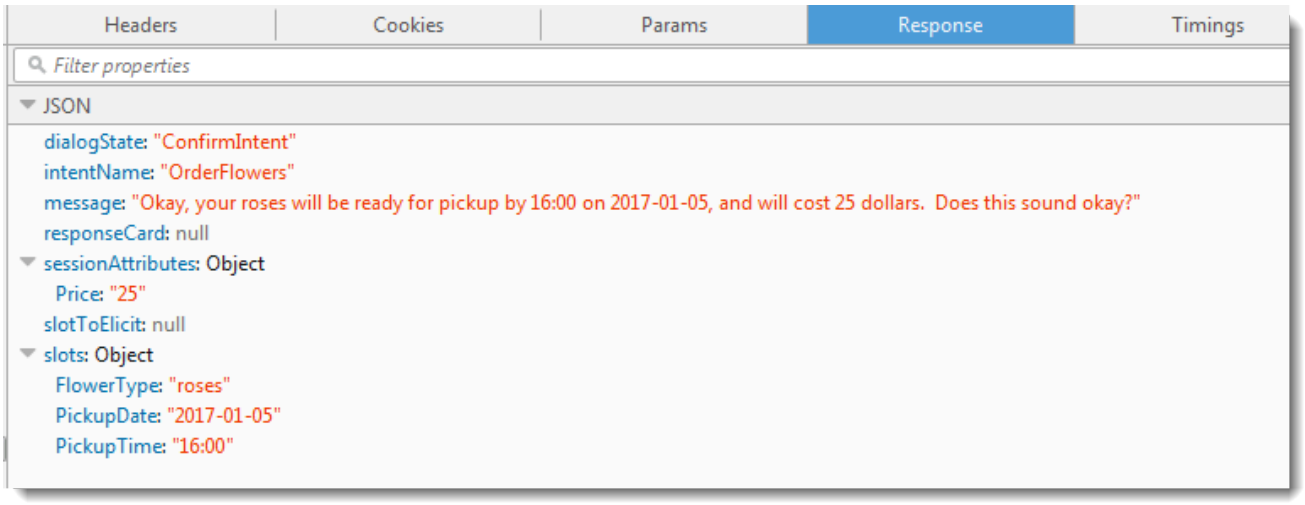
c. Entsprechend dem`invocationSource` Wert von`DialogCodeHook` führt die Lambda-Funktion eine Benutzerdatenvalidierung durch. Es erkennt, dass der`PickupDate` Slot-Wert gültig ist, und gibt die folgende Antwort an Amazon Lex zurück.

```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        }
    }
}
```

Beachten Sie Folgendes:

- `sessionAttributes`: Keine Änderung am Sitzungsattribut.
- `dialogAction.type`: Wird auf `Delegate` gesetzt. Die Benutzerdaten waren gültig, sodass die Lambda-Funktion Amazon Lex anweist, die nächste Vorgehensweise zu wählen.

d. Zu diesem Zeitpunkt weiß Amazon Lex, dass alle Slot-Daten vorliegen. Diese Absicht wurde mit einer Bestätigungsaufforderung konfiguriert. Daher sendet Amazon Lex die folgende Antwort an den Benutzer und bittet ihn um Bestätigung, bevor die Absicht erfüllt wird:



Der Client zeigt einfach die Mitteilung in der Antwort an und wartet auf die Antwort des Benutzers.

5. Benutzer: Ja

a. Der Kunde sendet die folgendePostText Anfrage an Amazon Lex:

```
POST /bot/OrderFlowers/alias/$LATEST/user/ignw84y6seypre4xly5rimopuri2xwnd/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "inputText": "yes",
    "sessionAttributes": {
        "Price": "25"
    }
}
```

b. Amazon Lex interpretiert das`inputText` im Kontext der Bestätigung der aktuellen Absicht. Amazon Lex geht davon aus, dass der Benutzer mit der Bestellung fortfahren möchte. Dieses Mal ruft Amazon Lex die Lambda-Funktion auf, um die Absicht zu erfüllen, indem das folgende Ereignis gesendet wird, das das An für das`invocationSource` Ereignis festlegt, das an die Lambda-Funktion gesendet wird.`FulfillmentCodeHook` Amazon Lex gibt auch die`confirmationStatus` Messlatte vor`Confirmed`.

```
{
    "messageVersion": "1.0",
    "invocationSource": "FulfillmentCodeHook",
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {
        "Price": "25"
    },
    "bot": {
        "name": "OrderFlowersCustomWithRespCard",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        },
        "confirmationStatus": "Confirmed"
    }
}
```

Beachten Sie Folgendes:

- `invocationSource`— Dieses Mal hat Amazon Lex diesen Wert auf gesetzt`FulfillmentCodeHook` und die Lambda-Funktion angewiesen, die Absicht zu erfüllen.
- `confirmationStatus`: Wird auf `Confirmed` gesetzt.

c. Dieses Mal erfüllt die Lambda-Funktion die`OrderFlowers` Absicht und gibt die folgende Antwort zurück:

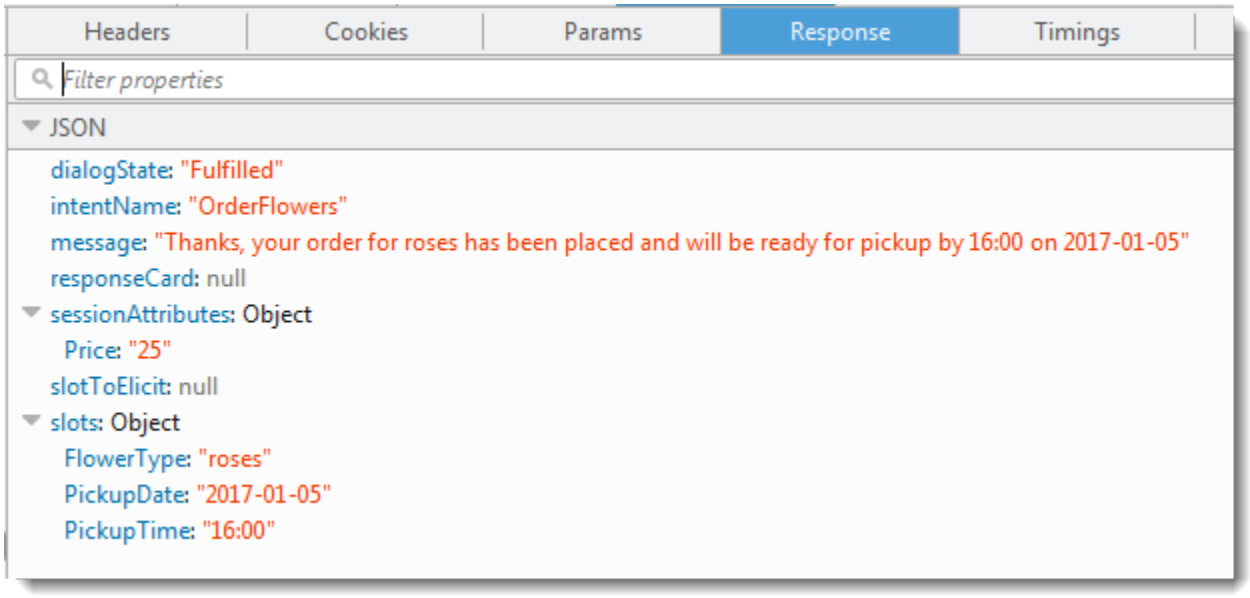
```
{
    "sessionAttributes": {
        "Price": "25"
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}
```

Beachten Sie Folgendes:

- Legt fest`dialogAction.type` — Die Lambda-Funktion legt diesen Wert auf fest`Close` und weist Amazon Lex an, keine Benutzerantwort zu erwarten.
- `dialogAction.fulfillmentState`: ist auf Fulfilled eingestellt und enthält eine geeignete `message` zur Übermittlung an den Benutzer.

d. Amazon Lex`fulfillmentState` überprüft Sie die folgende Antwort an den Kunden.

Amazon Lex gibt dann Folgendes an den Kunden zurück:



Beachten Sie:

- `dialogState`— Amazon Lex legt diesen Wert auf fest`fulfilled`.
- `message`— ist dieselbe Meldung, die die Lambda-Funktion geliefert hat.

Der Client zeigt die Mitteilung an.

6. Testen Sie nun den Bot erneut. Um einen neuen (Benutzer-)Kontext zu etablieren, wählen Sie den Link Clear im Testfenster aus. Geben Sie jetzt ungültige Slot-Daten für die Absicht `OrderFlowers` an. Dieses Mal führt die Lambda-Funktion die Datenüberprüfung durch, setzt den ungültigen Slot-Datenwert auf Null zurück und fordert Amazon Lex auf, den Benutzer zur Eingabe gültiger Daten aufzufordern. Versuchen Sie zum Beispiel Folgendes:

   - Jasmin als Blumenart (zählt nicht zu den unterstützten Blumenarten)
   - Gestern als den Tag, an dem Sie die Blumen abholen möchten
   - Nachdem Sie Ihre Bestellung aufgegeben haben, geben Sie eine andere Blumensorte ein, statt zur Bestätigung der Bestellung mit "Ja" zu antworten. Als Reaktion darauf aktualisiert die Lambda-Funktion das Attribut`Price` in the session, sodass die Gesamtzahl der Blumenbestellungen konstant bleibt.

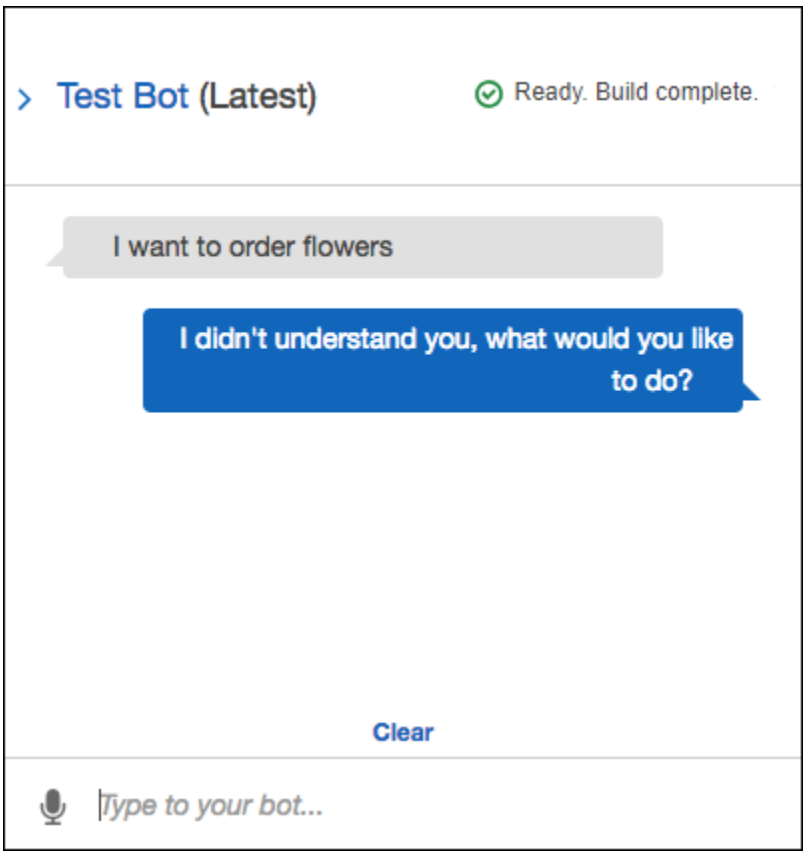
   Die Lambda-Funktion führt auch die Fulfillment-Aktivität aus.

Nächster Schritt

Schritt 6: Aktualisieren der Absichtskonfiguration zum Hinzufügen einer Äußerung (Konsole)

Schritt 6: Aktualisieren der Absichtskonfiguration zum Hinzufügen einer Äußerung (Konsole)

Der Bot `OrderFlowers` ist mit nur zwei Äußerungen konfiguriert. Dadurch stehen Amazon Lex begrenzte Informationen zur Verfügung, um ein Modell für maschinelles Lernen zu erstellen, das die Absicht des Benutzers erkennt und darauf reagiert. Versuchen Sie, wie im folgenden Testfenster „Ich möchte Blumen bestellen“ einzugeben. Amazon Lex erkennt den Text nicht und antwortet mit „Ich habe Sie nicht verstanden, was möchten Sie tun?“ Sie können das Modell für maschinelles Lernen durch Hinzufügen weiterer Äußerungen verbessern.

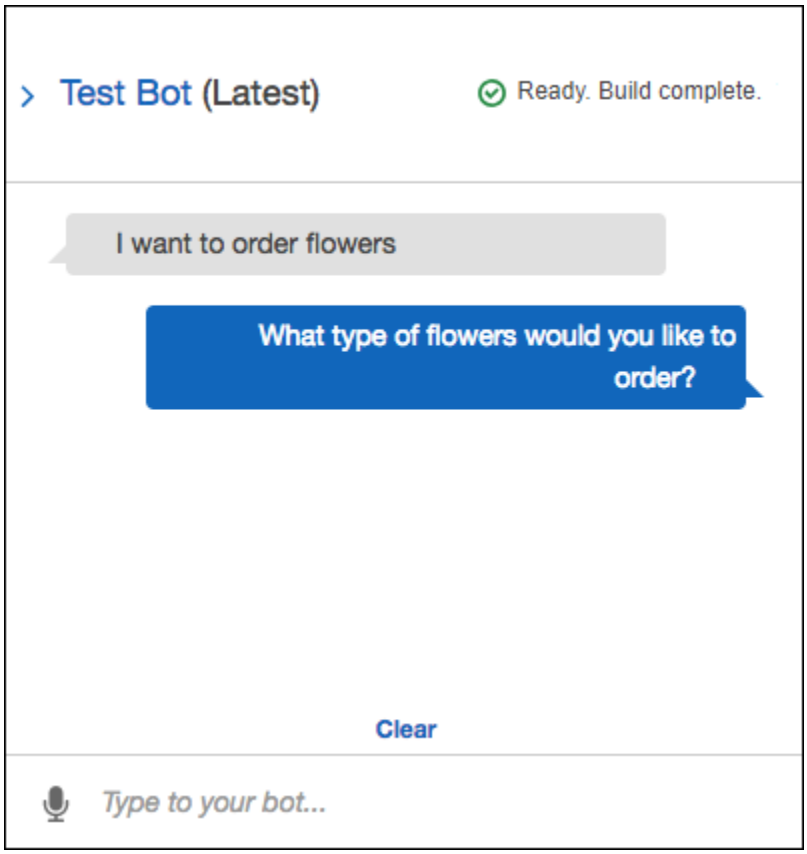


Jede Äußerung, die Sie hinzufügen, bietet Amazon Lex weitere Informationen darüber, wie Sie Ihren Benutzern antworten können. Sie müssen keine exakte Äußerung hinzufügen. Amazon Lex verallgemeinert anhand der von Ihnen bereitgestellten Beispiele, um sowohl exakte Übereinstimmungen als auch ähnliche Eingaben zu erkennen.

So fügen Sie eine Äußerung hinzu (Konsole)

1. Fügen Sie der Absicht die Äußerung „Ich möchte Blumen“ hinzu, indem Sie sie wie in der folgenden Abbildung in den Abschnitt Beispieläußerungen des Absichtseditors eingeben und dann auf das Plussymbol neben der neuen Äußerung klicken.

2. Erstellen Sie den Bot, um die Änderung zu übernehmen. Wählen Sie Build und dann erneut Build.
3. Testen Sie den Bot, um zu bestätigen, dass er die neue Äußerung erkennt. Geben Sie im Testfenster wie in der folgenden Abbildung „Ich möchte Blumen bestellen“ ein. Amazon Lex erkennt den Satz und antwortet mit „Welche Art von Blumen möchten Sie bestellen?“.



Nächster Schritt

Schritt 7 (optional): Bereinigen (Konsole)

Schritt 7 (optional): Bereinigen (Konsole)

Löschen Sie jetzt die Ressourcen, die Sie erstellt haben, und bereinigen Sie das Konto.

Sie können nur Ressourcen löschen, die nicht genutzt werden. Im Allgemeinen sollten Sie Ressourcen in der folgenden Reihenfolge löschen:

- Löschen Sie Bots, um Absicht-Ressourcen freizugeben.
- Löschen Sie Absichten, um Slot-Typ-Ressourcen freizugeben.

- Löschen Sie Slot-Typen zuletzt.

So bereinigen Sie Ihr Konto (Konsole)

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie Sie Sie Sie Sie Sie Amazon Lex Sie Sie Sie unter https://console.aws.amazon.com/lex/.
2. Aktivieren Sie in der Liste der Bots das Kontrollkästchen neben OrderFlowers.
3. Um den Bot zu löschen, wählen Sie Delete und dann im Bestätigungsdialogfeld Continue aus.
4. Wählen Sie im linken Bereich Intents aus.
5. Wählen Sie aus der Liste der Absichten OrderFlowersIntent aus.
6. Um die Absicht zu löschen, wählen Sie Delete und dann im Bestätigungsdialogfeld Continue aus.
7. Wählen Sie im linken Bereich Slot types aus.
8. Wählen Sie in der Liste der Slot-Typen Flowers aus.
9. Um den Slot-Typ zu löschen, wählen Sie Delete und dann im Bestätigungsdialogfeld Continue aus.

Sie haben alle Amazon Lex Lex-Ressourcen, die Sie erstellt haben, entfernt und Ihr Konto bereinigt. Falls gewünscht, können Sie die Lambda-Konsole verwenden, um die in dieser Übung verwendete Lambda-Funktion zu löschen.

Übung 2: Erstellen Sie einen benutzerdefinierten Amazon Lex Lex-Bot

In dieser Übung verwenden Sie die Amazon Lex-Konsole, um einen benutzerdefinierten Bot zu erstellen, der Pizza bestellt (`OrderPizzaBot`). Sie konfigurieren den Bot, indem Sie eine benutzerdefinierte Absicht (`OrderPizza`) hinzufügen, benutzerdefinierte Slot-Typen definieren und die Slots angeben, die zum Erfüllen der Pizzabestellung notwendig sind (Kruste, Größe der Pizza usw.). Weitere Informationen zu Steckplatzarten und Steckplätzen finden Sie unter Amazon Lex — Funktionsweise.

Themen

- Schritt 1: Erstellen einer Lambda-Funktion
- Schritt 2: Erstellen eines Bots
- Schritt 3: Erstellen und Testen des Bots
- Schritt 4 (Optional): Bereinigen

Schritt 1: Erstellen einer Lambda-Funktion

Erstellen Sie zunächst eine Lambda-Funktion, die eine Pizzabestellung erfüllt. Sie geben diese Funktion in Amazon Lex an, den Sie im nächsten Abschnitt erstellen.

So erstellen Sie eine Lambda-Funktion:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie Create function (Funktion erstellen).
3. Wählen Sie auf der Seite Create function die Option Author from scratch.

   Sie wählen die Option zum völligen Neuerstellen der Funktion aus, da Sie einen benutzerdefinierten Code verwenden, der in dieser Übung bereitgestellt wird, um eine Lambda-Funktion zu erstellen.

   Gehen Sie wie folgt vor:

   a. Geben Sie den Namen (`PizzaOrderProcessor`) ein.
   b. Wählen Sie für die Runtime (Laufzeit) die neueste Version von Node.js aus.
   c. Wählen Sie unter Role Create new role from template(s) aus.
   d. Geben Sie einen neuen Rollennamen ein (`PizzaOrderProcessorRole`).
   e. Wählen Sie Create function (Funktion erstellen).
4. Führen Sie auf der Seite function (Funktion konfigurieren) die folgenden Schritte aus:

   Wählen Sie im Abschnitt Function code die Option Edit code inline aus, kopieren Sie den folgenden Node.js-Funktionscode, und fügen Sie ihn in das Fenster ein.

```
'use strict';

// Close dialog with the customer, reporting fulfillmentState of Failed or
 Fulfilled ("Thanks, your pizza will arrive in 20 minutes")
function close(sessionAttributes, fulfillmentState, message) {
    return {
        sessionAttributes,
        dialogAction: {
            type: 'Close',
            fulfillmentState,
            message,
```

```
        },
    };
}

// --------------- Events -----------------------

function dispatch(intentRequest, callback) {
    console.log(`request received for userId=${intentRequest.userId}, intentName=
${intentRequest.currentIntent.name}`);
    const sessionAttributes = intentRequest.sessionAttributes;
    const slots = intentRequest.currentIntent.slots;
    const crust = slots.crust;
    const size = slots.size;
    const pizzaKind = slots.pizzaKind;

    callback(close(sessionAttributes, 'Fulfilled',
    {'contentType': 'PlainText', 'content': `Okay, I have ordered your ${size}
 ${pizzaKind} pizza on ${crust} crust`}));

}

// --------------- Main handler -----------------------

// Route the incoming request based on intent.
// The JSON body of the request is provided in the event slot.
export const handler = (event, context, callback) => {
    try {
        dispatch(event,
            (response) => {
                callback(null, response);
            });
    } catch (err) {
        callback(err);
    }
};
```

5. Wählen Sie Save (Speichern) aus.

Lambda-Funktion anhand von Beispielereignisationsdaten

Testen Sie die Lambda-Funktion in der Konsole, indem Sie sie anhand von Beispielereignisdaten manuell aufrufen.

Lambda-Funktion testen:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie auf der Lambda-Funktionsseite die Lambda-Funktion (`PizzaOrderProcessor`).
3. Wählen Sie auf der Funktionsseite in der Liste der Testereignisse die Option Configure test events aus.
4. Führen Sie auf der Seite Configure test event die folgenden Schritte aus:

   a. Wählen Sie Create new test event aus.

   b. Geben Sie im Feld Event name einen Namen für das Ereignis ein (`PizzaOrderProcessorTest`).

   c. Kopieren Sie das folgende Amazon Lex Lex-Ereignis in das Fenster.

```
{
  "messageVersion": "1.0",
  "invocationSource": "FulfillmentCodeHook",
  "userId": "user-1",
  "sessionAttributes": {},
  "bot": {
    "name": "PizzaOrderingApp",
    "alias": "$LATEST",
    "version": "$LATEST"
  },
  "outputDialogMode": "Text",
  "currentIntent": {
    "name": "OrderPizza",
    "slots": {
      "size": "large",
      "pizzaKind": "meat",
      "crust": "thin"
    },
    "confirmationStatus": "None"
  }
}
```

5. Wählen Sie Create (Erstellen) aus.

AWS Lambda erstellt den Test und Sie gelangen zur Funktionsseite zurück. Wählen Sie Test und Lambda führt Ihre Lambda-Funktion aus.

Wählen Sie im Ergebnisfeld die Option Details aus. In der Konsole wird die folgende Ausgabe im Bereich Execution result angezeigt.

```
{
  "sessionAttributes": {},
  "dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled",
    "message": {
      "contentType": "PlainText",
      "content": "Okay, I have ordered your large meat pizza on thin crust."
    }
}
```

Nächster Schritt

Schritt 2: Erstellen eines Bots

Schritt 2: Erstellen eines Bots

In diesem Schritt erstellen Sie einen Bot zur Bearbeitung von Pizzabestellungen.

Themen

- Erstellen des Bots
- Erstellen einer Absicht
- Erstellen von Slot-Typen
- Konfigurieren der Absicht
- Konfigurieren des -Bots

Erstellen des Bots

Erstellen Sie den Bot `PizzaOrderingBot` mit den minimal erforderlichen Informationen. Später fügen Sie dem Bot eine Absicht und eine Aktion, die der Benutzer ausführen möchte, hinzu.

So erstellen Sie den Bot

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Erstellen eines Bots.

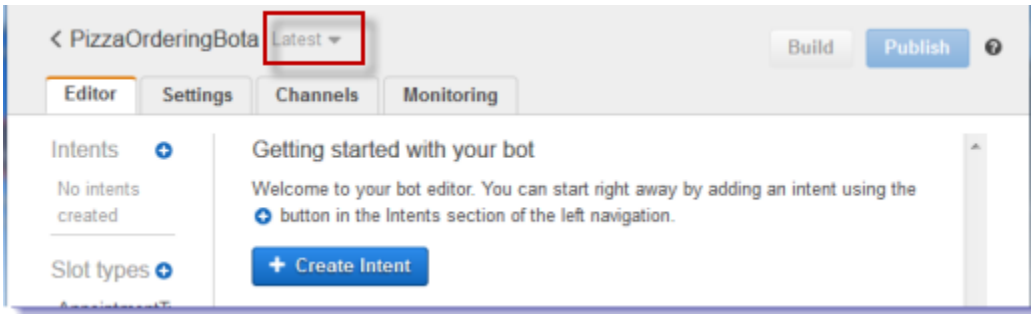
a. Wenn Sie Ihren ersten Bot erstellen, wählen Sie Get Started aus. Wählen Sie andernfalls Bots und dann Create aus.

b. Wählen Sie auf der Seite Create your Lex bot die Option Custom bot aus und stellen Sie die folgenden Informationen bereit:

   - Name des Bots: PizzaOrderingBot
   - Sprache: Wähle die Sprache und das Gebietsschema für deinen Bot.
   - Output voice: Salli
   - Session timeout : 5 Minuten.
   - COPPA: Wählen Sie die entsprechende Antwort.
   - Speicherung von Benutzeräußerungen: Wählen Sie die entsprechende Antwort aus.

c. Wählen Sie Create (Erstellen) aus.

   Die Konsole sendet an Amazon Lex eine Anfrage zum Erstellen eines neuen Bots. Amazon Lex legt die Bot-Version auf fest$LATEST. Nach dem Erstellen des Bots zeigt Amazon Lex die Registerkarte „Bot-Editor“, wie in der folgenden Abbildung dargestellt:



   - Neben dem Bot-Namen in der Konsole ist die Latest Version des Bots zu sehen. Neue Amazon Lex Lex-Ressourcen haben$LATEST als Version. Weitere Informationen finden Sie unter Versioning und Aliasnamen.
   - Da Sie keine Absichten oder Slot-Typen erstellt haben, werden auch keine aufgeführt.
   - Build und Publish sind Aktivitäten auf Bot-Ebene. Nach der Konfiguration des gesamten Bots erfahren Sie mehr über diese Aktivitäten.

Nächster Schritt

Erstellen einer Absicht

Erstellen einer Absicht

Erstellen Sie nun die Absicht `OrderPizza`, eine Aktion, die der Benutzer durchführen möchte, sowie die Informationen, die mindestens erforderlich sind. Sie fügen Slot-Typen zur Absicht hinzu und konfigurieren die Absicht später.

So erstellen Sie eine Absicht

1. Wählen Sie in der Amazon Lex-Konsole das Pluszeichen (+) neben Intents und wählen Sie dann Create New Intent.
2. Geben Sie im Dialogfeld Create intent den Namen der Absicht ein (`OrderPizza`) und wählen Sie dann Add aus.

Die Konsole sendet eine Anfrage an Amazon Lex, um die`OrderPizza` Absicht zu erstellen. In diesem Beispiel erstellen Sie Slots für die Absicht, nachdem Sie Slot-Typen erstellt haben.

Nächster Schritt

Erstellen von Slot-Typen

Erstellen von Slot-Typen

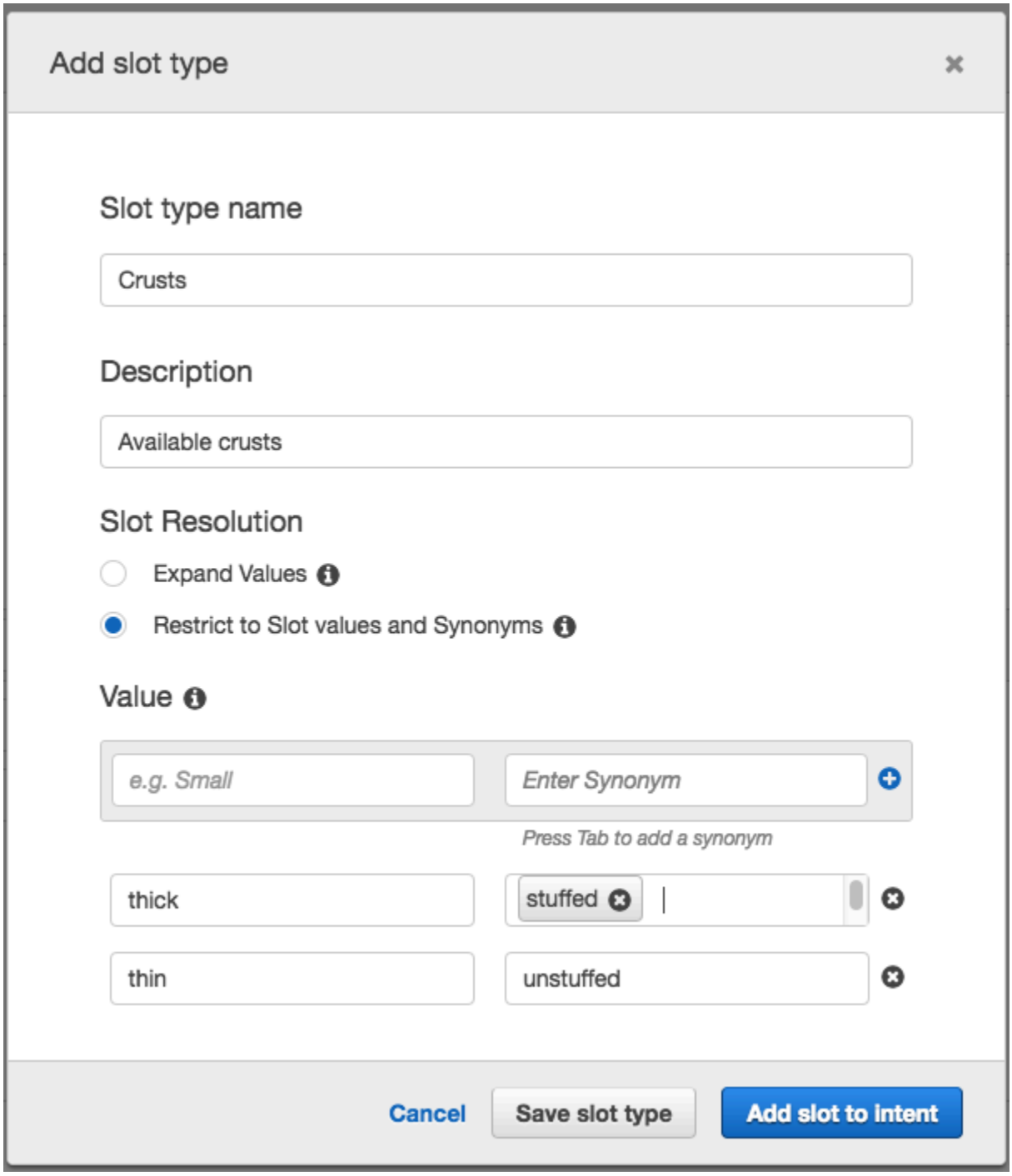
Erstellen Sie die Slot-Typen oder Parameterwerte, die die Absicht `OrderPizza` verwendet.

So erstellen Sie Slot-Typen

1. Wählen Sie im linken Menü das Pluszeichen (+) neben Slot types aus.
2. Fügen Sie im Dialogfeld Add slot type Folgendes hinzu:

   - Slot type name – Krusten
   - Description – Verfügbare Krusten
   - Wählen Sie Restrict to Slot values and Synonyms aus.
   - Wert — Typ**`thick`**. Drücken Sie die Tabulatortaste und geben Sie in das Feld Synonym **`stuffed`** ein. Klicken Sie auf das Pluszeichen (+). Geben Sie **`thin`** ein und klicken Sie erneut auf das Pluszeichen (+).

   Das Dialogfeld sollte wie das folgende Bild aussehen:



3. Wählen Sie Add slot to intent aus.
4. Wählen Sie auf der Seite Intent Required aus. Ändern Sie den Namen des Slots von **`slotOne`** in **`crust`**. Ändern Sie die Eingabeaufforderung in **`What kind of crust would you like?`**.
5. Wiederholen Sie Step 1 bis Step 4 mittels der Werte der folgenden Tabelle:

Name	Beschreibung	Werte	Slot-Name	Telefonansage
Größen	Verfügbare Größen	klein, mittel, groß	size	Welche Größe Pizza?
PizzaKind	Verfügbare Pizzen	veg, Käse	pizzaKind	Möchten Sie eine Pizza mit Käse oder mit Gemüse?

Nächster Schritt

Konfigurieren der Absicht

Konfigurieren der Absicht

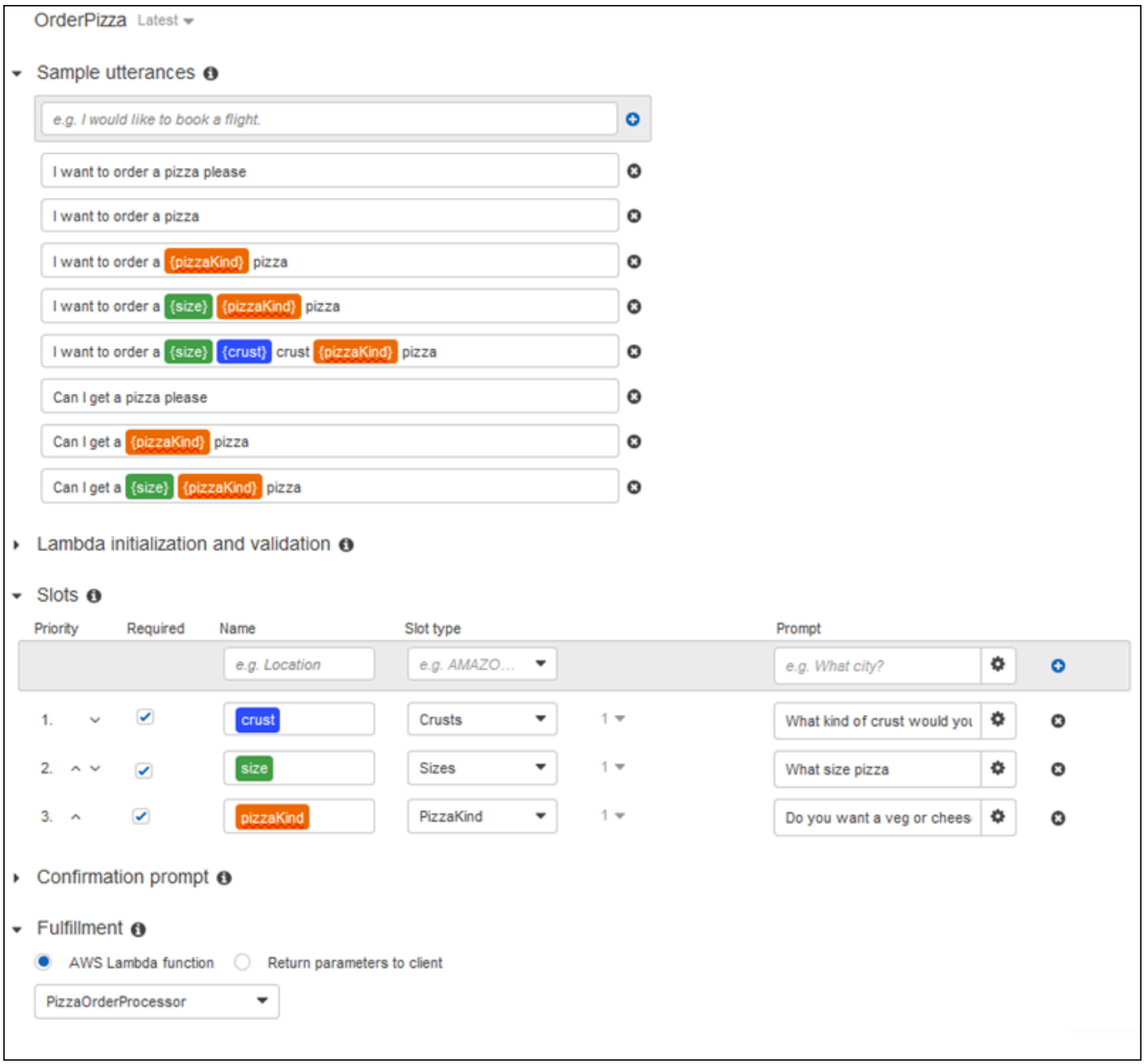
Konfigurieren Sie die Absicht `OrderPizza`, um die Anforderung des Benutzers, Pizza zu bestellen, zu erfüllen.

So konfigurieren Sie eine Absicht

- Konfigurieren Sie auf der OrderPizzaKonfigurationsseite die Absicht wie folgt:
  - Beispiel für Äußerungen — Geben Sie die folgenden Zeichenfolgen ein. Die Slot-Namen stehen in geschweiften Klammern {}.
    - Ich möchte Pizza bestellen, bitte
    - Ich möchte eine Pizza bestellen
    - Ich möchte eine {Pizza Art} Pizza bestellen
    - Ich möchte eine Pizza {Pizza Art} bestellen
    - Ich möchte eine {Pizza Art} Pizza mit Kruste.
    - Kann ich eine Pizza haben, bitte
    - Kann ich eine {Pizza Art} Pizza haben
    - Kann ich eine {Pizza Art} Pizza haben
  - Lambda initialization and validation – Ändern Sie die Standardeinstellung nicht.
  - Confirmation prompt – Ändern Sie die Standardeinstellung nicht.

- Erfüllung — die folgenden Aufgaben ausführen:
  - Wählen Sie AWS Lambda Funktion.
  - Wählen Sie **`PizzaOrderProcessor`**.
  - Wenn das Dialogfeld „Berechtigung zur Lambda-Funktion hinzufügen“ angezeigt wird, wählen Sie OK, um der`OrderPizza` Absicht die Berechtigung zum Aufrufen der`PizzaOrderProcessor` Lambda-Funktion zu erteilen.
  - Lassen Sie None ausgewählt.

Die Absicht sollte wie folgt aussehen:



Nächster Schritt

[Konfigurieren des -Bots]

Konfigurieren des -Bots

Konfigurieren Sie die Fehlerbehandlung für den Bot `PizzaOrderingBot`.

1. Navigieren Sie zum Bot `PizzaOrderingBot`. Wählen Sie „Editor“ und dann „Fehlerbehandlung“, wie in der folgenden Abbildung dargestellt:

2. Verwenden Sie die Registerkarte Editor zur Konfiguration der Fehlerbehandlung.

   - Informationen, die Sie in den Clarification Prompts (Klärungsaufforderungen) angeben, werden der Bot-Konfiguration clarificationPrompt zugeordnet.

     Wenn Amazon Lex die Absicht des Benutzers nicht ermitteln kann, gibt der Service eine Antwort mit dieser Meldung zurück.
   - Informationen, die Sie in der Hang-up (Formulierung zum Auflegen)-Phrase bereitstellen, werden der Bot-Konfiguration abortStatement zugeordnet.

     Wenn der Service die Absicht des Benutzers nach einer bestimmten Anzahl aufeinanderfolgender Anfragen nicht ermitteln kann, gibt Amazon Lex eine Antwort mit dieser Meldung zurück.

   Lassen Sie die übrigen Standardwerte.

Nächster Schritt

Schritt 3: Erstellen und Testen des Bots

Schritt 3: Erstellen und Testen des Bots

Stellen Sie sicher, dass der Bot funktioniert, indem Sie diesen erstellen und testen.

So erstellen und testen Sie den Bot

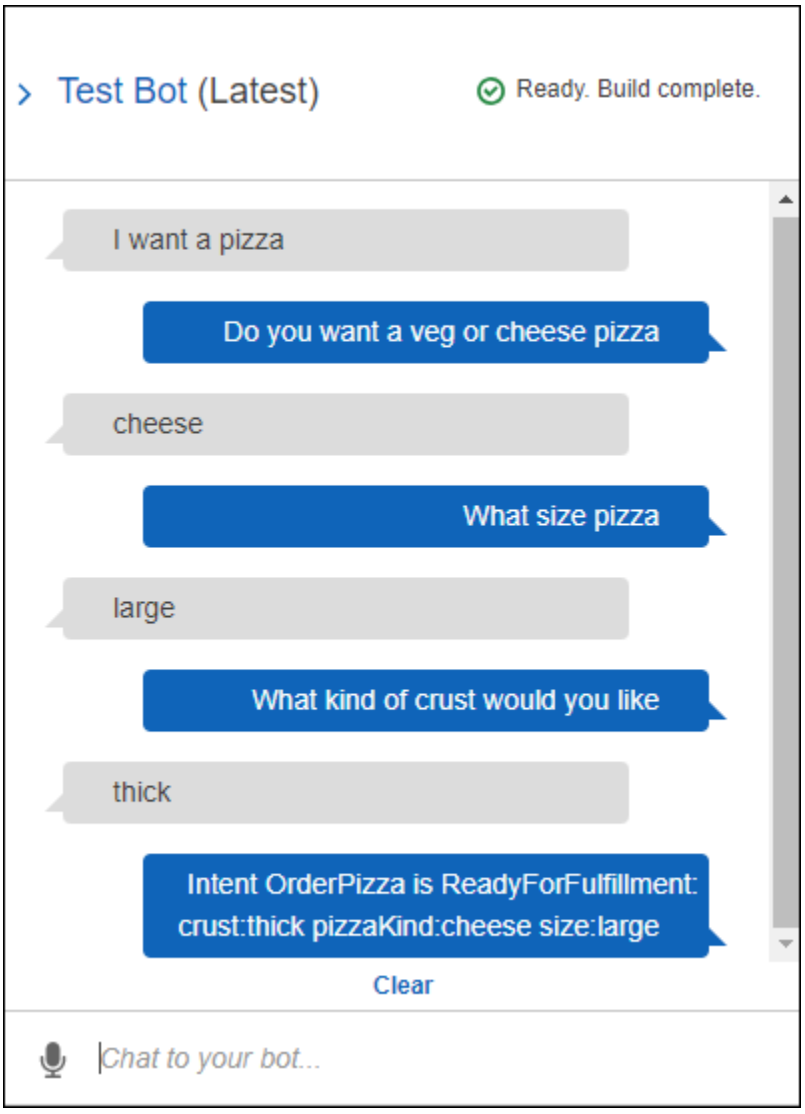
1. Wählen Sie zum Erstellen des Bots `PizzaOrderingBot` die Option Build aus.

   Amazon Lex erstellt ein Modell für maschinelles Lernen für den Bot. Wenn Sie den Bot testen, verwendet die Konsole die Runtime-API, um die Benutzereingaben zurück an Amazon Lex zu senden. Amazon Lex verwendet dann das Modell des maschinellen Lernens, um die Benutzereingabe zu interpretieren.

   Die Erstellung kann einige Zeit in Anspruch nehmen.

2. Um den Bot zu testen, beginnen Sie im Fenster Test Bot mit Ihrem Amazon Lex Lex-Bot zu kommunizieren.

   - Sie könnten beispielsweise das Folgende sagen oder eingeben:



- Nutzen Sie zum Testen des Bots die Beispiel-Äußerungen, die Sie in der Absicht `OrderPizza` konfiguriert haben. Die folgende Äußerung ist beispielsweise eine Äußerung, die Sie für die Absicht `PizzaOrder` konfiguriert haben:

  ```
  I want a {size} {crust} crust {pizzaKind} pizza
  ```

  Geben Sie zum Testen Folgendes ein:

  ```
  I want a large thin crust cheese pizza
  ```

  Wenn Sie „Ich möchte eine Pizza bestellen“ eingeben, erkennt Amazon Lex die Absicht (`OrderPizza`). Dann fragt Amazon Lex nach Slot-Informationen.

  Nachdem Sie alle Slot-Informationen angegeben haben, ruft Amazon Lex die Lambda-Funktion auf, die Sie für die Absicht konfiguriert haben.

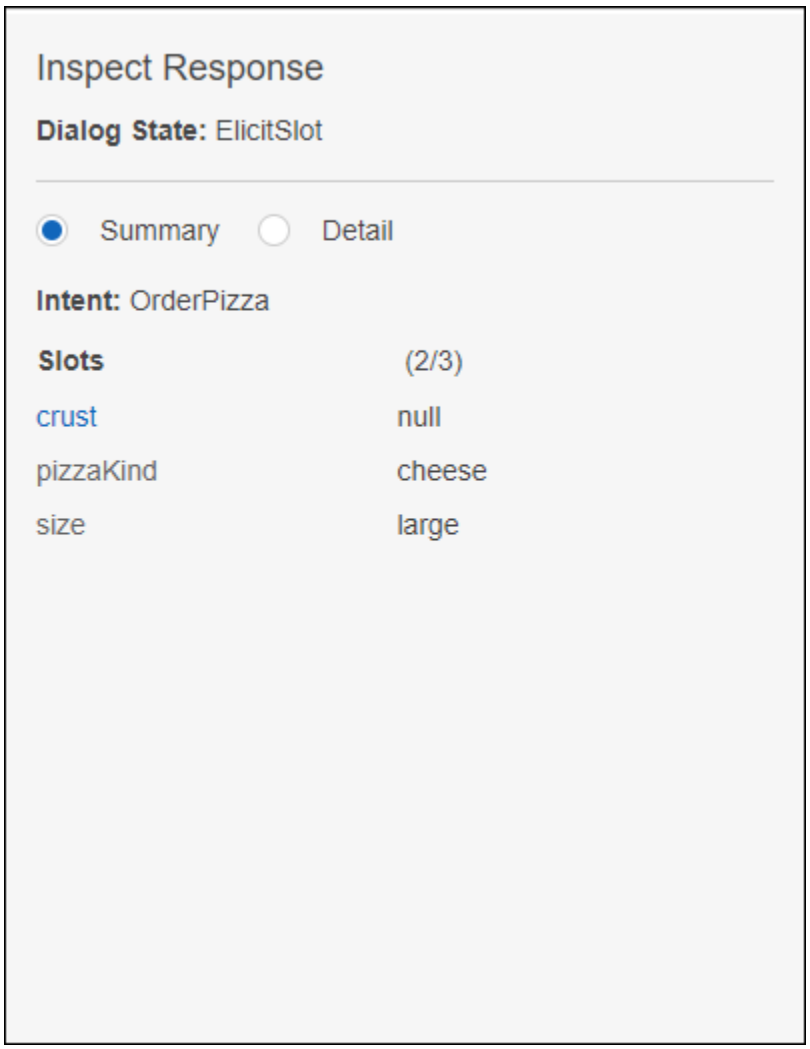
  Die Lambda-Funktion gibt eine Nachricht („Okay, ich habe Ihre bestellt...“) an Amazon Lex zurück, die Amazon Lex an Sie zurücksendet.

Untersuchen der Antwort

Unter dem Chatfenster befindet sich ein Bereich, in dem Sie die Antwort von Amazon Lex überprüfen können. Der Bereich enthält umfassende Informationen über den Status Ihres Bots, der sich ändert, wenn Sie mit dem Bot interagieren. Der Inhalt der Fenster zeigt Ihnen den aktuellen Zustand des Vorgangs.

- Dialogstatus — Der aktuelle Status der Konversation mit dem Benutzer. `ElicitIntent`, `ElicitSlot`, `ConfirmIntent` oder `Fulfilled` sind möglich.

- Zusammenfassung — Zeigt eine vereinfachte Ansicht des Dialogs an, in der die Slot-Werte für die zu erfüllende Absicht angezeigt werden, sodass Sie den Informationsfluss verfolgen können. Es werden der Name der Absicht, die Anzahl der Slots und die Anzahl der gefüllten Slots gezeigt sowie eine Liste aller Slots einschließlich zugehöriger Werte. Sehen Sie das folgende Bild:



- Detail — Zeigt die rohe JSON-Antwort des Chatbots an, um Ihnen einen tieferen Einblick in die Bot-Interaktion und den aktuellen Status des Dialogs zu geben, während Sie Ihren Chatbot testen und debuggen. Wenn Sie etwas in das Chat-Fenster eingeben, zeigt der Prüfbereich die JSON-Antwort der PostText-Operation. Wenn Sie etwas sagen, zeigt der Prüfbereich die Antwort-Header der PostContent-Operation an. Sehen Sie das folgende Bild:

Nächster Schritt

Schritt 4 (Optional): Bereinigen

Schritt 4 (Optional): Bereinigen

Löschen Sie die Ressourcen, die Sie erstellt haben, und bereinigen Sie Ihr Konto, um weitere Kosten für die von Ihnen erstellten Ressourcen zu vermeiden.

Sie können nur Ressourcen löschen, die nicht genutzt werden. Sie können beispielsweise keinen Slot-Typ löschen, auf den von einer Absicht verwiesen wird. Sie können keine Absicht löschen, auf die von einem Bot verwiesen wird.

Löschen Sie Ressourcen in der folgenden Reihenfolge:

- Löschen Sie Bots, um Absicht-Ressourcen freizugeben.

- Löschen Sie Absichten, um Slot-Typ-Ressourcen freizugeben.
- Löschen Sie Slot-Typen zuletzt.

So bereinigen Sie Ihr Konto

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Wählen Sie aus der Liste der Bots PizzaOrderingBot aus.
3. Wählen Sie zum Löschen des Bots Delete und dann Continue aus.
4. Wählen Sie im linken Bereich Intents aus.
5. Wählen Sie aus der Liste der Absichten OrderPizza aus.
6. Wählen Sie zum Löschen der Absicht Delete und dann Continue aus.
7. Wählen Sie im linken Menü Slot types aus.
8. Wählen Sie aus der Liste der Slot-Typen Crusts aus.
9. Wählen Sie zum Löschen des Slot-Typs Delete und dann Continue aus.
10. Wiederholen Sie Step 8 und Step 9 für die Slot-Typen `Sizes` und `PizzaKind`.

Sie haben alle Ressourcen, die Sie erstellt haben, entfernt und Ihr Konto bereinigt.

Nächste Schritte

- Veröffentlichen einer Version und Erstellen eines Aliasnamens
- Erstellen Sie einen Amazon Lex Lex-Bot mit demAWS Command Line Interface

Übung 3: Eine Version veröffentlichen und einen Aliasnamen generieren

In den Einstiegsübungen 1 und 2 haben Sie einen Bot erstellt und getestet. In dieser Übung führen Sie folgende Aufgaben aus:

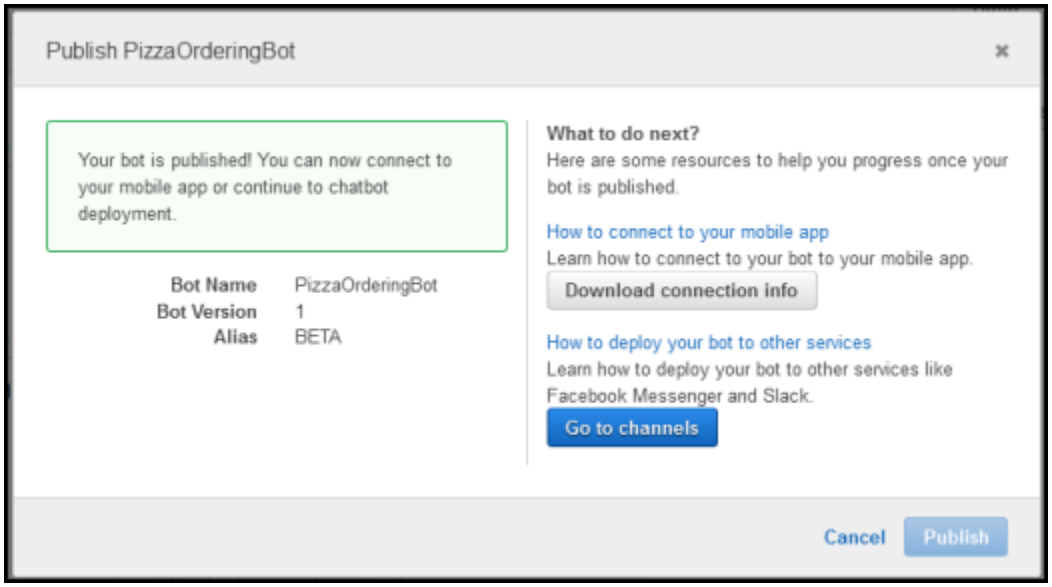
- Veröffentlichen Sie eine neue Version des Bots. Amazon Lex erstellt eine Snapshot-Kopie der`$LATEST` Version, um eine neue Version zu veröffentlichen.
- Erstellen Sie einen Alias, der auf die neue Version zeigt.

Weitere Informationen zu Versioning und Aliasnamen finden Sie unter Versioning und Aliasnamen.

Führen Sie die folgenden Schritte aus, um eine Version eines Bots, den Sie für diese Übung erstellt haben, zu veröffentlichen:

1. Wählen Sie in der Amazon Lex Lex-Konsole einen der Bots aus, die Sie erstellt haben.

   Überprüfen Sie, ob die Konsole `$LATEST` als Bot-Version neben dem Bot-Namen anzeigt.
2. Wählen Sie Publish.
3. Geben Sie im Assistenten Publish ***botname (Bot-Name veröffentlichen)*** den Aliasnamen **`BETA`** ein, und wählen Sie dann Publish (Veröffentlichen).
4. Stellen Sie sicher, dass in der Amazon Lex Lex-Konsole die neue Version neben dem Bot-Namen angezeigt wird, wie in der folgenden Abbildung dargestellt.



Nachdem Sie jetzt einen funktionierenden Bot mit veröffentlichter Version und Aliasnamen haben, können Sie den Bot in Ihrer mobilen Anwendung bereitstellen oder mit Facebook Messenger integrieren. Ein Beispiel finden Sie unter Integration eines Amazon Lex Lex-Bot mit Facebook Messenger.

Schritt 4: Erste Schritte (AWS CLI)

In diesem Schritt verwenden Sie dieAWS CLIum einen Amazon Lex -Bot zu erstellen, zu testen und zu modifizieren. Um diese Übungen abschließen zu können, benötigen Sie Kenntnisse zur Verwendung der Befehlszeile und einen Texteditor. Weitere Informationen finden Sie unterSchritt 2: Richten Sie das ein AWS Command Line Interface

- Übung 1 — Erstellen und Testen eines Amazon Lex -Bots. Die Übung umfasst alle JSON-Objekte, die Sie zur Erstellung eines benutzerdefinierten Slot-Typs, einer Absicht und eines Bots benötigen. Weitere Informationen finden Sie unterAmazon Lex — Funktionsweise
- Übung 2: Aktualisieren des in Übung 1 erstellten Bots, um eine weitere Beispieläußerung hinzuzufügen. Amazon Lex verwendet Beispieläußerungen zum Erstellen des Modells für maschinelles Lernen für den Bot.
- Übung 3: Aktualisieren des in Übung 1 erstellten Bots, um eine Lambda-Funktion zum Validieren der Benutzereingabe und zum Erfüllen der Absicht hinzuzufügen.
- Übung 4: Veröffentlichen einer Version von Slot-Typ-, Absicht- und Bot-Ressourcen, die in Übung 1 erstellt wurden. Eine Version ist ein Snapshot einer Ressource und kann nicht geändert.
- Übung 5: Erstellen eines Alias für den in Übung 1 erstellten Bot.
- Übung 6: Bereinigen des Kontos durch Löschen des Slot-Typs, der Absicht und des Bots, die in Übung 1 erstellt wurden, sowie des in Übung 5 erstellten Alias.

Themen

Übung 1: Erstellen eines Amazon Lex Bots (AWS CLI)

Grundsätzlich gilt beim Erstellen von Bots:

1. Erstellen Sie Slot-Typen, um die Daten zu definieren, mit denen der Bot dann arbeitet.
2. Erstellen Sie Absichten, die die vom Bot unterstützten Benutzeraktionen definieren. Verwenden Sie die zuvor erstellten benutzerdefinierten Slot-Typen (Parameter), die für die Absicht benötigt werden.
3. Erstellen Sie einen Bot, der die von Ihnen definierten Absichten verwendet.

In dieser Übung erstellen und testen Sie mit der CLI einen neuen Amazon Lex Lex-Bot. Verwenden Sie die JSON-Strukturen, die wir zum Erstellen des Bots bereitstellen. Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

Themen

Schritt 1: Erstellen einer serviceverknüpften Rolle (AWS CLI)

Amazon Lex geht davon ausAWS Identity and Access ManagementAufzurufende serviceverknüpfte RollenAWS-Dienste im Namen Ihrer Bots. Die Rollen im Konto sind mit Amazon-Lex-Anwendungsfällen verknüpft und haben vordefinierte Berechtigungen. Weitere Informationen finden Sie unter Verwendung von serviceverknüpfte Rollen für Amazon Lex .

Wenn Sie bereits mit der Konsole einen Amazon Lex Lex-Bot erstellt haben, wurde die serviceverknüpfte Rolle automatisch erstellt. Fahren Sie mit fort Schritt 2: Erstellen eines benutzerdefinierten Slot-Typs (AWS CLI).

So erstellen Sie eine serviceverknüpfte Rolle (AWS CLI)

1. Geben Sie in der AWS CLI folgenden Befehl ein:

   ```
   aws iam create-service-linked-role --aws-service-name lex.amazonaws.com
   ```

2. Prüfen Sie die Richtlinie mit dem folgenden Befehl:

   ```
   aws iam get-role --role-name AWSServiceRoleForLexBots
   ```

   Die Antwort lautet:

   ```
   {
       "Role": {
           "AssumeRolePolicyDocument": {
   ```

```
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lex.amazonaws.com"
                    }
                }
            ]
        },
        "RoleName": "AWSServiceRoleForLexBots",
        "Path": "/aws-service-role/lex.amazonaws.com/",
        "Arn": "arn:aws:iam::account-id:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
}
```

Nächster Schritt

Schritt 2: Erstellen eines benutzerdefinierten Slot-Typs (AWS CLI)

Schritt 2: Erstellen eines benutzerdefinierten Slot-Typs (AWS CLI)

Erstellen Sie einen benutzerdefinierten Slot-Typ mit Enumerationswerten für die Blumen, die bestellt werden können. Dieser Typ wird im nächsten Schritt verwendet, wenn Sie die `OrderFlowers`-Absicht erstellen. Ein Slot-Typ definiert die möglichen Werte für einen Slot (Parameter) der Absicht.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

So erstellen Sie einen benutzerdefinierten Slot-Typ (AWS CLI)

1. Erstellen Sie eine Textdatei mit dem Namen **`FlowerTypes.json`**. Kopieren Sie den JSON-Code aus FlowerTypes.json in die Textdatei.
2. Rufen Sie über die AWS CLI die PutSlotType-Operation zum Erstellen des Slot-Typs auf. Das Beispiel ist für Unix, Linux und macOS formatiert. Ersetzen Sie unter Windows den umgekehrten Schrägstrich (\), das Unix-Fortsetzungszeichen, am Ende jeder Zeile durch ein Caret-Zeichen oder Zirkumflex (^).

```
aws lex-models put-slot-type \
    --region region \
```

```
    --name FlowerTypes \
    --cli-input-json file://FlowerTypes.json
```

Die Antwort des Servers:

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "checksum": "checksum",
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Nächster Schritt

Schritt 3: Erstellen einer Absicht (AWS CLI)

FlowerTypes.json

Der folgende Code repräsentiert die JSON-Daten, die zum Erstellen des benutzerdefinierten Slot-Typs `FlowerTypes` benötigt werden:

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
```

```
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "description": "Types of flowers to pick up"
}
```

Schritt 3: Erstellen einer Absicht (AWS CLI)

Erstellen Sie eine Absicht für den Bot `OrderFlowersBot` und übergeben Sie drei Slots (Parameter). Mit den Slots kann der Bot die Absicht erfüllen:

- `FlowerType` ist ein benutzerdefinierter Slot-Typ, der angibt, welche Blumensorten bestellt werden können.
- `AMAZON.DATE` und `AMAZON.TIME` sind integrierte Slot-Typen zum Abrufen von Datum und Zeit der Blumenlieferung vom Benutzer.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

So erstellen Sie die **`OrderFlowers`**-Absicht (AWS CLI)

1. Erstellen Sie eine Textdatei mit dem Namen **`OrderFlowers.json`**. Kopieren Sie den JSON-Code aus OrderFlowers.json in die Textdatei.
2. Rufen Sie in der AWS CLI die Operation PutIntent zum Erstellen der Absicht auf. Das Beispiel ist für Unix, Linux und macOS formatiert. Ersetzen Sie unter Windows den umgekehrten Schrägstrich (\), das Unix-Fortsetzungszeichen, am Ende jeder Zeile durch ein Caret-Zeichen oder Zirkumflex (^).

   ```
   aws lex-models put-intent \
      --region region \
      --name OrderFlowers \
      --cli-input-json file://OrderFlowers.json
   ```

   Der Server antwortet wie folgt:

   ```
   {
   ```

```
   "confirmationPrompt": {
       "maxAttempts": 2,
       "messages": [
           {
               "content": "Okay, your {FlowerType} will be ready for pickup by
{PickupTime} on {PickupDate}.  Does this sound okay?",
               "contentType": "PlainText"
           }
       ]
   },
   "name": "OrderFlowers",
   "checksum": "checksum",
   "version": "$LATEST",
   "rejectionStatement": {
       "messages": [
           {
               "content": "Okay, I will not place your order.",
               "contentType": "PlainText"
           }
       ]
   },
   "createdDate": timestamp,
   "lastUpdatedDate": timestamp,
   "sampleUtterances": [
       "I would like to pick up flowers",
       "I would like to order some flowers"
   ],
   "slots": [
       {
           "slotType": "AMAZON.TIME",
           "name": "PickupTime",
           "slotConstraint": "Required",
           "valueElicitationPrompt": {
               "maxAttempts": 2,
               "messages": [
                   {
                       "content": "Pick up the {FlowerType} at what time on
{PickupDate}?",
                       "contentType": "PlainText"
                   }
               ]
           },
           "priority": 3,
           "description": "The time to pick up the flowers"
```

```
        },
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers would you like to
order?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "$LATEST",
            "sampleUtterances": [
                "I would like to order {FlowerType}"
            ],
            "description": "The type of flowers to pick up"
        },
        {
            "slotType": "AMAZON.DATE",
            "name": "PickupDate",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What day do you want the {FlowerType} to be
picked up?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 2,
            "description": "The date to pick up the flowers"
        }
    ],
    "fulfillmentActivity": {
        "type": "ReturnIntent"
    },
    "description": "Intent to order a bouquet of flowers for pick up"
```

```
}
```

Nächster Schritt

[Schritt 4: Erstellen eines Bots (AWS CLI)]

OrderFlowers.json

Der folgende Code repräsentiert die JSON-Daten, die zum Erstellen der `OrderFlowers`-Absicht benötigt werden:

```
{
    "confirmationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "Okay, your {FlowerType} will be ready for pickup by
 {PickupTime} on {PickupDate}.  Does this sound okay?",
                "contentType": "PlainText"
            }
        ]
    },
    "name": "OrderFlowers",
    "rejectionStatement": {
        "messages": [
            {
                "content": "Okay, I will not place your order.",
                "contentType": "PlainText"
            }
        ]
    },
    "sampleUtterances": [
        "I would like to pick up flowers",
        "I would like to order some flowers"
    ],
    "slots": [
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
```

```
                    {
                        "content": "What type of flowers would you like to order?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "$LATEST",
            "sampleUtterances": [
                "I would like to order {FlowerType}"
            ],
            "description": "The type of flowers to pick up"
        },
        {
            "slotType": "AMAZON.DATE",
            "name": "PickupDate",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What day do you want the {FlowerType} to be picked
up?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 2,
            "description": "The date to pick up the flowers"
        },
        {
            "slotType": "AMAZON.TIME",
            "name": "PickupTime",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "Pick up the {FlowerType} at what time on
{PickupDate}?",
                        "contentType": "PlainText"
                    }
                ]
            },
```

```
            "priority": 3,
            "description": "The time to pick up the flowers"
        }
    ],
    "fulfillmentActivity": {
        "type": "ReturnIntent"
    },
    "description": "Intent to order a bouquet of flowers for pick up"
}
```

Schritt 4: Erstellen eines Bots (AWS CLI)

Der Bot `OrderFlowersBot` hat eine Absicht, die im vorherigen Schritt erstellte Absicht `OrderFlowers`. Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows `"\$LATEST"` in `$LATEST`.

So erstellen Sie den **`OrderFlowersBot`**-Bot (AWS CLI)

1. Erstellen Sie eine Textdatei mit dem Namen **`OrderFlowersBot.json`**. Kopieren Sie den JSON-Code aus OrderFlowersBot.json in die Textdatei.
2. Rufen Sie in der AWS CLI die Operation PutBot zum Erstellen des Bots auf. Das Beispiel ist für Unix, Linux und macOS formatiert. Ersetzen Sie unter Windows den umgekehrten Schrägstrich (\), das Unix-Fortsetzungszeichen, am Ende jeder Zeile durch ein Caret-Zeichen oder Zirkumflex (^).

   ```
   aws lex-models put-bot \
       --region region \
       --name OrderFlowersBot \
       --cli-input-json file://OrderFlowersBot.json
   ```

Die Antwort des Servers folgt. Beim Erstellen oder Aktualisieren des Bots wird dem Feld `status` der Wert `BUILDING` zugewiesen. Das gibt an, dass der Bot nicht betriebsbereit ist. Um zu ermitteln, ob der Bot einsatzbereit ist, verwenden Sie die Operation GetBot im nächsten Schritt.

```
{
    "status": "BUILDING",
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "processBehavior": "BUILD",
    "description": "Bot to order flowers on the behalf of a user"
}
```

3. Um zu bestimmen, ob der neue Bot betriebsbereit ist, führen Sie den folgenden Befehl aus. Wiederholen Sie diesen Befehl, bis das Feld `status` den Wert `READY` zurückgibt. Das Beispiel ist für Unix, Linux und macOS formatiert. Ersetzen Sie unter Windows den umgekehrten Schrägstrich (\), das Unix-Fortsetzungszeichen, am Ende jeder Zeile durch ein Caret-Zeichen oder Zirkumflex (^).

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST"
```

Suchen Sie in der Antwort nach dem Feld `status`.

```
{
    "status": "READY",

    ...

}
```

Nächster Schritt

[Schritt 5: Testen eines Bots (AWS CLI)]

OrderFlowersBot.json

Der folgende Code stellt die JSON-Daten bereit, die zum Erstellen des`OrderFlowers`Amazon Lex Bot:

```
{
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "abortStatement": {
        "messages": [
            {
```

```
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Schritt 5: Testen eines Bots (AWS CLI)

Zum Testen des Bots können Sie einen text- oder einen sprachbasierten Test verwenden.

Themen

- Testen des Bots mittels Texteingabe (AWS CLI)
- Testen des Bots mittels Spracheingabe (AWS CLI)

Testen des Bots mittels Texteingabe (AWS CLI)

Verwenden Sie die Operation PostText, um die ordnungsgemäße Funktion des Bots mittels Texteingabe zu prüfen. Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Laufzeit-Service-Kontingente.

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows `"\$LATEST"` zu `$LATEST`, und ersetzen Sie den umgekehrten Schrägstrich (\) am Ende jeder Zeile durch ein Caret-Zeichen (^).

So verwenden Sie Text zum Testen des Bots (AWS CLI)

1. Starten Sie in der AWS CLI eine Konversation mit dem Bot `OrderFlowersBot`. Das Beispiel ist für Unix, Linux und macOS formatiert. Ersetzen Sie unter Windows den umgekehrten Schrägstrich (\), das Unix-Fortsetzungszeichen, am Ende jeder Zeile durch ein Caret-Zeichen oder Zirkumflex (^).

   ```
   aws lex-runtime post-text \
       --region region \
       --bot-name OrderFlowersBot \
       --bot-alias "\$LATEST" \
       --user-id UserOne \
       --input-text "i would like to order flowers"
   ```

   Amazon Lex erkennt die Absicht des Benutzers und startet eine Konversation, indem folgende Antwort zurückgegeben wird:

   ```
   {
       "slotToElicit": "FlowerType",
       "slots": {
           "PickupDate": null,
           "PickupTime": null,
           "FlowerType": null
       },
       "dialogState": "ElicitSlot",
       "message": "What type of flowers would you like to order?",
       "intentName": "OrderFlowers"
   }
   ```

2. Führen Sie die folgenden Befehle aus, um die Konversation mit dem Bot zu beenden.

   ```
   aws lex-runtime post-text \
       --region region \
       --bot-name OrderFlowersBot \
       --bot-alias "\$LATEST" \
       --user-id UserOne \
       --input-text "roses"
   ```

   ```
   aws lex-runtime post-text  \
       --region region \
       --bot-name OrderFlowersBot \
   ```

```
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "tuesday"
```

```
aws lex-runtime post-text  \
    --region region \
    --bot-name OrderFlowersBot --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "10:00 a.m."
```

```
aws lex-runtime post-text  \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "yes"
```

Nachdem Sie die Bestellung bestätigt haben, sendet Amazon Lex eine Erfüllungsantwort, um die Konversation abzuschließen:

```
{
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    },
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers"
}
```

Nächster Schritt

Testen des Bots mittels Spracheingabe (AWS CLI)

Testen des Bots mittels Spracheingabe (AWS CLI)

Verwenden Sie die Operation PostContent, um den Bot mit Audiodateien zu testen. Sie generieren die Audiodateien mit Text-To-Speech-Operationen von Amazon Polly.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle Amazon Lex und Amazon Polly ausgeführt werden. Eine Liste der Regionen für Amazon Lex finden Sie unterLaufzeit-Service-Kontingenteaus. Eine Liste der Regionen für Amazon Polly finden Sie unterAWS-Regionen und -EndpunkteimAllgemeine Amazon Web Services Services-Referenzaus.

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows "\$LATEST" zu $LATEST, und ersetzen Sie den umgekehrten Schrägstrich (\) am Ende jeder Zeile durch ein Caret-Zeichen (^).

So verwenden Sie eine Spracheingabe zum Testen des Bots (AWS CLI)

1. In derAWS CLIerstellen, indem Sie Amazon Polly eine Audiodatei verwenden. Das Beispiel ist für Unix, Linux und macOS formatiert. Ersetzen Sie unter Windows den umgekehrten Schrägstrich (\), das Unix-Fortsetzungszeichen, am Ende jeder Zeile durch ein Caret-Zeichen oder Zirkumflex (^).

   ```
   aws polly synthesize-speech \
       --region region \
       --output-format pcm \
       --text "i would like to order flowers" \
       --voice-id "Salli" \
       IntentSpeech.mpg
   ```

2. Führen Sie den folgenden Befehl aus, um die Audiodatei an Amazon Lex zu senden. Amazon Lex speichert die Audiodaten aus der Antwort in der angegebenen Ausgabedatei.

   ```
   aws lex-runtime post-content \
       --region region \
       --bot-name OrderFlowersBot \
       --bot-alias "\$LATEST" \
       --user-id UserOne \
       --content-type "audio/l16; rate=16000; channels=1" \
       --input-stream IntentSpeech.mpg \
       IntentOutputSpeech.mpg
   ```

   Amazon Lex antwortet mit einer Anfrage für den ersten Slot. Die Audiodaten werden in der angegebenen Ausgabedatei gespeichert.

```
{
    "contentType": "audio/mpeg",
    "slotToElicit": "FlowerType",
    "dialogState": "ElicitSlot",
    "intentName": "OrderFlowers",
    "inputTranscript": "i would like to order some flowers",
    "slots": {
        "PickupDate": null,
        "PickupTime": null,
        "FlowerType": null
    },
    "message": "What type of flowers would you like to order?"
}
```

3. Bestellen Sie Rosen, indem Sie die folgende Audiodatei erstellen und an Amazon Lex senden:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "roses" \
    --voice-id "Salli" \
    FlowerTypeSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream FlowerTypeSpeech.mpg \
    FlowerTypeOutputSpeech.mpg
```

4. Legen Sie das Lieferdatum fest, indem Sie die folgende Audiodatei erstellen und an Amazon Lex senden:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "tuesday" \
    --voice-id "Salli" \
    DateSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream DateSpeech.mpg \
    DateOutputSpeech.mpg
```

5. Legen Sie die Lieferzeit fest, indem Sie die folgende Audiodatei erstellen und an Amazon Lex senden:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "10:00 a.m." \
    --voice-id "Salli" \
    TimeSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream TimeSpeech.mpg \
    TimeOutputSpeech.mpg
```

6. Bestätigen Sie die Lieferung, indem Sie die folgende Audiodatei erstellen und an Amazon Lex senden:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "yes" \
    --voice-id "Salli" \
    ConfirmSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
```

```
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream ConfirmSpeech.mpg \
    ConfirmOutputSpeech.mpg
```

Nachdem Sie die Lieferung bestätigt haben, sendet Amazon Lex eine Antwort, die die Erfüllung der Absicht bestätigt:

```
{
    "contentType": "text/plain;charset=utf-8",
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers",
    "inputTranscript": "yes",
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    }
}
```

Nächster Schritt

Übung 2: Fügen Sie eine neue Äußerung hinzu (AWS CLI)

Übung 2: Fügen Sie eine neue Äußerung hinzu (AWS CLI)

Um das Modell für maschinelles Lernen zu verbessern, das Amazon Lex verwendet, um Anforderungen der Benutzer zu erkennen, fügen Sie dem Bot eine weitere Beispieläußerung hinzu.

Das Hinzufügen einer neuen Äußerung erfolgt in vier Schritten.

1. Verwenden derGetIntent-Vorgang, um eine Absicht von Amazon Lex abzurufen.
2. Aktualisieren Sie die Absicht.
3. Verwenden derPutIntentOperation, um die aktualisierte Absicht an Amazon Lex zurückzusenden.
4. Verwenden Sie die Operationen GetBot und PutBot, um einen Bot wiederherzustellen, der die Absicht verwendet.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

Die Antwort der Operation `GetIntent` enthält ein Feld namens `checksum`, das die spezifische Version der Absicht angibt. Sie müssen den Prüfsummenwert angeben, wenn Sie die Operation PutIntent zum Aktualisieren einer Absicht verwenden. Wenn Sie dies nicht tun, wird die folgende Fehlermeldung angezeigt:

```
An error occurred (PreconditionFailedException) when calling
the PutIntent operation: Intent intent name already exists.
If you are trying to update intent name you must specify the
checksum.
```

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows `"\$LATEST"` zu `$LATEST`, und ersetzen Sie den umgekehrten Schrägstrich (\) am Ende jeder Zeile durch ein Caret-Zeichen (^).

So aktualisieren Sie die Absicht **`OrderFlowers`** (AWS CLI)

1. In derAWS CLI, holen Sie die Absicht von Amazon Lex ab. Amazon Lex sendet die Ausgabe in eine Datei namens**`OrderFlowers-V2.json.`**

   ```
   aws lex-models get-intent \
       --region region \
       --name OrderFlowers \
       --intent-version "\$LATEST" > OrderFlowers-V2.json
   ```

2. Öffnen Sie **`OrderFlowers-V2.json`** in einem Texteditor.

   1. Suchen und löschen Sie `createdDate`, `lastUpdatedDate` und `version`.
   2. Fügen Sie Folgendes dem Feld `sampleUtterances` hinzu:

      ```
      I want to order flowers
      ```

   3. Speichern Sie die Datei.

3. Senden Sie die aktualisierte Absicht mit dem folgenden Befehl an Amazon Lex:

```
aws lex-models put-intent  \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V2.json
```

Amazon Lex sendet die folgende Antwort:

```
{
    "confirmationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "Okay, your {FlowerType} will be ready for pickup by
 {PickupTime} on {PickupDate}.  Does this sound okay?",
                "contentType": "PlainText"
            }
        ]
    },
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
    "rejectionStatement": {
        "messages": [
            {
                "content": "Okay, I will not place your order.",
                "contentType": "PlainText"
            }
        ]
    },
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "sampleUtterances": [
        "I would like to pick up flowers",
        "I would like to order some flowers",
        "I want to order flowers"
    ],
    "slots": [
        {
            "slotType": "AMAZON.TIME",
            "name": "PickupTime",
            "slotConstraint": "Required",
```

```
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "Pick up the {FlowerType} at what time on
{PickupDate}?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 3,
            "description": "The time to pick up the flowers"
        },
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers would you like to
order?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "$LATEST",
            "sampleUtterances": [
                "I would like to order {FlowerType}"
            ],
            "description": "The type of flowers to pick up"
        },
        {
            "slotType": "AMAZON.DATE",
            "name": "PickupDate",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What day do you want the {FlowerType} to be
picked up?",
```

```
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 2,
            "description": "The date to pick up the flowers"
        }
    ],
    "fulfillmentActivity": {
        "type": "ReturnIntent"
    },
    "description": "Intent to order a bouquet of flowers for pick up"
}
```

Nachdem die Absicht aktualisiert wurde, müssen Sie jeden Bot neu erstellen, der diese Absicht verwendet.

So erstellen Sie den Bot **`OrderFlowersBot`** (AWS CLI)

1. Rufen Sie in der AWS CLI die Definition des Bots `OrderFlowersBot` ab und speichern Sie sie mit dem folgenden Befehl in einer Datei:

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias "\$LATEST" > OrderFlowersBot-V2.json
   ```

2. Öffnen Sie **`OrderFlowersBot-V2.json`** in einem Texteditor. Entfernen Sie die Felder `createdDate`, `lastUpdatedDate`, `status` und `version`.
3. Fügen Sie in einem Texteditor die folgende Zeile in die Bot-Definition ein:

   ```
   "processBehavior": "BUILD",
   ```

4. Erstellen Sie in der AWS CLI eine neue Version des Bots, indem Sie folgenden Befehl ausführen:

   ```
   aws lex-models put-bot \
       --region region \
       --name OrderFlowersBot \
       --cli-input-json file://OrderFlowersBot-V2.json
   ```

Die Antwort des Servers:

```
{
    "status": "BUILDING",
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Nächster Schritt

Übung 3: Fügen Sie eine Lambda-Funktion hinzu (AWS CLI)

Übung 3: Fügen Sie eine Lambda-Funktion hinzu (AWS CLI)

Fügen Sie eine Lambda-Funktion hinzu, die Benutzereingaben validiert und die Benutzerabsicht für den Bot erfüllt.

Das Hinzufügen eines Lambda-Ausdrucks erfolgt in fünf Schritten.

1. Benutze das LambdaAddPermissionFunktion zum Aktivieren des`OrderFlowers`Absicht, das Lambda anzurufenAufrufenverwenden.
2. Verwenden derGetIntent-Vorgang, um die Absicht von Amazon Lex abzurufen.
3. Aktualisieren Sie die Absicht, indem Sie die Lambda-Funktion hinzufügen.
4. Verwenden derPutIntentOperation, um die aktualisierte Absicht an Amazon Lex zurückzusenden.
5. Verwenden Sie die Operationen GetBot und PutBot, um einen Bot wiederherzustellen, der die Absicht verwendet.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

Wenn Sie eine Lambda-Funktion einer Absicht hinzufügen, bevor Sie die`InvokeFunction`-Berechtigung erhalten Sie die folgende Fehlermeldung:

```
An error occurred (BadRequestException) when calling the
PutIntent operation: Lex is unable to access the Lambda
function Lambda function ARN in the context of intent
intent ARN.  Please check the resource-based policy on
the function.
```

Die Antwort der Operation `GetIntent` enthält ein Feld namens `checksum`, das die spezifische Version der Absicht angibt. Wenn Sie die Operation PutIntent zum Aktualisieren einer Absicht verwenden, müssen Sie den Prüfsummenwert angeben. Wenn Sie dies nicht tun, wird die folgende Fehlermeldung angezeigt:

```
        An error occurred (PreconditionFailedException) when calling
        the PutIntent operation: Intent intent name already exists.
        If you are trying to update intent name you must specify the
        checksum.
```

In dieser Übung wird die Lambda-Funktion ausÜbung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole)aus. Eine Anleitung zum Erstellen der Lambda-Funktion finden Sie unterSchritt 3: Erstellen einer Lambda-Funktion (Lambda-Funktion)aus.

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows "\$LATEST" in $LATEST.

So fügen Sie einer Absicht eine Lambda-Funktion hinzu

1. Fügen Sie in der AWS CLI der Absicht `InvokeFunction` die Berechtigung `OrderFlowers` hinzu:

   ```
   aws lambda add-permission \
       --region region \
       --function-name OrderFlowersCodeHook \
       --statement-id LexGettingStarted-OrderFlowersBot \
       --action lambda:InvokeFunction \
       --principal lex.amazonaws.com \
       --source-arn "arn:aws:lex:region:account ID:intent:OrderFlowers:*"
       --source-account account ID
   ```

   Lambda sendet die folgende Antwort:

   ```
   {
       "Statement": "{\"Sid\":\"LexGettingStarted-OrderFlowersBot\",
         \"Resource\":\"arn:aws:lambda:region:account ID:function:OrderFlowersCodeHook
   \",
         \"Effect\":\"Allow\",
         \"Principal\":{\"Service\":\"lex.amazonaws.com\"},
         \"Action\":[\"lambda:InvokeFunction\"],
         \"Condition\":{\"StringEquals\":
           {\"AWS:SourceAccount\": \"account ID\"},
   ```

```
        {\"AWS:SourceArn\":
          \"arn:aws:lex:region:account ID:intent:OrderFlowers:*\"}}}"
}
```

2. Rufen Sie die Absicht von Amazon Lex ab. Amazon Lex sendet die Ausgabe in eine Datei namens**`OrderFlowers-V3.json`**aus.

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V3.json
```

3. Öffnen Sie **`OrderFlowers-V3.json`** in einem Text-Editor.

   1. Suchen und löschen Sie `createdDate`, `lastUpdatedDate` und `version`.
   2. Aktualisieren Sie das Feld `fulfillmentActivity`:

```
   "fulfillmentActivity": {
       "type": "CodeHook",
       "codeHook": {
           "uri": "arn:aws:lambda:region:account
 ID:function:OrderFlowersCodeHook",
           "messageVersion": "1.0"
       }
   }
```

   3. Speichern Sie die Datei.

4. In derAWS CLI, senden Sie die aktualisierte Absicht an Amazon Lex:

```
aws lex-models put-intent \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V3.json
```

Erstellen Sie den Bot neu, nachdem die Absicht aktualisiert wurde.

So erstellen Sie den Bot **`OrderFlowersBot`**

1. Rufen Sie in der AWS CLI die Definition des Bots `OrderFlowersBot` ab und speichern Sie sie in einer Datei:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V3.json
```

2. Öffnen Sie **`OrderFlowersBot-V3.json`** in einem Texteditor. Entfernen Sie die Felder `createdDate`, `lastUpdatedDate`, `status` und `version`.
3. Fügen Sie im Texteditor die folgende Zeile in die Bot-Definition ein:

```
"processBehavior": "BUILD",
```

4. Erstellen Sie in der AWS CLI eine neue Version des Bots:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V3.json
```

Die Antwort des Servers:

```
{
    "status": "READY",
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
```

```
    "createdDate": timestamp,
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Nächster Schritt

Übung 4: Veröffentlichen einer Version (AWS CLI)

Übung 4: Veröffentlichen einer Version (AWS CLI)

Nun erstellen Sie eine Version des Bots, der in Übung 1 erstellt wurde. Eine Version ist ein Snapshot des Bots. Nach dem Erstellen einer Version kann diese nicht mehr geändert werden. Sie können immer nur die $LATEST-Version eines Bots aktualisieren. Weitere Informationen zu Versionen erhalten Sie unter Versioning und Aliasnamen.

Bevor Sie eine Version eines Bots veröffentlichen, müssen Sie die vom Bot verwendete Absicht veröffentlichen. Entsprechend müssen Sie die Slot-Typen veröffentlichen, auf die diese Absichten verweisen. Im Allgemeinen gehen Sie folgendermaßen vor, um eine Version eines Bots zu veröffentlichen:

1. Veröffentlichen Sie eine Version eines Slot-Typs mit der Operation CreateSlotTypeVersion.
2. Veröffentlichen Sie eine Version einer Absicht mit der Operation CreateIntentVersion.
3. Veröffentlichen Sie eine Version eines Bots mit der Operation CreateBotVersion.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

Themen

Schritt 1: Veröffentlichen des Slot-Typs (AWS CLI)

Bevor Sie eine Version einer Absicht veröffentlichen können, die einen Slot-Typ verwendet, müssen Sie eine Version des betreffenden Slot-Typs veröffentlichen. In diesem Beispiel veröffentlichen Sie den Slot-Typ `FlowerTypes`.

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows `"\$LATEST"` zu `$LATEST`, und ersetzen Sie den umgekehrten Schrägstrich (\) am Ende jeder Zeile durch ein Caret-Zeichen (^).

So veröffentlichen Sie einen Slot-Typ (AWS CLI)

1. Rufen Sie in der AWS CLI die neueste Version des Slot-Typs ab:

   ```
   aws lex-models get-slot-type \
       --region region \
       --name FlowerTypes \
       --slot-type-version "\$LATEST"
   ```

   Die Antwort von Amazon Lex folgt. Notieren Sie die Prüfsumme der aktuellen Fassung der `$LATEST`-Version.

   ```
   {
       "enumerationValues": [
           {
               "value": "tulips"
           },
           {
               "value": "lilies"
           },
   ```

```
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "checksum": "checksum",
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

2. Veröffentlichen Sie eine Version des Slot-Typs. Verwenden Sie die im vorherigen Schritt notierte Prüfsumme.

```
aws lex-models create-slot-type-version \
    --region region \
    --name FlowerTypes \
    --checksum "checksum"
```

Die Antwort von Amazon Lex folgt. Notieren Sie für den nächsten Schritt die Versionsnummer.

```
{
    "version": "1",
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Nächster Schritt

Schritt 2: Veröffentlichen der Absicht (AWS CLI)

Schritt 2: Veröffentlichen der Absicht (AWS CLI)

Bevor Sie eine Absicht veröffentlichen können, müssen Sie alle Slot-Typen veröffentlichen, auf die die Absicht verweist. Bei den Slot-Typen muss es sich um nummerierte Versionen handeln, nicht um die `$LATEST`-Version.

Aktualisieren Sie zunächst die Absicht `OrderFlowers`, damit sie die Version des Slot-Typs `FlowerTypes` verwendet, die im vorherigen Schritt veröffentlicht wurde. Veröffentlichen Sie dann eine neue Version der `OrderFlowers`-Absicht.

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows `"\$LATEST"` zu `$LATEST`, und ersetzen Sie den umgekehrten Schrägstrich (\) am Ende jeder Zeile durch ein Caret-Zeichen (^).

So veröffentlichen Sie eine Version einer Absicht (AWS CLI)

1. Rufen Sie in der AWS CLI die `$LATEST`-Version der `OrderFlowers`-Absicht ab und speichern Sie sie in einer Datei:

   ```
   aws lex-models get-intent \
       --region region \
       --name OrderFlowers \
       --intent-version "\$LATEST" > OrderFlowers_V4.json
   ```

2. Öffnen Sie die Datei **`OrderFlowers_V4.json`** in einem Texteditor. Löschen Sie die Felder `createdDate`, `lastUpdatedDate` und `version`. Suchen Sie den Slot-Typ `FlowerTypes` und ändern Sie die Version in die Versionsnummer, die Sie im vorherigen Schritt notiert haben. Das folgende Fragment der Datei **`OrderFlowers_V4.json`** zeigt die Position der Änderung:

   ```
           {
               "slotType": "FlowerTypes",
               "name": "FlowerType",
               "slotConstraint": "Required",
               "valueElicitationPrompt": {
   ```

```
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "version",
            "sampleUtterances": []
        },
```

3. Speichern Sie in der AWS CLI die Version der Absicht:

```
aws lex-models put-intent \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers_V4.json
```

4. Ermitteln Sie die Prüfsumme der neuesten Version der Absicht:

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4a.json
```

Das folgende Fragment der Antwort zeigt die Prüfsumme der Absicht. Notieren Sie den Wert für den nächsten Schritt.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
```

5. Veröffentlichen einer neuen Version der Absicht:

```
aws lex-models create-intent-version \
    --region region \
    --name OrderFlowers \
    --checksum "checksum"
```

Das folgende Fragment der Antwort zeigt die neue Version der Absicht. Notieren Sie für den nächsten Schritt die Versionsnummer.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "version",
```

Nächster Schritt

Schritt 3: Veröffentlichen des Bots (AWS CLI)

Schritt 3: Veröffentlichen des Bots (AWS CLI)

Nachdem alle vom Bot verwendeten Slot-Typen und Absichten verwendet wurden, können Sie den Bot veröffentlichen.

Aktualisieren Sie den Bot `OrderFlowersBot`, damit er die Absicht `OrderFlowers` verwendet, die im vorherigen Schritt aktualisiert wurde. Veröffentlichen Sie dann eine neue Version des Bots `OrderFlowersBot`.

> **Note**
>
> Das folgende AWS CLI-Beispiel ist für Unix, Linux und macOS formatiert. Ändern Sie unter Windows "\$LATEST" zu $LATEST, und ersetzen Sie den umgekehrten Schrägstrich (\) am Ende jeder Zeile durch ein Caret-Zeichen (^).

So veröffentlichen Sie eine Version eines Bots (AWS CLI)

1. Rufen Sie in der AWS CLI die $LATEST-Version des `OrderFlowersBot`-Bots ab und speichern Sie sie in einer Datei:

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias "\$LATEST" > OrderFlowersBot_V4.json
   ```

2. Öffnen Sie die Datei **`OrderFlowersBot_V4.json`** in einem Texteditor. Löschen Sie die Felder `createdDate`, `lastUpdatedDate`, `status` und `version`. Suchen Sie die Absicht `OrderFlowers` und ändern Sie die Version in die Versionsnummer, die Sie im vorherigen Schritt notiert haben. Das folgende Fragment der Datei **`OrderFlowersBot_V4.json`** zeigt die Position der Änderung.

```
    "intents": [
        {
            "intentVersion": "version",
            "intentName": "OrderFlowers"
        }
```

3. Speichern Sie in der AWS CLI die neue Version des Bots. Notieren Sie sich die Versionsnummer, die durch den Aufruf von`put-bot` zurückgegeben wird.

```
aws lex-models put-bot \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot_V4.json
```

4. Ermitteln Sie die Prüfsumme der neuesten Version des Bots. Verwenden Sie die Versionsnummer, die in Schritt 3 zurückgegeben wird.

```
aws lex-models get-bot \
    --region region \
    --version-or-alias version \
    --name OrderFlowersBot > OrderFlowersBot_V4a.json
```

Das folgende Fragment der Antwort zeigt die Prüfsumme des Bots. Notieren Sie den Wert für den nächsten Schritt.

```
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
```

5. Veröffentlichen einer neuen Version des Bots:

```
aws lex-models create-bot-version \
    --region region \
    --name OrderFlowersBot \
    --checksum "checksum"
```

Das folgende Fragment der Antwort zeigt die neue Version des Bots.

```
    "checksum": "checksum",
    "abortStatement": {
        ...
```

```
    },
    "version": "1",
    "lastUpdatedDate": timestamp,
```

Nächster Schritt

[Übung 5: Erstellen eines Alias (AWS CLI)]

Übung 5: Erstellen eines Alias (AWS CLI)

Ein Alias ist ein Zeiger auf eine bestimmte Bot-Version. Mit einem Alias können Sie ganz einfach die Version aktualisieren, die Ihre Client-Anwendungen verwenden. Weitere Informationen finden Sie unter Versioning und Aliasnamen. Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

So erstellen Sie einen Alias (AWS CLI)

1. Rufen Sie in der AWS CLI die Version des Bots `OrderFlowersBot` ab, die Sie in Übung 4: Veröffentlichen einer Version (AWS CLI) erstellt haben.

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias version > OrderFlowersBot_V5.json
   ```

2. Öffnen Sie **`OrderFlowersBot_v5.json`** in einem Texteditor. Suchen Sie die Versionsnummer und notieren Sie sie.
3. Erstellen Sie in der AWS CLI den Bot-Alias:

   ```
   aws lex-models put-bot-alias  \
       --region region \
       --name PROD \
       --bot-name OrderFlowersBot \
       --bot-version version
   ```

   Die Antwort vom Server:

   ```
   {
       "name": "PROD",
   ```

```
    "createdDate": timestamp,
    "checksum": "checksum",
    "lastUpdatedDate": timestamp,
    "botName": "OrderFlowersBot",
    "botVersion": "1"
}}
```

Nächster Schritt

Übung 6: Bereinigen (AWS CLI)

Übung 6: Bereinigen (AWS CLI)

Löschen Sie die Ressourcen, die Sie erstellt haben, und bereinigen Sie das Konto.

Sie können nur Ressourcen löschen, die nicht genutzt werden. Im Allgemeinen sollten Sie Ressourcen in der folgenden Reihenfolge löschen.

1. Löschen Sie die Aliasse, um Bot-Ressourcen freizugeben.
2. Löschen Sie Bots, um Absicht-Ressourcen freizugeben.
3. Löschen Sie Absichten, um Slot-Typ-Ressourcen freizugeben.
4. Löschen Sie die Slot-Typen.

Um die Befehle in dieser Übung auszuführen, müssen Sie wissen, in welcher Region die Befehle ausgeführt werden. Eine Liste der Regionen finden Sie unter Kontingente des Modellbaus .

So bereinigen Sie Ihr Konto (AWS CLI)

1. Löschen Sie den Alias in der AWS CLI-Befehlszeile:

```
aws lex-models delete-bot-alias \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot
```

2. Löschen Sie den Bot in der AWS CLI-Befehlszeile:

```
aws lex-models delete-bot \
```

```
    --region region \
    --name OrderFlowersBot
```

3. Löschen Sie die Absicht in der AWS CLI-Befehlszeile:

```
aws lex-models delete-intent \
    --region region \
    --name OrderFlowers
```

4. Löschen Sie den Slot-Typ in der AWS CLI-Befehlszeile:

```
aws lex-models delete-slot-type \
    --region region \
    --name FlowerTypes
```

Sie haben alle Ressourcen, die Sie erstellt haben, entfernt und Ihr Konto bereinigt.

Versioning und Aliasnamen

Amazon Lex unterstützt das Veröffentlichen von Versionen von Bots, Absichten und Slot-Typen, sodass Sie die Implementierung kontrollieren können, die Ihre Clientanwendungen verwenden. Eine Version ist ein nummerierter Snapshot Ihrer Arbeit, den Sie zur Verwendung in verschiedenen Teilen Ihres Workflows, zum Beispiel Entwicklung, Beta-Bereitstellung und Produktion, veröffentlichen können.

Amazon LEX Bots unterstützen auch Aliase. Ein Alias ist ein Zeiger auf eine bestimmte Bot-Version. Mit einem Alias können Sie einfach die Version aktualisieren, die Ihre Clientanwendungen verwenden. Beispielsweise können Sie einen Alias auf Version 1 Ihres Bot zeigen lassen. Wenn Sie bereit sind, den Bot zu aktualisieren, veröffentlichen Sie Version 2 und ändern den Alias so, dass er auf die neue Version zeigt. Da Ihre Anwendungen den Alias anstelle einer bestimmten Version verwenden, erhalten alle Ihre Clients die neuen Funktionen, ohne dafür aktualisiert werden zu müssen.

Themen

Versioning

Wenn Sie eine Amazon Lex LEX-Ressource versionieren, erstellen Sie einen Snapshot der Ressource, sodass Sie die Ressource in der Form verwenden können, die sie beim Erstellen der Version hatte. Nachdem Sie eine Version erstellt haben, bleibt Sie, während Sie weiter an Ihrer Anwendung arbeiten, unverändert.

Die $LATEST Version

Wenn Sie einen Amazon LEX -Bot, eine Absicht oder einen Slot-Typ erstellen, gibt es nur eine Version, die`$LATEST`-Version

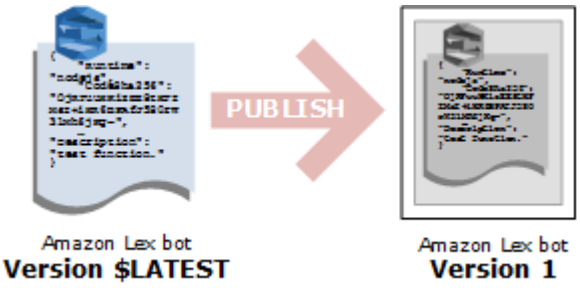
$LATEST ist die Arbeitsversion Ihrer Ressource. Sie können nur die $LATEST Version aktualisieren. Bis Sie Ihre erste Version veröffentlicht haben, ist die $LATEST Version die einzige Version der Ressource, die Sie haben.

Nur die $LATEST Version einer Ressource kann die $LATEST Version einer anderen Ressource nutzen. Beispielsweise kann die $LATEST Version eines Bots die $LATEST Version einer Absicht nutzen und die $LATEST Version einer Absicht kann die $LATEST Version eines Slot-Typs verwenden.

Die$LATESTDie Version Ihres Bots sollte nur für manuelle Tests verwendet werden. Amazon Lex begrenzt die Anzahl der Laufzeitanforderungen, die Sie an die$LATESTVersion des Bots.

Veröffentlichen einer Amazon Lex Lex-Ressourcenversion

Wenn Sie eine Ressource veröffentlichen, erstellt Amazon Lex eine Kopie der$LATEST-Version und speichert es als nummerierte Version. Die veröffentlichte Version kann nicht geändert werden.



Sie erstellen und veröffentlichen Versionen mithilfe der Amazon Lex LEX-Konsole oder derCreateBotVersionverwenden. Ein Beispiel finden Sie unter Übung 3: Eine Version veröffentlichen und einen Aliasnamen generieren.

Wenn Sie die $LATEST Version einer Ressource ändern, können Sie die neue Version veröffentlichen, um die Änderungen für Ihre Clientanwendungen bereitzustellen. Jedes Mal, wenn Sie eine Version veröffentlichen, kopiert Amazon Lex$LATEST-Version, um die neue Version zu

erstellen, erhöht die Versionsnummer um 1. Versionsnummern werden niemals wiederverwendet. Wenn Sie beispielsweise eine Ressource mit der Versionsnummer 10 entfernen und dann neu erstellen, ist die Versionsnummer, die Amazon Lex zuweist, die Nummer 11.

Bevor Sie einen Bot veröffentlichen können, müssen Sie diesen auf eine nummerierte Version einer Absicht verweisen, die er verwendet. Wenn Sie versuchen, eine neue Version eines Bots zu veröffentlichen, der die $LATEST-Version einer Absicht verwendet, gibt Amazon Lex einen HTTP-400-Fehler (Bad Request) zurück. Bevor Sie einen nummerierte Version einer Absicht veröffentlichen können, müssen Sie die Absicht auf eine nummerierte Version eines von ihr verwendeten Slot-Typs verweisen. Andernfalls erhalten Sie einen HTTP-400-Fehler (Bad Request).

> **Note**
>
> Amazon Lex veröffentlicht eine neue Version nur, wenn sich die zuletzt veröffentlichte Version von der$LATEST-Version Wenn Sie versuchen, das$LATEST-Version, ohne diese zu ändern, erstellt oder veröffentlicht Amazon Lex keine neue Version.

Aktualisieren einer Amazon Lex Lex-Ressource

Sie können nur die$LATEST-Version eines Amazon LEX Bots, einer -Absicht oder eines -Slot-Typs. Veröffentlichte Versionen können nicht geändert werden. Sie können jederzeit eine neue Version veröffentlichen, nachdem Sie eine Ressource in der Konsole oder über die Operationen CreateBotVersion, CreateIntentVersion oder CreateSlotTypeVersion aktualisiert haben.

Löschen einer Amazon Lex LEX-Ressource oder -Version

Amazon Lex unterstützt das Löschen einer Ressource oder Version mit der Konsole oder einer der folgenden API-Operationen:

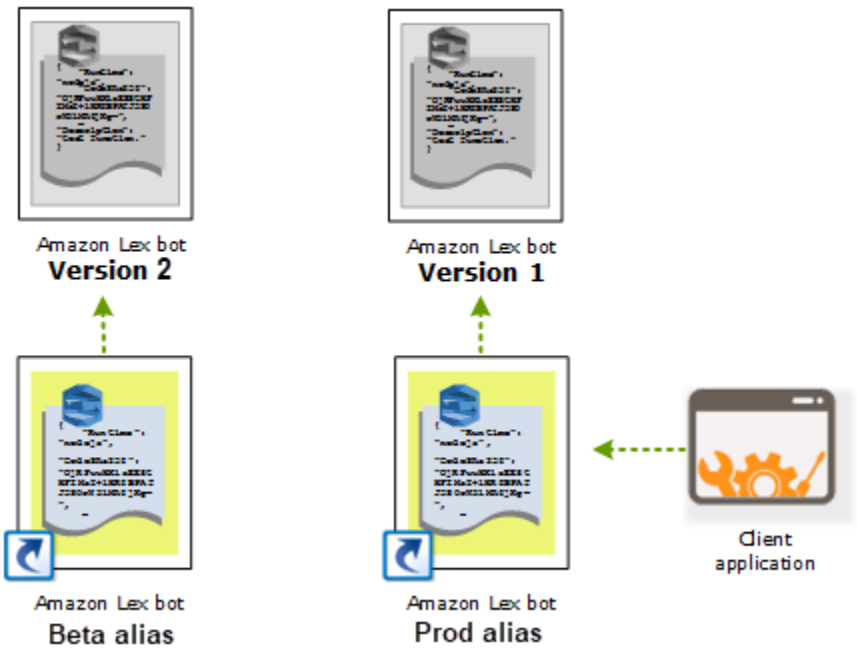
- DeleteBot

- DeleteBotVersion
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion

Aliasnamen

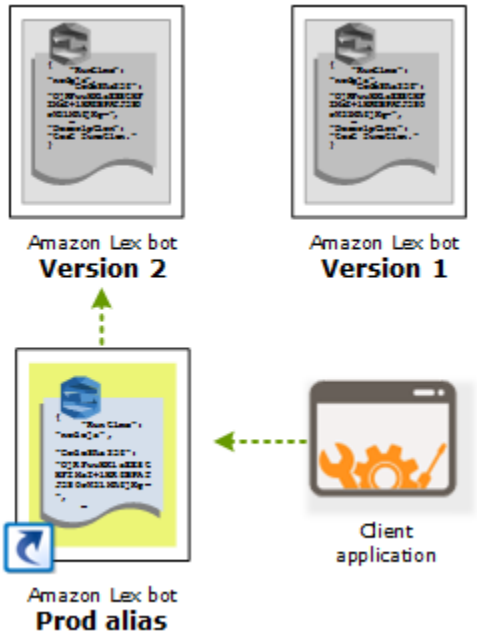
Ein Alias ist ein Zeiger auf eine bestimmte Version eines Amazon Lex LEX-Bots. Verwenden Sie einen Alias, um Clientanwendungen zu erlauben, eine bestimmte Version des Bots zu verwenden, ohne dass die Anwendung nachverfolgen muss, um welche Version es sich handelt.

Das folgende Beispiel zeigt zwei Versionen eines Amazon Lex Bots, Version 1 und Version 2. Jeder dieser Bot-Versionen ist ein Alias zugeordnet, BETA bzw. PROD. Clientanwendungen greifen auf den Bot mithilfe des PROD-Alias zu.



Wenn Sie eine zweite Version des Bots erstellen, können Sie den Alias mit der Konsole oder der PutBot-Operation so aktualisieren, dass er auf die neue Version des Bots zeigt. Wenn Sie den Alias ändern, verwenden alle Ihre Clientanwendungen die neue Version. Wenn es mit der neuen Version

ein Problem gibt, können Sie einfach zu der vorhergehenden Version zurückkehren, indem Sie den Alias so ändern, dass er auf diese Version zeigt.



> **Note**
>
> Obwohl Sie die $LATEST Version eines Bots in der Konsole testen können, empfehlen wir für den Fall, dass Sie einen Bot in Ihre Clientanwendung integrieren, zunächst eine Version zu veröffentlichen und einen Alias zu erstellen, der auf diese Version zeigt. Verwenden Sie den Alias in Ihrer Clientanwendung aus den Gründen, die in diesem Abschnitt erklärt werden. Wenn Sie einen Alias aktualisieren, wartet Amazon Lex bis alle aktuellen Sitzungen abgelaufen sind, ehe es die neue Version verwendet. Weitere Informationen zur Zeitbeschränkung bei einer Sitzung finden Sie unter the section called “Festlegen der Sitzungs-Zeitüberschreitung”.

Verwenden von Lambda-Funktionen

Sie könnenAWS Lambda-Funktionen zur Verwendung als Code-Haken für Ihren Amazon Lex. In der Konfiguration Ihrer Absicht können Sie Lambda-Funktionen dazu benutzen, um Initialisierung und Validierung, Erfüllung oder beides durchzuführen.

Wir empfehlen, dass Sie eine Lambda-Funktion als Code-Haken für den Bot verwenden. Ohne eine Lambda-Funktion gibt Ihr Bot die Absichtsinformationen zur Erfüllung an die Clientanwendung zurück.

Themen

Lambda-Funktions-Eingabe-Ereignis und Antwort-Format

In diesem Abschnitt wird die Struktur der Ereignisdaten beschrieben, die Amazon Lex für eine Lambda-Funktion bereitstellt. Nutzen Sie diese Informationen zum Analysieren der Eingabe der Lambda-Code. Es wird auch das Format der Antwort erklärt, die Amazon Lex als Rückgabe Ihrer Lambda-Funktion erwartet.

Themen

Eingabe-Ereignis-Format

Nachfolgend sehen Sie das allgemeine Format eines Amazon Lex -Ereignisses, das an eine Lambda-Funktion übergeben wird. Nutzen Sie diese Informationen zum Schreiben einer Lambda-Funktion.

> **Note**
>
> Das Eingabeformat kann sich auch ohne entsprechende Änderung der `messageVersion` ändern. Ihr Code sollte keinen Fehler ausgeben, wenn neue Felder vorhanden sind.

```
{
  "currentIntent": {
    "name": "intent-name",
    "nluIntentConfidenceScore": score,
    "slots": {
      "slot name": "value",
      "slot name": "value"
    },
    "slotDetails": {
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      },
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      }
    },
    "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)"
  },
  "alternativeIntents": [
    {
      "name": "intent-name",
      "nluIntentConfidenceScore": score,
      "slots": {
        "slot name": "value",
        "slot name": "value"
      },
      "slotDetails": {
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        },
```

```
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        }
      },
      "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)"
    }
 ],
 "bot": {
   "name": "bot name",
   "alias": "bot alias",
   "version": "bot version"
 },
 "userId": "User ID specified in the POST request to Amazon Lex.",
 "inputTranscript": "Text used to process the request",
 "invocationSource": "FulfillmentCodeHook or DialogCodeHook",
 "outputDialogMode": "Text or Voice, based on ContentType request header in runtime
API request",
 "messageVersion": "1.0",
 "sessionAttributes": {
    "key": "value",
    "key": "value"
 },
 "requestAttributes": {
    "key": "value",
    "key": "value"
 },
 "recentIntentSummaryView": [
   {
       "intentName": "Name",
       "checkpointLabel": Label,
       "slots": {
         "slot name": "value",
         "slot name": "value"
       },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)",
       "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
Close",
       "fulfillmentState": "Fulfilled or Failed",
```

```
        "slotToElicit": "Next slot to elicit"
    }
  ],
   "sentimentResponse": {
      "sentimentLabel": "sentiment",
      "sentimentScore": "score"
   },
   "kendraResponse": {
       Complete query response from Amazon Kendra
   },
   "activeContexts": [
        {
            "timeToLive": {
                "timeToLiveInSeconds": seconds,
                "turnsToLive": turns
            },
            "name": "name",
            "parameters": {
                "key name": "value"
            }
        }
    ]
}
```

Beachten Sie die folgenden zusätzlichen Informationen über die Ereignisfelder:

- currentIntent – Stellt die Absichtsfelder `name`, `slots`, `slotDetails` und `confirmationStatus` bereit.

  `nluIntentConfidenceScore`ist das Vertrauen, das Amazon Lex hat, dass die aktuelle Absicht diejenige ist, die der aktuellen Absicht des Benutzers am besten entspricht.

  `slots`ist eine Zuordnung von Slot-Namen, die für die Absicht konfiguriert werden, zu Slot-Werten, die Amazon Lex in der Benutzerkonversation erkannt hat. Ein Slot-Wert bleibt Null, bis der Benutzer einen Wert angibt.

Der Slot-Wert im Eingabeereignis stimmt möglicherweise nicht mit einem der für den Slot konfigurierten Werte überein. Wenn der Benutzer auf die Aufforderung "Welche Farbe soll Ihr Auto haben?" Mit „pizza“ gibt Amazon Lex „pizza“ als Slot-Wert zurück. Die Funktion sollte die Werte überprüfen, um sicherzustellen, dass sie im Kontext Sinn machen.

`slotDetails` bietet weitere Informationen zu einem Slot-Wert. Das `resolutions`-Array enthält eine Liste weiterer Werte, die für den Slot erkannt wurden. Jeder Slot kann maximal fünf Werte haben.

Das Feld `originalValue` enthält den Wert, der vom Benutzer für den Slot eingegeben wurde. Wenn der Slot-Typ so konfiguriert wurde, dass er den Wert der höchsten Auflösung als Slot-Wert zurückgibt, unterscheidet sich der `originalValue` möglicherweise von dem Wert im Feld `slots`.

`confirmationStatus` bietet dem Benutzer eine Antwort auf eine Bestätigungsaufforderung, falls eine vorhanden ist. Wenn zum Beispiel Amazon Lex fragt: „Möchten Sie eine große Käsepizza bestellen? , „ist der Wert dieses Felds je nach Antwort des Benutzers`Confirmed`oder`Denied`aus. Andernfalls ist der Wert dieses Felds `None`.

Wenn der Benutzer die Absicht bestätigt, setzt Amazon Lex dieses Feld auf`Confirmed`aus. Wenn der Benutzer die Absicht verneint, setzt Amazon Lex diesen Wert auf`Denied`aus.

In der Antwort zur Bestätigung führt eine Benutzeräußerung möglicherweise zur Aktualisierung von Slots. So könnte der Benutzer zum Beispiel sagen „Ja, ändere Größe zu Mittel.“ In diesem Fall hat das nachfolgende Lambda-Ereignis den aktualisierten Slot-Wert:`PizzaSize`stellen Sie auf`medium`aus. Amazon Lex setzt`confirmationStatus`zu`None`geändert hat, weshalb der Benutzer einige Slot-Daten geändert hat, weshalb die Lambda-Funktion eine Validierung der Benutzerdaten durchführen muss.

- AlternativeIntents— Wenn Sie Konfidenzwerte aktivieren, gibt Amazon Lex bis zu vier alternative Absichten zurück. Jede Absicht enthält einen Punktestand, der das Vertrauen angibt, das Amazon Lex hat, dass die Absicht die richtige Absicht ist, basierend auf der Äußerung des Benutzers.

  Der Inhalt der alternativen Absichten entspricht dem Inhalt des`currentIntent`field. Weitere Informationen finden Sie unter Verwenden von Zuverlässigkeitswert .

- bot – Informationen über den Bot, der die Anforderung verarbeitet hat.
  - `name` – Der Name des Bots, der die Anforderung verarbeitet hat.
  - `alias` – Der Alias der Bot-Version, mit dem die Anforderung verarbeitet wurde.
  - `version` – Die Version des Bots, mit der die Anforderung verarbeitet wurde.

- userId— Dieser Wert wird von der Clientanwendung angegeben. Amazon Lex gibt es an die Lambda-Funktion weiter.

- inputTranscript – Der Text, der für die Verarbeitung der Anforderung verwendet wurde.

  Wurde Text eingegeben, enthält das Feld `inputTranscript` den vom Benutzer eingegebenen Text.

  Wurde ein Audio-Stream eingegeben, enthält das Feld `inputTranscript` den aus dem Audio-Stream extrahierten Text. Dies ist der Text, der tatsächlich verarbeitet wurde, um Absichten und Slot-Werte zu erkennen.

- invocationSource: Um anzugeben, warum Amazon Lex die Lambda-Funktion aufruft, wird dies auf einen der folgenden Werte eingestellt:
  - `DialogCodeHook`: Amazon Lex stellt diesen Wert ein, um die Lambda-Funktion anzuweisen, die Funktion zu initialisieren und die Dateneingabe des Benutzers zu validieren.

Wenn die Absicht konfiguriert ist, um eine Lambda-Funktion als einen Code-Haken zur Initialisierung und Validierung aufzurufen, ruft Amazon Lex bei jeder Benutzereingabe (Äußerung) die angegebene Lambda-Funktion bei jeder Benutzereingabe (Äußerung) auf, nachdem Amazon Lex die Absicht verstanden hat.

> **Note**
>
> Wenn die Absicht nicht klar ist, kann Amazon Lex die Lambda-Funktion nicht aufrufen.

- `FulfillmentCodeHook`— Amazon Lex setzt diesen Wert ein, um die Lambda-Funktion anzuweisen, eine Absicht zu erfüllen.

  Wenn die Absicht konfiguriert ist, um eine Lambda-Funktion als Code-Haken zur Erfüllung aufzurufen, stellt Amazon Lex die`invocationSource`auf diesen Wert erst dann, wenn alle Slot-Daten zur Erfüllung der Absicht vorliegen.

In der Konfiguration Ihrer Absicht können Sie zwei unterschiedliche Lambda-Funktionen haben, um Benutzerdaten zu initialisieren und zu validieren und um die Absicht zu erfüllen. Eine Lambda-Funktion kann auch beides machen. In diesem Fall kann Ihre Lambda-Funktion`invocationSource`Wert, um dem richtigen Code-Pfad zu folgen.

- OutputDialogMode: Für jede Benutzereingabe sendet der Client die Anfrage an Amazon Lex mithilfe einer der API-Operationen zur Laufzeit,PostContentoderPostTextaus. Mithilfe der Anforderungsparameter legt Amazon Lex fest, ob es sich bei der Antwort an den Client um Text oder Sprache handelt, und stellt dieses Feld entsprechend ein.

  Die Lambda-Funktion kann diese Informationen verwenden, um eine entsprechende Mitteilung zu generieren. Wenn der Client beispielsweise eine Sprachantwort erwartet, kann Ihre Lambda-Funktion SSML (Speech Synthesis Markup Language) anstelle von Text zurückgeben.

- MessageVersion: Die Version der Mitteilung, die das Format der Ereignisdaten, die in die Lambda-Funktion eingehen, und das erwartete Format der Antwort von einer Lambda-Funktion identifiziert.

  > **Note**
  >
  > Sie konfigurieren diesen Wert, wenn Sie eine Absicht definieren. In der aktuellen Implementierung, nur Mitteilung Version 1.0 wird unterstützt. Daher nimmt die Konsole 1.0 als Standardwert an und zeigt die Mitteilungsversion nicht.

- sessionAttributes— Anwendungsspezifische Sitzungsattribute, die der Client in der Anforderung sendet. Wenn Sie möchten, dass Amazon Lex diese in die Antwort an den Client einschließt, sollte Ihre Lambda-Funktion diese in der Antwort an Amazon Lex zurücksenden. Weitere Informationen finden Sie unterFestlegen von Sitzungsattributen

- requestAttributes— Anforderungsspezifische Attribute, die der Client in der Anforderung sendet. Verwenden Sie Anforderungsattribute zur Weitergabe von Informationen, die nicht während der ganzen Sitzung erhalten bleiben müssen. Wenn keine Anforderungsattribute vorhanden sind, ist der Wert Null. Weitere Informationen finden Sie unterFestlegen von Anforderungsattributen

- RecentIntentSummaryView— Informationen über den Stand einer Absicht. Sie können Informationen zu den letzten drei Absichten anzeigen. Sie können diese Informationen verwenden, um Werte in der Absicht festzulegen oder zu einer vorherigen Absicht zurückzukehren. Weitere Informationen finden Sie unter Verwalten von Sitzungen mit der Amazon Lex API .

- SentimentResponse— Das Ergebnis einer Amazon Comprehend-Stimmentanalyse der letzten Äußerung. Sie können diese Informationen verwenden, um den Konversationsfluss Ihres Bots je nach der vom Benutzer ausgedrückten Stimmung zu verwalten. Weitere Informationen finden Sie unter Stimmungsanalyse .

- KendraResponse— Das Ergebnis einer Abfrage eines Amazon Kendra-Index. Nur in der Eingabe zu einem Erfüllungscode-Hook vorhanden und nur, wenn die Absicht die integrierte Absicht

`AMAZON.KendraSearchIntent` erweitert. Das Feld enthält die gesamte Antwort aus der Amazon Kendra-Suche. Weitere Informationen finden Sie unter AMAZON.KendraSearchIntent .

- ActiveContexts— Ein oder mehrere Kontexte, die während dieser Runde eines Gesprächs mit dem Benutzer aktiv sind.
  - TimeToLive— Die Dauer oder Anzahl der Windungen im Gespräch mit dem Benutzer, dass der Kontext aktiv bleibt.
  - Name— Der Name des Kontexts.
  - Parametereine Liste von Schlüssel/Wert-Paaren enthält den Namen und Wert der Slots aus der Absicht, die den Kontext aktiviert hat.

  Weitere Informationen finden Sie unter Absichtskontext festlegen .

Reaktion-Format

Amazon Lex erwartet eine Antwort einer Lambda-Funktion im folgenden Format:

```
{
  "sessionAttributes": {
    "key1": "value1",
    "key2": "value2"
    ...
  },
  "recentIntentSummaryView": [
    {
       "intentName": "Name",
       "checkpointLabel": "Label",
       "slots": {
         "slot name": "value",
         "slot name": "value"
        },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
Close",
        "fulfillmentState": "Fulfilled or Failed",
        "slotToElicit": "Next slot to elicit"
    }
  ],
```

```
  "activeContexts": [
     {
       "timeToLive": {
          "timeToLiveInSeconds": seconds,
          "turnsToLive": turns
       },
       "name": "name",
       "parameters": {
         "key name": "value"
       }
     }
  ],
  "dialogAction": {
    "type": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or Close",
    Full structure based on the type field. See below for details.
  }
}
```

Die Antwort besteht aus vier Feldern. Die`sessionAttributes`,`recentIntentSummaryView`, und`activeContexts`Die Felder sind optional`dialogAction`ist ein Pflichtfeld. Der Inhalt von Feld `dialogAction` ist abhängig vom Wert in Feld `type`. Details hierzu finden Sie unter dialogAction.

sessionAttributes

Optional. Wenn Sie das Feld `sessionAttributes` einschließen, kann es leer sein. Wenn Ihre Lambda-Funktion keine Sitzungsattribute zurückgibt, ist der letzte bekannte`sessionAttributes`, die über die API- oder Lambda-Funktion übergeben wird, bleiben erhalten. Weitere Informationen finden Sie im Zusammenhang mit den Operationen PostContent und PostText.

```
  "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"
  }
```

RecentIntentSummaryView

Optional. Wenn diese Option enthalten ist, werden Werte für eine oder mehrere aktuelle Absichten festgelegt. Sie können Informationen für bis zu drei Absichten aufnehmen. Beispielsweise können Sie Werte für vorherige Absichten basierend auf Informationen festlegen, die von der aktuellen Absicht erfasst werden. Die Informationen in der Zusammenfassung müssen für die Absicht gültig sein. Beispielsweise muss der Absichtsname eine Absicht im Bot sein. Wenn Sie einen Steckplatzwert in

die Zusammenfassungsansicht einschließen, muss der Steckplatz in der Absicht vorhanden sein. Wenn Sie `recentIntentSummaryView` nicht in Ihre Antwort aufnehmen, bleiben alle Werte für die letzten Absichten unverändert. Weitere Informationen finden Sie beim PutSession-Vorgang oder dem IntentSummary-Datentyp.

```
"recentIntentSummaryView": [
    {
       "intentName": "Name",
       "checkpointLabel": "Label",
       "slots": {
         "slot name": "value",
         "slot name": "value"
        },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
 Close",
        "fulfillmentState": "Fulfilled or Failed",
        "slotToElicit": "Next slot to elicit"
    }
 ]
```

ActiveContexts

Optional. Wenn diese Option enthalten ist, wird der Wert für einen oder mehrere Kontexte festgelegt. Sie können beispielsweise einen Kontext einbeziehen, um eine oder mehrere Absichten zu erstellen, die diesen Kontext als Eingabe haben, die für die nächste Runde der Konversation anerkannt werden kann.

Alle aktiven Kontexte, die nicht in der Antwort enthalten sind, haben ihre Time-to-Live-Werte verringert und sind möglicherweise bei der nächsten Anfrage weiterhin aktiv.

Wenn Sie eine Live-Zeit von 0 für einen Kontext angeben, der in das Eingabeereignis aufgenommen wurde, ist er bei der nächsten Anforderung inaktiv.

Weitere Informationen finden Sie unter Absichtskontext festlegen .

dialogAction

Erforderlich. Die`dialogAction`Das Feld führt Amazon Lex zur nächsten Aktion und beschreibt, was vom Benutzer zu erwarten ist, nachdem Amazon Lex eine Antwort an den Client zurückgegeben hat.

Das Feld `type` gibt die nächste Vorgehensweise an. Es legt auch fest, welche anderen Felder die Lambda-Funktion als Teil der`dialogAction`Wert.

- `Close`- Informiert Amazon Lex darüber, keine Antwort vom Benutzer zu erwarten. Beispiel: Für "Ihre Pizzabestellung wurde aufgenommen" ist keine Antwort erforderlich.

  Das Feld `fulfillmentState` ist ein Pflichtfeld. Amazon Lex verwendet diesen Wert, um die`dialogState`field imPostContentoderPostTextAntwort auf die Client-Anwendung. Die Felder `message` und `responseCard` sind optional. Wenn Sie keine Mitteilung festlegen, verwendet Amazon Lex die Abschiedsmitteilung oder die für die Absicht konfigurierte Follow-up-Mitteilung.

```
"dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled or Failed",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, Thanks, your pizza has
 been ordered."
    },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
 card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
       ]
     }
  }
```

- `ConfirmIntent`: Informiert Amazon Lex darüber, dass vom Benutzer ein „Ja“ oder „Nein“ als Antwort erwartet wird, um die aktuelle Absicht zu bestätigen oder zu verwerfen.

Sie müssen die Felder `intentName` und `slots` einschließen. Das Feld `slots` muss einen Eintrag für jeden ausgefüllten Slot für die angegebene Absicht enthalten. Sie brauchen im Feld `slots` keinen Eintrag für Slots anzugeben, die nicht ausgefüllt sind. Das Feld `message` muss enthalten sein, wenn das Feld `confirmationPrompt` der Absicht Null ist. Der Inhalt der`message`Das Feld, das von der Lambda-Funktion zurückgegeben wird, hat Vorrang vor`confirmationPrompt`In der Absicht angegeben. Das Feld `responseCard` ist optional.

```
"dialogAction": {
    "type": "ConfirmIntent",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, Are you sure you want a
 large pizza?"
    },
   "intentName": "intent-name",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
 card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
```

```
        ]
      }
  }
```

- `Delegate`— Weist Amazon Lex an, die nächste Aktion basierend auf der Bot-Konfiguration auszuwählen. Wenn die Antwort keine Sitzungsattribute enthält, behält Amazon Lex die vorhandenen Attribute bei. Wenn Sie möchten, dass ein Slot-Wert null ist, brauchen Sie das Slot-Feld nicht in die Anforderung einzuschließen. Sie erhalten die Ausnahme `DependencyFailedException`, wenn die Erfüllungsfunktion die Dialogaktion `Delegate` zurückgibt, ohne dass Slots entfernt werden.

  Die Felder `kendraQueryRequestPayload` und `kendraQueryFilterString` sind optional und werden nur verwendet, wenn die Absicht von der integrierten Absicht `AMAZON.KendraSearchIntent` abgeleitet wird. Weitere Informationen finden Sie unter AMAZON.KendraSearchIntent.

  ```
  "dialogAction": {
   "type": "Delegate",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "kendraQueryRequestPayload": "Amazon Kendra query",
   "kendraQueryFilterString": "Amazon Kendra attribute filters"
  }
  ```

- `ElicitIntent`: Informiert Amazon Lex darüber, dass vom Benutzer als Antwort eine Äußerung erwartet wird, die eine Absicht beinhaltet. "Ich möchte eine große Pizza" gibt beispielsweise `OrderPizzaIntent` an. Die Äußerung „groß“ reicht jedoch für Amazon Lex für Amazon Lex aus, um die Absicht des Benutzers zu erkennen.

  Die Felder `message` und `responseCard` sind optional. Wenn Sie keine Mitteilung bereitstellen, verwendet Amazon Lex eine der Aufforderungen zur Klärung des Bots. Wenn keine Klärungsaufforderung definiert ist, gibt Amazon Lex eine 400 Bad Request-Ausnahme zurück.

  ```
  {
    "dialogAction": {
  ```

```
   "type": "ElicitIntent",
   "message": {
     "contentType": "PlainText or SSML or CustomPayload",
     "content": "Message to convey to the user. For example, What can I help you
with?"
   },
   "responseCard": {
     "version": integer-value,
     "contentType": "application/vnd.amazonaws.card.generic",
     "genericAttachments": [
         {
            "title":"card-title",
            "subTitle":"card-sub-title",
            "imageUrl":"URL of the image to be shown",
            "attachmentLinkUrl":"URL of the attachment to be associated with the
card",
            "buttons":[
                {
                   "text":"button-text",
                   "value":"Value sent to server on button click"
                }
             ]
          }
      ]
   }
}
```

- `ElicitSlot`— Informiert Amazon Lex darüber, dass vom Benutzer als Antwort ein Slot-Wert erwartet wird.

  Die Felder `intentName`, `slotToElicit` und `slots` sind erforderlich. Die Felder `message` und `responseCard` sind optional. Wenn Sie keine Mitteilung festlegen, verwendet Amazon Lex eine der Aufforderungen zur Eingabe eines Slots, die für den Slot konfiguriert sind.

```
 "dialogAction": {
   "type": "ElicitSlot",
   "message": {
     "contentType": "PlainText or SSML or CustomPayload",
     "content": "Message to convey to the user. For example, What size pizza would
you like?"
```

```
    },
  "intentName": "intent-name",
  "slots": {
     "slot-name": "value",
     "slot-name": "value",
     "slot-name": "value"
  },
  "slotToElicit" : "slot-name",
  "responseCard": {
     "version": integer-value,
     "contentType": "application/vnd.amazonaws.card.generic",
     "genericAttachments": [
         {
            "title":"card-title",
            "subTitle":"card-sub-title",
            "imageUrl":"URL of the image to be shown",
            "attachmentLinkUrl":"URL of the attachment to be associated with the
card",
            "buttons":[
                {
                   "text":"button-text",
                   "value":"Value sent to server on button click"
                }
             ]
          }
      ]
    }
 }
```

Amazon Lex undAWS LambdaBlueprints

Die Amazon Lex Lex-Konsole bietet vorkonfigurierte Beispiel-Bots (Bot-Pläne genannt), mit denen Sie schnell einen Bot in der Konsole erstellen und testen können. Für jeden dieser Bot-Pläne werden auch Lambda-Funktionspläne bereitgestellt. Diese Pläne enthalten Beispiel-Code, der mit ihren entsprechenden Bots funktioniert. Mit diesen Mustern können Sie schnell einen Bot erstellen, der mit einer Lambda-Funktion als Code-Haken konfiguriert ist, und die durchgehende Einrichtung prüfen, ohne erst Code schreiben zu müssen.

Sie können die folgenden Amazon Lex Bot-Pläne und die entsprechendenAWS Lambda-Funktionspläne als Code-Haken für Bots:

- Amazon Lex -`OrderFlowers`
  - AWS Lambda-Bot: Plan -`lex-order-flowers-python`
- Amazon Lex -`ScheduleAppointment`
  - AWS Lambda-Bot: Plan -`lex-make-appointment-python`
- Amazon Lex -`BookTrip`
  - AWS Lambda-Bot: Plan -`lex-book-trip-python`

Weitere Informationen dazu, wie Sie einen Bot mit einem Plan erstellen und ihn so konfigurieren, dass er eine Lambda-Funktion als Code-Haken verwendet, finden Sie unterÜbung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole)aus. Ein Beispiel für die Verwendung anderer Pläne finden Sie unter Weitere Beispiele: Amazon Lex-Bots erstellen.

Aktualisieren eines Blueprints für ein bestimmtes Gebietsschema

Wenn Sie einen Blueprint in einem anderen Gebietsschema als Englisch (USA) (DE) verwenden, müssen Sie den Namen aller Absichten aktualisieren, um das Gebietsschema einzuschließen. Wenn zum Beispiel`OrderFlowers`-Bot: Plan:

- Suchen Sie die`dispatch`Funktion am Ende des Lambda-Funktionscodes.
- In der`dispatch`aktualisieren Sie den Namen der Absicht, das von Ihnen verwendete Gebietsschema einzuschließen. Wenn Sie beispielsweise das englische (australische) (en-AU) Gebietsschema verwenden, ändern Sie die Zeile:

  ```
  if intent_name == 'OrderFlowers':
  ```

  to

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

Andere Blueprints verwenden andere Absichtsnamen, sie sollten wie oben aktualisiert werden, bevor Sie sie verwenden.

Die Bereitstellung von Amazon Lex Bots

Dieser Abschnitt bietet Beispiele für das Bereitstellen von Amazon Lex Bots auf verschiedenen Messaging-Plattformen und in mobilen Anwendungen.

Themen

Das Installieren eines Amazon Lex Bots auf einer Messaging-Plattform

In diesem Abschnitt wird erklärt, wie Amazon Lex Bots auf Facebook-, Slack- und Twilllo-Messaging-Plattformen bereitgestellt werden.

> **Note**
>
> Beim Speichern der Facebook-, Slack- oder Twilllo-Konfigurationen nutzt Amazon LexAWS Key Management ServiceVom Kunden verwaltete Schlüssel zum Verschlüsseln der Informationen. Beim ersten Erstellen eines Channels für eine dieser Messaging-Plattformen legt Amazon Lex einen vom Kunden verwalteten Standardschlüssel (`aws/lex`) enthalten. Alternativ können Sie auch Ihren eigenen vom Kunden verwalteten Schlüssel mitAWS KMSaus. Dadurch haben Sie mehr Flexibilität und können Schlüssel erstellen, rotieren und deaktivieren. Sie können auch Zugriffskontrollen definieren und die Verschlüsselungsschlüssel zum Schützen Ihrer Daten prüfen. Weitere Informationen finden Sie im AWS Key Management Service-Entwicklerhandbuch.

Wenn eine Messaging-Plattform eine Anforderung an Amazon Lex sendet, enthält die Plattform spezifische Informationen als Anforderungsattribut für Ihre Lambda-Funktion. Verwenden Sie diese Attribute, um das Verhalten Ihres Bots anzupassen. Weitere Informationen finden Sie unter Festlegen von Anforderungsattributen .

Alle Attribute haben den Namespace `x-amz-lex:` als Präfix. Beispielsweise hat das Attribut `user-id` den Namen `x-amz-lex:user-id`. Es gibt gemeinsame Attribute, die von allen Messaging-

Plattformen zusätzlich zu den Attributen gesendet werden, die für eine bestimmte Plattform spezifisch sind. Die folgende Tabelle listet die Anforderungsattribute auf, die Messaging-Plattformen an die Lambda-Funktion Ihres Bots senden.

Gemeinsame Anforderungsattribute

Attribut	Beschreibung
`channel-id`	Die Channel-Endpunktkennung von Amazon Lex.
`channel-name`	Der Channel-Name von Amazon Lex.
`channel-type`	Einer der folgenden Werte: • `Facebook` • `Kik` • `Slack` • `Twilio-SMS`
`webhook-endpoint-url`	Der Amazon Lex -Endpunkt für den Channel.

Facebook-Anforderungsattribute

Attribut	Beschreibung
`user-id`	Die Facebook-Kennung des Absenders. Siehe https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.
`facebook-page-id`	Der Facebook-Seitenkennung des Empfängers. Siehe https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.

Kik-Anforderungsattribute

Attribut	Beschreibung
`kik-chat-id`	Die Kennung für das Gespräch, an dem Ihr Bot beteiligt ist. Weitere Informationen finden Sie unter https://dev.kik.com/#/docs/messaging#message-formats.
`kik-chat-type`	Der Gesprächstyp, von dem die Nachricht stammt. Weitere Informationen finden Sie unter https://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-id`	Eine UUID, die die Nachricht bezeichnet. Weitere Informationen finden Sie unter https://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-type`	Nachrichtentyp Weitere Informationen finden Sie unter https://dev.kik.com/#/docs/messaging#message-types.

Twilio-Anforderungsattribute

Attribut	Beschreibung
`user-id`	Die Telefonnummer des Absenders („Von“). Siehe https://www.twilio.com/docs/api/rest/message.
`twilio-target-phone-number`	Die Telefonnummer des Empfängers („An“). Siehe https://www.twilio.com/docs/api/rest/message.

Slack-Anforderungsattribute

Attribut	Beschreibung
`user-id`	Die Slack-Benutzerkennung. Siehe https://api.slack.com/types/user.
`slack-team-id`	Die Kennung des Teams, das die Nachricht gesendet hat. Siehe https://api.slack.com/methods/team.info.

Attribut	Beschreibung
`slack-bot-token`	Das Entwickler-Token, das dem Bot Zugriff auf die Slack-APIs gewährt. Siehe https://api.slack.com/docs/token-types.

Integration eines Amazon Lex Lex-Bot mit Facebook Messenger

Diese Übung zeigt, wie Sie Facebook Messenger in Ihren Amazon Lex Lex-Bot integrieren. Führen Sie die folgenden Schritte aus:

1. Erstellen Sie einen Amazon Lex Lex-Bot
2. Erstellen einer Facebook-Anwendung
3. Integrieren Sie Facebook Messenger in Ihren Amazon Lex Lex-Bot
4. Validieren der Integration

Themen

Schritt 1: Erstellen eines Amazon-Lex-Bot

Wenn Sie nicht bereits über einen Amazon-Lex-Bot haben, erstellen Sie eine. In diesem Thema gehen wir davon aus, dass Sie den Bot verwenden, den Sie in der Erste-Schritte-Übung 1 erstellt haben. Sie können jedoch jeden der in diesem Handbuch bereitgestellten Beispiel-Bots verwenden. Informationen zur Erste-Schritte-Übung 1 finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).

1. Erstellen Sie einen Amazon Lex Lex-Bot. Detaillierte Anweisungen finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).
2. Installieren Sie den Bot und erstellen Sie einen Alias. Detaillierte Anweisungen finden Sie unter Übung 3: Eine Version veröffentlichen und einen Aliasnamen generieren.

Schritt 2: Erstellen von einer Facebook-Anwendung

Klicken Sie auf dem Facebook-Developer- Portal, erstellen Sie eine Facebook-Anwendung und eine Facebook-Seite. Für Instruktionen, siehe schneller Start in den Unterlagen vom Facebook Messenger-Plattform. Notieren Sie das Folgende:

- Den App Secret (App-Schlüssel) für die Facebook-App
- Das Page Access Token (Seitenzugriffstoken) für die Facebook-Seite

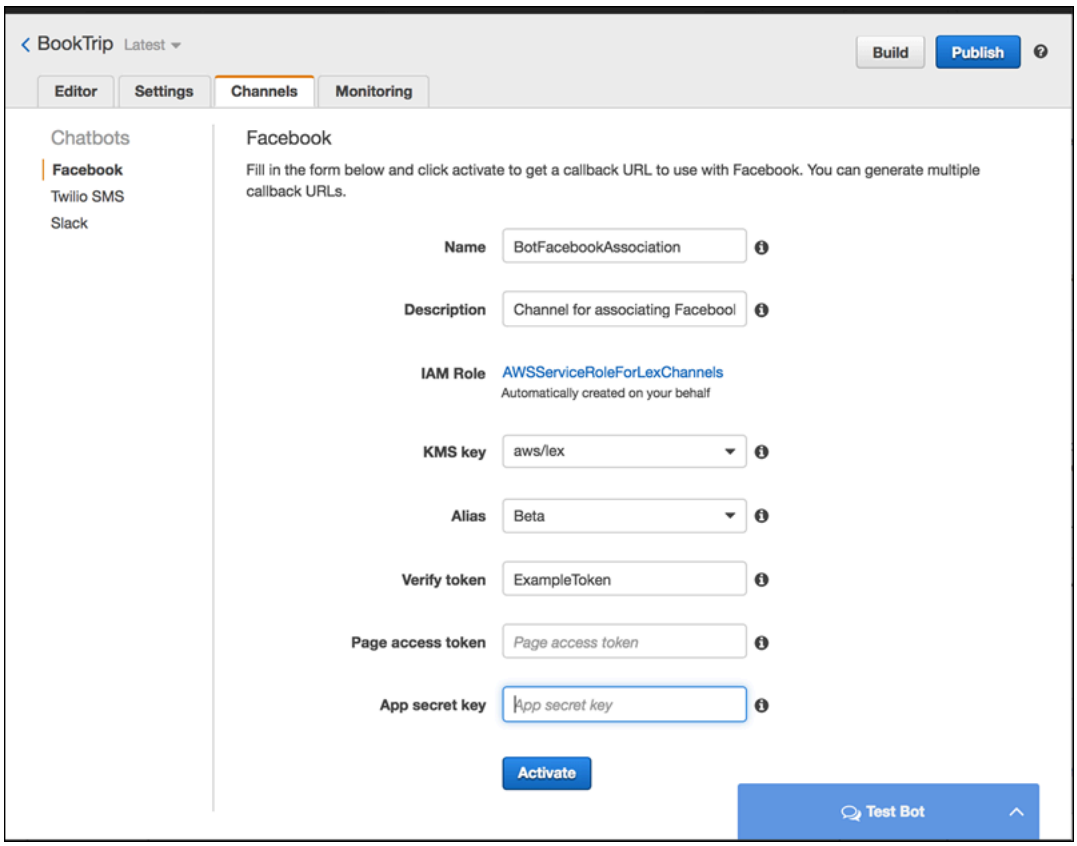
Schritt 3: Integrieren Sie den Facebook Messenger mit dem Amazon Lex Bot

In diesem Abschnitt integrieren Sie Facebook Messenger in Ihren Amazon Lex Lex-Bot.

Nachdem Sie diesen Schritt abgeschlossen haben, bietet die Konsole einen Rückruffunktion-URL. Notieren Sie diesen URL.

Integrieren von Facebook Messenger in Ihren Bot

1. a. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.
   b. Wählen Sie Ihren Amazon Lex Lex-Bot.
   c. Wählen Sie Channels aus.
   d. Wählen Sie Facebook unter Chatbots aus. Die Konsole zeigt die Facebook-Integration-Seite.
   e. Führen Sie auf der Seite für die Facebook-Integration Folgendes aus:
      - Geben Sie folgenden Namen ein: `BotFacebookAssociation`.
      - Wählen Sie in der Dropdown-Liste KMS key aws/lex aus.
      - Wählen Sie für Alias den Bot-Alias aus.
      - Geben Sie für Verify token ein Token ein. Das könnte jene Zeichenfolge sein, die Sie wählen (z.B. `ExampleToken`). Sie verwenden das Token später im Facebook-Developer-Portal, wenn Sie den Webhook generieren.
      - Geben Sie für Page access token das Token ein, das Sie in Schritt 2 von Facebook erhalten haben.
      - Geben Sie für App secret key den Schlüssel ein, den Sie in Schritt 2 von Facebook erhalten haben.



f. Wählen Sie Activate.

Die Konsole erstellt die Bot-Channel-Zuordnung und gibt eine Rückruffunktion-URL zurück. Notieren Sie diesen URL.

2. Auf dem Facebook-Developer-Portal, wählen Sie Ihre App.
3. Wählen Sie das Messenger-Produkt und Setup webhooks im Abschnitt Webhooks der Seite aus.

   Für Instruktionen, siehe schneller Start in den Unterlagen vom Facebook Messenger-Plattform.
4. Gehen Sie auf derWebhook-Seite des Abonnement-Assistenten wie folgt vor:

   - Geben Sie als Rückruf-URL die zu Beginn des Verfahrens in der Amazon Lex-Konsole angegebene Rückruf-URL ein.
   - Geben Sie für Verify Token dasselbe Token ein, das Sie in Amazon Lex verwendet haben.
   - Wählen Sie Subscription Fields (messages, messaging_postbacks und messaging_optins) aus.
   - Wählen Sie Verify and Save aus. Dadurch wird ein Handschlag zwischen Facebook und Amazon Lex ausgelöst.

5. Aktivieren Sie Webhooks-Integration. Wählen Sie die Seite, die Sie erstellt haben, und anschließend subscribe aus.

> **Note**
>
> Wenn Sie einen Webhook aktualisieren oder erneut erstellen, müssen Sie das Abonnement für die Seite erst beenden und die Seite anschließend erneut abonnieren.

Schritt 4: Testen der Integration

Sie können jetzt eine Konversation über Facebook Messenger mit Ihrem Amazon Lex Lex-Bot beginnen.

1. Öffnen Sie Ihre Facebook-Seite und wählen Sie Message aus.
2. Verwenden Sie im Messenger-Fenster die Beispiel-Äußerung, die unter Schritt 1: Erstellen eines Amazon-Lex-Botts (Amazon-Lex-Basisberechtigungen) bereitgestellt wird.

Integration eines Amazon Lex Lex-Bot mit Kik

Diese Übung enthält Anweisungen für die Integration eines Amazon Lex Lex-Bot in die Kik-Messaging-Anwendung. Führen Sie die folgenden Schritte aus:

1. Erstellen Sie einen Amazon Lex Lex-Bot.
2. Erstellen eines Kik-Bots mithilfe der Kik-App und -Website.
3. Integrieren Sie Ihren Amazon Lex Lex-Bot mithilfe der Amazon Lex Lex-Konsole in den Kik-Bot.
4. Führen Sie mithilfe von Kik eine Konversation mit Ihrem Amazon Lex Lex-Bot, um die Verbindung zwischen Ihrem Amazon Lex Lex-Bot und Kik zu testen.

Themen

Schritt 1: Erstellen eines Amazon-Lex-Bot

Wenn Sie nicht bereits über einen Amazon-Lex-Bot verfügen, erstellen und implementieren Sie einen. In diesem Thema gehen wir davon aus, dass Sie den Bot verwenden, den Sie in der Erste-Schritte-Übung 1 erstellt haben. Sie können jedoch jeden der in diesem Handbuch bereitgestellten Beispiel-Bots verwenden. Informationen zur Erste-Schritte-Übung 1 finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).

1. Erstellen Sie einen Amazon Lex Lex-Bot. Detaillierte Anweisungen finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).
2. Installieren Sie den Bot und erstellen Sie einen Alias. Detaillierte Anweisungen finden Sie unter Übung 3: Eine Version veröffentlichen und einen Aliasnamen generieren.

Nächster Schritt

Schritt 2: Erstellen eines Kik-Bots

Schritt 2: Erstellen eines Kik-Bots

In diesem Schritt verwenden Sie die Kik-Benutzeroberfläche zum Erstellen eines Kik-Bots. Sie verwenden Informationen, die bei der Erstellung des Bots generiert wurden, um ihn mit Ihrem Amazon Lex Lex-Bot zu verbinden.

1. Falls noch nicht geschehen, laden Sie die Kik-App herunter, installieren Sie sie, und melden Sie sich für ein Kik-Konto an. Wenn Sie ein Konto haben, melden Sie sich an.
2. Öffnen Sie die Kik-Website unter https://dev.kik.com/. Lassen Sie das Browser-Fenster geöffnet.
3. Wählen Sie in der Kik-App das Zahnradsymbol aus, um die Einstellungen zu öffnen, und wählen Sie anschließend Your Kik Code aus.
4. Scannen Sie den Kik-Code auf der Kik-Website, um den Botsworth-Chatbot zu öffnen. Wählen Sie Yes aus, um das Bot-Dashboard zu öffnen.
5. Wählen Sie in der Kik-App Create a Bot aus. Folgen Sie den Anweisungen zum Erstellen Ihres Kik-Bots.
6. Wenn der Bot erstellt wurde, wählen Sie in Ihrem Browser Configuration aus. Stellen Sie sicher, dass Ihr neuer Bot ausgewählt ist.
7. Notieren Sie den Bot-Namen und den API-Schlüssel für den nächsten Abschnitt.

Nächster Schritt

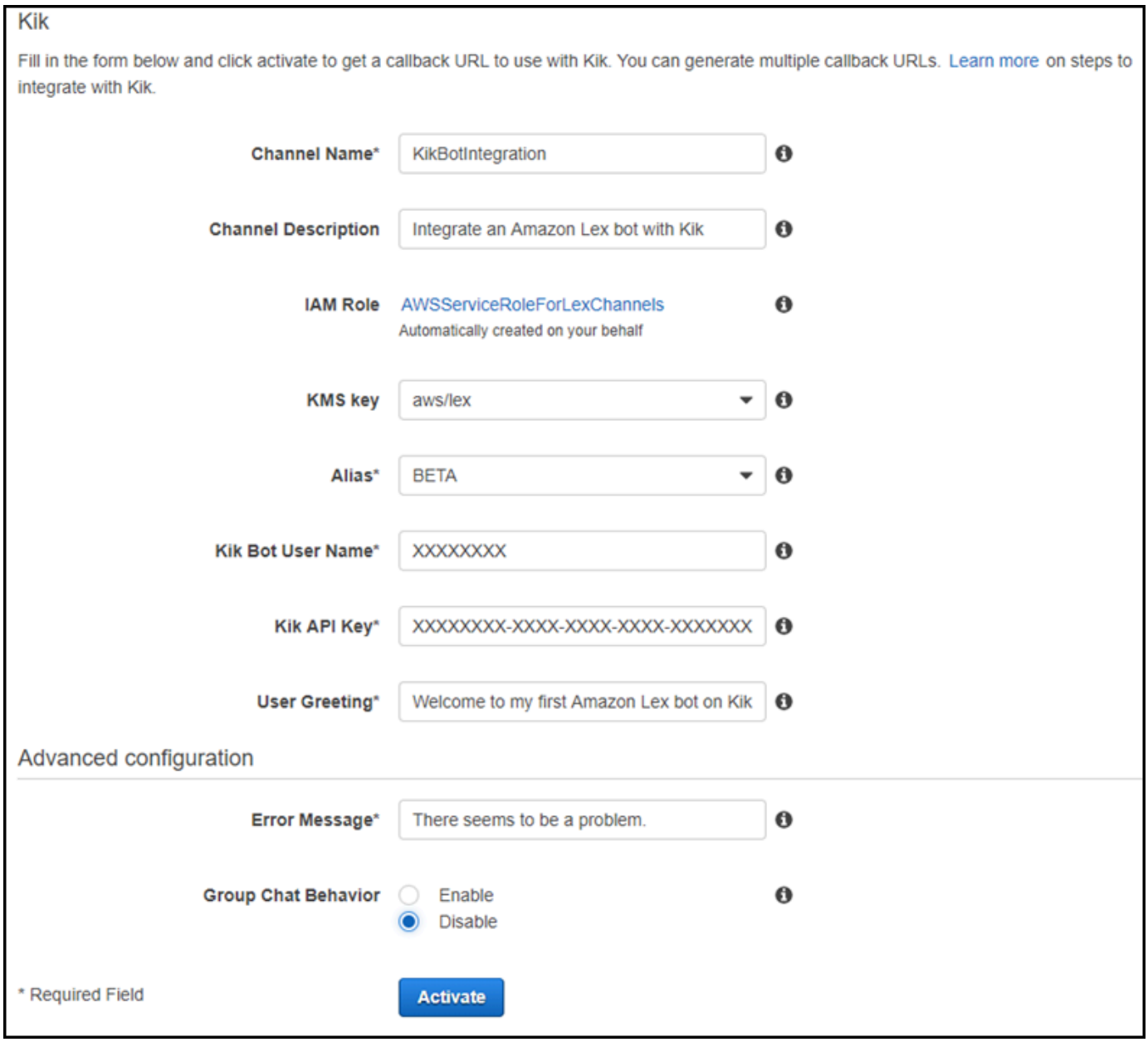
Schritt 3: Integrieren Sie den Kik Bot in den Amazon Lex Bot

Schritt 3: Integrieren Sie den Kik Bot in den Amazon Lex Bot

Nachdem Sie einen Amazon Lex-Bot und einen Kik-Bot erstellt haben, können Sie in Amazon Lex eine Kanalzuordnung zwischen ihnen erstellen. Wenn die Zuordnung aktiviert ist, richtet Amazon Lex automatisch eine Rückruf-URL mit Kik ein.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Wählen Sie den Amazon Lex Lex-Bot aus, den Sie in Schritt 1 erstellt haben.
3. Wählen Sie die Registerkarte Channels aus.
4. Wählen Sie im Bereich Channels die Option Kik aus.
5. Geben Sie auf der Kik-Seite Folgendes an:

   - Geben Sie einen Namen. Zum Beispiel `BotKikIntegration`.
   - Geben Sie eine Beschreibung ein.
   - Wählen Sie "aws/lex" aus dem Dropdown-Menü KMS key aus.
   - Wählen Sie unter Aliasein Alias aus dem Dropdown-Menü aus.
   - Geben Sie unter Kik bot user name den Namen ein, den Sie dem Bot bei Kik gegeben haben.
   - Geben Sie unter Kik API key den API-Schlüssel ein, der dem Bot bei Kik zugewiesen wurde.
   - Geben Sie unter User greeting die Grußformel ein, die der Bot senden soll, wenn ein Benutzer zum ersten Mal mit ihm kommuniziert.
   - Geben Sie unter Error message eine Fehlermeldung ein, die dem Benutzer angezeigt werden soll, wenn ein Teil der Unterhaltung nicht verstanden wurde.
   - Wählen Sie unter Group chat behavior eine der folgenden Optionen aus:
     - Enable – Erlaubt es, dass die gesamte Chat-Gruppe in einem einzigen Gespräch mit Ihrem Bot interagiert.
     - Disable – Beschränkt die Unterhaltung auf einen Benutzer in der Chat-Gruppe.
   - Wählen Sie Activate aus, um die Zuordnung zu erstellen und sie mit dem Kik-Bot zu verknüpfen.



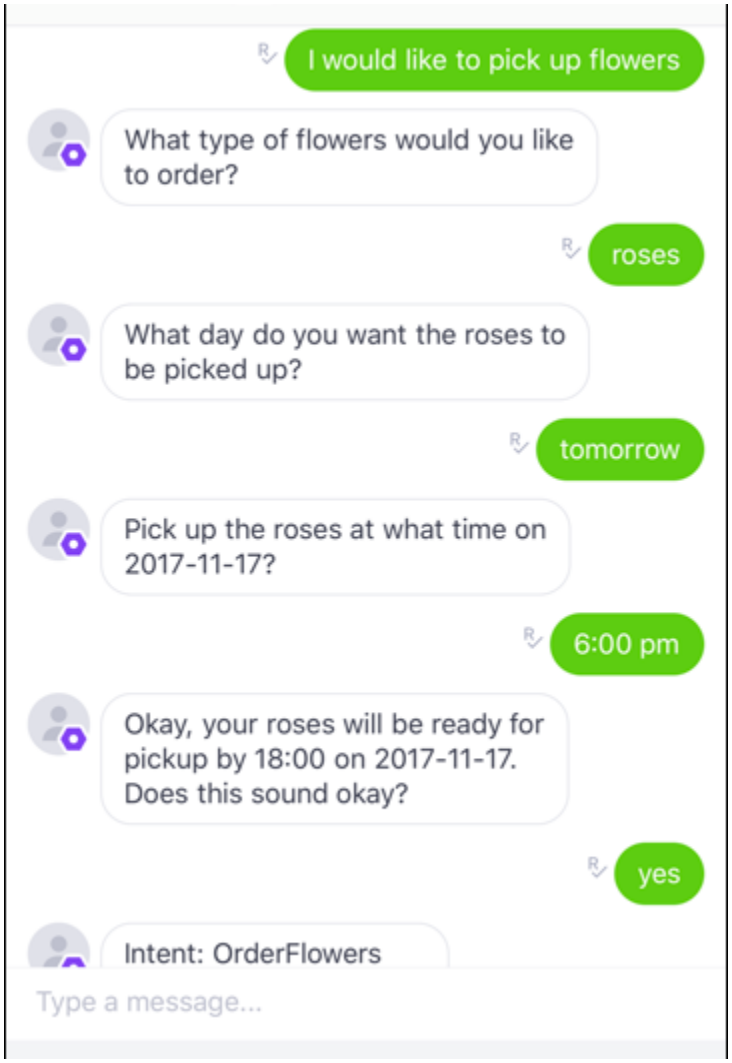
Nächster Schritt

Schritt 4: Testen der Integration

Schritt 4: Testen der Integration

Nachdem Sie eine Verknüpfung zwischen Ihrem Amazon Lex Lex-Bot und Kik erstellt haben, können Sie die Verknüpfung mit der Kik-App testen.

1. Starten Sie die Kik-App und melden Sie sich an. Wählen Sie den Bot aus, den Sie erstellt haben.
2. Sie können den Bot mit Folgendem testen:



Wenn Sie jede Phrase eingeben, antwortet Ihr Amazon Lex Lex-Bot über Kik mit der Aufforderung, die Sie für jeden Slot erstellt haben.

Integration eines Amazon Lex Lex-Bot mit Slack

Diese Übung enthält Anweisungen für die Integration eines Amazon Lex Lex-Bot in die Slack-Messaging-Anwendung. Führen Sie die folgenden Schritte aus:

1. Erstellen Sie einen Amazon Lex Lex-Bot.
2. Erstellen Sie eine Slack-Messaging-Anwendung.
3. Integrieren Sie die Slack-Anwendung in Ihren Bot Amazon Lex.

4. Testen Sie die Integration, indem Sie eine Konversation mit Ihrem Amazon Lex Lex-Bot führen. Sie senden Nachrichten mit der Slack-Anwendung und testen in einem Browserfenster.

Themen

Schritt 1: Erstellen Sie einen Amazon-Lex-Bot

Wenn Sie noch keinen Amazon-Lex-Bot haben, erstellen Sie einen und implementieren Sie ihn. In diesem Thema gehen wir davon aus, dass Sie den Bot verwenden, den Sie in der Erste-Schritte-Übung 1 erstellt haben. Sie können jedoch jeden der in diesem Handbuch bereitgestellten Beispiel-Bots verwenden. Informationen zur Erste-Schritte-Übung 1 finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).

1. Erstellen Sie einen Amazon Lex Lex-Bot. Detaillierte Anweisungen finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).
2. Installieren Sie den Bot und erstellen Sie einen Alias. Detaillierte Anweisungen finden Sie unter Übung 3: Eine Version veröffentlichen und einen Aliasnamen generieren.

Nächster Schritt

Schritt 2: Anmelden für Slack und Erstellen eines Slack-Teams

Schritt 2: Anmelden für Slack und Erstellen eines Slack-Teams

Melden Sie sich für ein Slack-Konto an und erstellen Sie ein Slack-Team. Anweisungen hierzu finden Sie unter Verwenden von Slack. Im nächsten Abschnitt erstellen Sie eine Slack-Anwendung, die jedes Slack-Team installieren kann.

Nächster Schritt

Schritt 3: Erstellen einer Slack-Anwendung

Schritt 3: Erstellen einer Slack-Anwendung

In diesem Abschnitt führen Sie folgenden Aufgaben aus:

1. Erstellen Sie eine Slack-Anwendung auf der Slack-API-Konsole
2. Konfigurieren Sie die Anwendung, um interaktives Messaging zu Ihrem Bot hinzuzufügen:

Am Ende dieses Abschnitts erhalten Sie die Anmeldeinformationen für die Anwendung (Client-ID, Clientschlüssel und Verifizierungs-Token). Im nächsten Abschnitt verwenden Sie diese Informationen, um die Bot-Kanal-Zuordnung in der Amazon Lex-Konsole zu konfigurieren.

1. Melden Sie sich an der Slack-API-Konsole unter http://api.slack.com an.
2. Erstellen Sie eine -Anwendung.

   Nachdem Sie die Anwendung erfolgreich erstellt haben, zeigt Slack die Seite Basic Information für die Anwendung an.
3. Konfigurieren Sie die Anwendungsfunktionen wie folgt:

   - Wählen Sie im linken Menü Interaktivität und Kurzbefehle.
     - Aktivieren Sie die Option, um interaktive Komponenten zu aktivieren.
     - Geben Sie im Feld Request URL eine gültige URL an. Sie können beispielsweise die Datei **`https://slack.com`** verwenden.

       > **Note**
       >
       > Geben Sie zunächst eine gültige URL ein, sodass Sie das Verifizierungs-Token erhalten, das Sie im nächsten Schritt benötigen. Sie aktualisieren diese URL, nachdem Sie die Bot-Channel-Zuordnung in der Amazon Lex-Konsole hinzugefügt haben.

     - Wählen Sie Save Changes.
4. Klicken Sie im linken Menü unter Settings auf Basic Information. Notieren Sie die folgenden Anwendungs-Anmeldeinformationen:

   - Client-ID

- Clientschlüssel
- Verifizierungstoken

Nächster Schritt

Schritt 4: Integrieren Sie die Slack-Anwendung in den Amazon Lex Lex-Bot

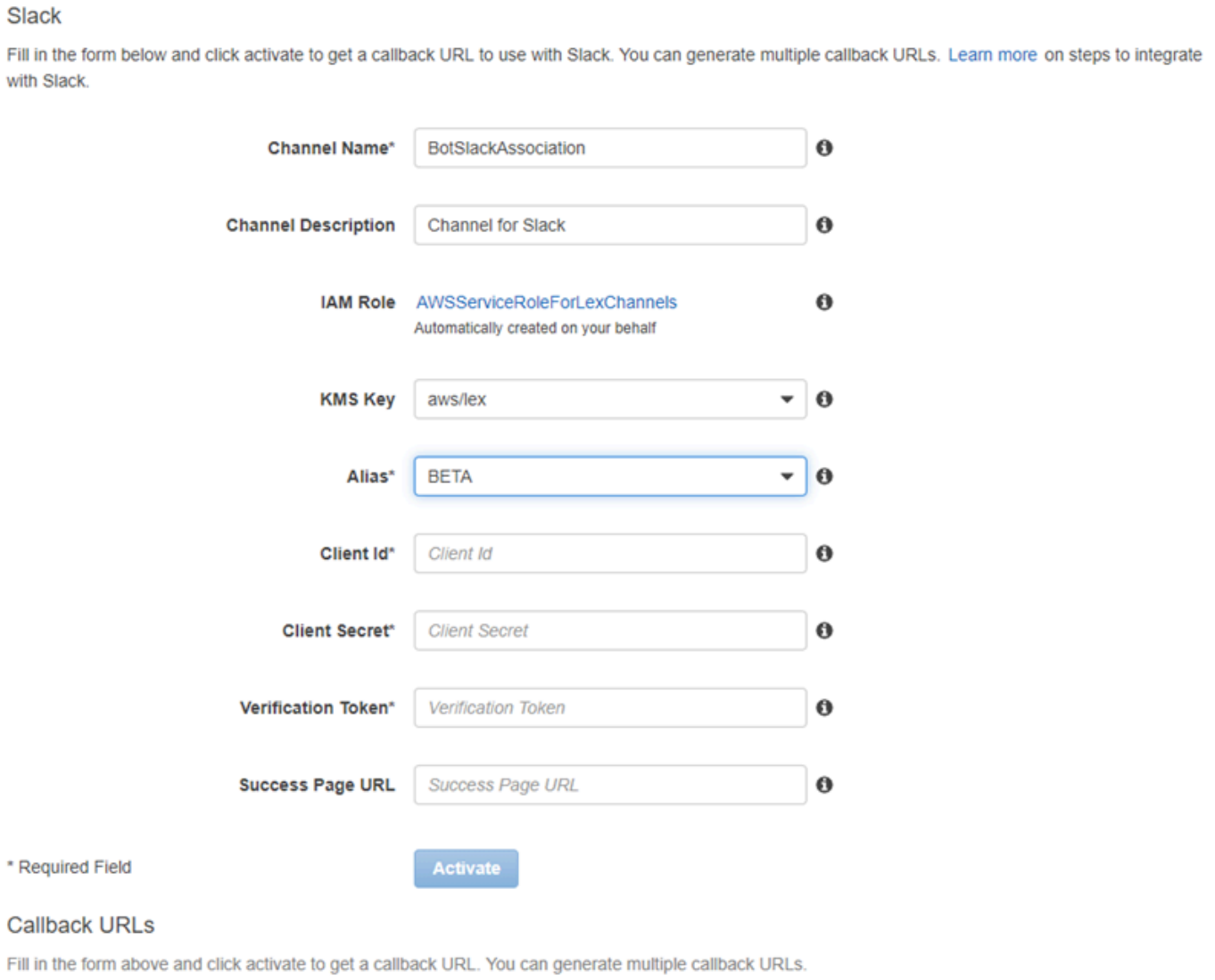
Schritt 4: Integrieren Sie die Slack-Anwendung in den Amazon Lex Lex-Bot

Da Sie nun über die Anmeldeinformationen für die Slack-Anwendung verfügen, können Sie die Anwendung in Ihren Amazon Lex Lex-Bot integrieren. Um die Slack-Anwendung mit Ihrem Bot zu verknüpfen, fügen Sie in Amazon Lex eine Bot-Channel-Zuordnung hinzu.

Aktivieren Sie in der Amazon Lex Lex-Konsole eine Bot-Channel-Zuordnung, um den Bot mit Ihrer Slack-Anwendung zu verknüpfen. Wenn die Bot-Channel-Zuordnung aktiviert ist, gibt Amazon Lex zwei URLs zurück (Postback-URL und OAuth-URL). Notieren Sie diese URLs, da Sie sie später benötigen.

Um die Slack-Anwendung in Ihren Amazon Lex Lex-Bot zu integrieren

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Wählen Sie den Amazon-Lex-Bot aus, den Sie in Schritt 1 erstellt haben.
3. Wählen Sie die Registerkarte Channels aus.
4. Wählen Sie im linken Menü Slack aus.
5. Geben Sie auf der Seite Slack Folgendes an:
   - Geben Sie einen Namen. Zum Beispiel `BotSlackIntegration`.
   - Wählen Sie "aws/lex" aus dem Dropdown-Menü KMS key aus.
   - Wählen Sie für Alias den Bot-Alias aus.
   - Geben Sie die im vorherigen Schritt notierte Client-ID, den Clientschlüssel und das Verifizierungs-Token ein. Dies sind die Anmeldeinformationen der Slack-Anwendung.



6. Wählen Sie Activate.

   Die Konsole erstellt die Bot-Channel-Zuordnung und gibt zwei URLs zurück (Postback-URL und OAuth-URL). Notieren Sie diese. Im nächsten Abschnitt aktualisieren Sie die Konfiguration Ihrer Slack-Anwendung, um diese Endpunkte wie folgt zu verwenden:

   - Die Postback-URL ist der Endpunkt des Amazon Lex Lex-Bot, der Slack-Ereignisse abhört. Sie verwenden diese URL:
     - als Anforderungs-URL in der Funktion Event Subscriptions der Slack Anwendung.
     - um den Platzhalter-Wert für die Anforderungs-URL in der Funktion Interactive Messages der Slack Anwendung zu ersetzen.

- Die OAuth-URL ist der Endpunkt Ihres Amazon Lex Lex-Bots für einen OAuth-Handshake mit Slack.

Nächster Schritt

Schritt 5: Abschließen der Slack-Integration

Schritt 5: Abschließen der Slack-Integration

In diesem Abschnitt schließen Sie die Integration der Slack-Anwendung mit der Slack-API-Konsole ab.

1. Melden Sie sich an der Slack-API-Konsole unter http://api.slack.com an. Wählen Sie die App aus, die Sie in Schritt 3: Erstellen einer Slack-Anwendung erstellt haben.
2. Aktualisieren Sie die Funktion OAuth & Permissions wie folgt:

   a. Wählen Sie im linken Menü OAuth & Permissions (OAuth und Berechtigungen) aus.

   b. Fügen Sie im Abschnitt Umleitungs-URLs die OAuth-URL hinzu, die Amazon Lex im vorherigen Schritt bereitgestellt hat. Wählen Sie Add a new Redirect URL und dann Save URLs aus.

   c. Fügen Sie im Abschnitt Bot-Token-Bereiche zwei Berechtigungen hinzu, indem Sie auf die Schaltfläche OAuth-Geltungsbereich hinzufügen klicken. Filtern Sie die Liste nach dem folgenden Text:

      - **`chat:write`**
      - **`team:read`**
3. Aktualisieren Sie die Funktion „Interaktivität und Kurzbefehle“, indem Sie den Wert für die Anfrage-URL auf die Postback-URL aktualisieren, die Amazon Lex im vorherigen Schritt bereitgestellt hat. Geben Sie die in Schritt 4 gespeicherte Postback-URL ein und wählen Sie Save Changes (Änderungen speichern).
4. Abonnieren Sie die Funktion Event Subscriptions- wie folgt:

   - Aktivieren Sie Ereignisse, indem Sie die Option On auswählen.
   - Stellen Sie den Wert für die Anforderungs-URL auf die Postback-URL ein, die Amazon Lex im vorherigen Schritt bereitgestellt hat.
   - Abonnieren Sie im Abschnitt Subscribe to Bot Events das Bot-Ereignis `message.im`, um den direkten Nachrichtenaustausch zwischen dem Endbenutzer und dem Slack-Bot zu aktivieren.

- Speichern Sie die Änderungen.

5. Aktivieren Sie das Senden von Nachrichten auf der Registerkarte „Nachrichten“ wie folgt:

   - Wählen Sie im linken Menü App Home aus.
   - Wählen Sie im Abschnitt „Tabs anzeigen“ die Option Benutzern das Senden von Slash-Befehlen und -Nachrichten vom Nachrichten-Tab aus ermöglichen.

Nächster Schritt

Schritt 6: Testen der Integration

Schritt 6: Testen der Integration

Verwenden Sie jetzt ein Browserfenster, um die Integration von Slack mit Ihrem Amazon Lex Lex-Bot zu testen.

1. Wählen Sie Manage Distribution unter Settings aus. Wählen Sie Add to Slack aus, um die Anwendung zu installieren. Autorisieren Sie den Bot dazu, auf Nachrichten zu reagieren.
2. Sie werden an Ihr Slack-Team weitergeleitet. Wählen Sie im linken Menü im Abschnitt Direct Messages Ihren Bot aus. Wenn Sie Ihren Bot nicht sehen, klicken Sie auf das Plus-Symbol (+) neben Direct Messages, um danach zu suchen.
3. Nehmen Sie an einem Chat mit Ihrer Slack-Anwendung teil, die mit dem Amazon Lex Lex-Bot verknüpft ist. Ihr Bot antwortet jetzt auf Mitteilungen.

   Wenn Sie den Bot über Erste-Schritte-Übung 1 erstellt haben, können Sie die Beispielunterhaltung verwenden, die in dieser Übung bereitgestellt wurde. Weitere Informationen finden Sie unter Schritt 4: Fügen Sie die Lambda-Funktion als Code-Hook hinzu (Konsole).

Integration eines Amazon Lex Lex-Bot mit Twilio Programmable SMS

Diese Übung enthält Anweisungen zur Integration eines Amazon Lex-Bots in den Twilio Simple Messaging Service (SMS). Führen Sie die folgenden Schritte aus:

1. Erstellen Sie einen Amazon Lex Lex-Bot
2. Integrieren Sie programmierbare Twilio-SMS in Ihren Bot Amazon Lex
3. Nehmen Sie an einer Interaktion mit dem Amazon Lex Lex-Bot teil, indem Sie das Setup mithilfe des SMS-Dienstes auf Ihrem Mobiltelefon testen.

4. Testen der Integration

Themen

Schritt 1: Erstellen eines Amazon-Lex-Bot

Wenn Sie noch kein -Programm Amazon Lex, erstellen Sie eine. In diesem Thema gehen wir davon aus, dass Sie den Bot verwenden, den Sie in der Erste-Schritte-Übung 1 erstellt haben. Sie können jedoch jeden der in diesem Handbuch bereitgestellten Beispiel-Bots verwenden. Informationen zur Erste-Schritte-Übung 1 finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).

1. Erstellen Sie einen Amazon Lex Lex-Bot. Detaillierte Anweisungen finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).
2. Installieren Sie den Bot und erstellen Sie einen Alias. Detaillierte Anweisungen finden Sie unter Übung 3: Eine Version veröffentlichen und einen Aliasnamen generieren.

Schritt 2: Erstellen eines Twilio SMS-Kontos

Melden Sie sich für ein Twilio-Konto an und erfassen Sie die folgenden Kontoinformationen:

- ACCOUNT SID
- AUTH TOKEN

Anweisungen zur Anmeldung finden Sie unter https://www.twilio.com/console.

Schritt 3: Integrieren Sie den Twilio Messaging Service Endpoint in den Amazon Lex Lex-Bot

Um Twilio in Ihren Amazon Lex Lex-Bot zu integrieren

1. Um den Amazon Lex Lex-Bot mit Ihrem programmierbaren Twilio-SMS-Endpunkt zu verknüpfen, aktivieren Sie die Bot-Kanalzuordnung in der Amazon Lex Lex-Konsole. Wenn die Bot-Channel-Zuordnung aktiviert wurde, gibt Amazon Lex eine Rückruf-URL zurück. Notieren Sie sich diesen Callback-URL, da Sie ihn später benötigen.

   a. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.

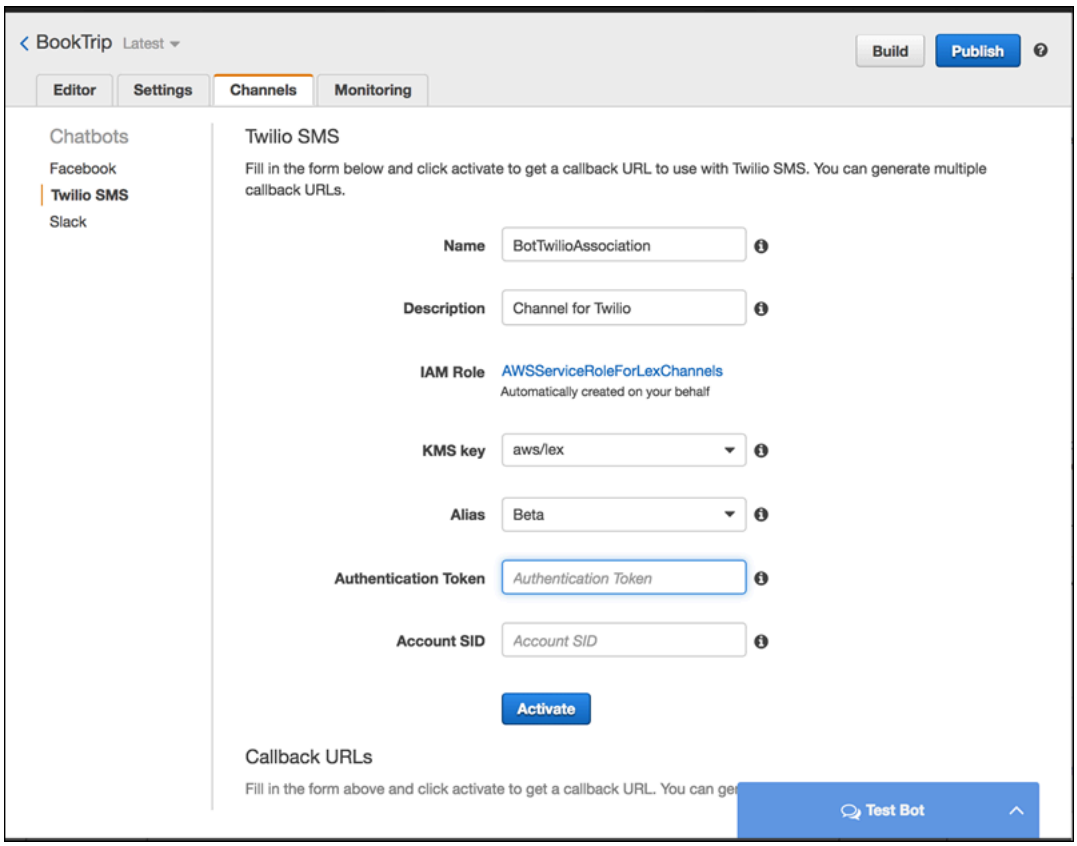
   b. Wählen Sie den Amazon-Lex-Bot aus, den Sie in Schritt 1 erstellt haben.

   c. Wählen Sie die Registerkarte Channels aus.

   d. Wählen Sie im Abschnitt Chatbots Twilio SMS aus.

   e. Geben Sie auf der Seite Twilio SMS die folgenden Informationen an:

      - Geben Sie einen Namen. Zum Beispiel `BotTwilioAssociation`.
      - Wählen Sie "aws/lex" in der Dropdown-Liste KMS key aus.
      - Wählen Sie für Alias den Bot-Alias aus.
      - Geben Sie für Authentication Token das Authentifizierungs-Token für Ihr Twilio-Konto ein.
      - Geben Sie für Account SID die Konto-SID für Ihr Twilio-Konto ein.



f. Wählen Sie Activate.

Die Konsole erstellt die Bot-Channel-Zuordnung und gibt eine Rückruffunktion-URL zurück. Notieren Sie diesen URL.

2. Verbinden Sie auf der Twilio-Konsole den Twilio-SMS-Endpunkt mit dem Amazon Lex Lex-Bot.

   a. Melden Sie sich bei der Twilio-Konsole an unter https://www.twilio.com/console.

   b. Wenn Sie keinen Twilio SMS-Endpunkt haben, erstellen Sie ihn.

   c. Aktualisieren Sie die Konfiguration der Eingangseinstellungen des Messaging-Dienstes, indem Sie den Wert REQUEST URL auf die Rückruf-URL setzen, die Amazon Lex im vorherigen Schritt bereitgestellt hat.

Schritt 4: Testen der Integration

Verwenden Sie Ihr Mobiltelefon, um die Integration zwischen Twilio SMS und Ihrem Bot zu testen.

So testen Sie die Integration

1. Melden Sie sich bei der Twilio-Konsole unter https://www.twilio.com/console an und gehen Sie wie folgt vor:

   a. Überprüfen Sie, ob eine Twilio-Nummer dem Messaging-Service unter Manage Numbers zugeordnet ist.

      Sie senden Nachrichten an diese Nummer und interagieren von Ihrem Handy aus per SMS mit dem Amazon Lex Lex-Bot.

   b. Vergewissern Sie sich, dass Ihr Mobiltelefon als verifizierte Anrufer-ID aufgeführt ist.

      Ist dies nicht der Fall, folgen Sie den Anweisungen auf der Twilio-Konsole, um das Mobiltelefon zu aktivieren, das Sie zum Testen verwenden möchten.

      Jetzt können Sie Ihr Mobiltelefon verwenden, um Nachrichten an den Twilio-SMS-Endpunkt zu senden, der dem Amazon Lex Lex-Bot zugeordnet ist.

2. Senden Sie von Ihrem Mobiltelefon aus Nachrichten an die Twilio-Nummer.

   Der Amazon Lex Lex-Bot reagiert. Wenn Sie den Bot über Erste-Schritte-Übung 1 erstellt haben, können Sie die Beispielunterhaltung verwenden, die in dieser Übung bereitgestellt wurde. Weitere Informationen finden Sie unter Schritt 4: Fügen Sie die Lambda-Funktion als Code-Hook hinzu (Konsole).

Die Bereitstellung eines Amazon Lex Bots in mobilen Anwendungen

benutzenAWS Amplifykönnen Sie Ihre Amazon Lex Bots mit mobilen oder Webanwendungen integrieren. Weitere Informationen finden Sie unterInteraktionen - Erste SchritteimAWS AmplifyDokumenteaus.

Importieren und Exportieren von Amazon Lex Lex-Bots, Absichten und Slot-Typen

Sie können einen Bot, eine Absicht oder einen Slot-Typ importieren oder exportieren. Wenn Sie zum Beispiel einen Bot mit einer Kollegin in einem anderen AWS-Konto teilen wollen, können Sie ihn exportieren und ihn dann an sie senden. Wenn Sie einem Bot mehrere Äußerungen hinzufügen möchten, können Sie ihn exportieren, die Äußerungen hinzufügen und ihn dann zurück in Ihr Konto importieren.

Sie könnenExportBots, Absichten und Slot-Typen in Amazon Lex (um sie zu teilen oder zu ändern), oder in ein Alexa Skill-Format. Sie könneneinführennur im Amazon Lex Lex-Format.

Wenn Sie eine Ressource exportieren, müssen Sie sie in einem Format exportieren, das kompatibel mit dem Service ist, an den Sie exportieren, Amazon Lex Lex-Format oder das Alexa Skills Kit. Wenn Sie einen Bot in Amazon Lex-Format exportieren, können Sie ihn zurück in Ihr Konto importieren, oder ein Amazon Lex-Benutzer in einem anderen Konto kann ihn in sein Konto importieren. Sie können einen Bot auch in einem Format exportieren, das mit einer Alexa-Skill kompatibel ist. Anschließend können Sie den Import der Bot unter Verwendung des Alexa Skills Kit importieren, damit Ihr Bot in Alexa verfügbar ist. Weitere Informationen finden Sie unter In eine Alexa-Skill exportieren .

Wenn Sie einen Bot, eine Absicht oder einen Slot-Typ exportieren, werden seine Ressourcen in eine JSON-Datei geschrieben. Um einen Bot, eine Absicht oder einen Slot-Typ zu exportieren, können Sie entweder die Amazon Lex Lex-Konsole oder dieGetExportverwenden. Importieren eines Bot, einer Absicht oder eines Slot-Typs unter Verwendung der StartImport.

Themen

Exportieren und Importieren im Amazon Lex Lex-Format

Um Bots, Absichten und Slot-Typen aus Amazon Lex zu importieren, um sie später wieder in Amazon Lex zu importieren, erstellen Sie eine JSON-Datei im Amazon Lex Lex-Format. Sie können Ihre

Ressourcen in dieser Datei bearbeiten und sie dann wieder zurück in Amazon Lex importieren. Sie können beispielsweise einer Absicht Äußerungen hinzufügen und die geänderte Absicht wieder zurück in Ihr Konto importieren. Sie können das JSON-Format auch verwenden, um eine Ressource zu teilen. Sie können beispielsweise einen Bot aus einer AWS-Region exportieren und ihn dann in eine andere Region importieren. Sie können die JSON-Datei auch an einen Kollegen senden, um einen Bot zu teilen.

Themen

Exportieren im Amazon Lex Format

Exportieren Ihrer Amazon Lex Lex-Bots, Absichten und Slot-Typen in einem Format, das Sie in ein -Format importieren könnenAWSKonto. Sie können die folgenden Ressourcen exportieren:

- Ein Bot, einschließlich aller von dem Bot verwendeten Absichten und benutzerdefinierter Slot-Typen
- Eine Absicht, einschließlich aller von der Absicht verwendeten Slot-Typen
- Ein benutzerdefinierter Slot-Typ, einschließlich aller Werte für den Slot-Typ

Sie können nur eine nummerierte Version einer Ressource exportieren. Es ist nicht möglich, die `$LATEST` Version einer Ressource zu exportieren.

Der Export ist ein asynchroner Prozess. Wenn der Export abgeschlossen ist, erhalten Sie eine von Amazon S3 S3-vorsignierte URL. Die URL gibt den Speicherort des ZIP-Archivs an, das die exportierte Ressource im JSON-Format enthält.

Sie verwenden entweder die Konsole oder die GetExport-Operation, um Bots, Absichten und benutzerdefinierte Slot-Typen zu exportieren.

Der Prozess zum Exportieren ist für Bots, Absichten oder Slot-Typen identisch. Ersetzen Sie in den folgenden Verfahren den Bot durch die Absicht oder den Slot-Typ.

Exportieren eines Bots

So exportieren Sie einen Bot

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie Botsund anschließend den zu exportierenden Bot.
3. Wählen Sie im Menü Actions (Aktionen) die Option Export.
4. Wählen Sie im Dialogfeld Export Bot (Bot exportieren) die Version des zu exportierenden Bots aus. Für Platform (Plattform) wählen Sie Amazon Lex.
5. Wählen Sie Export aus.
6. Laden Sie das .zip-Archiv herunter und speichern Sie es.

Amazon Lex exportiert den Bot in eine JSON-Datei, die im .zip-Archiv enthalten ist. Um den Bot zu aktualisieren, ändern Sie den JSON-Text und importieren ihn dann wieder zurück in Amazon Lex.

Nächster Schritt

Importieren in Amazon Lex Format

Importieren in Amazon Lex Format

Nachdem Sie eine Ressource in eine JSON-Datei im Amazon Lex Lex-Format exportiert haben, können Sie die JSON-Datei mit der Ressource in eine oder mehrere importierenAWSKonten. Sie können beispielsweise einen Bot exportieren und ihn dann in eine andere AWS-Region importieren. Sie können den Bot an eine Kollegin senden, sodass Sie ihn in ihr Konto importieren kann.

Wenn Sie einen Bot, eine Absicht oder einen Slot-Typ importieren, müssen Sie entscheiden, ob Sie die `$LATEST` Version einer Ressource beim Import überschreiben wollen, wie z. B. für eine Absicht oder einen Slot-Typ, oder ob der Import fehlschlagen soll, wenn Sie die in Ihrem Konto vorhandene Ressource beibehalten wollen. Wenn Sie beispielsweise eine bearbeitete Version einer Ressource in Ihr Konto hochladen wollen, würden Sie auswählen, die `$LATEST` Version zu überschreiben. Wenn Sie eine Ressource hochladen, die Ihnen ein Kollege gesendet hat, können Sie festlegen, dass der Import fehlschlagen soll, wenn es Ressourcenkonflikte gibt, sodass Ihre eigenen Ressourcen nicht ersetzt werden.

Beim Importieren einer Ressource, gelten die Berechtigungen, die dem Benutzer zugewiesen wurden, der die Importanfrage stellt. Der Benutzer muss über Berechtigungen für alle Ressourcen

in dem Konto verfügen, auf die sich der Import auswirkt. Der Benutzer muss zudem über die Berechtigung für die Operationen GetBot, PutBot, GetIntent PutIntent, GetSlotType und PutSlotType verfügen. Weitere Informationen zu Berechtigungen finden Sie unter So funktioniert Amazon Lex mit IAM.

Der Import meldet Fehler, die während der Verarbeitung auftreten. Einige Fehler werden gemeldet, bevor der Import beginnt, andere werden während des Importvorgangs gemeldet. Hat beispielsweise das Konto, das eine Absicht importiert, nicht die Berechtigung für den Aufruf einer Lambda-Funktion, die die Absicht verwendet, schlägt der Import fehl, bevor Änderungen an den Slot-Typen oder Absichten vorgenommen werden. Schlägt ein Import während des Importvorgangs fehl, wird die `$LATEST` Version einer Absicht oder eines Slot-Typs importiert, bevor der fehlgeschlagene Prozess geändert wird. Änderungen an der `$LATEST` Version können nicht rückgängig gemacht werden.

Wenn Sie eine Ressource importieren, werden alle abhängigen Ressourcen in die `$LATEST` Version der Ressource importiert und erhalten dann eine nummerierte Version. Verwendet beispielsweise ein Bot eine Absicht, erhält die Absicht eine nummerierte Version. Verwendet eine Absicht einen benutzerdefinierten Slot-Typ, erhält der Slot-Typ eine nummerierte Version.

Eine Ressource wird nur einmal importiert. Enthält der Bot beispielsweise die Absicht `OrderPizza` und die Absicht `OrderDrink`, die beide den benutzerdefinierten Slot-Typ `Size` verwenden, wird der Slot-Typ `Size` einmal importiert und für beide Absichten verwendet.

> **Note**
>
> Wenn du deinen Bot mit de`enableModelImprovements`parameter set auf`false`, müssen Sie die ZIP-Datei öffnen, die die Bot-Definition enthält, und ände`enableModelImprovements`parameter an`true`in den folgenden Regionen:
>
> - Asien-Pazifik (Singapur): (ap-southeast-1)
> - Asien-Pazifik (Tokyo) (ap-northeast-1)
> - EU (Frankfurt): (eu-central-1)
> - EU (London): (eu-west-2)

Der Prozess zum Importieren ist für Bots, Absichten oder benutzerdefinierter Slot-Typen identisch. Ersetzen Sie in den folgenden Verfahren den Bot nach Bedarf durch die Absicht oder den Slot-Typ.

Importieren eines Bots

So importieren Sie einen Bot

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie Botsund anschließend den zu importierenden Bot. Beim Importieren eines neuen Bots überspringen Sie diesen Schritt.
3. Wählen Sie für Actions (Aktionen) die Option Import.
4. Für Import Bot (Bot importieren) wählen Sie das .zip-Archiv, das die JSON-Datei mit dem zu importierenden Bot enthält. Wenn Sie Konflikte beim Zusammenführen erkennen wollen, bevor die Zusammenführung stattfindet, wählen Sie Notify me of merge conflicts (Benachrichtigung bei Konflikten bei der Zusammenführung). Wenn Sie die Konfliktprüfung deaktivieren, wird die `$LATEST` Version aller von dem Bot verwendeten Ressourcen überschrieben.
5. Wählen Sie Import. Wenn Sie gewählt haben, im Falle von Konflikten bei der Zusammenführung benachrichtigt zu werden, und Konflikte vorliegen, wird ein Dialogfeld angezeigt, in dem diese Konflikte aufgelistet werden. Um die `$LATEST` Version aller Konflikte erzeugenden Ressourcen zu überschreiben, wählen Sie Overwrite and continue (Überschreiben und Fortsetzen). Um den Import zu beenden, wählen Sie Cancel (Abbrechen).

Jetzt können Sie den Bot in Ihrem Konto testen.

JSON-Format für das Importieren und Exportieren

Die folgenden Beispiele zeigen die JSON-Struktur für das Exportieren und Importieren von Slot-Typen, Absichten und Bots im Amazon Lex Lex-Format.

Slot-Typ-Struktur

Nachfolgend ist die JSON-Struktur für benutzerdefinierte Slot-Typen gezeigt. Verwenden Sie diese Struktur, wenn Sie Slot-Typen importieren oder exportieren, und wenn Sie Absichten exportieren, die von benutzerdefinierten Slot-Typen abhängig sind.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
```

```
  },
  "resource": {
    "name": "slot type name",
    "version": "version number",
    "enumerationValues": [
      {
        "value": "enumeration value",
        "synonyms": []
      },
      {
        "value": "enumeration value",
        "synonyms": []
      }
    ],
    "valueSelectionStrategy": "ORIGINAL_VALUE or TOP_RESOLUTION"
  }
}
```

Absicht-Struktur

Nachfolgend ist die JSON-Struktur für Absichten gezeigt. Verwenden Sie diese Struktur, wenn Sie Absichten und Bots importieren oder exportieren, die von einer Absicht abhängig sind.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "description": "intent description",
    "rejectionStatement": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ]
    },
    "name": "intent name",
    "version": "version number",
    "fulfillmentActivity": {
      "type": "ReturnIntent or CodeHook"
```

```
    },
    "sampleUtterances": [
      "string",
      "string"
    ],
    "slots": [
      {
        "name": "slot name",
        "description": "slot description",
        "slotConstraint": "Required or Optional",
        "slotType": "slot type",
        "valueElicitationPrompt": {
          "messages": [
            {
              "contentType": "PlainText or SSML or CustomPayload",
              "content": "string"
            }
          ],
          "maxAttempts": value
        },
        "priority": value,
        "sampleUtterances": []
      }
    ],
    "confirmationPrompt": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        },
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ],
      "maxAttempts": value
    },
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Slot-Typ-Struktur.
    ]
  }
}
```

Bot-Struktur

Nachfolgend ist die JSON-Struktur für Bots gezeigt. Verwenden Sie diese Struktur, wenn Sie Bots importieren oder exportieren.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "name": "bot name",
    "version": "version number",,
    "nluIntentConfidenceThreshold": 0.00-1.00,
    "enableModelImprovements": true | false,
    "intents": [
        List of intent JSON structures.
        For more information, see Absicht-Struktur.
    ],
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Slot-Typ-Struktur.
    ],
    "voiceId": "output voice ID",
    "childDirected": boolean,
    "locale": "en-US",
    "idleSessionTTLInSeconds": timeout,
    "description": "bot description",
    "clarificationPrompt": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ],
      "maxAttempts": value
    },
    "abortStatement": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
```

```
        }
      ]
    }
  }
}
```

In eine Alexa-Skill exportieren

Sie können Ihr Bot-Schema in einem Format exportieren, das mit einer Alexa-Skill kompatibel ist. Nach dem Exportieren des Bots in eine JSON-Datei laden Sie ihn über den Skill Builder in Alexa.

So exportieren Sie einen Bot und sein Schema (Interaktionsmodell)

1. Melden Sie sich beim anAWS Management Consoleund öffnen Sie die Amazon Lex Lex-Konsole unterhttps://console.aws.amazon.com/lex/aus.
2. Wählen Sie den zu exportierenden Bot aus.
3. Wählen Sie für Actions (Aktionen) die Option Export.
4. Wählen Sie die Version des zu exportierenden Bots aus. Wählen Sie für das Format die Option Alexa Skills Kit und dann Export.
5. Wenn ein Download-Dialogfeld angezeigt wird, wählen Sie einen Speicherort für die Datei aus, und klicken Sie dann auf Save (Speichern).

Die heruntergeladene Datei ist ein .zip-Archiv, das eine Datei enthält, die nach dem exportierten Bot benannt ist. Sie enthält die Informationen, die erforderlich sind, um den Bot als Alexa-Skill zu importieren.

> **Note**
>
> Amazon Lex und das Alexa Skills Kit unterscheiden sich wie folgt:
>
> - Sitzungsattribute, die durch eckige Klammern ([]) gekennzeichnet sind, werden vom Alexa Skills Kit nicht unterstützt. Sie müssen Aufforderungen aktualisieren, die Sitzungsattribute verwenden.
> - Satzzeichen werden vom Alexa Skills Kit nicht unterstützt. Sie müssen Äußerungen aktualisieren, die Satzzeichen enthalten.

So laden Sie den Bot in eine Alexa Skill

1. Melden Sie sich Entwicklerportal unter https://developer.amazon.com/ an.
2. Wählen Sie auf der Seite Alexa Skills die Option Create Skill (Skill erstellen) aus.
3. Geben Sie auf der Seite Create a new skill (Neuen Skill erstellen) einen Namen und die Standardsprache für den Skill ein. Stellen Sie sicher, dass als Skill-Modell Custom (Benutzerdefiniert) ausgewählt ist, und wählen Sie anschließend Create skill (Skill erstellen).
4. Achten Sie darauf, dass Start from scratch (Von Grund auf neu starten) ausgewählt ist, und wählen Sie anschließend Choose (Wählen) aus.
5. Wählen Sie im linken Menü JSON Editor aus. Ziehen Sie die aus Amazon Lex exportierte JSON-Datei mit der Maus in den JSON-Editor.
6. Wählen Sie Save Model (Modell speichern), um Ihr Interaktionsmodell zu speichern.

Nachdem Sie das Schema in eine Alexa Skill hochgeladen haben, können Sie die Änderungen vornehmen, die erforderlich sind, um die Skill mit Alexa auszuführen. Weitere Informationen zum Erstellen einer Alexa-Skill finden Sie unter Use Skill Builder (Beta) im Alexa Skills Kit.

Weitere Beispiele: Amazon Lex-Bots erstellen

Die folgenden Abschnitte enthalten zusätzliche Amazon Lex-Übungen mit step-by-step Anweisungen.

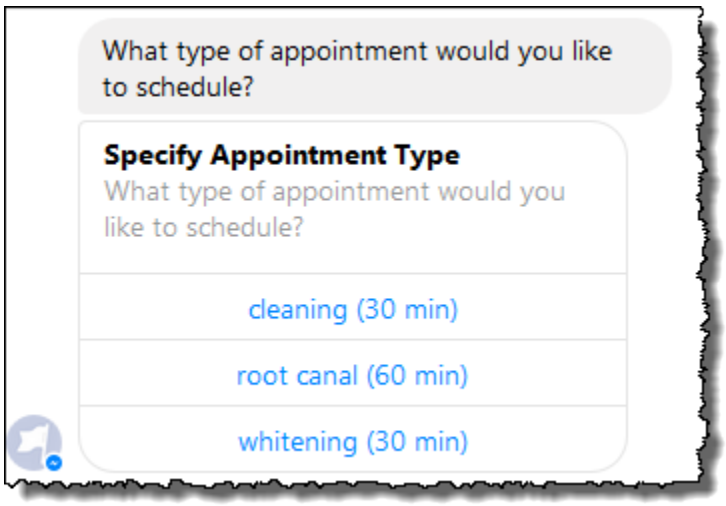
Themen

Termin vereinbaren

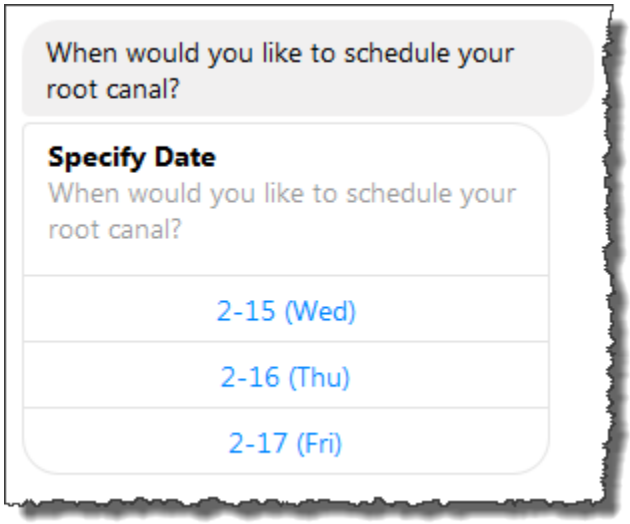
Der Beispiel-Bot in dieser Übung plant Termine für eine Zahnarztpraxis. Das Beispiel veranschaulicht auch die Verwendung von Antwortkarten, um Benutzereingaben mit Schaltflächen zu erhalten. Insbesondere veranschaulicht das Beispiel die dynamische Erzeugung von Antwortkarten zur Laufzeit.

Sie können Antwortkarten zum Zeitpunkt der Erstellung konfigurieren (auch statische Antwortkarten genannt) oder dynamisch in einer AWS Lambda-Funktion generieren. In diesem Beispiel verwendet der Bot die folgenden Antwortkarten:

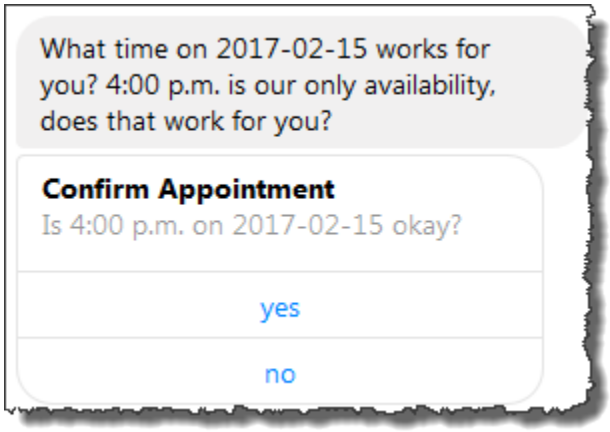
- Eine Antwortkarte, die Schaltflächen für Terminarten auflistet. In der folgenden Abbildung finden Sie ein Beispiel:



- Eine Antwortkarte, die Schaltflächen für Termindaten auflistet. In der folgenden Abbildung finden Sie ein Beispiel:



- Eine Antwortkarte, die Schaltflächen zur Bestätigung einer vorgeschlagenen Terminuhrzeit auflistet. In der folgenden Abbildung finden Sie ein Beispiel:



Die verfügbaren Termindaten und -uhrzeiten variieren, daher ist es erforderlich, Antwortkarten zur Laufzeit zu generieren. Sie verwenden eine AWS Lambda-Funktion, um diese Antwortkarten dynamisch zu generieren. Die Lambda-Funktion gibt Antwortkarten als Antwort an Amazon Lex zurück. Amazon Lex fügt die Antwortkarte in seiner Antwort an den Kunden bei.

Wenn ein Client (z. B. Facebook Messenger) Antwortkarten unterstützt, kann der Benutzer entweder aus der Liste der Schaltflächen auswählen oder die Antwort eingeben. Andernfalls gibt der Benutzer lediglich die Antwort ein.

Außer der im vorherigen Beispiel angezeigten Schaltfläche können Antwortkarten auch Bilder, Anhänge und andere nützliche Informationen enthalten. Informationen über Antwortkarten finden Sie unter Antwort-Karten.

In dieser Übung führen Sie folgende Aufgaben aus:

- Erstellen und testen Sie einen Bot (mithilfe des ScheduleAppointment Blueprints). Für diese Übung verwenden Sie einen Bot-Plan, um den Bot schnell einzurichten und zu testen. Eine Liste der verfügbaren Pläne finden Sie unter Amazon Lex undAWS LambdaBlueprints. Dieser Bot ist mit einer Absicht (`MakeAppointment`) vorkonfiguriert.

- Erstellen und testen Sie eine Lambda-Funktion (mithilfe des von Lambda bereitgestellten lex-make-appointment-python Blueprints). Sie konfigurieren die `MakeAppointment` Absicht, diese Lambda-Funktion als Code-Hook für Initialisierungs-, Validierungs- und Erfüllungsaufgaben zu verwenden.

  > **Note**
  >
  > Die bereitgestellte Lambda-Beispielfunktion zeigt eine dynamische Konversation, die auf der simulierten Verfügbarkeit eines Zahnarzttermins basiert. In einer realen Anwendung würden Sie Termine mithilfe eines realen Kalenders vereinbaren.

- Aktualisieren Sie die `MakeAppointment` Intent-Konfiguration, um die Lambda-Funktion als Code-Hook zu verwenden. Testen Sie dann die durchgehende Erfahrung.
- Veröffentlichen Sie den Bot zur Terminplanung im Facebook Messenger, damit Sie die Antwortkarten in Aktion sehen können (der Client in der Amazon Lex-Konsole unterstützt derzeit keine Antwortkarten).

In den folgenden Abschnitten finden Sie Übersichtsinformationen über die Pläne, die Sie in dieser Übung verwenden.

Themen

Überblick über den Bot-Blueprint () ScheduleAppointment

Der ScheduleAppointment Blueprint, den Sie verwenden, um einen Bot für diese Übung zu erstellen, ist wie folgt vorkonfiguriert:

- Slot-Typen: Ein benutzerdefinierter Slot-Typ namens `AppointmentTypeValue` mit den Aufzählungswerten: `root canal`, `cleaning` und `whitening`.
- Absicht: Eine Absicht (`MakeAppointment`), die wie folgt vorkonfiguriert ist:
  - Slots: Die Absicht ist mit den folgenden Slots konfiguriert:
    - Slot `AppointmentType`, von dem benutzerdefinierten Slot-Typ `AppointmentTypes`.
    - Slot `Date`, von dem integrierten Slot-Typ `AMAZON.DATE`.
    - Slot `Time`, von dem integrierten Slot-Typ `AMAZON.TIME`.
  - Äußerungen: Die Absicht ist mit den folgenden Äußerungen vorkonfiguriert:
    - „Ich möchte einen Termin buchen.“
    - „Buche einen Termin.“
    - „Buche eine {AppointmentType}“

    Wenn der Benutzer eine dieser Anweisungen ausgibt, stellt Amazon Lex fest, dass dies die Absicht `MakeAppointment` ist, und verwendet dann die Eingabeaufforderungen, um Slot-Daten abzurufen.
  - Aufforderungen: Die Absicht ist mit folgenden Aufforderungen vorkonfiguriert:
    - Aufforderung für den Slot `AppointmentType`: „Welche Art von Termin möchten Sie planen?“

- Aufforderung zur Eingabe des `Date` Termins — „Wann sollte ich Ihre {AppointmentType} einplanen?“
- Aufforderung zur Eingabe `Time` des Termins — „Zu welcher Uhrzeit möchten Sie den {AppointmentType} einplanen?“ und

  „Zu welcher Uhrzeit am ?“
- Bestätigungsaufforderung: „{Uhrzeit} ist verfügbar, soll ich fortfahren und Ihren Termin buchen?“
- Stornierungsmitteilung: „Okay, ich werde den Termin nicht planen.“

Überblick über den Lambda Function Blueprint () lex-make-appointment-python

Die Lambda-Funktion blueprint (lex-make-appointment-python) ist ein Code-Hook für Bots, die Sie mithilfe des ScheduleAppointment Bot-Blueprints erstellen.

Dieser Blueprint-Code für Lambda-Funktionen kann sowohl Initialisierungs-/Validierungs- als auch Erfüllungsaufgaben ausführen.

- Der Lambda-Funktionscode zeigt eine dynamische Konversation, die auf einem Beispiel für die Verfügbarkeit eines Zahnarzttermins basiert (in echten Anwendungen können Sie einen Kalender verwenden). Für den Tag oder das Datum, das der Benutzer angibt, wird der Code wie folgt konfiguriert:
  - Wenn keine Termine verfügbar sind, gibt die Lambda-Funktion eine Antwort zurück, in der Amazon Lex angewiesen wird, den Benutzer zur Eingabe eines anderen Tages oder Datums aufzufordern (indem der `dialogAction` Typ auf gesetzt wird. `ElicitSlot)` Weitere Informationen finden Sie unter Reaktion-Format.
  - Wenn an dem angegebenen Tag oder Datum nur ein Termin verfügbar ist, schlägt die Lambda-Funktion die verfügbare Uhrzeit in der Antwort vor und weist Amazon Lex an, eine Benutzerbestätigung einzuholen, indem das `dialogAction` in der Antwort auf `ConfirmIntent` gesetzt wird. Dies veranschaulicht, wie Sie die Benutzererfahrung verbessern können, indem sie die verfügbare Uhrzeit für einen Termin proaktiv vorschlagen.
  - Wenn mehrere Termine verfügbar sind, gibt die Lambda-Funktion als Antwort an Amazon Lex eine Liste der verfügbaren Zeiten zurück. Amazon Lex gibt dem Client eine Antwort mit der Nachricht der Lambda-Funktion zurück.

- Als Erfüllungscode-Hook gibt die Lambda-Funktion eine zusammenfassende Meldung zurück, die angibt, dass ein Termin geplant ist (d. h. die Absicht ist erfüllt).

> **Note**
>
> In diesem Beispiel wird gezeigt, wie Antwortkarten verwendet werden. Die Lambda-Funktion erstellt eine Antwortkarte und gibt sie an Amazon Lex zurück. Die Antwortkarte listet verfügbare Tage und Uhrzeiten als Schaltflächen zur Auswahl auf. Wenn Sie den Bot mit dem von der Amazon Lex-Konsole bereitgestellten Client testen, können Sie die Antwortkarte nicht sehen. Um sie zu sehen, müssen Sie den Bot mit einer Messaging-Plattform integrieren, wie etwa Facebook Messenger. Detaillierte Anweisungen finden Sie unter Integration eines Amazon Lex Lex-Bot mit Facebook Messenger. Weitere Informationen über Antwortkarten finden Sie unter Verwalten von Mitteilungen .

Wenn Amazon Lex die Lambda-Funktion aufruft, übergibt es Ereignisdaten als Eingabe. Eines der Ereignisfelder ist`invocationSource`, das die Lambda-Funktion verwendet, um zwischen einer Eingabevalidierung und einer Erfüllungsaktivität zu wählen. Weitere Informationen finden Sie unter Eingabe-Ereignis-Format.

Nächster Schritt

Schritt 1: Erstellen Sie einen Amazon Lex-Bot

Schritt 1: Erstellen Sie einen Amazon Lex-Bot

In diesem Abschnitt erstellen Sie einen Amazon Lex-Bot mithilfe des ScheduleAppointment Blueprints, der in der Amazon Lex-Konsole bereitgestellt wird.

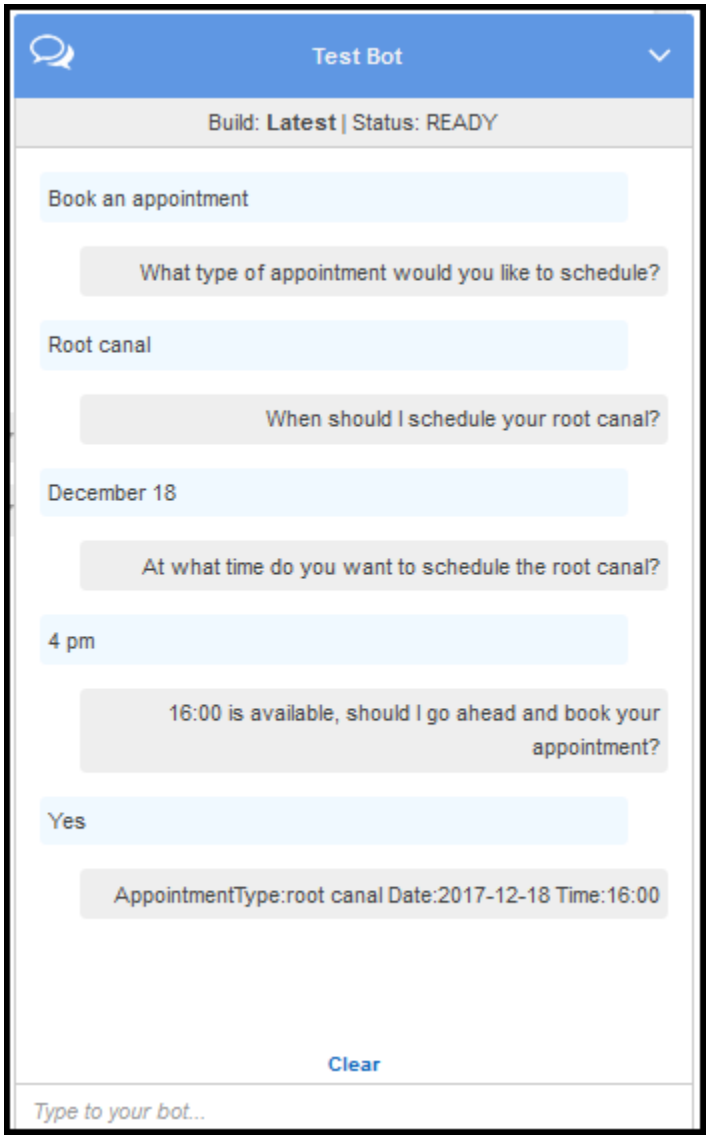
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Klicken Sie auf der Seite Bots auf die Option Create.
3. Führen Sie auf der Seite Create your Lex bot die folgenden Schritte aus:
   - Wählen Sie den Plan ScheduleAppointment aus.
   - Belassen Sie den Standard-Botnamen (ScheduleAppointment).
4. Wählen Sie Create (Erstellen) aus.

In diesem Schritt wird der Bot gespeichert und erstellt. Die Konsole sendet während des Erstellungsprozesses die folgenden Anfragen an Amazon Lex:

- Erstelle eine neue Version der Slot-Typen (aus der $LATEST Version). Weitere Informationen über in diesem Bot-Plan definierte Slot-Typen finden Sie unter Überblick über den Bot-Blueprint () ScheduleAppointment.
- Erstelle eine Version der Absicht `MakeAppointment` (von der $LATEST Version). In einigen Fällen sendet die Konsole eine Anforderung für die API-Operation `update`, bevor eine neue Version erstellt wird.
- Aktualisieren Sie die $LATEST Version des Bots.

  Derzeit erstellt Amazon Lex ein Modell für maschinelles Lernen für den Bot. Wenn Sie den Bot in der Konsole testen, verwendet die Konsole die Runtime-API, um Benutzereingaben an Amazon Lex zurückzusenden. Amazon Lex verwendet dann das Modell für maschinelles Lernen, um die Benutzereingabe zu interpretieren.

5. Die Konsole zeigt den ScheduleAppointment Bot. Prüfen Sie auf der Registerkarte Editor die Details der vorkonfigurierten Absicht (`MakeAppointment`).
6. Testen Sie den Bot im Testfenster. Verwenden Sie folgenden Screenshot, um eine Testunterhaltung mit Ihrem Bot zu führen:



Beachten Sie Folgendes:

- Aus der anfänglichen Benutzereingabe („Buche einen Termin“) erschließt der Bot die Absicht (`MakeAppointment`).
- Der Bot verwendet dann die konfigurierten Aufforderungen, um Slot-Daten vom Benutzer zu erhalten.
- Der Bot-Plan hat die Absicht `MakeAppointment` mit der folgenden Bestätigungsaufforderung konfiguriert:

```
{Time} is available, should I go ahead and book your appointment?
```

Nachdem der Benutzer alle Steckplatzdaten angegeben hat, sendet Amazon Lex eine Antwort an den Client mit einer Bestätigungsaufforderung als Nachricht zurück. Der Client zeigt die Mitteilung für den Benutzer an:

```
16:00 is available, should I go ahead and book your appointment?
```

Beachten Sie, dass der Bot alle Werte für Datum und Uhrzeit des Termins akzeptiert, da Sie keinen Code haben, um die Benutzerdaten zu initialisieren oder zu validieren. Im nächsten Abschnitt fügen Sie dazu eine Lambda-Funktion hinzu.

Nächster Schritt

Schritt 2: Erstellen Sie eine Lambda-Funktion

Schritt 2: Erstellen Sie eine Lambda-Funktion

In diesem Abschnitt erstellen Sie eine Lambda-Funktion mithilfe eines Blueprints (lex-make-appointment-python), der in der Lambda-Konsole bereitgestellt wird. Sie testen die Lambda-Funktion auch, indem Sie sie mithilfe von Amazon Lex-Beispielereignisdaten aufrufen, die von der Konsole bereitgestellt werden.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie Create a Lambda function (Eine Lambda-Funktion erstellen) aus.
3. Geben Sie unter Blueprint auswählen ein, um den Blueprint **`lex`** zu finden, und wählen Sie dann den Blueprint aus. lex-make-appointment-python
4. Konfigurieren Sie die Lambda-Funktion wie folgt.
   - Geben Sie den Namen der Lambda-Funktion (`MakeAppointmentCodeHook`) ein.
   - Wählen Sie für die Rolle Create a new role from template(s) aus und geben Sie einen Rollennamen ein.
   - Lassen Sie andere übrigen Standardwerte.
5. Wählen Sie Create Function.

6. Wenn Sie ein anderes Gebietsschema als Englisch (USA) (en-US) verwenden, aktualisieren Sie die Intentnamen wie unter beschrieben. Aktualisieren eines Blueprints für ein bestimmtes Gebietsschema
7. Testen Sie die Lambda-Funktion.

   a. Wählen Sie Actionsund anschließend Configure Test Event aus.

   b. Wählen Sie aus der Liste Sample event template Lex-Make Appointment (preview) aus. Dieses Beispielereignis verwendet das Amazon Lex-Anforderungs-/Antwortmodell, wobei die Werte so eingestellt sind, dass sie einer Anfrage Ihres Amazon Lex-Bot entsprechen. Informationen zum Amazon Lex-Anforderungs-/Antwortmodell finden Sie unter. Verwenden von Lambda-Funktionen

   c. Klicken Sie auf Save and Test (Speichern und Testen).

   d. Stellen Sie sicher, dass die Lambda-Funktion erfolgreich ausgeführt wurde. Die Antwort entspricht in diesem Fall dem Amazon Lex-Antwortmodell.

Nächster Schritt

Schritt 3: Aktualisieren der Absicht: Konfigurieren eines Code-Hakens

Schritt 3: Aktualisieren der Absicht: Konfigurieren eines Code-Hakens

In diesem Abschnitt aktualisieren Sie die Konfiguration der `MakeAppointment` Absicht, die Lambda-Funktion als Code-Hook für die Validierungs- und Erfüllungsaktivitäten zu verwenden.

1. Wählen Sie in der Amazon Lex-Konsole den ScheduleAppointment Bot aus. Die Konsole zeigt die MakeAppointmentAbsicht. Ändern Sie die Konfiguration der Absicht wie folgt:

> **Note**
>
> Sie können nur die $LATEX-Versionen aller Amazon Lex-Ressourcen aktualisieren, einschließlich der Intents. Stellen Sie sicher, dass die Version der Absicht auf $LATEST eingestellt ist. Sie haben noch keine Version Ihres Bots veröffentlicht, daher sollte immer noch die $LATEST Version in der Konsole sein.

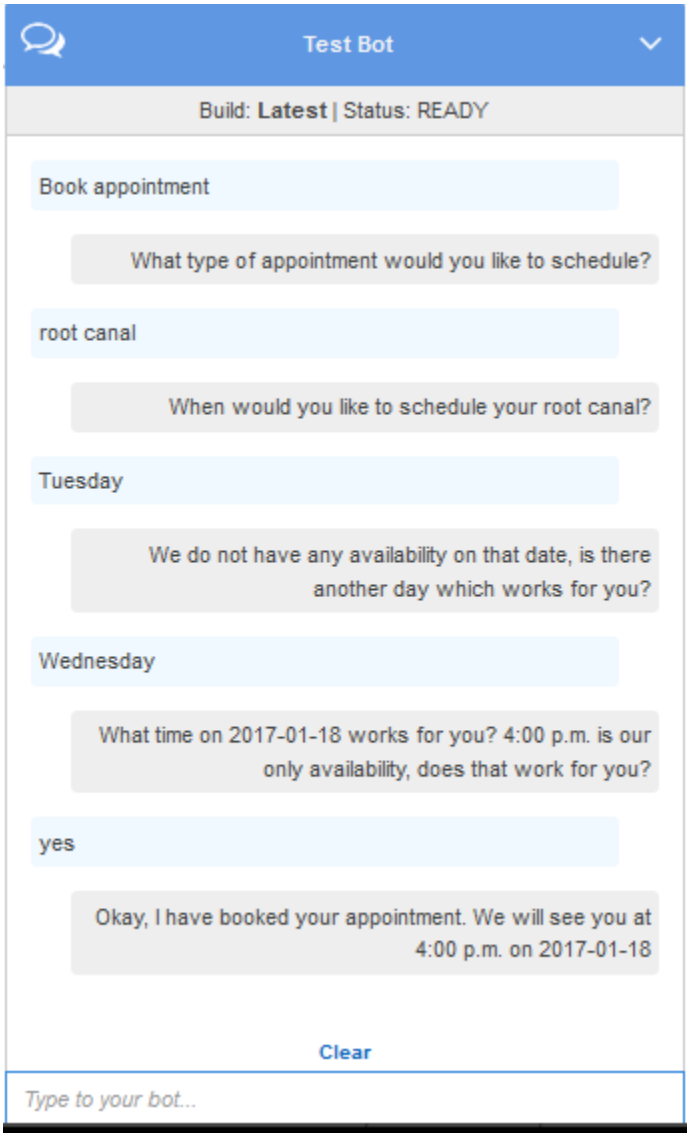
a. Wählen Sie im Abschnitt Optionen den Initialisierungs- und Validierungscode-Hook und wählen Sie dann die Lambda-Funktion aus der Liste aus.

b. Wählen Sie im Abschnitt Versand die AWS Lambda-Funktion und dann die Lambda-Funktion aus der Liste aus.

c. Wählen Sie Goodbye message aus und geben Sie eine Mitteilung ein.

2. Wählen Sie Save und dann Build aus.

3. Testen Sie den Bot wie in der folgenden Abbildung:



Nächster Schritt

Schritt 4: Bereitstellen des Bot auf der Facebook Messenger-Plattform

Schritt 4: Bereitstellen des Bot auf der Facebook Messenger-Plattform

Im vorherigen Abschnitt haben Sie den ScheduleAppointment Bot mithilfe des Clients in der Amazon Lex-Konsole getestet. Derzeit unterstützt die Amazon Lex-Konsole keine Antwortkarten. Um die dynamisch generierten Antwortkarten, die der Bot unterstützt, zu testen, stellen Sie den Bot auf der Facebook Messenger-Plattform bereit und testen Sie ihn.

Detaillierte Anweisungen finden Sie unter Integration eines Amazon Lex Lex-Bot mit Facebook Messenger.

Nächster Schritt

Informationsfluss im Detail

Informationsfluss im Detail

Der Bot-Plan `ScheduleAppointment` veranschaulicht vor allem die Verwendung von dynamisch generierten Antwortkarten. Die Lambda-Funktion in dieser Übung enthält Antwortkarten in ihrer Antwort auf Amazon Lex. Amazon Lex enthält die Antwortkarten in seiner Antwort an den Kunden. In diesem Abschnitt wird Folgendes erklärt:

- Datenfluss zwischen dem Kunden und Amazon Lex.

In diesem Abschnitt wird davon ausgegangen, dass der Client mithilfe der `PostText` Runtime-API Anfragen an Amazon Lex sendet und die Anforderungs-/Antwortdetails entsprechend anzeigt. Weitere Informationen zur `PostText` finden Sie unter PostText.

> **Note**
>
> Ein Beispiel für den Informationsfluss zwischen dem Kunden und Amazon Lex, bei dem der Client die `PostContent` API verwendet, finden Sie unterSchritt 2a (optional): Prüfen der Details des Informationsflusses gesprochener Inhalte (Konsole) .

- Datenfluss zwischen Amazon Lex und der Lambda-Funktion. Weitere Informationen finden Sie unter Lambda-Funktions-Eingabe-Ereignis und Antwort-Format.

> **Note**
>
> Im Beispiel wird davon ausgegangen, dass Sie den Facebook Messenger-Client verwenden, der in der Anfrage keine Sitzungsattribute an Amazon Lex weitergibt. Entsprechend zeigen die in diesem Abschnitt gezeigten Beispielanforderungen leere `sessionAttributes` an. Wenn Sie den Bot mit dem in der Amazon Lex-Konsole bereitgestellten Client testen, enthält der Client die Sitzungsattribute.

In diesem Abschnitt wird beschrieben, was nach jeder Benutzereingabe geschieht.

1. Benutzer: Gibt **`Book an appointment`** ein.

   a. Der Client (Konsole) sendet die folgende PostContent-Anforderung an Amazon Lex:

   ```
   POST /bot/ScheduleAppointment/alias/$LATEST/
   user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book appointment",
      "sessionAttributes":{}
   }
   ```

   Sowohl die Anforderungs-URI als auch der Text stellen Amazon Lex Informationen zur Verfügung:

   - Anforderungs-URI — Gibt den Botnamen (ScheduleAppointment), den Bot-Alias ($LATEST) und die Benutzernamen-ID an. Der abschließende `text` zeigt an, dass es eine `PostText` (nicht `PostContent`)-API-Anforderung ist.
   - Anforderungsinhalt: Enthält die Benutzereingabe (`inputText`) und leere `sessionAttributes`.

   b. Anhand des `inputText` erkennt Amazon Lex die Absicht (`MakeAppointment`). Der Dienst ruft die Lambda-Funktion auf, die als Code-Hook konfiguriert ist, um die Initialisierung und

Validierung durchzuführen, indem das folgende Ereignis übergeben wird. Details hierzu finden Sie unter Eingabe-Ereignis-Format.

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "bijt6rovckwecnzesbthrr1d7lv3ja3n",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Zusätzlich zu den vom Kunden gesendeten Informationen enthält Amazon Lex auch die folgenden Daten:

- `currentIntent`: Liefert aktuelle Informationen zur Absicht.
- `invocationSource`— Gibt den Zweck des Lambda-Funktionsaufrufs an. In diesem Fall besteht der Zweck darin, die Initialisierung und Validierung von Benutzerdaten durchzuführen. (Amazon Lex weiß, dass der Benutzer noch nicht alle Steckplatzdaten angegeben hat, um die Absicht zu erfüllen.)
- `messageVersion`— Derzeit unterstützt Amazon Lex nur die Version 1.0.

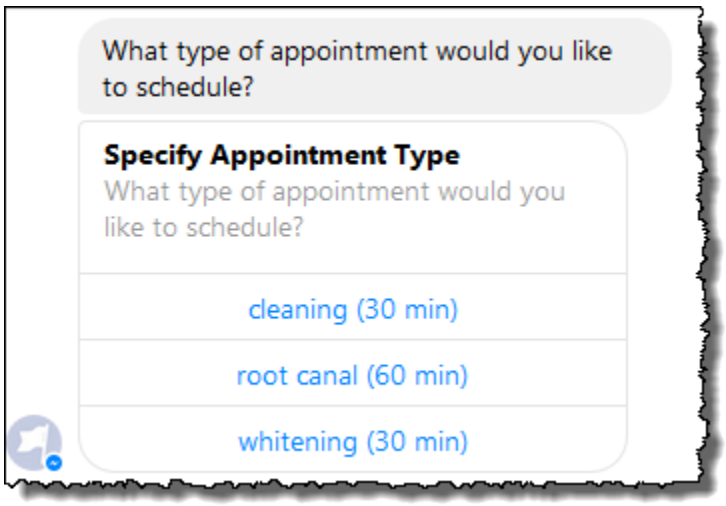
c. Derzeit sind alle Slot-Werte Null (es gibt nichts zu validieren). Die Lambda-Funktion gibt die folgende Antwort an Amazon Lex zurück und weist den Service an, Informationen für den Slot abzurufen. `AppointmentType` Weitere Informationen über das Antwortformat finden Sie unter Reaktion-Format.

```
{
```

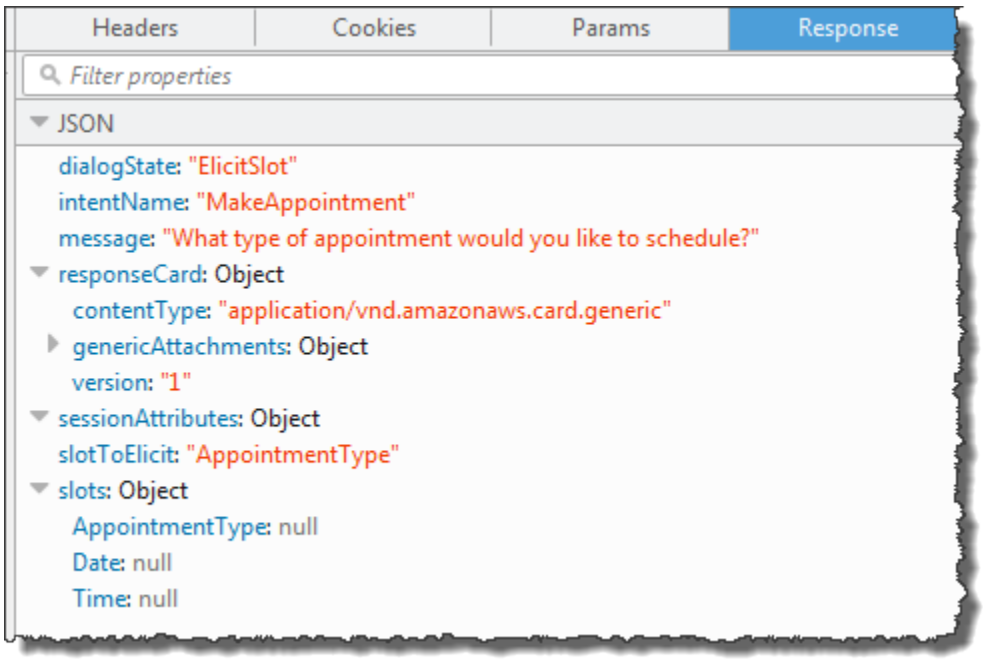
```
    "dialogAction": {
        "slotToElicit": "AppointmentType",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "cleaning (30 min)",
                            "value": "cleaning"
                        },
                        {
                            "text": "root canal (60 min)",
                            "value": "root canal"
                        },
                        {
                            "text": "whitening (30 min)",
                            "value": "whitening"
                        }
                    ],
                    "subTitle": "What type of appointment would you like to
 schedule?",
                    "title": "Specify Appointment Type"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

Die Antwort enthält die Felder `dialogAction` und `sessionAttributes`. Unter anderem gibt das Feld `dialogAction` die folgenden Felder zurück:

- `type`— Wenn Sie dieses Feld auf setzen`ElicitSlot`, weist die Lambda-Funktion Amazon Lex an, den Wert für den im Feld angegebenen Slot abzurufen. `slotToElicit` Die Lambda-Funktion bietet auch eine Möglichkeit`message`, sie dem Benutzer zu vermitteln.
- `responseCard`— Identifiziert eine Liste möglicher Werte für den `AppointmentType` Slot. Ein Client, der Antwortkarten unterstützt (z. B. der Facebook Messenger), zeigt eine Antwortkarte an, damit der Benutzer einen Termintyp auswählen kann, wie in der folgenden Abbildung dargestellt:



d. Wie aus der Antwort `dialogAction.type` der Lambda-Funktion hervorgeht, sendet Amazon Lex die folgende Antwort an den Client zurück:



Der Kunde liest die Antwort und zeigt dann die Meldung an: „Welche Art von Termin möchten Sie vereinbaren?“ und die Antwortkarte an (wenn der Client Antwortkarten unterstützt).

2. Benutzer: Je nach Client hat der Benutzer zwei Optionen:

   - Wenn die Antwortkarte angezeigt wird, wählen Sie Wurzelkanal (60 Minuten) aus oder geben **`root canal`** ein.
   - Wenn der Client keine Antwortkarten unterstützt, geben Sie **`root canal`** ein.

   a. Der Client sendet die folgende `PostText` Anfrage an Amazon Lex (aus Gründen der Lesbarkeit wurden Zeilenumbrüche hinzugefügt):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "root canal",
    "sessionAttributes": {}
}
```

   b. Amazon Lex ruft die Lambda-Funktion zur Überprüfung von Benutzerdaten auf, indem das folgende Ereignis als Parameter gesendet wird:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "bijt6rovckwecnzesbthrr1d7lv3ja3n",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Beachten Sie in den Ereignisdaten Folgendes:

- `invocationSource` ist weiterhin `DialogCodeHook`. In diesem Schritt validieren wir nur Benutzerdaten.
- Amazon Lex legt das `AppointmentType` Feld im `currentIntent.slots` Slot auf fest`root canal`.
- Amazon Lex übergibt einfach das `sessionAttributes` Feld zwischen dem Client und der Lambda-Funktion.

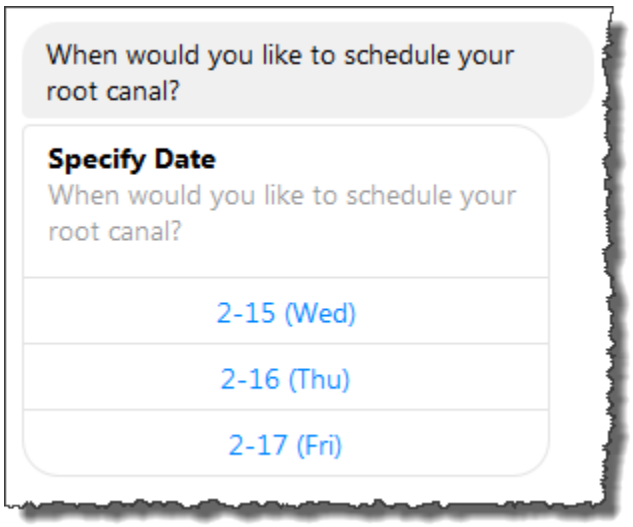
c. Die Lambda-Funktion validiert die Benutzereingabe und gibt die folgende Antwort an Amazon Lex zurück. Sie weist den Service an, einen Wert für das Termindatum zu ermitteln.

```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
```

```
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
                            "value": "Wednesday, February 15, 2017"
                        },
                        {
                            "text": "2-16 (Thu)",
                            "value": "Thursday, February 16, 2017"
                        },
                        {
                            "text": "2-17 (Fri)",
                            "value": "Friday, February 17, 2017"
                        },
                        {
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "subTitle": "When would you like to schedule your root
 canal?",
                    "title": "Specify Date"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "When would you like to schedule your root canal?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

Die Antwort enthält wieder die Felder `dialogAction` und `sessionAttributes`. Unter anderem gibt das Feld `dialogAction` die folgenden Felder zurück:

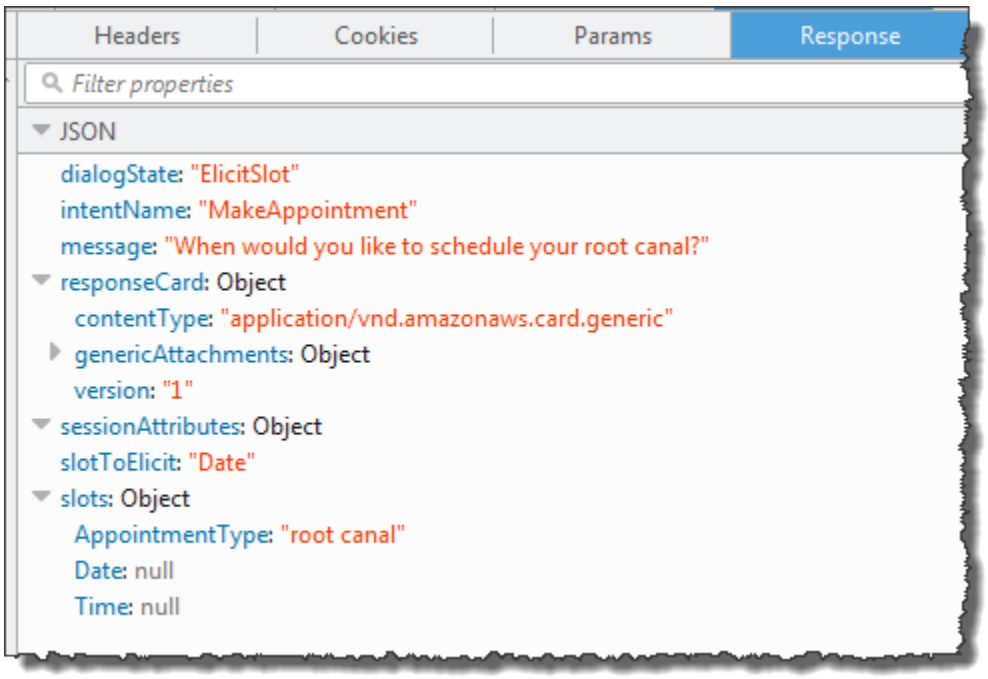
- `type`— Wenn Sie dieses Feld auf setzen`ElicitSlot`, weist die Lambda-Funktion Amazon Lex an, den Wert für den im Feld angegebenen Slot abzurufen. `slotToElicit` Die Lambda-Funktion bietet auch eine Möglichkeit`message`, sie dem Benutzer zu vermitteln.
- `responseCard`— Identifiziert eine Liste möglicher Werte für den `Date` Slot. Ein Client, der Antwortkarten unterstützt (z. B. Facebook Messenger), zeigt eine Antwortkarte an, auf der der Benutzer ein Termindatum auswählen kann, wie in der folgenden Abbildung dargestellt:



Obwohl die Lambda-Funktion fünf Daten zurückgegeben hat, hat der Client (Facebook Messenger) ein Limit von drei Schaltflächen für eine Antwortkarte. Daher sehen Sie nur die ersten drei Werte im Screenshot.

Diese Daten sind in der Lambda-Funktion fest codiert. In einer Produktionsanwendung würden Sie eventuell einen Kalender verwenden, um verfügbare Daten in Echtzeit zu erhalten. Da die Daten dynamisch sind, müssen Sie die Antwortkarte dynamisch in der Lambda-Funktion generieren.

d. Amazon Lex bemerkt das `dialogAction.type` und gibt die folgende Antwort an den Client zurück, die Informationen aus der Antwort der Lambda-Funktion enthält.



Der Client zeigt die Nachricht: When would you like to schedule your root canal? (Wann möchten Sie Ihren Wurzelkanal einplanen?) und die Antwortkarte an (wenn der Client Antwortkarten unterstützt).

3. Benutzer: Gibt **`Thursday`** ein.

   a. Der Client sendet die folgende `PostText` Anfrage an Amazon Lex (aus Gründen der Lesbarkeit wurden Zeilenumbrüche hinzugefügt):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "Thursday",
    "sessionAttributes": {}
}
```

   b. Amazon Lex ruft die Lambda-Funktion zur Überprüfung von Benutzerdaten auf, indem das folgende Ereignis als Parameter gesendet wird:

```
{
    "currentIntent": {
        "slots": {
```

```
            "AppointmentType": "root canal",
            "Date": "2017-02-16",
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Beachten Sie in den Ereignisdaten Folgendes:

- `invocationSource` ist weiterhin `DialogCodeHook`. In diesem Schritt validieren wir nur die Benutzerdaten.
- Amazon Lex legt das `Date` Feld im `currentIntent.slots` Slot auf fest`2017-02-16`.
- Amazon Lex leitet das einfach `sessionAttributes` zwischen dem Client und der Lambda-Funktion weiter.

c. Die Lambda-Funktion validiert die Benutzereingabe. Diesmal stellt die Lambda-Funktion fest, dass am angegebenen Datum keine Termine verfügbar sind. Es gibt die folgende Antwort an Amazon Lex zurück und weist den Service an, erneut einen Wert für das Termindatum zu ermitteln.

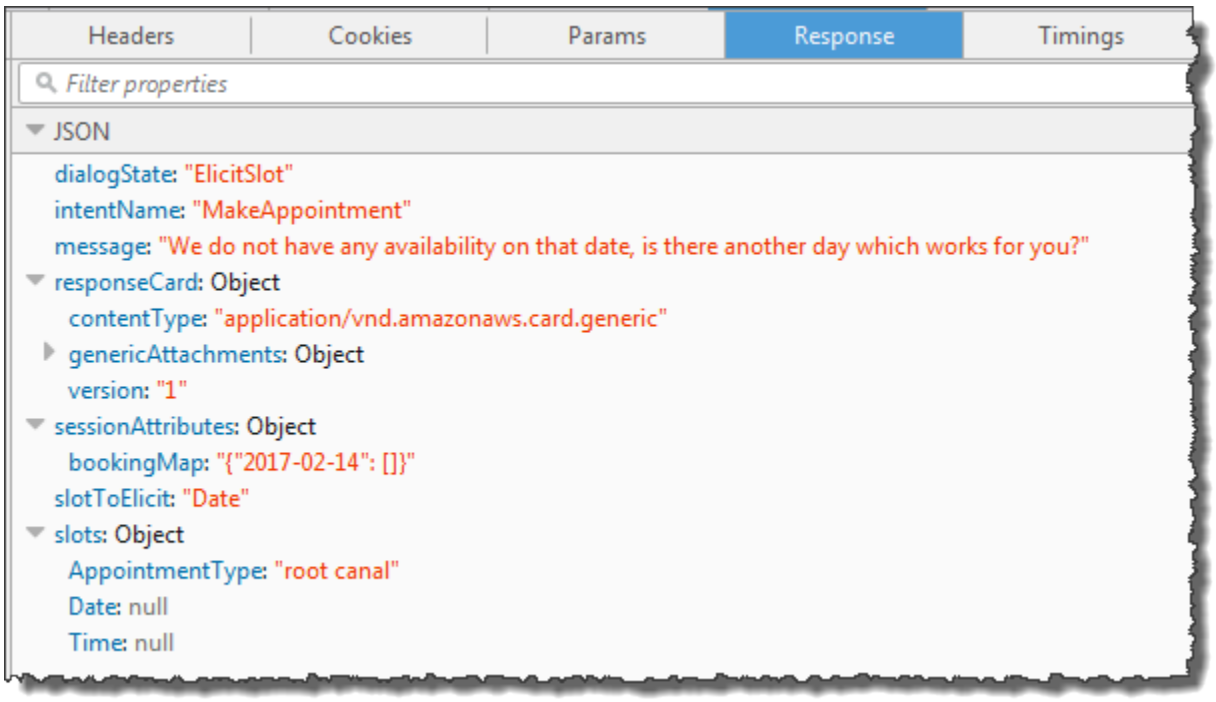
```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
```

```
                            "value": "Wednesday, February 15, 2017"
                        },
                        {
                            "text": "2-17 (Fri)",
                            "value": "Friday, February 17, 2017"
                        },
                        {
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "subTitle": "When would you like to schedule your root
 canal?",
                    "title": "Specify Date"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "We do not have any availability on that date, is there
 another day which works for you?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {
     "bookingMap": "{\"2017-02-16\": []}"
   }
}
```

Die Antwort enthält wieder die Felder `dialogAction` und `sessionAttributes`. Unter anderem gibt die `dialogAction` die folgenden Felder zurück:

- `dialogAction` field:
  - `type`— Die Lambda-Funktion setzt diesen Wert auf `ElicitSlot` und setzt das `slotToElicit` Feld auf zurück. `Date` Die Lambda-Funktion bietet auch eine entsprechende Funktion`message`, um sie dem Benutzer zu vermitteln.
  - `responseCard`: Gibt eine Liste von Werten für den Slot `Date` zurück.
- `sessionAttributes`- Diesmal enthält die Lambda-Funktion das `bookingMap` Sitzungsattribut. Sein Wert ist das gewünschte Datum des Termins und verfügbare Termine (ein leeres Objekt zeigt an, dass kein Termin verfügbar ist).

d. Amazon Lex bemerkt das `dialogAction.type` und gibt die folgende Antwort an den Client zurück, die Informationen aus der Antwort der Lambda-Funktion enthält.



Der Client zeigt die Nachricht an: Wir haben keinen Termin zu diesem Datum, gibt es einen anderen Tag, der für Sie in Frage kommt? und die Antwortkarte an (wenn der Client Antwortkarten unterstützt).

4. Benutzer: Je nach Client hat der Benutzer zwei Optionen:
   - Wenn die Antwortkarte angezeigt wird, wählen Sie 2-15 (Wed) (2-15 (Mit)) aus oder geben **`Wednesday`** ein.
   - Wenn der Client keine Antwortkarten unterstützt, geben Sie **`Wednesday`** ein.

a. Der Client sendet die folgende `PostText` Anfrage an Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "Wednesday",
    "sessionAttributes": {
   }
}
```

> **Note**
>
> Der Facebook Messenger-Client setzt keine Sitzungsattribute. Wenn Sie Sitzungsstatus zwischen Anfragen beibehalten möchten, müssen Sie dies in der Lambda-Funktion tun. In einer realen Anwendung müssen Sie diese Sitzungsattribute möglicherweise in einer Backend-Datenbank speichern.

b. Amazon Lex ruft die Lambda-Funktion zur Überprüfung von Benutzerdaten auf, indem das folgende Ereignis als Parameter gesendet wird:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "DialogCodeHook",
```

```
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {
    }
}
```

Amazon Lex wurde aktualisiert, `currentIntent.slots` indem der `Date` Slot auf gesetzt wurde`2017-02-15`.

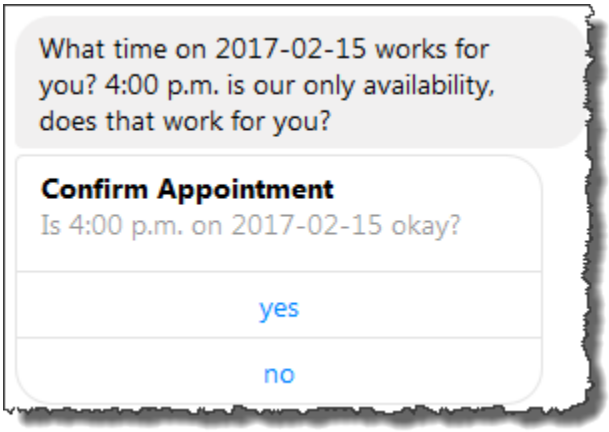
c. Die Lambda-Funktion validiert die Benutzereingabe und gibt die folgende Antwort an Amazon Lex zurück und weist Amazon Lex an, den Wert für die Terminzeit abzurufen.

```
{
    "dialogAction": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": "16:00"
        },
        "message": {
            "content": "What time on 2017-02-15 works for you? 4:00 p.m. is our
 only availability, does that work for you?",
            "contentType": "PlainText"
        },
        "type": "ConfirmIntent",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "yes",
                            "value": "yes"
                        },
                        {
                            "text": "no",
                            "value": "no"
                        }
                    ],
                    "subTitle": "Is 4:00 p.m. on 2017-02-15 okay?",
                    "title": "Confirm Appointment"
                }
            ],
```

```
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        }
    },
    "sessionAttributes": {
        "bookingMap": "{\"2017-02-15\": [\"10:00\", \"16:00\", \"16:30\"]}"
    }
}
```

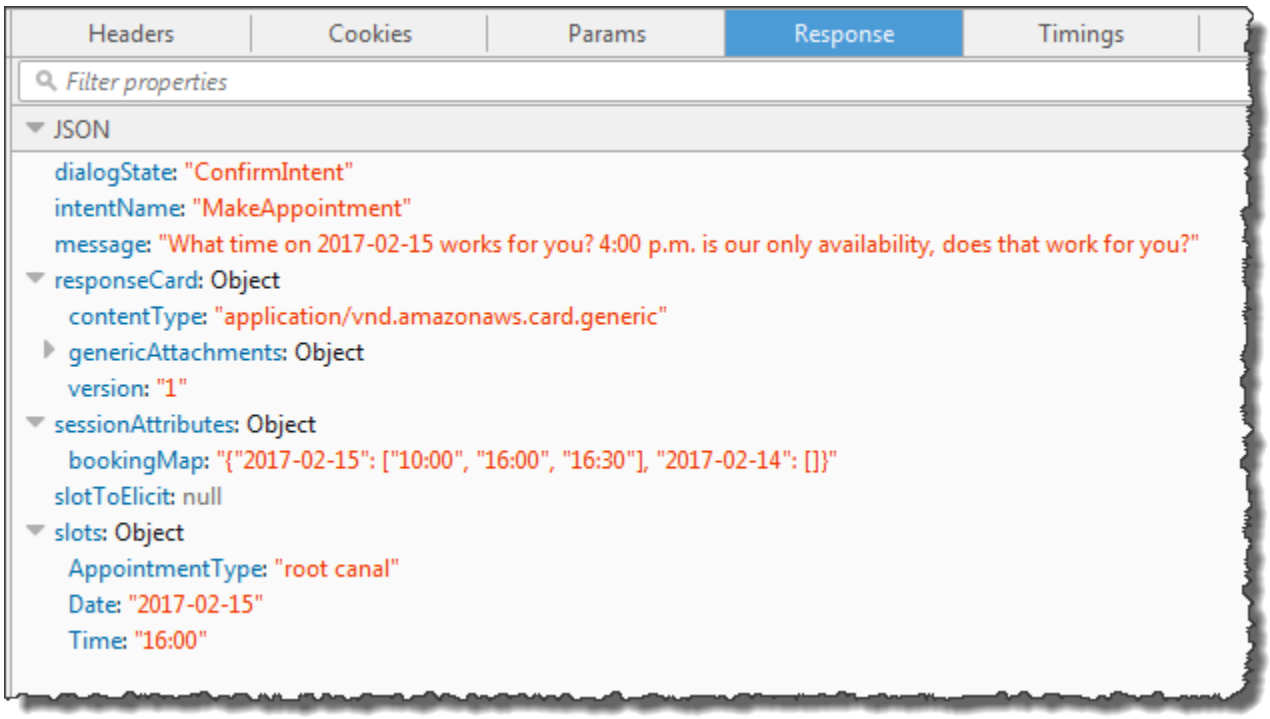
Die Antwort enthält wieder die Felder `dialogAction` und `sessionAttributes`. Unter anderem gibt die `dialogAction` die folgenden Felder zurück:

- `dialogAction` field:
  - `type`— Die `Lambda` Funktion legt diesen Wert auf fest und `ConfirmIntent` weist Amazon Lex an, vom Benutzer eine Bestätigung der in der `message` vorgeschlagenen Terminzeit einzuholen.
  - `responseCard`— Gibt eine Liste mit Ja/Nein-Werten zurück, aus denen der Benutzer wählen kann. Wenn der Client Antwortkarten unterstützt, zeigt er die Antwortkarte an, wie in folgendem Beispiel gezeigt:



- `sessionAttributes`- Die Lambda-Funktion setzt das `bookingMap` Sitzungsattribut mit seinem Wert auf das Termindatum und die verfügbaren Termine an diesem Datum. In diesem Beispiel sind es Termine von je 30 Minuten. Für eine Wurzelbehandlung, die eine Stunde dauert, kann nur der Termin um 16 Uhr gebucht werden.

d. Wie `dialogAction.type` in der Antwort der Lambda-Funktion angegeben, gibt Amazon Lex die folgende Antwort an den Client zurück:



Der Client zeigt die Meldung an: Welche Uhrzeit am 15.02.2017 funktioniert für Sie? 16:00 Uhr ist unsere einzige Verfügbarkeit, funktioniert das für Sie?

5. Benutzer: Wählt **`yes`**.

   Amazon Lex ruft die Lambda-Funktion mit den folgenden Ereignisdaten auf. Da der Benutzer geantwortet hat**`yes`**, legt Amazon Lex das `confirmationStatus` bis `Confirmed` und das `Time` Feld `currentIntent.slots` auf fest`4 p.m`.

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": "16:00"
        },
        "name": "MakeAppointment",
        "confirmationStatus": "Confirmed"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
```

```
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "FulfillmentCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {
   }
}
```

Da der bestätigt `confirmationStatus` wird, verarbeitet die Lambda-Funktion die Absicht (bucht einen Zahnarzttermin) und gibt die folgende Antwort an Amazon Lex zurück:

```
{
    "dialogAction": {
        "message": {
            "content": "Okay, I have booked your appointment.  We will see you at
 4:00 p.m. on 2017-02-15",
            "contentType": "PlainText"
        },
        "type": "Close",
        "fulfillmentState": "Fulfilled"
    },
    "sessionAttributes": {
        "formattedTime": "4:00 p.m.",
        "bookingMap": "{\"2017-02-15\": [\"10:00\"]}"
    }
}
```

Beachten Sie Folgendes:

- Die Lambda-Funktion hat die `sessionAttributes` aktualisiert.
- `dialogAction.type`eist auf gesetzt`Close`, was Amazon Lex anweist, keine Benutzerantwort zu erwarten.
- `dialogAction.fulfillmentState` wird auf `Fulfilled` gesetzt, wodurch angezeigt wird, dass die Absicht erfolgreich erfüllt wurde.

Der Kunde zeigt die Meldung an: Okay, ich habe Ihren Termin gebucht. Wir sehen uns am 15.02.2017 um 16:00 Uhr.

Reise buchen

Dieses Beispiel veranschaulicht das Erstellen eines Bots, der konfiguriert wurde, um mehrere Absichten zu unterstützen. Das Beispiel veranschaulicht auch, wie Sie Sitzungsattribute für die gemeinsame Nutzung von Informationen für mehrere Absichten verwenden können. Nachdem Sie den Bot erstellt haben, verwenden Sie einen Testclient in der Amazon Lex Lex-Konsole, um den Bot zu testen (BookTrip). Der Client verwendet denPostText Runtime-API-Vorgang, um für jede Benutzereingabe Anfragen an Amazon Lex zu senden.

Der BookTrip Bot in diesem Beispiel ist mit zwei Absichten (BookHotel und BookCar) konfiguriert. Angenommen zum Beispiel, ein Benutzer bucht zuerst ein Hotel. Während der Interaktion gibt der Benutzer Informationen an wie z. B. Anreisedaten, Position und Anzahl der Nächte. Nachdem die Absicht erfüllt ist, kann der Client diese Informationen mithilfe von Sitzungsattributen speichern. Weitere Informationen über Sitzungsattribute finden Sie unter PostText.

Angenommen, dass der Benutzer jetzt damit fortfährt, ein Auto zu buchen. Mithilfe der Informationen, die der Benutzer in der vorherigen BookHotel Absicht angegeben hat (d. h. Zielort sowie An- und Abreisetag), initialisiert der Code-Hook (Lambda-Funktion), den Sie für die Initialisierung und Validierung der BookCar Absicht konfiguriert haben, die Slot-Daten für die BookCar Absicht (d. h. Ziel, Abholort, Abholdatum und Rückgabedatum). Dies veranschaulicht, wie die gemeinsame Nutzung von Informationen für mehrere Absichten dazu beiträgt, Bots zu erstellen, die dynamische Unterhaltungen mit dem Benutzer führen können.

In diesem Beispiel verwenden wir die folgenden Sitzungsattribute. Nur der Client und die Lambda-Funktion können Sitzungsattribute festlegen und aktualisieren. Amazon Lex leitet diese Daten nur zwischen dem Client und der Lambda-Funktion aus. Amazon Lex verwaltet oder ändert keine Sitzungsattribute.

- `currentReservation`— Enthält Slot-Daten für eine laufende Reservierung und andere relevante Informationen. Folgendes ist zum Beispiel eine Beispielanforderung des Clients an Amazon Lex. Das Sitzungsattribut `currentReservation` wird im Anforderungstext gezeigt.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
```

```
    "inputText":"Chicago",
    "sessionAttributes":{
        "currentReservation":"{\"ReservationType\":\"Hotel\",
                               \"Location\":\"Moscow\",
                               \"RoomType\":null,
                               \"CheckInDate\":null,
                               \"Nights\":null}"
    }
}
```

- `lastConfirmedReservation`— Enthält ähnliche Informationen für eine frühere Absicht, falls vorhanden. Wenn der Benutzer beispielsweise ein Hotel gebucht hat und dann gerade dabei ist, ein Auto zu buchen, speichert dieses Sitzungsattribut Slot-Daten für die vorherige BookHotel Absicht.

- `confirmationContext`— Die Lambda-Funktion stellt dies so ein, dass einige der Slot-Daten auf der Grundlage von Slot-Daten aus der vorherigen Reservierung (falls vorhanden) vorab aufgefüllt werden.`AutoPopulate` Dies ermöglicht die gemeinsame Nutzung von Informationen für mehrere Absichten. Wenn der Benutzer beispielsweise zuvor ein Hotel gebucht hat und jetzt ein Auto buchen möchte, kann Amazon Lex den Benutzer auffordern, zu bestätigen (oder abzulehnen), dass das Auto für dieselbe Stadt und dieselben Daten wie seine Hotelreservierung gebucht wird.

In dieser Übung verwenden Sie Blueprints, um einen Amazon Lex Lex-Bot und eine Lambda-Funktion zu erstellen. Weitere Informationen über Pläne finden Sie unter Amazon Lex undAWS LambdaBlueprints.

Nächster Schritt

Schritt 1: Überprüfen der in dieser Übung verwendeten Pläne

Schritt 1: Überprüfen der in dieser Übung verwendeten Pläne

Themen

- Überblick über den Bot-Blueprint (BookTrip)

- Überblick über den Lambda-Funktionsbauplan (lex-book-trip-python)

Überblick über den Bot-Blueprint (BookTrip)

Der Plan (BookTrip), den Sie zum Erstellen eines Bot verwenden, bietet folgende Vorkonfiguration:

- Slot-Typen: Zwei benutzerdefinierte Slot-Typen:
  - `RoomTypes` mit Aufzählungswerten: `king`, `queen` und `deluxe`, zur Verwendung in der Absicht `BookHotel`.
  - `CarTypes` mit Aufzählungswerten: `economy`, `standard`, `midsize`, `full size`, `luxury` und `minivan`, zur Verwendung in der Absicht `BookCar`.

- Absicht 1 (BookHotel) — Es ist wie folgt vorkonfiguriert:
  - Vorkonfigurierte Slots
    - `RoomType`, von dem benutzerdefinierten Slot-Typ `RoomTypes`
    - `Location`, von dem integrierten Slot-Typ `AMAZON.US_CITY`
    - `CheckInDate`, von dem integrierten Slot-Typ `AMAZON.DATE`
    - `Nights`, von dem integrierten Slot-Typ `AMAZON.NUMBER`
  - Vorkonfigurierte Äußerungen
    - „Buche ein Hotel“
    - „Ich möchte eine Hotelreservierung vornehmen“
    - „Buche einen Aufenthalt für in“

    Wenn der Benutzer eine dieser Optionen ausspricht, bestimmt Amazon Lex, dass dies die Absicht`BookHotel` ist, und fordert den Benutzer dann zur Eingabe von Slot-Daten auf.
  - Vorkonfigurierte Aufforderungen
    - Aufforderung für den Slot `Location`: „In welcher Stadt werden Sie bleiben?“
    - Aufforderung für den Slot `CheckInDate`: „An welchem Tag möchten Sie anreisen?“
    - Aufforderung für den Slot `Nights`; „Wie viele Nächte werden Sie bleiben?“
    - Aufforderung für den Slot `RoomType`: „Welche Art von Zimmer möchten Sie buchen: Queen, King oder Deluxe?“
    - Bestätigungserklärung — „Okay, ich habe dich für eine {Nächte} -Übernachtung in {Location} ab {CheckInDate} gebucht. Soll ich die Reservierung vornehmen?“

    - Ablehnung: „Okay, ich habe Ihre laufende Reservierung abgebrochen.“

- Absicht 2 (BookCar) — Es ist wie folgt vorkonfiguriert:
  - Vorkonfigurierte Slots
    - `PickUpCity`, von dem integrierten Typ `AMAZON.US_CITY`
    - `PickUpDate`, von dem integrierten Typ `AMAZON.DATE`
    - `ReturnDate`, von dem integrierten Typ `AMAZON.DATE`
    - `DriverAge`, von dem integrierten Typ `AMAZON.NUMBER`
    - `CarType`, von dem benutzerdefinierten Typ `CarTypes`
  - Vorkonfigurierte Äußerungen
    - „Buche ein Auto“
    - „Reserviere ein Auto“
    - „Nimm eine Autoreservierung vor“

    Wenn der Benutzer eine dieser Angaben ausspricht, bestimmt BookCar Amazon Lex die Absicht und fordert den Benutzer dann zur Eingabe von Slot-Daten auf.
  - Vorkonfigurierte Aufforderungen
    - Aufforderung für den Slot `PickUpCity`: „In welcher Stadt möchten Sie ein Auto mieten?“
    - Aufforderung für den Slot `PickUpDate`: „An welchem Tag möchten Sie die Miete beginnen?“
    - Aufforderung für den Slot `ReturnDate`: „An welchem Tag möchten Sie dieses Auto zurückgeben?“
    - Aufforderung für den Slot `DriverAge`: „Wie alt ist der Fahrer für diese Miete?“
    - Aufforderung zur Eingabe`CarType` des Zeitplans — „Welchen Autotyp möchten Sie mieten? Unsere beliebtesten Optionen sind Economy, Mittelklasse und Luxusklasse.“
    - Bestätigungserklärung — „Okay, ich habe dich für eine {CarType} Vermietung in {PickUpCity} von {PickUpDate} bis {ReturnDate} gebucht. Soll ich die Reservierung vornehmen?“
    - Ablehnung: „Okay, ich habe Ihre laufende Reservierung abgebrochen.“

Überblick über den Lambda-Funktionsbauplan (lex-book-trip-python)

Zusätzlich zu dem Bot-Plan bietet AWS Lambda einen Plan (lex-book-trip-python), den Sie als Code Haken mit dem Bot-Plan verwenden können. Eine Liste der Bot-Blueprints und der entsprechenden Lambda-Funktions-Blueprints finden Sie unterAmazon Lex undAWS LambdaBlueprints.

Wenn Sie einen Bot mithilfe des BookTrip Blueprints erstellen, aktualisieren Sie die Konfiguration sowohl der Absichten (BookCar als auch BookHotel), indem Sie diese Lambda-Funktion als Code-Hook sowohl für die Initialisierung/Validierung der Benutzerdateneingabe als auch für die Erfüllung der Absichten hinzufügen.

Dieser bereitgestellte Lambda-Funktionscode ist ein Beispiel für eine dynamische Unterhaltung, in der zuvor bekannte (in Sitzungsattributen gespeicherte) Informationen über einen Benutzer verwendet werden, um Slot-Werte für eine Absicht zu initialisieren. Weitere Informationen finden Sie unter Verwaltung von Konversation-Kontext.

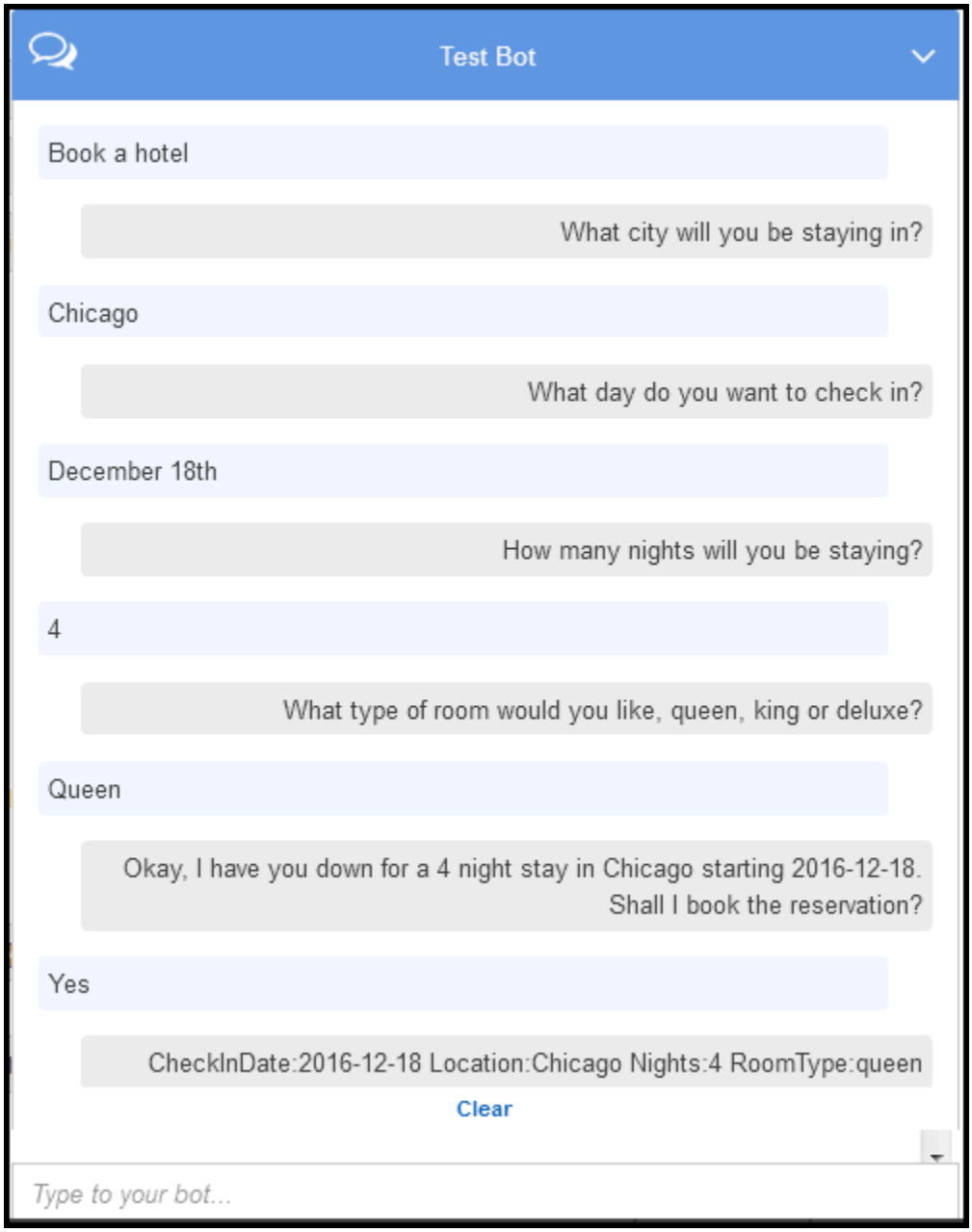
Nächster Schritt

Schritt 2: Erstellen eines Erstellen eines Amazon-Lex-Bot

Schritt 2: Erstellen eines Erstellen eines Amazon-Lex-Bot

In diesem Abschnitt erstellen Sie einen Amazon Lex Lex-Bot (BookTrip).

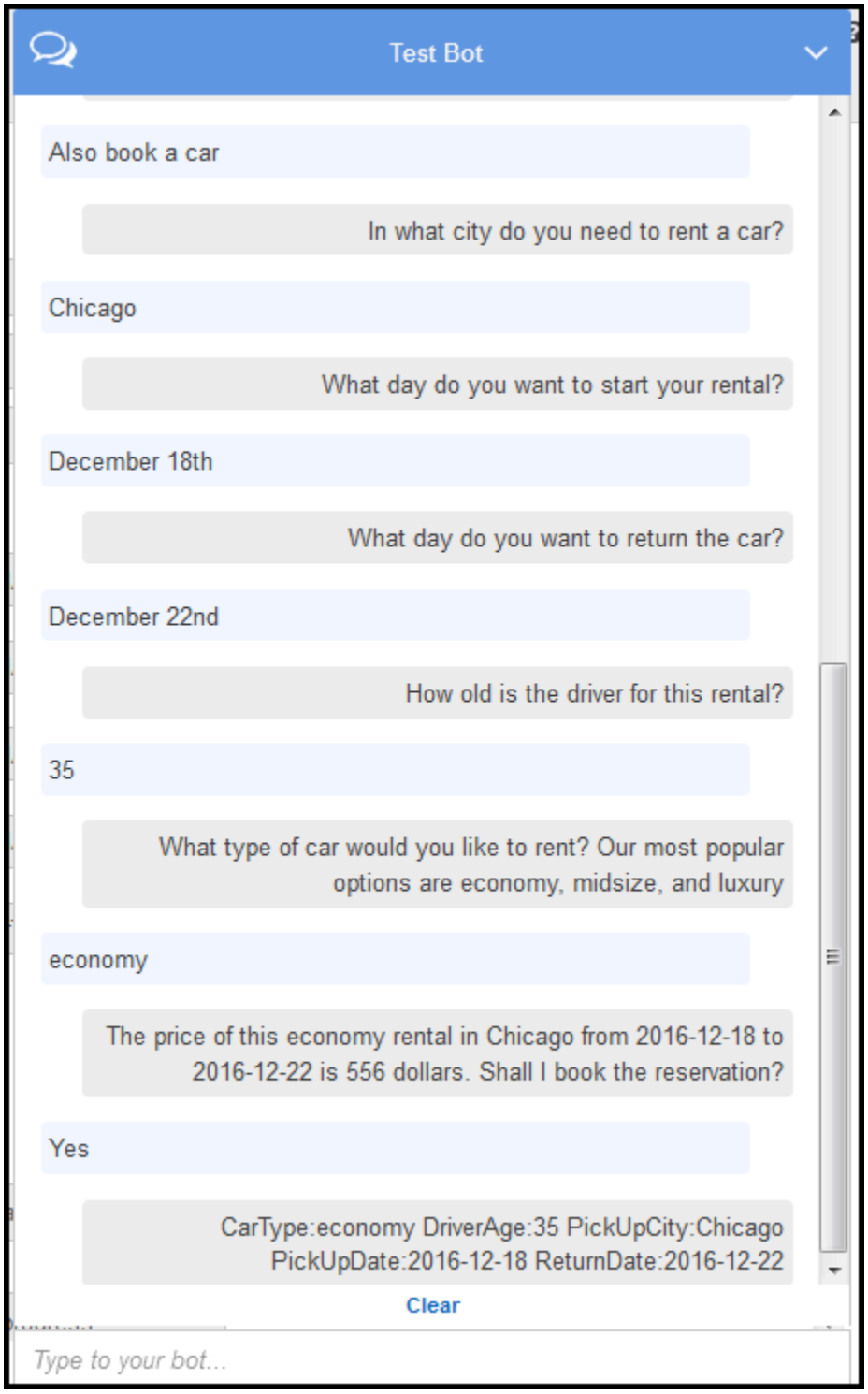
1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon Lex Konsole unter https://console.aws.amazon.com/lex/
2. Klicken Sie auf der Seite Bots auf die Option Create.
3. Wählen Sie auf der Seite Create your Lex bot
   - Wählen Sie einen BookTrip Plan.
   - Belassen Sie den Standard-Bot-Namen (BookTrip).
4. Wählen Sie Create (Erstellen) aus. Die Konsole sendet eine Reihe von Anfragen an Amazon Lex, um den Bot zu erstellen. Beachten Sie Folgendes:
5. Die Konsole zeigt den BookTrip Bot. Überprüfen Sie auf der Registerkarte Editor die Details der vorkonfigurierten Absichten (BookCar und BookHotel).
6. Testen Sie den Bot im Testfenster. Verwenden Sie das Folgende, um eine Konversation mit Ihrem Bot zu testen:



Aus der ersten Benutzereingabe („Hotel buchen“) leitet Amazon Lex die Absicht ab (BookHotel). Der Bot verwendet dann die in dieser Absicht vorkonfigurierten Aufforderungen, um vom Benutzer Slot-Daten zu erfragen. Nachdem der Benutzer alle Slot-Daten eingegeben hat, sendet Amazon Lex eine Antwort an den Client mit einer Nachricht zurück, die alle Benutzereingaben als Nachricht enthält. Der Client zeigt die Mitteilung in der Antwort wie angezeigt an.

```
CheckInDate:2016-12-18 Location:Chicago Nights:5 RoomType:queen
```

Nun setzt ihr das Gespräch fort und versucht im folgenden Gespräch ein Auto zu buchen.



Beachten Sie,

- Dieses Mal erfolgt keine Validierung der Benutzerdaten. Zum Beispiel können Sie jede beliebige Stadt angeben, in der Sie ein Hotel buchen möchten.

- Sie geben einige der gleichen Informationen erneut an (Ziel, Stadt der Abholung, Abholungsdatum, Rückgabedatum), um ein Auto zu buchen. In einer dynamischen Unterhaltung sollte Ihr Bot einige dieser Informationen aufgrund vorheriger Angaben des Benutzers für die Hotelbuchung initialisieren.

Im nächsten Abschnitt erstellen Sie eine Lambda-Funktion, die einen Teil der Validierung und Initialisierung der Benutzerdaten mithilfe der gemeinsamen Nutzung von Informationen für mehrere Absichten über Sitzungsattribute übernimmt. Dann aktualisieren Sie die Absichtskonfiguration, indem Sie die Lambda-Funktion als Code-Haken hinzufügen, um Initialisierung/Validierung der Benutzereingaben durchzuführen und die Absicht zu erfüllen.

Nächster Schritt

Schritt 3: Erstellen einer Lambda-Funktion

Schritt 3: Erstellen einer Lambda-Funktion

In diesem Abschnitt erstellen Sie eine Lambda-Funktion mithilfe eines in derAWS Lambda Konsole bereitgestellten Blueprint (lex-book-trip-python). Sie testen die Lambda-Funktion auch, indem Sie sie mithilfe von Beispielereignisdaten aufrufen, die von der Konsole bereitgestellt werden.

Diese Lambda-Funktion ist in Python geschrieben.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie Create function (Funktion erstellen).
3. Wählen Sie Use a blueprint (Vorlage verwenden). Geben Sie **`lex`** ein, um die Vorlage zu suchen, und wählen Sie die `lex-book-trip-python`-Vorlage aus.
4. Wählen Sie Configure the Lambda-Funktion wie folgt aus.
   - Geben Sie einen Lambda-Funktionsnamen (`BookTripCodeHook`) ein.
   - Wählen Sie für die Rolle Create a new role from template(s) aus und geben Sie einen Rollennamen ein.
   - Übernehmen Sie im Übrigen die Standardwerte.
5. Wählen Sie Create function (Funktion erstellen).

6. Wenn Sie ein anderes Gebietsschema als Englisch (US) (en-US) verwenden, aktualisieren Sie die Absichtsnamen wie unter beschriebenAktualisieren eines Blueprints für ein bestimmtes Gebietsschema.
7. Lambda-Funktion. Sie rufen die Lambda-Funktion zweimal auf und verwenden dabei Beispieldaten sowohl für die Buchung eines Autos als auch für die Buchung eines Hotels.

   a. Wählen Sie Configure test event (Testereignis konfigurieren) aus der Dropdownliste Select a test event (Ein Testereignis auswählen) aus.
   b. Wählen Sie Amazon Lex Book Hotel aus der Liste Sample event template (Beispielereignisvorlage) aus.

      Dieses Beispielereignis entspricht dem Anforderungs-/Antwortmodell von Amazon Lex. Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen.
   c. Klicken Sie auf Save and Test (Speichern und Testen).
   d. Stellen Sie sicher, dass die Lambda-Funktion Die Antwort entspricht in diesem Fall dem Amazon Lex Lex-Antwortmodell.
   e. Wiederholen Sie den Schritt. Wählen Sie diesmal Amazon Lex Book Car aus der Liste Sample event template (Beispielereignisvorlage) aus. Die Lambda-Funktion verarbeitet die Fahrzeugreservierung.

Nächster Schritt

Schritt 4: Fügen Sie die Lambda-Funktion als Code-Hook hinzu

Schritt 4: Fügen Sie die Lambda-Funktion als Code-Hook hinzu

In diesem Abschnitt aktualisieren Sie die Konfigurationen von BookCar und BookHotel intents, indem Sie die Lambda-Funktion als Code-Hook für Initialisierung/Validierung und Fulfillment-Aktivitäten hinzufügen. Stellen Sie sicher, dass Sie die $LATEST-Version der Intents wählen, da Sie nur die $LATEST-Version Ihrer Amazon Lex-Ressourcen aktualisieren können.

1. Wählen Sie in der Amazon Lex-Konsole den BookTripBot aus.
2. Wählen Sie auf der Registerkarte Editor die BookHotelAbsicht aus. Aktualisieren Sie die Absichtskonfiguration wie folgt:

   a. Stellen Sie sicher, dass die Absichtsversion (neben dem Absichtsnamen) $LATEST ist.

b. Fügen Sie die Lambda-Funktion wie folgt als Initialisierungs- und Validierungscode-Hook hinzu:

   - Wählen Sie im Abschnitt Options Initialization and validation code hook aus.
   - Wählen Sie Ihre Lambda-Funktion aus.

c. Fügen Sie die Lambda-Funktion wie folgt hinzu:

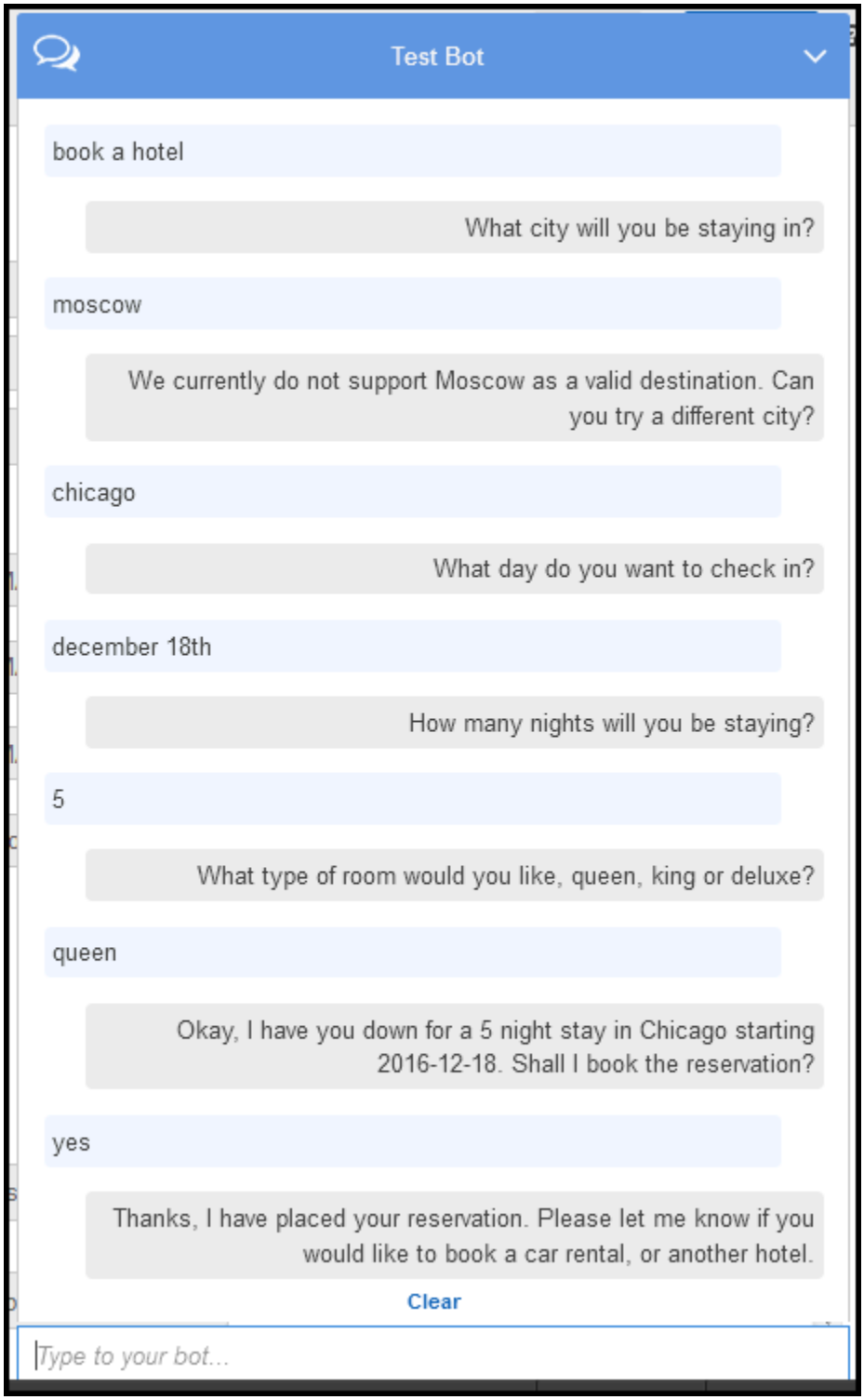
   - Wählen Sie im Abschnitt Fulfillment AWS Lambda function aus.
   - Wählen Sie Ihre Lambda-Funktion aus.
   - Wählen Sie Goodbye message aus und geben Sie eine Mitteilung ein.

d. Wählen Sie Save (Speichern) aus.

3. Wählen Sie auf der Registerkarte Editor die BookCar Absicht aus. Befolgen Sie den vorhergehenden Schritt, um Ihre Lambda-Funktion als Code-Haken für Validierung und Erfüllung hinzuzufügen.

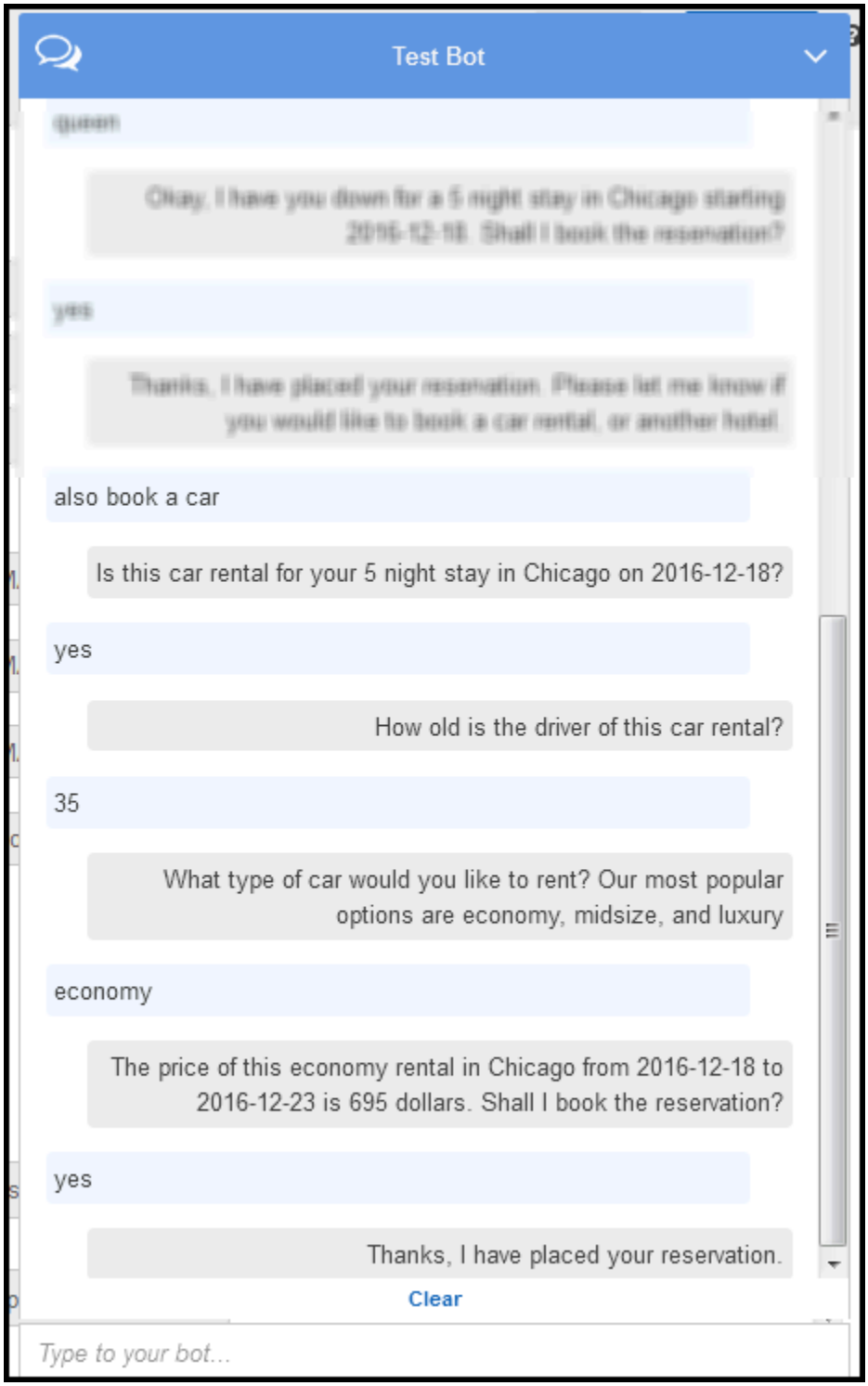
4. Wählen Sie Build aus. Die Konsole sendet eine Reihe von Anfragen an Amazon Lex, um die Konfigurationen zu speichern.

5. Testen Sie den Bot. Da Sie nun über eine Lambda-Funktion verfügen, die die Initialisierung, Validierung und Erfüllung der Benutzerdaten durchführt, können Sie den Unterschied in der Benutzerinteraktion in der folgenden Konversation erkennen:



Weitere Hinweise zum Datenfluss vom Client (Konsole) zu Amazon Lex und von Amazon Lex zur Lambda-Funktion finden Sie unterDatenfluss: Absicht zur Buchung eines Hotels.

6. Konversation und Buchung eines Autos, wie im folgenden Image:



Wenn Sie ein Auto buchen möchten, sendet der Kunde (Konsole) eine Anfrage an Amazon Lex, die die Sitzungsattribute (aus der vorherigen Konversation BookHotel) enthält. Amazon Lex übergibt diese Informationen an die Lambda-Funktion, die dann einige der BookCar Slot-Daten (d. h.,, und) initialisiert (das heißt PickUpDate ReturnDate, sie füllt sie vorab aus PickUpCity).

> **Note**
>
> Dies veranschaulicht, wie Sitzungsattribute dazu dienen können, Kontext über Absichten hinweg aufrechtzuerhalten. Der Konsolenclient zeigt den Link Clear im Testfenster an, über das ein Benutzer alle vorherigen Sitzungsattribute löschen kann.

Weitere Hinweise zum Datenfluss vom Client (Konsole) zu Amazon Lex und von Amazon Lex zur Lambda-Funktion finden Sie unterDatenfluss: Absicht Auto buchen.

Informationsfluss im Detail

In dieser Übung haben Sie mithilfe des in der Amazon Lex Lex-Konsole bereitgestellten Testfensterclients eine Konversation mit dem Amazon Lex BookTrip Lex-Bot geführt. In diesem Abschnitt wird Folgendes erklärt:

- Der Datenfluss zwischen dem Kunden und Amazon Lex.

  In diesem Abschnitt wird davon ausgegangen, dass der Client mithilfe der`PostText` Runtime-API Anfragen an Amazon Lex sendet, und die Anfrage- und Antwortdetails werden entsprechend angezeigt. Weitere Informationen zur `PostText` finden Sie unter PostText.

  > **Note**
  >
  > Ein Beispiel für den Informationsfluss zwischen dem Client und Amazon Lex, bei dem der Client die`PostContent` API verwendet, finden Sie unterSchritt 2a (optional): Prüfen der Details des Informationsflusses gesprochener Inhalte (Konsole) .

- Der Datenfluss zwischen Amazon Lex und der Lambda-Funktion. Weitere Informationen finden Sie unter Lambda-Funktions-Eingabe-Ereignis und Antwort-Format.

Themen

- Datenfluss: Absicht zur Buchung eines Hotels
- Datenfluss: Absicht Auto buchen

Datenfluss: Absicht zur Buchung eines Hotels

In diesem Abschnitt wird erklärt, was nach jeder Benutzereingabe geschieht.

1. Benutzer: „Buche ein Hotel“

   a. Der Client (Konsole) sendet die folgende PostText-Anforderung an Amazon Lex:

   ```
   POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book a hotel",
      "sessionAttributes":{}
   }
   ```

   Sowohl die Anfrage-URI als auch der Text stellen Informationen für Amazon Lex bereit:

   - Anforderungs-URI — Stellt den Botnamen (BookTrip), den Bot-Alias ($LATEST) und den Benutzernamen bereit. Der abschließende `text` zeigt an, dass es sich um eine `PostText`-API-Anforderung handelt (und nicht um `PostContent`).
   - Anforderungsinhalt: Enthält die Benutzereingabe (`inputText`) und leere `sessionAttributes`. Anfänglich ist dies ein leeres Objekt und die Lambda-Funktion legt zuerst die Sitzungsattribute fest.

   b. Anhand des`inputText` erkennt Amazon Lex die Absicht (BookHotel). Diese Absicht wird mit einer Lambda-Funktion als Code-Hook für die Initialisierung/Validierung von Benutzerdaten konfiguriert. Daher ruft Amazon Lex diese Lambda-Funktion auf, indem es die folgenden Informationen als Ereignisparameter übergibt (siehe Eingabe-Ereignis-Format):

   ```
   {
      "messageVersion":"1.0",
      "invocationSource":"DialogCodeHook",
      "userId":"wch89kjqcpkds8seny7dly5x3otq68j3",
   ```

```
   "sessionAttributes":{
   },
   "bot":{
      "name":"BookTrip",
      "alias":null,
      "version":"$LATEST"
   },
   "outputDialogMode":"Text",
   "currentIntent":{
      "name":"BookHotel",
      "slots":{
         "RoomType":null,
         "CheckInDate":null,
         "Nights":null,
         "Location":null
      },
      "confirmationStatus":"None"
   }
}
```

Zusätzlich zu den vom Kunden gesendeten Informationen enthält Amazon Lex auch die folgenden zusätzlichen Daten:

- `messageVersion`— Derzeit unterstützt Amazon Lex nur die Version 1.0.
- `invocationSource`— Gibt den Zweck des Aufrufs einer Lambda-Funktion an. In diesem Fall erfolgt die Initialisierung und Validierung der Benutzerdaten (Amazon Lex weiß derzeit, dass der Benutzer nicht alle Slot-Daten zur Erfüllung der Absicht angegeben hat).
- `currentIntent`: Alle Slot-Werte werden auf Null gesetzt.

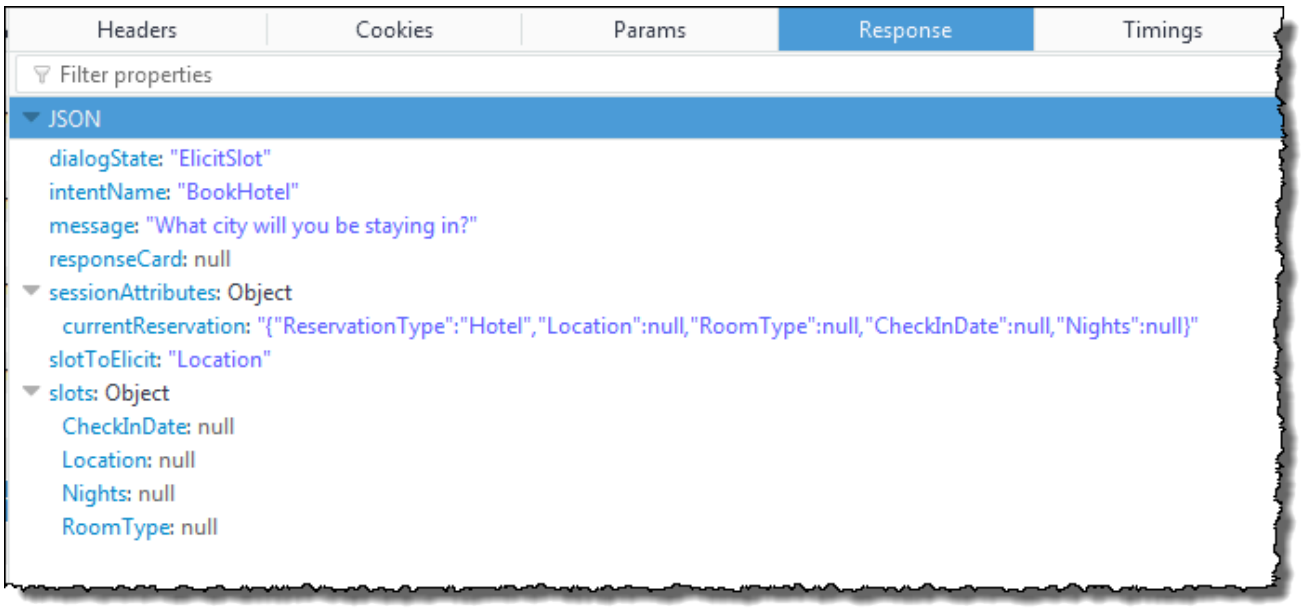
c. Zu diesem Zeitpunkt sind alle Slot-Werte auf Null gesetzt. Die Lambda-Funktion muss nichts validieren. Die Lambda-Funktion gibt die folgende Antwort an Amazon Lex zurück. Weitere Informationen über das Antwortformat finden Sie unter Reaktion-Format.

```
{
   "sessionAttributes":{
      "currentReservation":"{\"ReservationType\":\"Hotel\",\"Location\":null,
\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
   },
   "dialogAction":{
      "type":"Delegate",
      "slots":{
         "RoomType":null,
```

```
            "CheckInDate":null,
            "Nights":null,
            "Location":null
        }
    }
}
```

> **Note**
>
> - `currentReservation`— Die Lambda-Funktion beinhaltet dieses Sitzungsattribut. Sein Wert ist eine Kopie der aktuellen Slot-Informationen und des Reservierungstyps.
>
> Nur die Lambda-Funktion und der Client können diese Sitzungsattribute aktualisieren. Amazon Lex übergibt diese Werte einfach.
> - `dialogAction.type`— Wenn Sie diesen Wert auf festlegen`Delegate`, delegiert die Lambda-Funktion die Verantwortung für die nächste Vorgehensweise an Amazon Lex.
>
> Wenn die Lambda-Funktion bei der Benutzerdatenüberprüfung etwas entdeckt, weist sie Amazon Lex an, was als Nächstes zu tun ist.

d. Gemäß dem`dialogAction.type` entscheidet Amazon Lex über die nächste Vorgehensweise — das Abrufen von Daten vom Benutzer für den`Location` Slot. Es wählt eine der Aufforderungsmitteilungen ("In welcher Stadt werden Sie bleiben?") gemäß der Konfiguration der Absicht für diesen Slot aus und sendet dann die folgende Antwort an den Benutzer:



Die Sitzungsattribute werden an den Client übergeben.

Der Client liest die Antwort und zeigt dann die Mitteilung an: „In welcher Stadt werden Sie bleiben?“

2. Benutzer: „Moskau“

   a. Der Client sendet die folgende`PostText` Anfrage an Amazon Lex (aus Gründen der Lesbarkeit wurden Zeilenumbrüche hinzugefügt):

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Moscow",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":null,
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

Zusätzlich zu dem `inputText` fügt der Client die gleichen `currentReservation`-Sitzungsattribute hinzu, die er erhalten hat.

b. Amazon Lex interpretiert das zuerst`inputText` im Kontext der aktuellen Absicht (der Service erinnert sich, dass er den bestimmten Benutzer um Informationen zum`Location` Slot gebeten hat). Es aktualisiert den Slot-Wert für die aktuelle Absicht und ruft die Lambda-Funktion mithilfe des folgenden Ereignisses auf:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":null,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Moscow"
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `invocationSource` ist weiterhin `DialogCodeHook`. In diesem Schritt validieren wir nur Benutzerdaten.
> - Amazon Lex übergibt lediglich das Sitzungs-Atx an die Lambda-Funktion an die Lambda-Funktion.

- Denn`currentIntent.slots` Amazon Lex hat den`Location` Slot aktualisiert auf`Moscow`.

c. Die Lambda-Funktion führt die Benutzerdatenvalidierung durch und stellt fest, dass`Moscow` es sich um einen ungültigen Standort handelt.

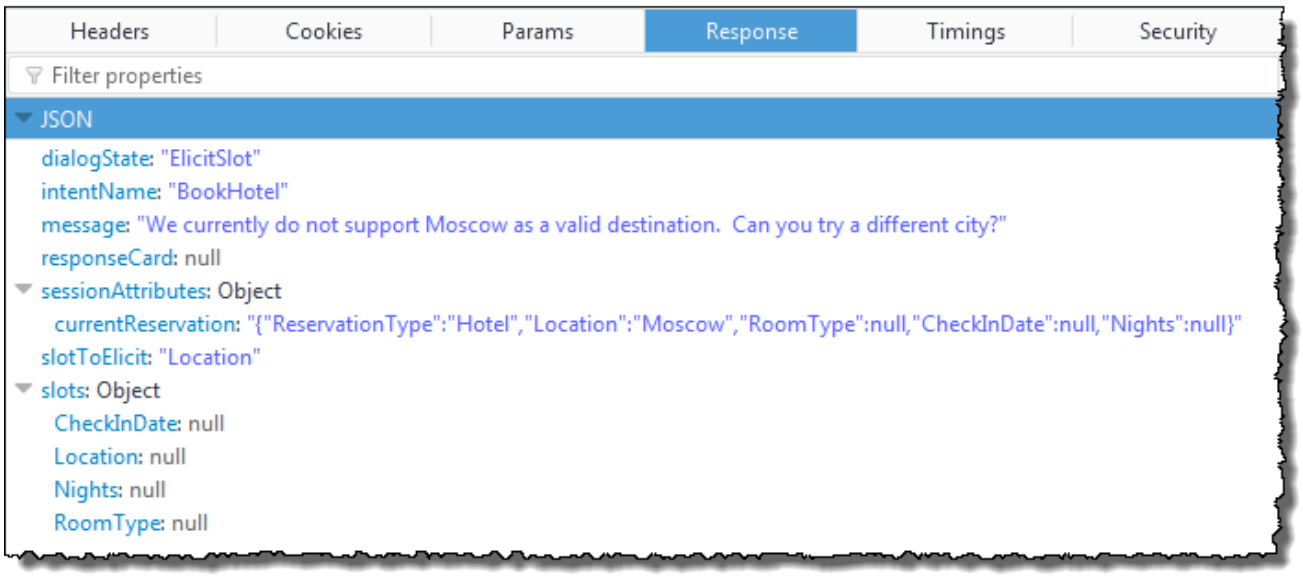
> **Note**
>
> Die Lambda-Funktion in dieser Übung hat eine einfache Liste gültiger Städte und`Moscow` ist nicht in der Liste enthalten. In einer Produktionsanwendung würden Sie eventuell eine Backend-Datenbank verwenden, um diese Information zu erhalten.

Es setzt den Slot-Wert wieder auf Null zurück und weist Amazon Lex an, den Benutzer erneut zur Eingabe eines anderen Werts aufzufordern, indem die folgende Antwort gesendet wird:

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Moscow\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "ElicitSlot",
        "intentName": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": null
        },
        "slotToElicit": "Location",
        "message": {
            "contentType": "PlainText",
            "content": "We currently do not support Moscow as a valid
 destination.  Can you try a different city?"
        }
    }
}
```

> **Note**
>
> - `currentIntent.slots.Location` wird auf Null zurückgesetzt.
> - `dialogAction.type`ist auf gesetzt`ElicitSlot`, was Amazon Lex anweist, den Benutzer erneut aufzufordern, indem er Folgendes angibt:
>   - `dialogAction.slotToElicit`: Slot, für den Daten vom Benutzer erfragt werden
>   - `dialogAction.message`: Eine an den Benutzer zu übermittelnde `message`

d. Amazon Lex bemerkt das`dialogAction.type` und leitet die Informationen in der folgenden Antwort an den Kunden weiter:



Der Client zeigt die Mitteilung einfach an: „Wir unterstützen derzeit Moskau als gültiges Ziel nicht. Können Sie es mit einer anderen Stadt versuchen?“

3. Benutzer: „Chicago“

a. Der Kunde sendet die folgende`PostText` Anfrage an Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Chicago",
```

```
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Moscow\",
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

b. Amazon Lex kennt den Kontext, in dem es Daten für den`Location` Slot abgerufen hat. In diesem Kontext weiß es, dass der Wert `inputText` für den Slot `Location` ist. Anschließend wird die Lambda-Funktion aufgerufen, indem das folgende Ereignis gesendet wird:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":Moscow,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        },
        "confirmationStatus": "None"
    }
}
```

Amazon Lex hat das aktualisiert,`currentIntent.slots` indem der`Location` Slot auf gesetzt wurde`Chicago`.

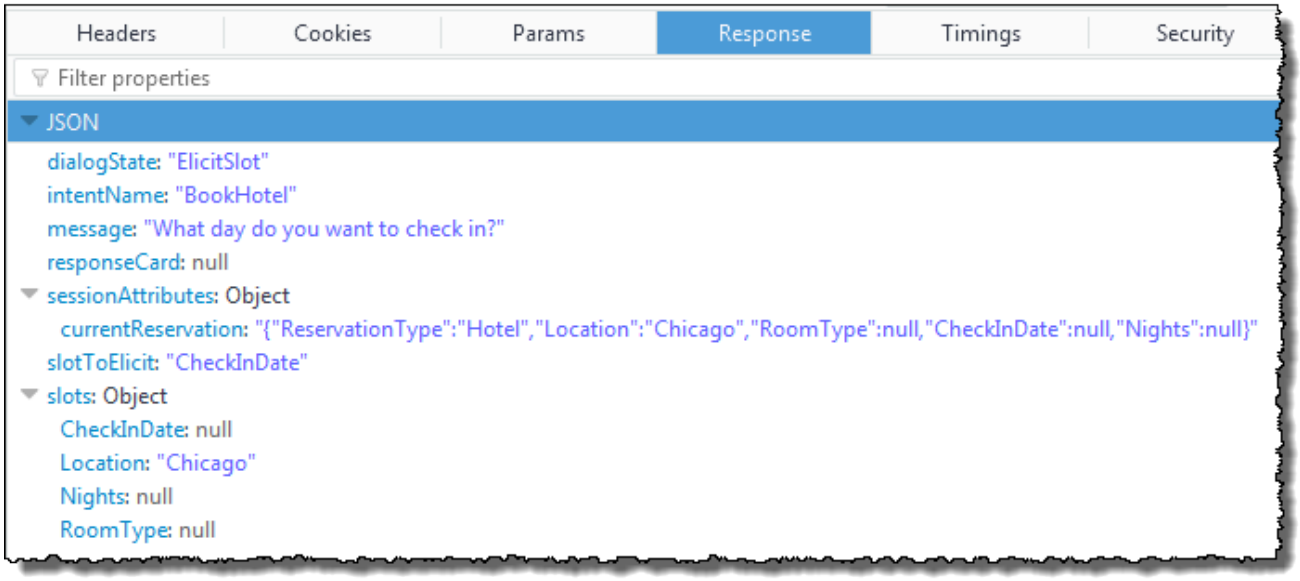
c. Entsprechend dem`invocationSource` Wert von`DialogCodeHook` führt die Lambda-Funktion eine Benutzerdatenvalidierung durch. Es wird`Chicago` als gültiger Slot-Wert erkannt, aktualisiert das Sitzungsattribut entsprechend und gibt dann die folgende Antwort an Amazon Lex zurück.

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        }
    }
}
```

> **Note**
>
> - `currentReservation`— Die Lambda-Funktion aktualisiert dieses Sitzungsattribut, indem sie das`Location` auf setzt`Chicago`.
> - `dialogAction.type`: Wird auf `Delegate` gesetzt. Die Benutzerdaten waren gültig, und die Lambda-Funktion weist Amazon Lex an, die nächste Vorgehensweise zu wählen.

d. Laut`dialogAction.type` Amazon Lex wählt Amazon Lex die nächste Vorgehensweise. Amazon Lex weiß, dass mehr Steckplatzdaten benötigt werden, und wählt entsprechend der gewünschten Konfiguration den nächsten unbesetzten Steckplatz (`CheckInDate`) mit der höchsten Priorität aus. Es wählt eine der Aufforderungsmitteilungen ("An welchem Tag

möchten Sie einchecken?") gemäß der Konfiguration der Absicht für diesen Slot aus und sendet dann die folgende Antwort an den Kunden:

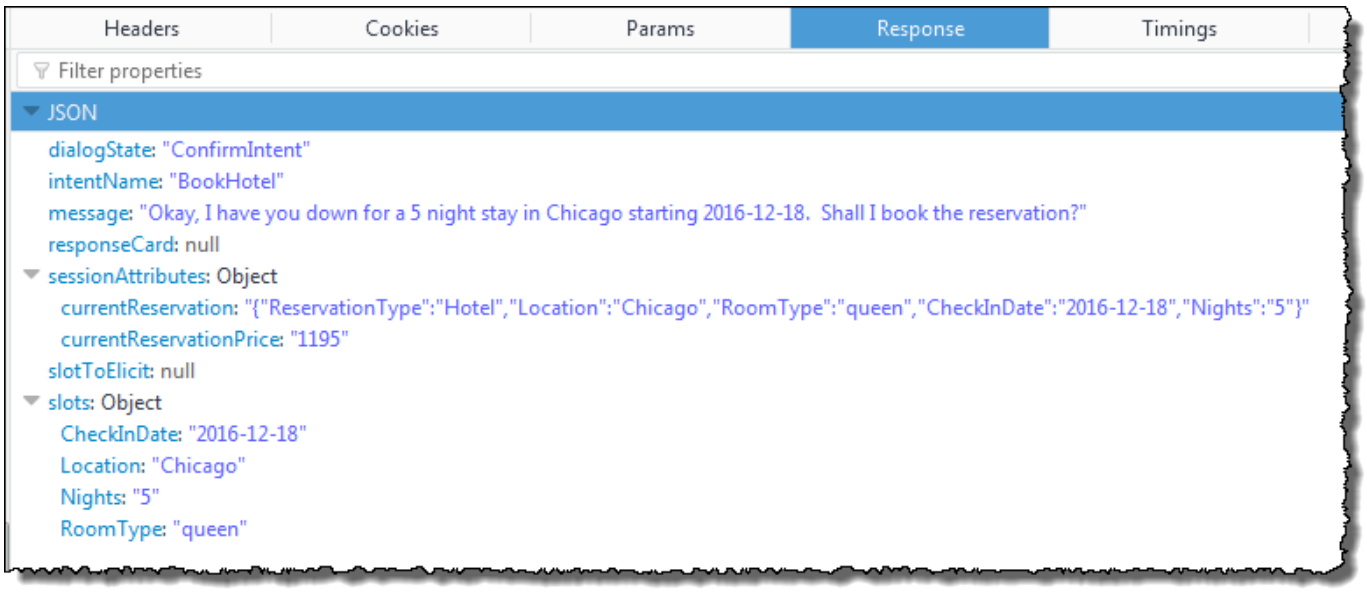


Der Client zeigt die Nachricht an: „An welchem Tag möchten Sie anreisen?“

4. Die Benutzerinteraktion wird fortgesetzt — der Benutzer stellt Daten bereit, die Lambda-Funktion validiert Daten und delegiert dann die nächste Vorgehensweise an Amazon Lex. Schließlich stellt der Benutzer alle Slot-Daten zur Verfügung, die Lambda-Funktion validiert alle Benutzereingaben, und dann erkennt Amazon Lex, dass alle Slot-Daten vorliegen.

   > **Note**
   >
   > In dieser Übung berechnet die Lambda-Funktion den Preis der Hotelreservierung, nachdem der Benutzer alle Slot-Daten angegeben hat, und gibt ihn als weiteres Sitzungsattribut (`currentReservationPrice`) zurück.

   Zu diesem Zeitpunkt ist die Absicht bereit, erfüllt zu werden, aber die BookHotel Absicht ist mit einer Bestätigungsaufforderung konfiguriert, die eine Bestätigung durch den Benutzer erfordert, bevor Amazon Lex die Absicht erfüllen kann. Daher sendet Amazon Lex die folgende Nachricht an den Kunden und bittet ihn um Bestätigung, bevor das Hotel gebucht wird:



Der Client zeigt die Nachricht an: „Okay, ich habe einen Aufenthalt für 5 Nächte in Chicago ab 18.12.2016 notiert. Soll ich die Reservierung vornehmen?“

5. Benutzer: „Ja“

   a. Der Kunde sendet die folgende`PostText` Anfrage an Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Yes",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Chicago\",
                              \"RoomType\":\"queen\",
                              \"CheckInDate\":\"2016-12-18\",
                              \"Nights\":\"5\"}",
      "currentReservationPrice":"1195"
   }
}
```

   b. Amazon Lex interpretiert das`inputText` im Kontext der Bestätigung der aktuellen Absicht. Amazon Lex geht davon aus, dass der Benutzer mit der Reservierung fortfahren möchte. Dieses Mal ruft Amazon Lex die Lambda-Funktion auf, um die Absicht zu erfüllen, indem das folgende Ereignis gesendet wird. Wenn das

Ereignis`invocationSource` `FulfillmentCodeHook` auf gesetzt wird, wird es an die Lambda-Funktion gesendet. Amazon Lex gibt auch die`confirmationStatus` Messlatte vor`Confirmed`.

```
{
    "messageVersion": "1.0",
    "invocationSource": "FulfillmentCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights\":
\"5\"}",
        "currentReservationPrice": "956"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": "queen",
            "CheckInDate": "2016-12-18",
            "Nights": "5",
            "Location": "Chicago"
        },
        "confirmationStatus": "Confirmed"
    }
}
```

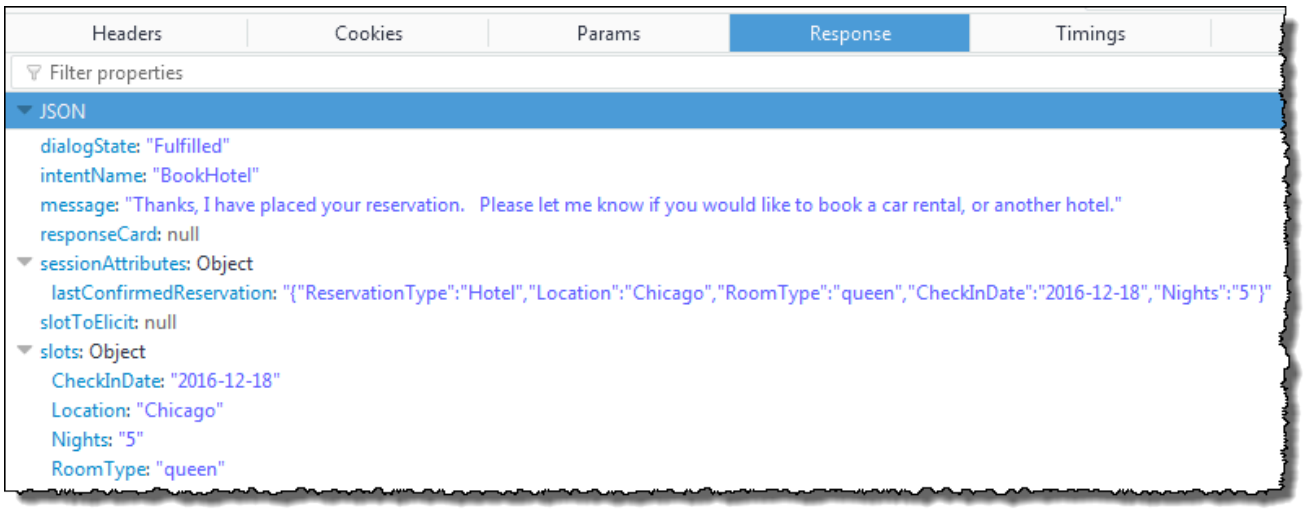
> **Note**
>
> - `invocationSource`— Dieses Mal hat Amazon Lex diesen Wert auf festgelegt`FulfillmentCodeHook` und die Lambda-Funktion angewiesen, die Absicht zu erfüllen.
> - `confirmationStatus`: Wird auf `Confirmed` gesetzt.

c. Dieses Mal erfüllt die Lambda-Funktion die BookHotel Absicht, Amazon Lex schließt die Reservierung ab und gibt dann die folgende Antwort zurück:

```
{
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, I have placed your reservation.   Please let me
 know if you would like to book a car rental, or another hotel."
        }
    }
}
```

> **Note**
>
> - `lastConfirmedReservation`— Ist ein neues Sitzungsattribut, das die Lambda-Funktion hinzugefügt hat (anstelle von`currentReservation`,`currentReservationPrice`).
> - `dialogAction.type`— Die Lambda-Funktion legt diesen Wert auf fest`Close`, was darauf hinweist, dass Amazon Lex keine Benutzerantwort erwartet.
> - `dialogAction.fulfillmentState`: Ist auf `Fulfilled` eingestellt und enthält eine geeignete `message` zur Übermittlung an den Benutzer.

d. Amazon Lex überprüft das`fulfillmentState` und sendet die folgende Antwort an den Kunden:



> **Note**
>
> - `dialogState`— Amazon Lex legt diesen Wert auf fest`Fulfilled`.
> - `message`— Ist dieselbe Meldung, die die Lambda-Funktion bereitgestellt hat.

Der Client zeigt die Mitteilung an.

Datenfluss: Absicht Auto buchen

Der BookTrip Bot in dieser Übung unterstützt zwei Absichten (BookHotel und BookCar). Nach der Buchung eines Hotels kann der Benutzer die Unterhaltung fortsetzen, um ein Auto zu buchen. Solange die Zeit der Sitzung noch nicht abgelaufen ist, sendet der Client bei jeder folgenden Anforderung weiter Sitzungsattribute (in diesem Beispiel die `lastConfirmedReservation`). Die Lambda-Funktion kann diese Informationen verwenden, um Slot-Daten für die BookCar Absicht zu initialisieren. Dies zeigt, wie Sie Sitzungsattribute für die gemeinsame Nutzung von Informationen für mehrere Absichten verwenden können.

Insbesondere wenn der Benutzer die BookCar Absicht auswählt, verwendet die Lambda-Funktion relevante Informationen im Sitzungsattribut, um die Steckplätze (PickUpDate ReturnDate, und PickUpCity) für die BookCar Absicht vorab auszufüllen.

> **Note**
>
> Die Amazon Lex Lex-Konsole bietet den Link Löschen, mit dem Sie alle vorherigen Sitzungsattribute löschen können.

Führen Sie die folgenden Schritte aus, um die Unterhaltung fortzusetzen.

1. Benutzer: „Buche ebenfalls ein Auto“

   a. Der Kunde sendet die folgende`PostText` Anfrage an Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"also book a car",
   "sessionAttributes":{
       "lastConfirmedReservation":""{\"ReservationType\":\"Hotel\",
                                    \"Location\":\"Chicago\",
                                    \"RoomType\":\"queen\",
                                    \"CheckInDate\":\"2016-12-18\",
                                    \"Nights\":\"5\"}"
   }
}
```

   Der Client schließt das Sitzungsattribut `lastConfirmedReservation` ein.

   b. Amazon Lex erkennt die Absicht (BookCar) aus dem`inputText`. Diese Absicht ist auch so konfiguriert, dass die Lambda-Funktion aufgerufen wird, um die Initialisierung und Validierung der Benutzerdaten durchzuführen. Amazon Lex die Lambda-Funktion mit dem folgenden Ereignis:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
```

```
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookCar",
        "slots": {
            "PickUpDate": null,
            "ReturnDate": null,
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": null
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `messageVersion`— Derzeit unterstützt Amazon Lex nur die Version 1.0.
> - `invocationSource`: Zeigt an, dass der Zweck des Aufrufs ist, Initialisierung und Validierung von Benutzerdaten durchzuführen.
> - `currentIntent`— Es enthält den Namen der Absicht und die Slots. Zu diesem Zeitpunkt sind alle Slot-Werte auf Null gesetzt.

c. Die Lambda-Funktion bemerkt alle Null-Slot-Werte, ohne dass etwas zu validieren ist. Sie verwendet jedoch Sitzungsattribute, um einige der Slot-Werte zu initialisieren (`PickUpDate`, `ReturnDate` und `PickUpCity`), und gibt dann die folgende Antwort zurück:

```
{
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
```

```
        "confirmationContext": "AutoPopulate"
    },
    "dialogAction": {
        "type": "ConfirmIntent",
        "intentName": "BookCar",
        "slots": {
            "PickUpCity": "Chicago",
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "CarType": null,
            "DriverAge": null
        },
        "message": {
            "contentType": "PlainText",
            "content": "Is this car rental for your 5 night stay in Chicago on
 2016-12-18?"
        }
    }
}
```

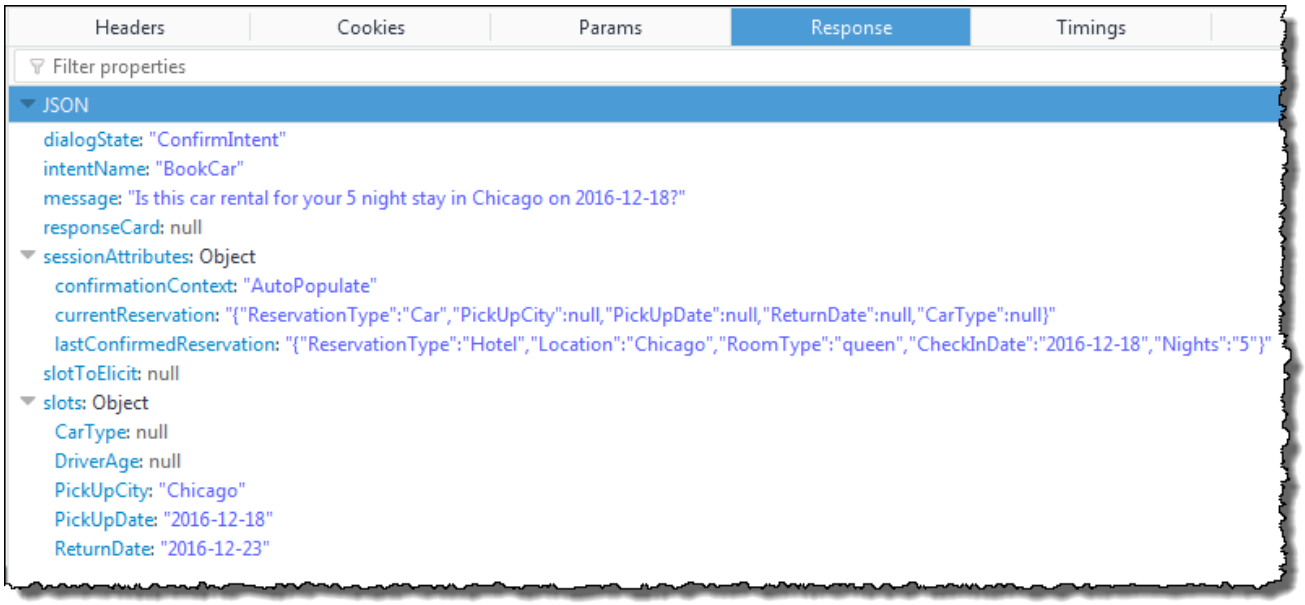
Note

- Zusätzlich zu dem`lastConfirmedReservation` enthält die Lambda-Funktion weitere Sitzungsattribute (`currentReservation`und`confirmationContext`).
- `dialogAction.type`ist auf gesetzt`ConfirmIntent`, wodurch Amazon Lex darüber informiert wird, dass vom Benutzer eine Antwort mit Ja, Nein erwartet wird (der confirmationContext ist auf gesetzt AutoPopulate, die Lambda-Funktion weiß, dass die Ja/Nein-Benutzerantwort dazu dient, eine Bestätigung des Benutzers über die von der Lambda-Funktion ausgeführte Initialisierung zu erhalten (auto aufgefüllte Slot-Daten).

  Die Lambda-Funktion enthält in der Antwort auch eine informative Meldung in der`dialogAction.message`, damit Amazon Lex an den Client zurückgesendet werden kann.

> **Note**
>
> Der Ausdruck `ConfirmIntent` (Wert des `dialogAction.type`) hat keinen Bezug zu einer Bot-Absicht. Im Beispiel verwendet die Lambda-Funktion diesen Begriff, um Amazon Lex anzuweisen, eine Ja/Nein-Antwort vom Benutzer zu erhalten.

d. Laut dem`dialogAction.type` gibt Amazon Lex die folgende Antwort an den Kunden zurück:



Der Client zeigt die Mitteilung an: „Ist dieser Mietwagen für Ihren 5-Nächte-Aufenthalt in Chicago am 18.12.2016?“

2. Benutzer: „Ja“

a. Der Kunde sendet die folgende`PostText` Anfrage an Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"yes",
   "sessionAttributes":{
      "confirmationContext":"AutoPopulate",
```

```
      "currentReservation":"{\"ReservationType\":\"Car\",
                             \"PickUpCity\":null,
                             \"PickUpDate\":null,
                             \"ReturnDate\":null,
                             \"CarType\":null}",
      "lastConfirmedReservation":"{\"ReservationType\":\"Hotel\",
                                   \"Location\":\"Chicago\",
                                   \"RoomType\":\"queen\",
                                   \"CheckInDate\":\"2016-12-18\",
                                   \"Nights\":\"5\"}"
   }
}
```

b. Amazon Lex liest den`inputText` und kennt den Kontext (hat den Benutzer gebeten, die auto Auffüllung zu bestätigen). Amazon Lex die Lambda-Funktion, indem es das folgende Ereignis auf:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "confirmationContext": "AutoPopulate",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookCar",
        "slots": {
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": "Chicago"
```

```
        },
        "confirmationStatus": "Confirmed"
    }
}
```

Da der Benutzer mit Ja geantwortet hat, legt Amazon Lex`confirmationStatus` die Einstellung fest`Confirmed`.

c. Aus dem`confirmationStatus` weiß die Lambda-Funktion, dass die vorab ausgefüllten Werte korrekt sind. Die Lambda-Funktion:

- Sie aktualisiert das Sitzungsattribut `currentReservation` auf den vorab ausgefüllten Slot-Wert.
- Sie setzt den `dialogAction.type` auf `ElicitSlot`.
- Sie setzt den Wert `slotToElicit` auf `DriverAge`.

Die folgende Antwort wird gesendet:

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":\"Chicago\",\"PickUpDate\":\"2016-12-18\",\"ReturnDate\":\"2016-12-22\",
\"CarType\":null}",
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "dialogAction": {
        "type": "ElicitSlot",
        "intentName": "BookCar",
        "slots": {
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": "Chicago"
        },
        "slotToElicit": "DriverAge",
        "message": {
            "contentType": "PlainText",
            "content": "How old is the driver of this car rental?"
```

```
        }
    }
}
```

d. Amazon Lex gibt folgende Antwort zurück:

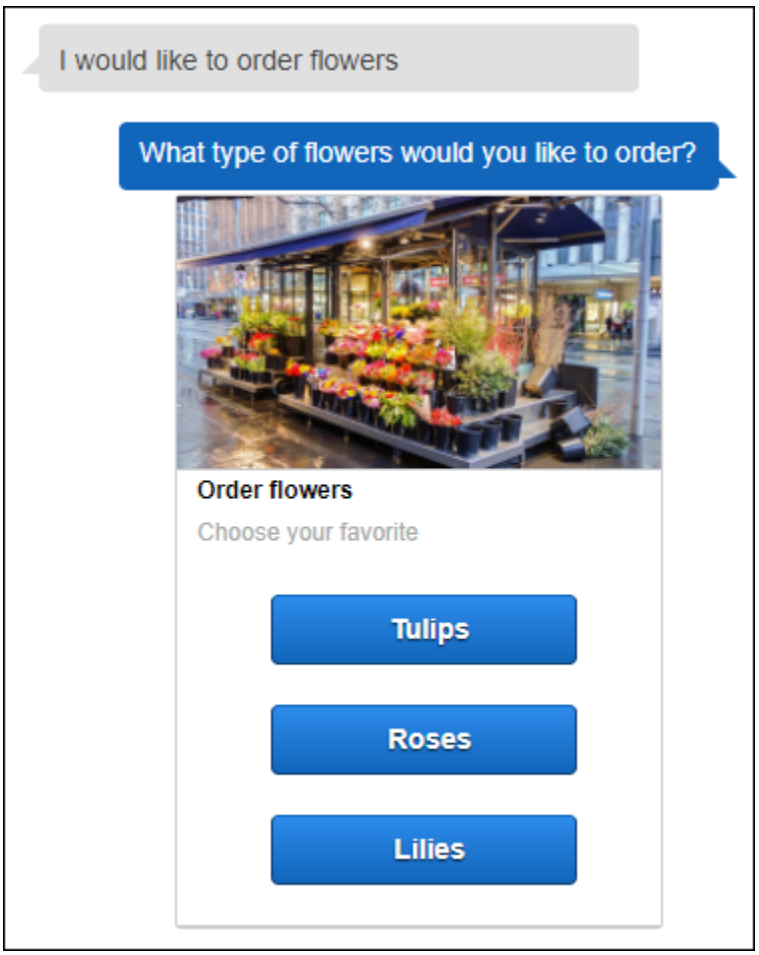
Der Kunde zeigt die Meldung „Wie alt ist der Fahrer dieser Autovermietung?“ und die Unterhaltung wird fortgesetzt.

Eine Antwortkarte verwenden

Diese Übung erweitert die Erste-Schritte-Übung 1 durch Hinzufügen einer Antwortkarte. Sie erstellen einen Bot, der die OrderFlowers Absicht unterstützt, und aktualisieren dann die Absicht, indem Sie eine Antwortkarte für den`FlowerType` Slot hinzufügen. Zusätzlich zu der folgenden Aufforderung für den Slot `FlowerType` kann der Benutzer die Art der Blumen auf der Antwortkarte auswählen:

```
What type of flowers would you like to order?
```

Folgendes ist die Antwortkarte:



Der Bot-Benutzer kann entweder den Text eingeben oder aus der Liste der Blumenarten auswählen. Diese Antwortkarte wird mit einem Bild konfiguriert, das im Client so angezeigt wird. Weitere Informationen über Antwortkarten finden Sie unter Antwort-Karten.

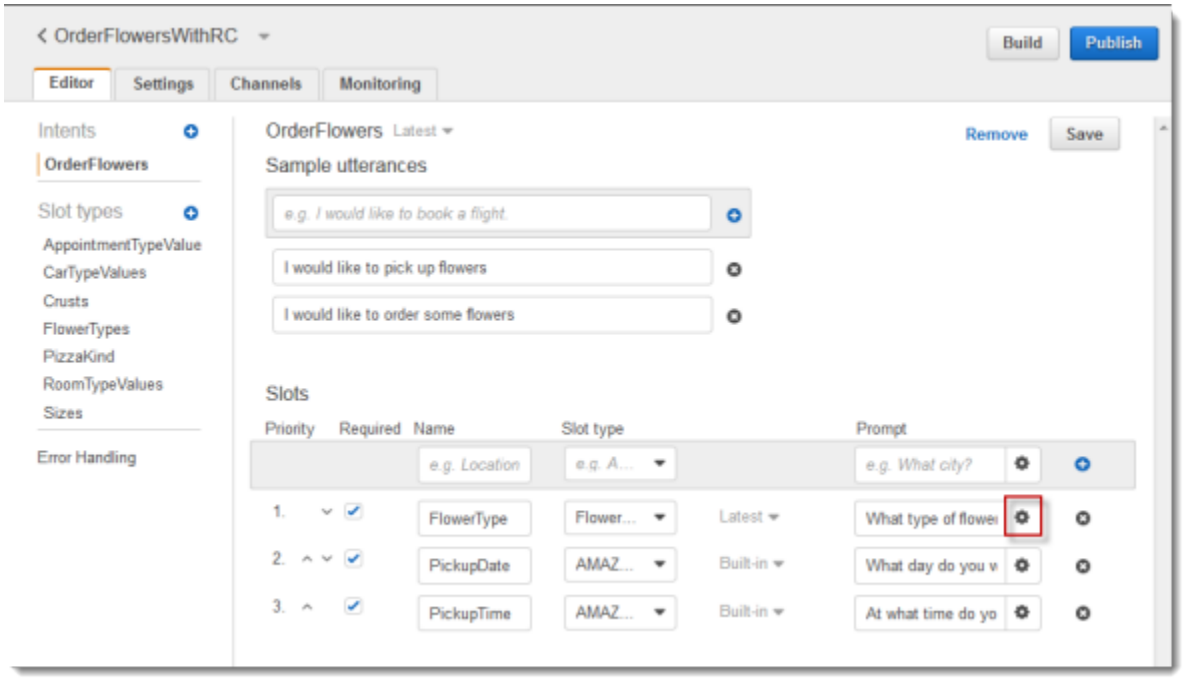
So erstellen und testen Sie einen Bot mit einer Antwortkarte:

1. Folgen Sie der ersten Übung „Erste Schritte“, um einen OrderFlowers Bot zu erstellen und zu testen. Sie müssen die Schritte 1, 2 und 3 abschließen. Sie müssen keine Lambda-Funktion hinzufügen, um die Antwortkarte zu testen. Detaillierte Anweisungen finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).
2. Aktualisieren Sie den Bot, indem Sie die Antwortkarte hinzufügen, und veröffentlichen Sie dann eine Version. Wenn Sie eine Version veröffentlichen, geben Sie einen Alias (BETA) an, der auf sie verweist.

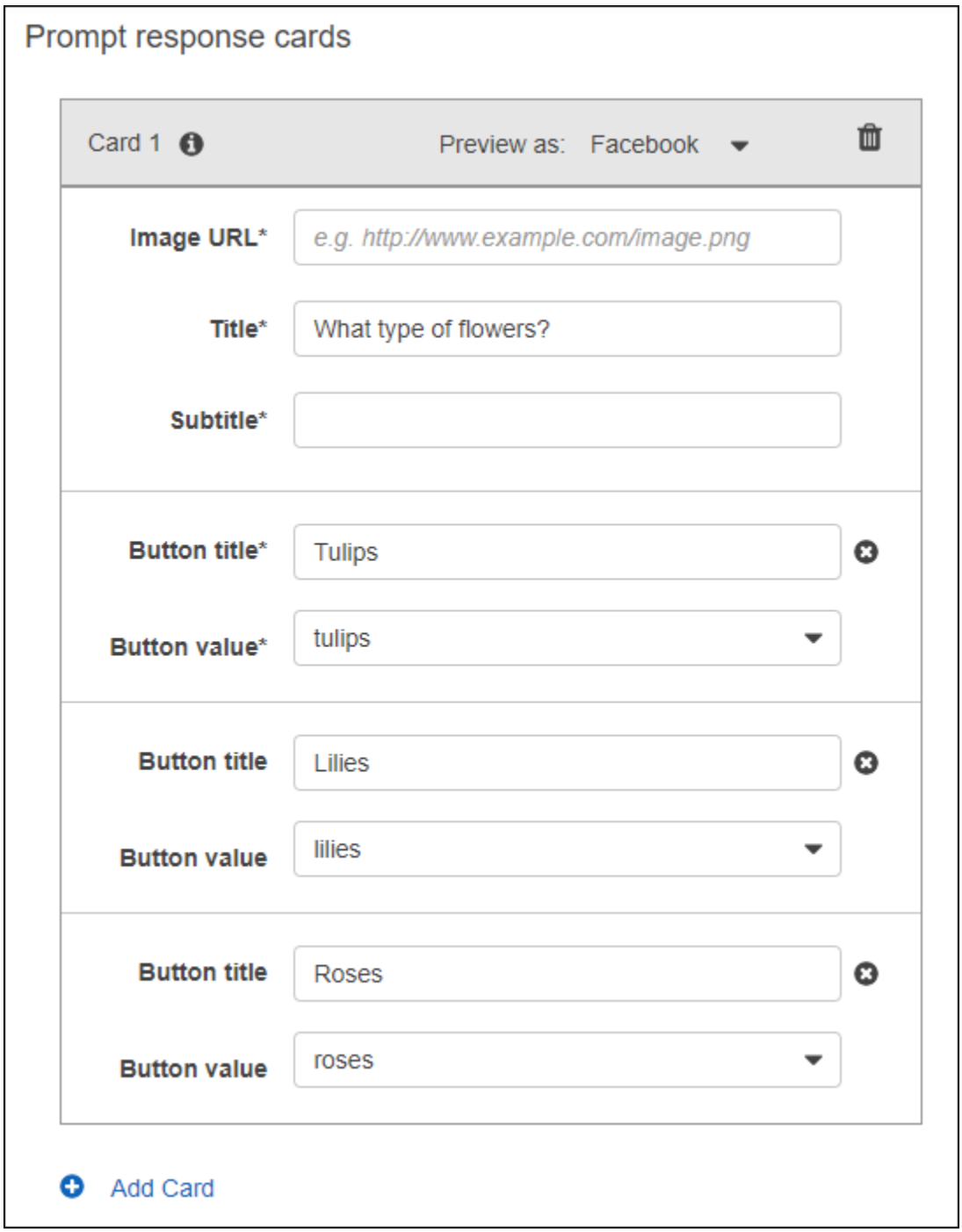
    a. Wählen Sie Ihren Bot in der Amazon-Lex-Konsole aus.

    b. Wählen Sie die Absicht `OrderFlowers` aus.

c. Wählen Sie das Zahnradsymbol für die Einstellungen neben der Aufforderung „Welche Art von Blumen“, um eine Antwortkarte für die zu konfigurieren`FlowerType`, wie in der folgenden Abbildung gezeigt.



d. Legen Sie einen Titel für die Karte fest und konfigurieren Sie drei Schaltflächen, wie im folgenden Screenshot gezeigt. Sie können der Antwortkarte optional ein Bild hinzufügen, vorausgesetzt, Sie haben eine Bild-URL. Wenn Sie Ihren Bot mithilfe von Twilio SMS bereitstellen, müssen Sie eine Bild-URL bereitstellen.



e. Wählen Sie Save (Speichern) aus, um die Antwortkarte zu speichern.

f. Wählen Sie Save intent (Absicht speichern) aus, um die Konfiguration der Absicht zu speichern.

g. Wählen Sie Build aus, um den Bot zu erstellen.

h. Wählen Sie zum Veröffentlichen einer Bot-Version Publish aus. Geben Sie BETA als Alias an, der auf die Bot-Version verweist. Weitere Informationen über Versioning finden Sie unter Versioning und Aliasnamen.

3. Stellen Sie den Bot auf einer Messaging-Plattform bereit:

   - Stellen Sie den Bot auf der Facebook Messenger-Plattform bereit und testen Sie die Integration. Detaillierte Anweisungen finden Sie unter Integration eines Amazon Lex Lex-Bot mit Facebook Messenger. Wenn Sie Blumen bestellen, zeigt das Mitteilungsfenster die Antwortkarte, sodass Sie eine Blumenart wählen können.
   - Stellen Sie den Bot auf der Slack-Plattform bereit und testen Sie die Integration. Detaillierte Anweisungen finden Sie unter Integration eines Amazon Lex Lex-Bot mit Slack . Wenn Sie Blumen bestellen, zeigt das Mitteilungsfenster die Antwortkarte, sodass Sie eine Blumenart wählen können.
   - Stellen Sie den Bot auf der Twilio SMS-Plattform bereit. Detaillierte Anweisungen finden Sie unter Integration eines Amazon Lex Lex-Bot mit Twilio Programmable SMS . Wenn Sie Blumen bestellen, zeigt die Mitteilung von Twilio das Bild der Antwortkarte. Twilio SMS unterstützt keine Schaltflächen in der Antwort.

Aktualisierung von Äußerungen

In dieser Übung fügen Sie den Äußerungen, die Sie in der Einstiegsübung 1 erstellt haben, weitere hinzu. Sie verwenden den Tab Monitoring in der Amazon Lex Lex-Konsole, um Äußerungen einzusehen, die Ihr Bot nicht erkannt hat. Zur Verbesserung der Kundenerfahrung Ihrer Benutzer fügen Sie diese Äußerungen zum Bot hinzu.

Unter den folgenden Bedingungen werden keine Äußerungsstatistiken generiert:

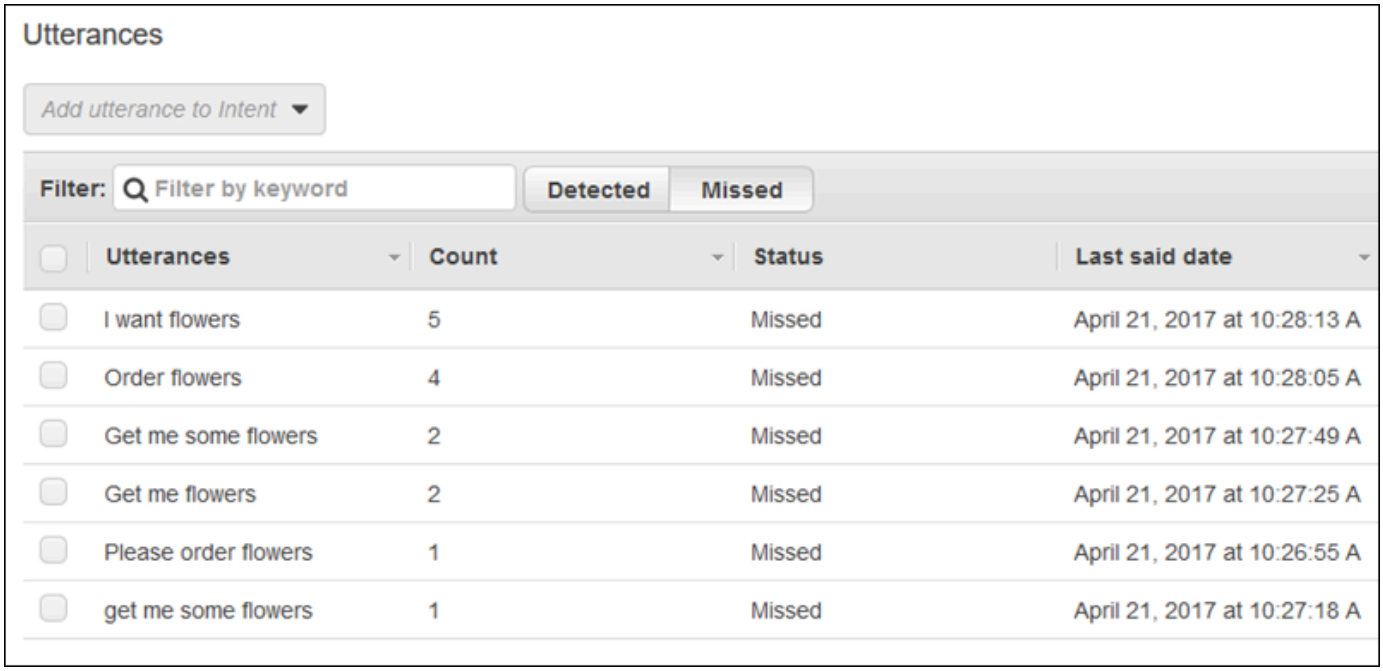
- Das `childDirected` Feld wurde auf true gesetzt, als der Bot erstellt wurde.
- Sie verwenden die Slot-Obfuscation mit einem oder mehreren Steckplätzen.
- Sie haben sich gegen die Teilnahme an der Verbesserung von Amazon Lex entschieden.

> **Note**
>
> Statistiken zu den Äußerungen werden einmal täglich erstellt. Sie sehen die nicht erkannte Äußerung, wie oft diese gehört wurde und das Datum und die Uhrzeit des letzten Hörens. Es kann bis zu 24 Stunden dauern, bis verpasste Äußerungen in der Konsole angezeigt werden.

Sie können Äußerungen für verschiedene Versionen Ihres Bots anzeigen. Um die Version Ihres Bots zu ändern, für das Sie Äußerungen anzeigen, wählen Sie eine andere Version aus der Dropdown-Liste neben dem Bot-Namen aus.

So zeigen Sie verpasste Äußerungen an und fügen diese zum Bot hinzu:

1. Befolgen Sie die Schritte der Einstiegsübung 1, um einen `OrderFlowers`-Bot zu erstellen und zu testen. Detaillierte Anweisungen finden Sie unter Übung 1: Erstellen Sie einen Amazon Lex Lex-Bot mithilfe eines Blueprints (Konsole).
2. Testen Sie den Bot, indem Sie die Äußerungen im Fenster Test Bot eingeben. Geben Sie jede Äußerung mehrmals ein. Der Beispiel-Bot erkennt die folgenden Äußerungen nicht:
   - Bestelle Blumen
   - Besorge mir Blumen
   - Bitte bestelle Blumen
   - Besorge mir einige Blumen
3. Warten Sie, bis Amazon Lex Nutzungsdaten zu den verpassten Äußerungen gesammelt hat. Daten zu den Äußerungen werden einmal pro Tag (in der Regel nachts) generiert.
4. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.
5. Wählen Sie den Bot `OrderFlowers` aus.
6. Wählen Sie die Registerkarte Monitoring und anschließend im linken Menü Utterances aus. Klicken Sie dann auf die Schaltfläche Missed. Im folgenden Bereich werden maximal 100 verpasste Äußerungen angezeigt.



7. Zum Auswählen verpasster Äußerungen, die Sie zum Bot hinzufügen möchten, aktivieren Sie das Kontrollkästchen neben der jeweiligen Äußerung. Zum Hinzufügen der Äußerung zur `$LATEST` Version der Absicht klicken Sie auf den Abwärtspfeil neben der Dropdown-Liste Add utterance to intent und wählen die Absicht aus.
8. Um den Bot neu zu erstellen, wählen Sie Build und dann erneut Build aus.
9. Prüfen Sie im Bereich Test Bot, ob der Bot die neuen Äußerungen erkennt.

Integration in eine Website

In diesem Beispiel integrieren Sie einen Bot per Text und Sprache in eine Website. Sie verwenden JavaScript und AWS-Services zur Entwicklung einer interaktiven Erfahrung für die Besucher Ihrer Website. Sie können aus diesen Beispielen auswählen, die im AWS AI-Blog dokumentiert sind:

- Stellen Sie eine Web-Benutzeroberfläche für Ihren Chatbot bereit— Zeigt eine Web-Benutzeroberfläche mit vollem Funktionsumfang, die einen Web-Client für Amazon Lex Lex-Chatbots bereitstellt. Sie können sie Beispiel verwenden, um mehr über Web-Clients zu erfahren, oder sie als Baustein für Ihre eigenen Anwendungen benutzen.

- "Greetings, visitor!" —Binden Sie Ihre Webbenutzer mit Amazon Lex— Zeigt, wie Sie Amazon Lex, dem AWS SDK for JavaScript in the Browser und Amazon Cognito eine dialogorientierte Benutzererfahrung für Ihre Website bereitstellen.

- Spracheingabe in einem Browser erfassen und an Amazon Lex senden— Zeigt das Einbetten eines sprachbasierten Chatbots in eine Website mit dem SDK for JavaScript in the Browser. Die Anwendung zeichnet Audiodaten auf, sendet diese an Amazon Lex und spielt die Antwort ab.

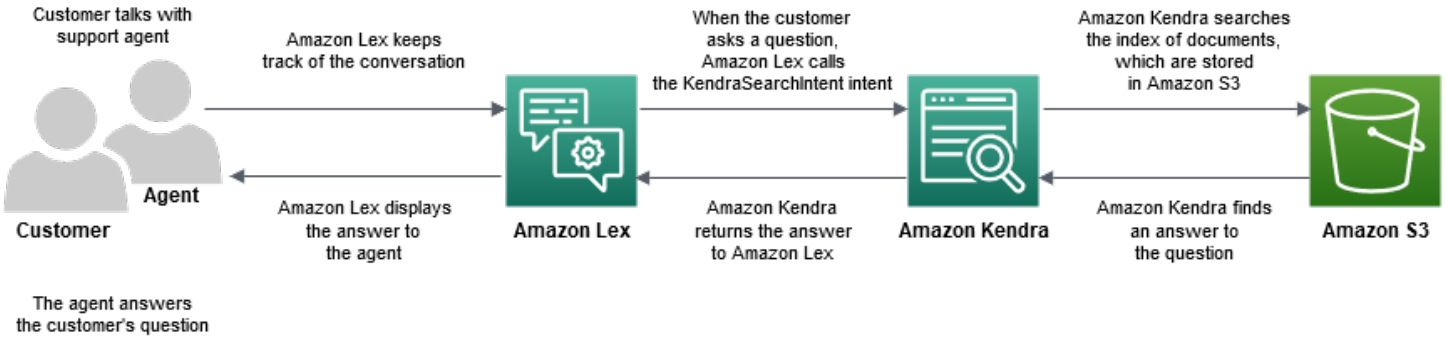
Assistentin des Call-Center-Agenten

In diesem Tutorial verwenden Sie Amazon Lex mit Amazon Kendra, um einen Bot zur Agentenunterstützung zu erstellen, der Kundenbetreuer unterstützt, und ihn als Webanwendung zu veröffentlichen. Amazon Kendra ist ein Suchdienst für Unternehmen, der maschinelles Lernen verwendet, um Dokumente zu durchsuchen, um Antworten zu finden. Weitere Informationen zu Amazon Kendra finden Sie im Amazon Kendra Developer Guide.

Amazon Lex Lex-Bots werden in Call Centern häufig als erste Anlaufstelle für Kunden eingesetzt. Ein Bot ist oft in der Lage, Kundenfragen zu lösen. Wenn ein Bot eine Frage nicht beantworten kann, leitet er die Konversation an einen Mitarbeiter des Kundensupports weiter.

In diesem Tutorial erstellen wir einen Amazon Lex Lex-Bot, mit dem Agenten Kundenanfragen in Echtzeit beantworten. Durch das Lesen der Antworten, die der Bot gibt, erspart es dem Agenten, manuell nach Antworten zu suchen.

Der Bot und die Webanwendung, die Sie in diesem Tutorial erstellen, helfen Agenten dabei, Kundenanfragen effizient und präzise zu beantworten, indem sie schnell die richtigen Ressourcen bereitstellen. Das folgende Diagramm zeigt, wie die Webanwendung funktioniert.



Wie das Diagramm zeigt, wird der Amazon Kendra Index in einem Amazon Simple Storage Service (Amazon S3) -Bucket gespeichert. Wenn Sie noch keinen S3-Bucket haben, können Sie einen

einrichten, wenn Sie den Amazon Kendra Kendra-Index erstellen. Zusätzlich zu Amazon S3 werden Sie Amazon Cognito für dieses Tutorial verwenden. Amazon Cognito verwaltet die Berechtigungen für die Bereitstellung des Bots als Webanwendung.

In diesem Tutorial erstellen Sie einen Amazon Kendra Kendra-Index, der Antworten auf Kundenfragen enthält, erstellen den Bot und fügen Absichten hinzu, die es ihm ermöglichen, Antworten auf der Grundlage der Konversation mit dem Kunden vorzuschlagen, Amazon Cognito für die Verwaltung von Zugriffsberechtigungen einzurichten und den Bot als Webanwendung bereitzustellen.

Geschätzte Zeit: 75 Minuten

Geschätzte Kosten: 2,50 USD pro Stunde für einen Amazon Kendra Kendra-Index und 0,75 USD für 1000 Amazon Lex Lex-Anfragen. Ihr Amazon Kendra Kendra-Index läuft weiter, nachdem Sie mit dieser Übung fertig sind. Löschen Sie es unbedingt, um unnötige Kosten zu vermeiden.

Hinweis: Stellen Sie sicher, dass Sie für alle in diesem Tutorial verwendeten Services dieselbe AWS-Region auswählen.

Themen

Schritt 1: Erstellen eines Amazon Kendra Kendra-Index

Erstellen Sie zunächst einen Amazon Kendra Kendra-Index mit Dokumenten, die Kundenfragen beantworten. Ein Index bietet eine Such-API für Kundenanfragen. Sie erstellen den Index aus Quelldokumenten. Amazon Kendra gibt Antworten, die es in indexierten Dokumenten findet, an den Bot zurück, der sie dem Agenten anzeigt.

Die Qualität und Genauigkeit der von Amazon Kendra vorgeschlagenen Antworten hängen von den Dokumenten ab, die Sie indexieren. Dokumente sollten Dateien enthalten, auf die der Agent häufig

zugreift und die in einem S3-Bucket gespeichert werden müssen. Sie können unstrukturierte und halbstrukturierte Daten in den Formaten .html, Microsoft Office (.doc, .ppt), PDF und Text indexieren.

Informationen zum Erstellen eines Amazon Kendra-Index finden Sie unter Erste Schritte mit einem S3-Bucket (Konsole) im Amazon Kendra Developer Guide.

Informationen zum Hinzufügen von Fragen und Antworten (FAQs) zur Beantwortung von Kundenanfragen finden Sie unter Hinzufügen von Fragen und Antworten im Amazon Kendra Developer Guide. Verwenden Sie für dieses Tutorial die Datei ML_FAQ.csv auf GitHub.

Nächster Schritt

Schritt 2: Erstellen Sie einen Amazon-Lex-Bot

Schritt 2: Erstellen Sie einen Amazon-Lex-Bot

Amazon Lex bietet eine Schnittstelle zwischen dem Call Center-Agenten und dem Amazon Kendra Kendra-Index. Es verfolgt die Konversation zwischen dem Agenten und dem Kunden und nennt die`AMAZON.KendraSearchIntent` Absicht auf der Grundlage der Fragen, die der Kunde stellt. Eine Absicht ist eine Aktion, die der Benutzer ausführen möchte.

Amazon Kendra durchsucht die indizierten Dokumente und gibt eine Antwort an Amazon Lex zurück, die im Bot angezeigt wird. Diese Antwort ist nur für den Agenten sichtbar.

Um einen Agenten-Assistenten-Bot zu erstellen

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Wählen Sie im Navigationsbereich Bots.
3. Wählen Sie Create (Erstellen) aus.
4. Wählen Sie Custom Bot und konfigurieren Sie den Bot.
    a. Botname — Geben Sie einen Namen ein, der den Zweck des Bots angibt, z.**`AgentAssistBot`** B.
    b. Stimme ausgeben — Wählen Sie „Keine“.
    c. Sitzungs-Timeout — Enter**5**.
    d. COPPA — Wähle Nein.
5. Wählen Sie Create (Erstellen) aus. Nach dem Erstellen des Bots zeigt Amazon Lex den Bot-Editor-Tab an.

Nächster Schritt

Schritt 3: Hinzufügen einer benutzerdefinierten und integrierten Absicht

Schritt 3: Hinzufügen einer benutzerdefinierten und integrierten Absicht

Eine Absicht stellt eine Aktion dar, die der Call Center-Agent vom Bot ausführen soll. In diesem Fall möchte der Agent, dass der Bot Antworten und hilfreiche Ressourcen vorschlägt, die auf der Konversation des Agenten mit dem Kunden basieren.

Amazon Lex hat zwei Arten von Absichten: benutzerdefinierte Absichten und integrierte Absichten. `AMAZON.KendraSearchIntent`ist eine eingebaute Absicht. Der Bot verwendet die`AMAZON.KendraSearchIntent` Absicht, den Index abzufragen und die von Amazon Kendra vorgeschlagenen Antworten anzuzeigen.

Der Bot in diesem Beispiel benötigt keine benutzerdefinierte Absicht. Um den Bot zu erstellen, müssen Sie jedoch mindestens eine benutzerdefinierte Absicht mit mindestens einer Beispieläußerung erstellen. Diese Absicht ist nur erforderlich, um Ihren Agentenassistenten-Bot zu erstellen. Es erfüllt keine andere Funktion. Die Äußerung der Absicht darf keine der Fragen beantworten, die der Kunde möglicherweise stellen könnte. Dadurch wird sichergestellt, dass der zur Beantwortung von Kundenanfragen aufgerufen`AMAZON.KendraSearchIntent` wird. Weitere Informationen finden Sie unter AMAZON.KendraSearchIntent.

Um die erforderliche benutzerdefinierte Absicht zu erstellen

1. Wählen Sie auf der Seite Getting started with your bot (Erste Schritte mit Ihrem Bot) die Option Create intent (Absicht erstellen).
2. Wählen Sie unter Add intent (Absicht hinzufügen) die Option Create intent (Absicht erstellen).
3. Geben Sie im Dialogfeld Absicht erstellen einen beschreibenden Namen für die Absicht ein, z.**`RequiredIntent`** B.
4. Geben Sie für Beispieläußerungen eine beschreibende Äußerung ein, z.**`Required utterance`** B.
5. Wählen Sie Save intent (Absicht speichern).

Um den AMAZON hinzuzufügen. KendraSearchIntent Absichts- und Antwortnachricht

1. Wählen Sie im Navigationsbereich das Pluszeichen (+) neben Intents aus.

2. Wählen Sie Nach vorhandenen Absichten suchen.
3. Geben Sie im Feld Suchabsichten den Text ein**`AMAZON.KendraSearchIntent`**, und wählen Sie ihn dann aus der Liste aus.
4. Geben Sie der Absicht einen aussagekräftigen Namen, z. B.**`AgentAssistSearchIntent`**, und wählen Sie dann Hinzufügen.
5. Wählen Sie im Absichtseditor Amazon Kendra query (Amazon Kendra-Abfrage), um die Abfrageoptionen zu öffnen.
6. Wählen Sie den Index aus, in dem die Absicht gesucht werden soll,
7. Fügen Sie im Abschnitt Antwort die folgenden drei Nachrichten zu einer Nachrichtengruppe hinzu.

```
I found an answer for the customer query: ((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 this answer will help the customer: ((x-amz-lex:kendra-search-response-
answer-1)).
```

8. Wählen Sie Save intent (Absicht speichern).
9. Wähle Build, um den Bot zu bauen.

Nächster Schritt

Schritt 4: Einrichten von Amazon Cognito

Schritt 4: Einrichten von Amazon Cognito

Für die Verwaltung von Berechtigungen und Benutzern für die Webanwendung müssen Sie Amazon Cognito einrichten. Amazon Cognito stellt sicher, dass die Webanwendung sicher ist und über eine Zugriffskontrolle verfügt. Amazon Cognito nutzt - mithilfe vonAWS Identitäten-Pools, um Benutzern den Zugriff auf andereAWS Services zu ermöglichen. Dieses Tutorial bietet Zugriff auf Amazon Lex.

Bei der Erstellung eines Identitäten-Pools stellt Ihnen Amazon Cognito -AWS Identity and Access Management (-) -Rollen für authentifizierte und nicht authentifizierte Benutzer bereit. Sie ändern die IAM-Rollen, indem Sie Richtlinien hinzufügen, die den Zugriff auf Amazon Lex gewähren.

Einrichten von Amazon Cognito

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon-Cognito-Cognito-Konsole unter https://console.aws.amazon.com/cognito/.
2. Klicken Sie auf Manage Identity Pools (Identitäten-Pools verwalten).
3. Wählen Sie Create new identity pool.
4. Konfigurieren Sie den Identitäten-Pool.

    a. Name des Identitätspools — Geben Sie einen Namen ein, der den Zweck des Pools angibt, z.**`BotPool`** B.

    b. Wählen Sie im Abschnitt Nicht authentifizierte Identitäten die Option Zugriff auf nicht authentifizierte Identitäten aktivieren aus.
5. Wählen Sie Create Pool.
6. Wählen Sie auf der Seite Identifizieren Sie die IAM-Rollen, die Sie mit Ihrem neuen Identitätspool verwenden möchten, die Option Details anzeigen aus.
7. Notieren Sie die Namen der - (-) -Rollen auf. Später werden Sie sie ändern.
8. Wählen Sie Allow.
9. Wählen Sie auf der Seite Erste Schritte mit Amazon Cognito für Plattform JavaScript.
10. Suchen Sie im Abschnitt „AWSAnmeldeinformationen abrufen“ die Identitätspool-ID und zeichnen Sie sie auf.
11. Um den Zugriff auf Amazon Lex zu ermöglichen, ändern Sie die authentifizierten und nicht authentifizierten IAM-Rollen.

    a. Melden Sie sich bei der AWS Management Console an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.

    b. Wählen Sie im Navigationsbereich unter Access Management die Option Roles aus.

    c. Geben Sie in das Suchfeld den Namen der authentifizierten - ((((((-Rolle ein und aktivieren Sie das Kontrollkästchen neben der Rolle.

        i. Wählen Sie Attach Policies (Richtlinien hinzufügen).

        ii. Geben Sie im Suchfeld das nebenstehende Feld ein**`AmazonLexRunBotsOnly`** und wählen Sie das Kontrollkästchen aus.

        iii. Wählen Sie Attach policy (Richtlinie anfügen) aus.

    d. Geben Sie den Namen der nicht authentifizierten - - Rolle in das Suchfeld ein und aktivieren Sie das Kontrollkästchen neben der Rolle.

i. Wählen Sie Attach Policies (Richtlinien hinzufügen).

ii. Geben Sie im Suchfeld das nebenstehende Feld ein**`AmazonLexRunBotsOnly`** und wählen Sie das Kontrollkästchen aus.

iii. Wählen Sie Attach policy (Richtlinie anfügen) aus.

Nächster Schritt

Schritt 5: Stellen Sie Ihren Bot als Webanwendung bereit

Schritt 5: Stellen Sie Ihren Bot als Webanwendung bereit

Um Ihren Bot als Webanwendung bereitzustellen

1. Laden Sie das Repository unter https://github.com/awsdocs/amazon-lex-developer-guide /blob/master/example_apps/agent_assistance_bot/ auf Ihren Computer herunter.
2. Navigieren Sie zum heruntergeladenen Repository und öffnen Sie die Datei index.html in einem Editor.
3. Nehmen Sie die folgenden Änderungen vor.

   a. Geben Sie im`AWS.config.credentials` Abschnitt Ihren Regionsnamen und Ihre Identitätspool-ID ein.

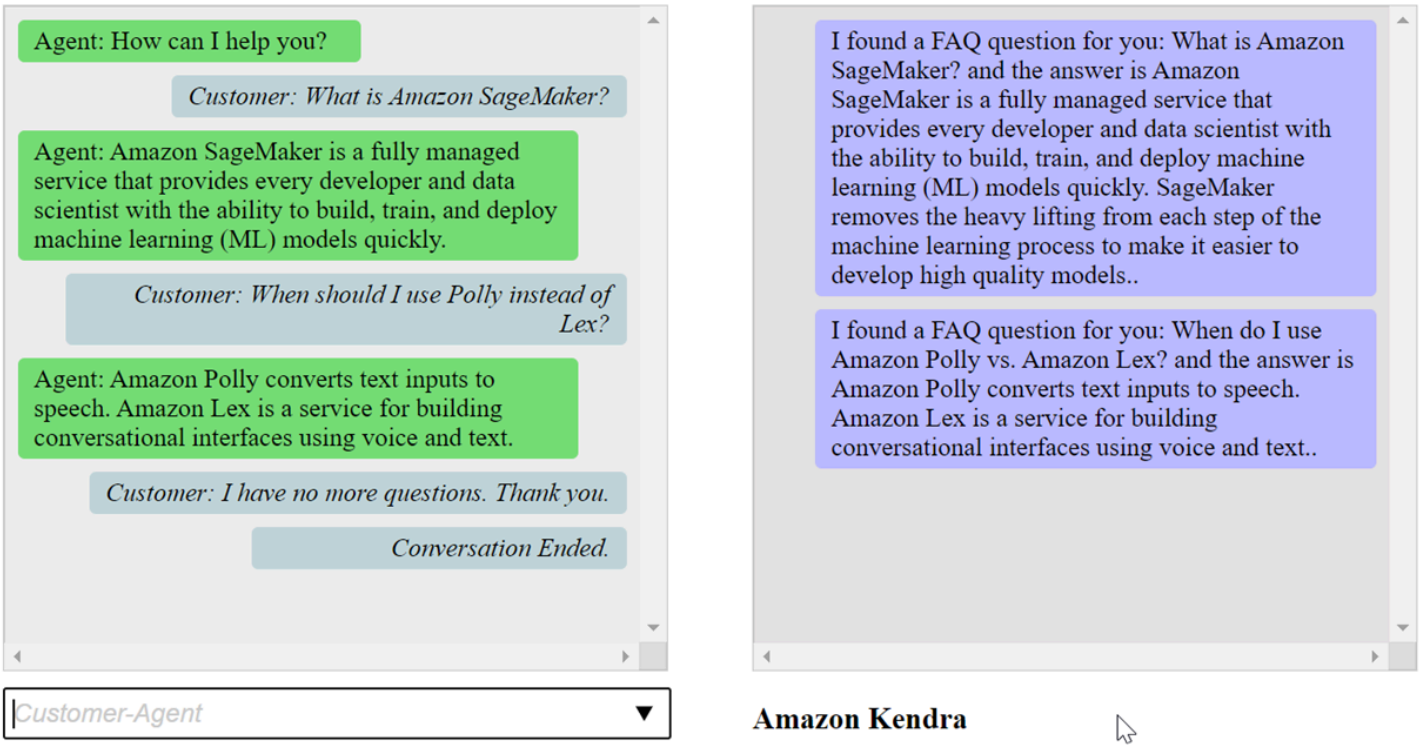
   b. Geben Sie in dem `Amazon Lex runtime parameters` Abschnitt den Bot-Namen ein.

   c. Speichern Sie die Datei.

Schritt 6: Benutze den Bot

Zu Demozwecken geben Sie dem Bot als Kunde und als Agent Eingaben. Um zwischen den beiden zu unterscheiden, beginnen die vom Kunden gestellten Fragen mit „Kunde:“ und die Antworten des Agenten beginnen mit „Agent:“. Sie können aus einem Menü mit vorgeschlagenen Eingaben wählen.

Starten Sie Ihre Webanwendung, indem Sie`index.html` sie öffnen, um mit Ihrem Bot eine Konversation zu führen, die der folgenden Abbildung ähnelt:



Die `pushChat()` Funktion in der Datei index.html wird im Folgenden erklärt.

```
            var endConversationStatement = "Customer: I have no more questions. Thank
you."
            // If the agent has to send a message, start the message with 'Agent'
            var inputText = document.getElementById('input');
            if (inputText && inputText.value && inputText.value.trim().length > 0 &&
inputText.value[0]=='Agent') {
                showMessage(inputText.value, 'agentRequest','conversation');
                inputText.value = "";
            }
            // If the customer has to send a message, start the message with 'Customer'
            if(inputText && inputText.value && inputText.value.trim().length > 0 &&
inputText.value[0]=='Customer') {
                // disable input to show we're sending it
                var input = inputText.value.trim();
                inputText.value = '...';
                inputText.locked = true;
                customerInput = input.substring(2);
```

```
                // Send it to the Lex runtime
                var params = {
                    botAlias: '$LATEST',
                    botName: 'KendraTestBot',
                    inputText: customerInput,
                    userId: lexUserId,
                    sessionAttributes: sessionAttributes
                };

                showMessage(input, 'customerRequest', 'conversation');
                if(input== endConversationStatement){
                    showMessage('Conversation
Ended.','conversationEndRequest','conversation');
                }
                lexruntime.postText(params, function(err, data) {
                    if (err) {
                        console.log(err, err.stack);
                        showMessage('Error:  ' + err.message + ' (see console for
details)', 'lexError', 'conversation1')
                    }

                    if (data &&input!=endConversationStatement) {
                        // capture the sessionAttributes for the next cycle
                        sessionAttributes = data.sessionAttributes;

                            showMessage(data, 'lexResponse', 'conversation1');
                    }
                    // re-enable input
                    inputText.value = '';
                    inputText.locked = false;
                });
            }
            // we always cancel form submission
            return false;
```

Wenn Sie als Kunde Eingaben bereitstellen, sendet die Amazon Lex-Laufzeit-API diese an Amazon Lex.

Die`showMessage(daText, senderRequest, displayWindow)` Funktion zeigt die Konversation zwischen dem Agenten und dem Kunden im Chatfenster an. Die von Amazon Kendra vorgeschlagenen Antworten werden in einem angrenzenden Fenster angezeigt. Die Konversation endet, wenn der Kunde sagt**“I have no more questions. Thank you.”**

Hinweis: Bitte löschen Sie Ihren Amazon Kendra Kendra-Index, wenn er nicht verwendet wird.

Einen Bot migrieren

Die Amazon Lex V2-API verwendet eine aktualisierte Informationsarchitektur, die eine vereinfachte Versionierung von Ressourcen und die Unterstützung mehrerer Sprachen in einem Bot ermöglicht. Weitere Informationen finden Sie im Migrationshandbuch im Amazon Lex V2-Entwicklerhandbuch.

Um diese neuen Funktionen nutzen zu können, müssen Sie Ihren Bot migrieren. Wenn Sie einen Bot migrieren, bietet Amazon Lex Folgendes:

- Bei der Migration werden Ihre benutzerdefinierten Absichten und Slot-Typen auf den Amazon Lex V2-Bot kopiert.
- Sie können demselben Amazon Lex V2-Bot mehrere Sprachen hinzufügen. In Amazon Lex V1 erstellen Sie für jede Sprache einen separaten Bot. Sie können mehrere Amazon Lex V1-Bots, die jeweils eine andere Sprache verwenden, auf einen Amazon Lex V2-Bot migrieren.
- Amazon Lex ordnet integrierte Amazon Lex V1-Slot-Typen und Intents den integrierten Amazon Lex V2-Slot-Typen und Intents zu. Wenn ein integriertes System nicht migriert werden kann, gibt Amazon Lex eine Nachricht zurück, in der Sie darüber informiert werden, was als Nächstes zu tun ist.

Der Migrationsprozess migriert Folgendes nicht:

- Aliasnamen
- Amazon Kendra-Indizes
- AWS Lambda-Funktionen
- Einstellungen von Gesprächsprotokollstruktur
- Messaging-Kanäle wie Slack
- Tags (Markierungen)

Um einen Bot zu migrieren, muss Ihr Benutzer oder Ihre Rolle über IAM-Berechtigungen für Amazon Lex- und Amazon Lex V2-API-Operationen verfügen. Die erforderlichen Berechtigungen finden Sie unterErlauben Sie einem Benutzer, einen Bot zu Amazon Lex V2-APIs zu migrieren.

Einen Bot migrieren (Konsole)

Verwenden Sie die Amazon Lex V1-Konsole, um die Struktur eines Bots auf einen Amazon Lex V2-Bot zu migrieren.

So verwenden Sie die Konsole, um einen Bot zur Amazon Lex V2-API zu migrieren

1. Melden Sie sich bei derAWS Management Console und öffnen Sie die Amazon-Lex-Konsole unter https://console.aws.amazon.com/lex/.
2. Wählen Sie im linken Menü das Migrationstool.
3. Wählen Sie aus der Liste der Bots den Bot aus, den Sie migrieren möchten, und wählen Sie dann Migrieren.
4. Wählen Sie die Version des Bots aus, den Sie migrieren möchten, und geben Sie dann den Namen des Bots ein, zu dem Sie migrieren möchten. Wenn Sie den Namen eines vorhandenen Amazon Lex V2-Bots eingeben, wird der Amazon Lex V1-Bot in die in den Details angezeigte Sprache migriert und überschreibt die Entwurfsversion der Sprache.
5. Wählen Sie Next (Weiter).
6. Wählen Sie die IAM-Rolle, die Amazon Lex verwendet, um die Amazon Lex V2-API-Version des Bots auszuführen. Sie können wählen, ob Sie eine neue Rolle mit den Mindestberechtigungen erstellen möchten, die für die Ausführung des Bots erforderlich sind, oder Sie können eine vorhandene IAM-Rolle wählen.
7. Wählen Sie Next (Weiter).
8. Überprüfen Sie die Einstellungen für die Migration. Wenn sie in Ordnung aussehen, wählen Sie Migration starten.

Nachdem Sie den Migrationsprozess gestartet haben, kehren Sie zur Startseite des Migrationstools zurück. Sie können den Fortschritt der Migration in der Protokoll-Tabellenstruktur überwachen. Wenn in der Spalte Migrationsstatus Abgeschlossen angezeigt wird, ist die Migration abgeschlossen.

Amazon Lex verwendet die`StartImport` Operation in der Amazon Lex V2-API, um den migrierten Bot zu importieren. In der Tabelle mit dem Importverlauf der Amazon Lex V2-Konsole wird für jede Migration ein Eintrag angezeigt.

Während der Migration findet Amazon Lex möglicherweise Ressourcen im Bot, die nicht migriert werden können. Sie erhalten eine Fehler- oder Warnmeldung für jede Ressource, die nicht migriert

werden kann. Jede Nachricht enthält einen Link zur Dokumentation, in der erklärt wird, wie das Problem behoben werden kann.

Migration einer Lambda-Funktion

Amazon Lex V2 ändert die Art und Weise, wie Lambda-Funktionen für einen Bot definiert werden. Es erlaubt nur eine Lambda-Funktion in einem Alias für jede Sprache in einem Bot. Weitere Informationen zur Migration Ihrer Lambda-Funktionen finden Sie unterMigration einer Lambda-Funktion von Amazon Lex V1 auf Amazon Lex V2.

Migrationsmeldungen

Während der Migration findet Amazon Lex möglicherweise Ressourcen, z. B. integrierte Slot-Typen, die nicht auf die entsprechende Amazon Lex V2-Ressource migriert werden kann. In diesem Fall gibt Amazon Lex eine Migrationsnachricht zurück, in der beschrieben wird, was passiert ist, und einen Link zur Dokumentation enthält, in der Sie erfahren, wie Sie das Migrationsproblem beheben können. In den folgenden Abschnitten werden die Probleme beschrieben, die bei der Migration eines Bots auftreten können, und wie Sie das Problem beheben können.

Themen

- Integrierte Absicht
- Typ mit integriertem Steckplatz
- Gesprächsprotokolle
- Mitteilungsgruppen
- Eingabeaufforderungen und Ausdrücke
- Weitere Funktionen von Amazon Lex V1struktur

Integrierte Absicht

Wenn Sie eine integrierte Absicht verwenden, die in Amazon Lex V2 nicht unterstützt wird, wird die Absicht einer benutzerdefinierten Absicht in Ihrem Amazon Lex V2-Bot zugeordnet. Die benutzerdefinierte Absicht enthält keine Äußerungen. Um die Absicht weiterhin zu verwenden, fügen Sie Beispieläußerungen hinzu.

Typ mit integriertem Steckplatz

Jedem migrierten Steckplatz, der einen Slot-Typ verwendet, der in Amazon Lex V2 nicht unterstützt wird, wird kein Slot-Typwert zugewiesen. Um diesen Slot zu verwenden:

- Erstellen Sie einen benutzerdefinierten Slot-Typ
- Fügen Sie Slot-Typwerte hinzu, die für den Slot-Typ erwartet werden
- Aktualisieren Sie den Slot, um den neuen benutzerdefinierten Slot-Typ zu verwenden

Gesprächsprotokolle

Durch die Migration werden die Konversationsprotokolleinstellungen des Amazon Lex V2-Bots nicht aktualisiert.

So konfigurieren Sie Konversationsprotokolle

1. Öffnen Sie die Amazon Lex V2-Konsole unter https://console.aws.amazon.com/lexv2.
2. Wählen Sie aus der Liste der Bots den Bot aus, dessen Konversationsprotokolle Sie konfigurieren möchten.
3. Wählen Sie im linken Menü Aliase und wählen Sie dann einen Alias aus der Liste aus.
4. Wählen Sie im Abschnitt Konversationsprotokolle die Option Konversationsprotokolle verwalten aus, um Konversationsprotokolle für den Bot-Alias zu konfigurieren.

Mitteilungsgruppen

Amazon Lex V2 unterstützt nur eine Nachricht und zwei alternative Nachrichten pro Nachrichtengruppe. Wenn Sie mehr als drei Nachrichten pro Nachrichtengruppe in einem Amazon Lex V1-Bot haben, werden nur die ersten drei Nachrichten migriert. Um mehr Nachrichten in einer Nachrichtengruppe zu verwenden, verwenden Sie eine Lambda-Funktion, um verschiedene Nachrichten auszugeben.

Eingabeaufforderungen und Ausdrücke

Amazon Lex V2 verwendet einen anderen Mechanismus zur Nachverfolgung, Klärung und zum Auflegen von Aufforderungen.

Verwenden Sie für Folgeanfragen die Übertragung von Kontexten, um nach der Erfüllung zu einer anderen Absicht zu wechseln.

Nehmen wir wir wir wir wir wir an, dass Sie beabsichtigen, einen Mietwagen zu buchen, der so konfiguriert ist, dass Sie einen Ereignismeldungen erhaltenbook_car_fulfilled. Wenn die Absicht erfüllt ist, setzt Amazon Lex die Ausgabekontextvariable aufbook_car_fulfilled. Dabook_car_fulfilled es sich um einen aktiven Kontext handelt, wird eine Absicht mitbook_car_fulfilled als Eingabekontext zur Erkennung in Betracht gezogen, sofern die Äußerung des Benutzers als Versuch erkannt wird, diese Absicht hervorzurufen. Sie können dies für Zwecke verwenden, die erst nach der Buchung eines Fahrzeugs sinnvoll sind, z. B. das Versenden einer Quittung per E-Mail oder die Änderung einer Reservierung.

Amazon Lex V2 unterstützt keine Aufforderungen zur Klärung und Ausdrücke zum Auflegen (Abbrucherklärungen). Amazon Lex V2-Bots enthalten eine standardmäßige Fallback-Absicht, die aufgerufen wird, wenn keine Absichten übereinstimmen. Um eine Aufforderung zur Klärung mit Wiederholungen zu senden, konfigurieren Sie eine Lambda-Funktion und aktivieren Sie den Dialogcode-Hook in der Fallback-Absicht. Die Lambda-Funktion kann eine Aufforderung zur Klärung als Antwort und den Wiederholungswert in einem Sitzungsattribut ausgeben. Wenn der Wiederholungswert die maximale Anzahl von Wiederholungen überschreitet, können Sie eine Aufhangphrase ausgeben und die Konversation schließen.

Weitere Funktionen von Amazon Lex V1struktur

Das Migrationstool unterstützt nur die Migration von Amazon Lex V1-Bots und ihren zugrundeliegenden Absichten, Slot-Typen und Slots. Weitere Funktionen finden Sie unter den folgenden Themen in der Dokumentation von Amazon Lex V2.

- Bot-Aliase: Aliase
- Bot-Kanäle: Einsatz eines Amazon Lex V2-Bots auf einer Messaging-Plattform
- Einstellungen für Konversationsprotokolle: Überwachung mit Konversationsprotokollen
- Amazon Kendra-Indizes: AMAZON. KendraSearchIntent
- Lambda-Funktionen: EineAWS Lambda Funktion verwenden
- Schlagworte: Ressourcen kennzeichnen

Migration einer Lambda-Funktion von Amazon Lex V1 auf Amazon Lex V2

Amazon Lex V2 erlaubt nur eine Lambda-Funktion für jede Sprache in einem Bot. Die Lambda-Funktion und ihre Einstellungen sind für den Bot-Alias konfiguriert, den Sie zur Laufzeit verwenden.

Die Lambda-Funktion wird in dieser Sprache in jeder Hinsicht aufgerufen, wenn Dialog- und Fulfillment-Code-Hooks für die Absicht aktiviert sind.

Amazon Lex V2 Lambda-Funktionen haben ein anderes Eingabe- und Ausgabe-Nachrichtenformat als Amazon Lex V1. Dies sind die Unterschiede im Eingabeformat der Lambda-Funktion.

- Amazon Lex V2 ersetzt die`currentIntent` und`alternativeIntents` -Strukturen durch die`interpretations` Struktur. Jede Interpretation enthält eine Absicht, den NLU-Konfidenzwert für die Absicht und eine optionale Stimmungsanalyse.
- Amazon Lex V2 verschiebt das`activeContexts`,`sessionAttributes` in Amazon Lex V1 auf die einheitliche`sessionState` Struktur. Diese Struktur bietet Informationen zum aktuellen Status der Konversation, einschließlich der ursprünglichen Ereignismeldungen.
- Amazon Lex V2 gibt das nicht zurück`recentIntentSummaryView`. Verwenden Sie stattdessen die Informationen in der`sessionState` Struktur.
- Die Amazon Lex V2-Eingabe stellt das`botId` und`localeId` im`bot` Attribut bereit.
- Die Eingabestruktur enthält ein`inputMode` Attribut, das Informationen über die Art der Eingabe bereitstellt: Text, Sprache oder DTMF.

Dies sind die Unterschiede im Ausgabeformat der Lambda-Funktion:

- Die`activeContexts` und`sessionAttributes` -Strukturen in Amazon Lex V1 werden durch die`sessionState` Struktur in Amazon Lex V2 ersetzt.
- Das`recentIntentSummaryView` ist nicht in der Ausgabe enthalten.
- Die Amazon Lex`dialogAction` V1-Struktur ist in zwei Strukturen aufgeteilt,`dialogAction` die Teil der`sessionState` Struktur sind und`messages` die`dialogAction.type` erforderlich`ElicitIntent` sind, wenn Amazon Lex wählt Nachrichten aus dieser Struktur aus, um sie dem Benutzer anzuzeigen.

Wenn Sie einen Bot mit den Amazon Lex V2-APIs erstellen, gibt es nur eine Lambda-Funktion pro Bot-Alias pro Sprache anstelle einer Lambda-Funktion für jede Absicht. Wenn Sie weiterhin separate

Funktionen verwenden möchten, können Sie eine Routerfunktion erstellen, die für jede Absicht eine separate Funktion aktiviert. Im Folgenden finden Sie eine Routerfunktion, die Sie für Ihre Anwendung verwenden oder ändern können.

```
import os
import json
import boto3

# reuse client connection as global
client = boto3.client('lambda')

def router(event):
    intent_name = event['sessionState']['intent']['name']
    fn_name = os.environ.get(intent_name)
    print(f"Intent: {intent_name} -> Lambda: {fn_name}")
    if (fn_name):
        # invoke lambda and return result
        invoke_response = client.invoke(FunctionName=fn_name, Payload =
 json.dumps(event))
        print(invoke_response)
        payload = json.load(invoke_response['Payload'])
        return payload
    raise Exception('No environment variable for intent: ' + intent_name)

def lambda_handler(event, context):
    print(event)
    response = router(event)
    return response
```

Liste der aktualisierten Felder

Die folgenden Tabellen enthalten detaillierte Informationen zu den aktualisierten Feldern in der Amazon Lex V2 Lambda-Anfrage und -Antwort. Sie können diese Tabellen verwenden, um Felder zwischen den Versionen zuzuordnen.

Anfrage

Die folgenden Felder wurden im Format der Lambda-Funktionsanforderung aktualisiert.

Aktive Kontexte

Die`activeContexts` Struktur ist jetzt Teil der`sessionState` Struktur.

V1struktur	Struktur von V2
Aktive Kontexte	SessionState.ActiveContexts
Aktive Kontexte [*]. timeToLive	sessionState.activeContexts [*]. timeToLive
Aktive Kontexte [*]. timeToLive. timeToLiv eInSeconds	sessionState.activeContexts [*]. timeToLive. timeToLiveInSeconds
Aktive Kontexte [*]. timeToLive. turnsToLive	sessionState.activeContexts [*]. timeToLive. turnsToLive
ActiveContexts [*] .name	sessionState.activeContexts [*] .name
activeContexts [*] .parameter	sessionState.activeContexts [*] .context-Attribute

Alternative Absichten

Die Interpretationsliste von Index 1 bis N enthält die Liste der von Amazon Lex V2 vorhergesagten alternativen Absichten sowie deren Konfidenzwerte. Das`recentIntentSummaryView` wird aus der Anforderungsstruktur in Amazon Lex V2 entfernt. Um die Details von zu sehen`recentIntentSummaryView`, verwenden Sie dieGetSession Operation.

V1struktur	Struktur von V2
Alternative Inhalte	Interpretationen [1: *]
recentIntentSummaryAnsicht	–

Bot

In Amazon Lex V2 haben Bots und Aliase Identifikatoren. Die Bot-ID ist Teil der Codehook-Eingabe. Die Alias-ID ist enthalten, aber nicht der Aliasname. Amazon Lex V2 unterstützt mehrere Gebietsschemas für denselben Bot, sodass die Gebietsschema-ID enthalten ist.

V1struktur	Struktur von V2
Bot	Bot
Bot.name	Bot.name
–	bot.id
bot.alias	–
–	Bot.AliasID
Bot-Version	Bot-Version
–	Bot.LocaleID

Aktuelle Absicht

Die`sessionState.intent` Struktur enthält die Details der aktiven Absicht. Amazon Lex V2 gibt außerdem eine Liste aller Intents, einschließlich alternativer Intents, in der`interpretations` Struktur zurück. Das erste Element in der Interpretationsliste ist immer dasselbe wie`sessionState.intent`.

V1struktur	Struktur von V2
currentIntent	sessionState.intention ODER Interpretationen [0] .intent
Aktueller Intent.Name	sessionState.Intent.Name ODER Interpret ationen [0] .intent.name
currentIntent. nluConfidenceScore	Interpretationen [0] .nluConfidence.Score

Aktion „Dialog“

Das`confirmationStatus` Feld ist jetzt Teil der`sessionState` Struktur.

V1struktur	Struktur von V2
Aktueller Intent.Bestätigungsstatus	sessionState.Intent.ConfirmationState ODER Interpretationen [0] .intent.ConfirmationState
–	sessionState.Intent.State ODER Interpret ationen [*] .intent.state

Amazon Kendra

Das`kendraResponse` Feld ist jetzt Teil der`sessionState` und`interpretations` -Strukturen.

V1struktur	Struktur von V2
kendraResponse	sessionState.Intent.kendraResponse ODER Interpretationen [0] .intent.kendraResponse

Stimmung

Die`sentimentResponse` Struktur wird in die neue`interpretations` Struktur verschoben.

V1struktur	Struktur von V2
sentimentResponse	Interpretationen [0] .sentimentResponse
Stimmungsantwort.SentimentLabel	Interpretationen [0] .sentimentResponse .Sentiment
Stimmungsantwort. Stimmungswert	Interpretationen [0] .sentimentResponse .sentimentScore

Slots

Amazon Lex V2 stellt ein einzelnes`slots` Objekt innerhalb der`sessionState.intent` Struktur bereit, das die aufgelösten Werte, den interpretierten Wert und den ursprünglichen Wert dessen, was der Benutzer gesagt hat, enthält. Amazon Lex V2 unterstützt auch Steckplätze mit mehreren Werten,

indem`slotShape` es die als`List` und die`values` Liste festlegt. Steckplätze mit einem Wert werden vom`value` Feld unterstützt, ihre Form wird angenommen`Scalar`.

V1struktur	Struktur von V2
Aktuell in TENT.SLOTS	sessionState.Intent.slots ODER Interpret ationen [0] .intent.slots
CurrentIntent.slots [*] .value	sessionState.Intent.slots [*] .value.interpreted Value ODER Interpretationen [0] .intent.slots [*] .value.interpretedValue
–	sessionState.Intent.slots [*] .value.shape ODER Interpretationen [0] .intent.slots [*] .shape
–	sessionState.Intent.slots [*] .values ODER Interpretationen [0] .intent.slots [*] .values
Aktuelle Intent.Slotdetails	sessionState.Intent.slots ODER Interpret ationen [0] .intent.slots
Aktuelle Intent.SlotDetails [*]. Auflösungen	sessionState.Intent.slots [*] .resolvedValues ODER Interpretationen [0] .intent.slots [*] .resolvedValues
Aktuelle Intent.SlotDetails [*]. Originalwert	sessionState.Intent.slots [*] .OriginalValue ODER Interpretationen [0] .intent.slots [*] .originalValue

Weitere

Das Amazon Lex`sessionId` V2-Feld ist dasselbe wie das`userId` Feld in Amazon Lex V1. Amazon Lex V2 sendet auch die`inputMode` des Anrufers: Text, DTMF oder Sprache.

V1struktur	Struktur von V2
userId	sessionId

V1struktur	Struktur von V2
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion
sessionAttributes	SessionState.Session-Attribute
requestAttributes	requestAttributes
–	Eingabemodus
–	originatingRequestId

Antwort

Die folgenden Felder wurden im Antwortnachrichtenformat der Lambda-Funktion geändert.

Aktive Kontexte

Die`activeContexts` Struktur wurde in die`sessionState` Struktur verschoben.

V1struktur	Struktur von V2
Aktive Kontexte	SessionState.ActiveContexts
Aktive Kontexte [*]. timeToLive	sessionState.activeContexts [*]. timeToLive
Aktive Kontexte [*]. timeToLive. timeToLiv eInSeconds	sessionState.activeContexts [*]. timeToLive. timeToLiveInSeconds
Aktive Kontexte [*]. timeToLive. turnsToLive	sessionState.activeContexts [*]. timeToLive. turnsToLive
ActiveContexts [*] .name	sessionState.activeContexts [*] .name

V1struktur	Struktur von V2
activeContexts [*] .parameter	sessionState.activeContexts [*] .context-Attribute

Aktion „Dialog“

Die`dialogAction` Struktur wurde in die`sessionState` Struktur verschoben. Sie können jetzt mehrere Nachrichten in einer Dialogaktion angeben, und die`genericAttachments` Struktur ist jetzt die`imageResponseCard` Struktur.

V1struktur	Struktur von V2
dialogAction	sessionState.DialogAction
DialogAction.Type	SessionState.DialogAction.Type
dialogAction. slotToElicit	SessionState.Intent.DialogAction. slotToElicit
DialogAction.Type.FulfillmentState	SessionState.Intent.State
DialogAction.Nachricht	messages
DialogAction.Message.ContentType	Nachrichten [*] .ContentType
DialogAction.Message.Inhalt	Nachrichten [*] .content
DialogAction.Antwortkarte	Nachrichten [*]. imageResponseCard
DialogAction.Antwortkarte.Version	–
DialogAction.ResponseCard.ContentType	Nachrichten [*] .ContentType
DialogAction.ResponseCard.Generische Anhänge	–
DialogAction.ResponseCard.GenericAttachments [*] .title	Nachrichten [*]. imageResponseCard.title

V1struktur	Struktur von V2
DialogAction.ResponseCard.GenericAttachments [*] .subtitle	Nachrichten [*]. imageResponseCard.untertitel
DialogAction.ResponseCard.GenericAttachments [*] .imageURL	Nachrichten [*]. imageResponseCard. imageUrl
DialogAction.ResponseCard.GenericAttachments [*] .buttons	Nachrichten [*]. imageResponseCard.tasten
DialogAction.ResponseCard.GenericAttachments [*] .buttons [*] .value	Nachrichten [*]. imageResponseCard.buttons [*] .value
DialogAction.ResponseCard.GenericAttachments [*] .buttons [*] .text	Nachrichten [*]. imageResponseCard.buttons [*] .text
dialogAction. kendraQueryRequestNutzlast	dialogAction. kendraQueryRequestNutzlast
dialogAction. kendraQueryFilterSchnur	dialogAction. kendraQueryFilterSchnur

Absichten und Slots

Absichts- und Slot-Felder, die Teil der`dialogAction` Struktur waren, sind jetzt Teil der`sessionState` Struktur.

V1struktur	Struktur von V2
DialogAction.IntentName	SessionState.Intent.Name
DialogAction.Spielautomaten	SessionState.Intent.Slots
DialogAction.Slots [*] .key	sessionState.Intent.slots [*] .key
DialogAction.Slots [*] .value	sessionState.Intent.Slots [*] .value.interpretedValue
–	sessionState.Intent.Slots [*] .value.shape

V1struktur	Struktur von V2
–	sessionState.Intent.slots [*] .values

Weitere

Die`sessionAttributes` Struktur ist jetzt Teil der`sessionState` Struktur. Die`recentIntentSummaryReview` Struktur wurde entfernt.

V1struktur	Struktur von V2
sessionAttributes	SessionState.Session-Attribute
recentIntentSummaryAnsicht	–

Sicherheit in Amazon Lex

Cloud-Sicherheit bei AWS hat höchste Priorität. Als - AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die entwickelt wurde, um die Anforderungen der sicherheitssensibelsten Organisationen zu erfüllen.

Sicherheit ist eine geteilte Verantwortung zwischen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist für den Schutz der Infrastruktur verantwortlich, die Services in der AWS Cloud ausführt AWS . stellt Ihnen AWS außerdem Services bereit, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS -Compliance-Programms getestet und überprüft. Informationen zu den Compliance-Programmen, die für Amazon Lex gelten, finden Sie unter AWS Im Rahmen des Compliance-Programms zugelassene -Services.
- Sicherheit in der Cloud – Ihre Verantwortung wird durch den - AWS Service bestimmt, den Sie verwenden. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation erläutert, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von Amazon Lex einsetzen können. Die folgenden Themen zeigen Ihnen, wie Sie Amazon Lex zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren. Sie erfahren auch, wie Sie andere AWS-Services verwenden, die Sie bei der Überwachung und Sicherung Ihrer Amazon Lex-Ressourcen unterstützen.

Themen

Datenschutz in Amazon Lex

Amazon Lex sammelt Kundeninhalte zur Fehlerbehebung und zur Verbesserung des Services. Kundeninhalte werden standardmäßig geschützt. Sie können Inhalte für einzelne Kunden mithilfe der Amazon Lex-API löschen.

Amazon Lex speichert vier Arten von Inhalten:

- Beispieläußerungen, die zum Erstellen und Trainieren eines Bots verwendet werden
- Kundenäußerungen, die Benutzer bei Interaktionen mit dem Bot äußern
- Sitzungsattribute, die während der Interaktion eines Benutzers mit einem Bot anwendungsspezifische Informationen bereitstellen
- Anforderungsattribute, die Informationen enthalten, die für eine einzelne Anforderung an einen Bot gelten

Jeder Amazon Lex-Bot, der für die Verwendung durch Kind konzipiert ist, unterliegt dem Online Privacy Protection Act (COPPA) des s. Sie teilen Amazon Lex mit, dass der Bot COPPA unterliegt, indem Sie die Konsole oder die Amazon Lex-API verwenden, um das `childDirected` Feld auf festzulegen`true`. Wenn das Feld `childDirected` auf auf `true` festgelegt ist, werden keine Benutzeräußerungen gespeichert.

Themen

- Verschlüsselung im Ruhezustand
- Verschlüsselung während der Übertragung
- Schlüsselverwaltung

Verschlüsselung im Ruhezustand

Amazon Lex verschlüsselt die gespeicherten Benutzeräußerungen.

Themen

- Beispieläußerungen
- Kundenäußerungen
- Sitzungsattribute
- Anforderungsattribute

Beispieläußerungen

Sie können während der Entwicklung eines Bots Beispieläußerungen für jede Absicht und jeden Slot bereitstellen. Sie können auch benutzerdefinierte Werte und Synonyme für Slots bereitstellen. Diese Informationen werden im Ruhezustand verschlüsselt und werden verwendet, um den Bot zu erstellen und die Benutzererfahrung zu erstellen.

Kundenäußerungen

Amazon Lex verschlüsselt Äußerungen, die Benutzer an Ihren Bot senden, es sei denn, das `childDirected` Feld ist auf gesetzt`true`.

Wenn das Feld `childDirected` auf auf `true` festgelegt ist, werden keine Benutzeräußerungen gespeichert.

Wenn das Feld `childDirected` auf `false` festgelegt ist (Standardeinstellung), werden Benutzeräußerungen verschlüsselt und 15 Tage für die Verwendung mit der Operation GetUtterancesView aufbewahrt. Um gespeicherte Äußerungen eines bestimmten Benutzers zu löschen, verwenden Sie die Operation DeleteUtterances.

Wenn Ihr Bot Spracheingaben akzeptiert, wird die Eingabe auf unbestimmte Zeit gespeichert. Amazon Lex verwendet sie, um die Fähigkeit Ihres Bots zu verbessern, auf Benutzereingaben zu reagieren.

Zum Löschen der gespeicherten Äußerungen eines bestimmten Benutzers verwenden Sie die Operation DeleteUtterances.

Sitzungsattribute

Sitzungsattribute enthalten anwendungsspezifische Informationen, die zwischen Amazon Lex und Clientanwendungen übergeben werden. Amazon Lex übergibt Sitzungsattribute an alle für einen Bot konfigurierten AWS Lambda Funktionen. Wenn eine Lambda-Funktion Sitzungsattribute hinzufügt oder aktualisiert, übergibt Amazon Lex die neuen Informationen an die Clientanwendung.

Sitzungsattribute bleiben in einem verschlüsselten Speicher für die Dauer der Sitzung erhalten. Sie können die Sitzung konfigurieren, sodass sie für mindestens eine Minute und höchstens 24 Stunden nach derletzten Äußerung des Benutzers aktiv bleibt. Die Standardsitzungsdauer beträgt 5 Minuten.

Anforderungsattribute

Anforderungsattribute enthalten anforderungsspezifische Informationen und gelten nur für die jeweils aktuelle Anforderung. Eine Client-Anwendung verwendet Anforderungsattribute, um zur Laufzeit Informationen an Amazon Lex zu senden.

Verwenden Sie Anforderungsattribute zur Weitergabe von Informationen, die nicht während der ganzen Sitzung erhalten bleiben müssen. Da Anforderungsattribute nicht anforderungsübergreifend erhalten bleiben, werden sie nicht gespeichert.

Verschlüsselung während der Übertragung

Amazon Lex verwendet das HTTPS-Protokoll, um mit Ihrer Client-Anwendung zu kommunizieren. Es verwendet HTTPS- und AWS-Signaturen, um mit anderen -Services wie Amazon Polly und im Namen AWS Lambda Ihrer Anwendung zu kommunizieren.

Schlüsselverwaltung

Amazon Lex schützt Ihre Inhalte vor unbefugter Verwendung mit internen Schlüsseln.

Identity and Access Management für Amazon Lex

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Amazon Lex-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Problembehandlung bei Identität und Zugriff auf Amazon Lex

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Amazon Lex ausführen.

Servicebenutzer — Wenn Sie den Amazon Lex Lex-Service für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Da Sie für Ihre Arbeit mehr Amazon Lex-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie auf eine Funktion in Amazon Lex nicht zugreifen können, finden Sie weitere Informationen unterProblembehandlung bei Identität und Zugriff auf Amazon Lex.

Service-Administrator — Wenn Sie in Ihrem Unternehmen für die Amazon Lex-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon Lex. Es ist Ihre Aufgabe, zu bestimmen, auf welche Funktionen und Ressourcen von Amazon Lex Ihre Servicebenutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM mit Amazon Lex verwenden kann, finden Sie unterSo funktioniert Amazon Lex mit IAM.

IAM-Administrator — Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf Amazon Lex zu verwalten. Beispiele für identitätsbasierte Amazon Lex Lex-Richtlinien, die Sie in IAM verwenden können, finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Amazon Lex

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen

Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, mit denen Sie Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch signieren können. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAM-Benutzerhandbuch unter AWS API-Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige verwenden Funktionen in anderen. AWS-Services AWS-Services Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon-EC2 aus oder speichert Objekte in Amazon-S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services

könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Anwendungen, die auf Amazon EC2 ausgeführt werden — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen AWS CLI . AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen

in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und

Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service Control Policies (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung

mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

So funktioniert Amazon Lex mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf Amazon Lex zu verwalten, sollten Sie sich darüber informieren, welche IAM-Funktionen für Amazon Lex verfügbar sind.

IAM-Funktionen, die Sie mit Amazon Lex verwenden können

IAM-Feature	Amazon Lex Lex-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja

IAM-Feature	Amazon Lex Lex-Unterstützung
Richtlinienbedingungsschlüssel (servicespezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Ja

Einen allgemeinen Überblick darüber, wie Amazon Lex und andere AWS Services mit den meisten IAM-Funktionen funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Services, die mit IAM funktionieren.

Identitätsbasierte Richtlinien für Amazon Lex

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon Lex

Beispiele für identitätsbasierte Richtlinien von Amazon Lex finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Amazon Lex

Ressourcenbasierte Richtlinien innerhalb von Amazon Lex

Unterstützt ressourcenbasierte Richtlinien	Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM-Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Politische Maßnahmen für Amazon Lex

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Amazon Lex-Aktionen finden Sie unter Von Amazon Lex definierte Aktionen in der Service Authorization Reference.

Richtlinienaktionen in Amazon Lex verwenden das folgende Präfix vor der Aktion:

```
lex
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "lex:Describe*"
```

Richtlinienressourcen für Amazon Lex

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Ein Amazon Lex Lex-Bot-Ressourcen-ARN hat das folgende Format.

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon Resource Names (ARNs) und AWS Service Namespaces.

Um beispielsweise den Bot `OrderFlowers` in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN.

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:OrderFlowers"
```

Um alle Bots anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*).

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:*"
```

Einige Amazon Lex Lex-Aktionen, z. B. zum Erstellen von Ressourcen, können nicht für eine bestimmte Ressource ausgeführt werden. In diesen Fällen müssen Sie den Platzhalter (*) verwenden.

```
"Resource": "*"
```

Eine Liste der Amazon Lex-Ressourcentypen und ihrer ARNs finden Sie unter Von Amazon Lex definierte Ressourcen in der Service Authorization Reference. Informationen darüber, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter Von Amazon Lex definierte Aktionen.

Schlüssel für Richtlinienbedingungen für Amazon Lex

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der Amazon Lex-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon Lex in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon Lex definierte Aktionen.

In der folgenden Tabelle sind die Amazon Lex Lex-Bedingungsschlüssel aufgeführt, die für Amazon Lex Lex-Ressourcen gelten. Sie können diese Schlüssel in `Condition` Elemente einer IAM-Berechtigungsrichtlinie aufnehmen.

ACLs in Amazon Lex

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon Lex

Unterstützt ABAC (Tags in Richtlinien)	Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Sie können Tags bestimmten Typen von Amazon Lex Lex-Ressourcen für die Autorisierung zuordnen. Um den Zugriff basierend auf Tags zu steuern, stellen Sie Tag-Informationen

im Bedingungselement einer Richtlinie unter Verwendung der Bedingungsschlüssel `lex:ResourceTag/${TagKey}`, `aws:RequestTag/${TagKey}` oder `aws:TagKeys` bereit.

Weitere Informationen zum Taggen von Amazon Lex Lex-Ressourcen finden Sie unterAmazon Lex Lex-Ressourcen kennzeichnen.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Verwenden Sie ein Tag, um auf eine Ressource zuzugreifen.

In der folgenden Tabelle sind die Aktionen und die entsprechenden Ressourcentypen für die tagbasierte Zugriffssteuerung aufgeführt. Jede Aktion wird basierend auf den Tags autorisiert, die dem entsprechenden Ressourcentyp zugeordnet sind.

Aktion	Ressourcentyp	Bedingungsschlüssel	Hinweise
CreateBotVersion	Bot	`lex:ResourceTag`	
DeleteBot	Bot	`lex:ResourceTag`	
DeleteBotAlias	alias	`lex:ResourceTag`	
DeleteBotChannelAssociation	channel	`lex:ResourceTag`	
DeleteBotVersion	Bot	`lex:ResourceTag`	
DeleteSession	Bot oder Alias	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn Alias auf `$LATEST` gesetzt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn sie mit anderen Aliasnamen verwendet werden.
DeleteUtterances	Bot	`lex:ResourceTag`	

Aktion	Ressourcentyp	Bedingungsschlüssel	Hinweise
GetBot	Bot oder Alias	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn `versionOrAlias` auf `$LATEST` oder numerische Version festgelegt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn sie mit Aliassen verwendet werden
GetBotAlias	alias	`lex:ResourceTag`	
GetBotChannelAssociation	Channel	`lex:ResourceTag`	
GetBotChannelAssociations	Channel	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn Alias auf „-“ gesetzt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn ein Bot-Alias angegeben wird
GetBotVersions	Bot	`lex:ResourceTag`	
GetExport	Bot	`lex:ResourceTag`	

Aktion	Ressourcentyp	Bedingungsschlüssel	Hinweise
GetSession	Bot oder Alias	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn Alias auf `$LATEST` gesetzt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn sie mit anderen Aliasnamen verwendet werden.
GetUtterancesView	Bot	`lex:ResourceTag`	
ListTagsForResource	Bot, Alias oder Channel	`lex:ResourceTag`	
PostContent	Bot oder Alias	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn Alias auf `$LATEST` gesetzt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn sie mit anderen Aliasnamen verwendet werden.

Aktion	Ressourcentyp	Bedingungsschlüssel	Hinweise
PostText	Bot oder Alias	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn Alias auf `$LATEST` gesetzt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn sie mit anderen Aliasnamen verwendet werden.
PutBot	Bot	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutBotAlias	alias	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutSession	Bot oder Alias	`lex:ResourceTag`	Verwendet Tags, die dem Bot zugeordnet sind, wenn Alias auf `$LATEST` gesetzt ist. Verwendet Tags, die dem angegebenen Alias zugeordnet sind, wenn sie mit anderen Aliasnamen verwendet werden.

Aktion	Ressourcentyp	Bedingungsschlüssel	Hinweise
StartImport	Bot	`lex:ResourceTag`	Basiert auf der Zugriffsrichtlinie für die `PutBot`-Operation. Tags und Berechtigungen, die für die `StartImport`-Operation spezifisch sind, werden ignoriert.
TagResource	Bot, Alias oder Channel	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
UntagResource	Bot, Alias oder Channel	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	

Temporäre Anmeldeinformationen mit Amazon Lex verwenden

Unterstützt temporäre Anmeldeinformationen	Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit IAM.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden

und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Sie erhalten temporäre Sicherheitsanmeldedaten, indem Sie AWS STS API-Operationen wie AssumeRoleoder GetFederationToken aufrufen.

Serviceübergreifende Prinzipalberechtigungen für Amazon Lex

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Amazon Lex

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann die Funktionalität von Amazon Lex beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon Lex Sie dazu anleitet.

Auswahl einer IAM-Rolle in Amazon Lex

Amazon Lex verwendet serviceverknüpfte Rollen, um Amazon Comprehend und Amazon Polly aufzurufen. Es verwendet Berechtigungen auf Ressourcenebene für Ihre Funktionen, um sie aufzurufen. AWS Lambda

Sie müssen eine IAM-Rolle angeben, um das Tagging von Konversationen zu aktivieren. Weitere Informationen finden Sie unter Erstellen einer IAM-Rolle und von Richtlinien für Konversationsprotokolle.

Servicebezogene Rollen für Amazon Lex

Unterstützt serviceverknüpfte Rollen	Ja

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Einzelheiten zum Erstellen oder Verwalten von serviceverknüpften Amazon Lex-Rollen finden Sie unterVerwendung von serviceverknüpfte Rollen für Amazon Lex.

Beispiele für identitätsbasierte Richtlinien für Amazon Lex

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Amazon Lex Lex-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu den von Amazon Lex definierten Aktionen und Ressourcentypen, einschließlich des Formats der ARNs für jeden Ressourcentyp, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Lex in der Service Authorization Reference.

Themen

- Bewährte Methoden für Richtlinien
- Verwenden der Amazon Lex Lex-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Alle Amazon Lex Bots löschen
- Erlauben Sie einem Benutzer, einen Bot zu Amazon Lex V2-APIs zu migrieren
- Verwenden Sie ein Tag, um auf eine Ressource zuzugreifen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Amazon Lex Lex-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder diese löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum

Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.

- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon Lex Lex-Konsole

Um auf die Amazon Lex Lex-Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon Lex Lex-Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

AWS adressiert viele gängige Anwendungsfälle durch die Bereitstellung eigenständiger IAM-Richtlinien, die von erstellt und verwaltet AWS werden. Diese Richtlinien werden als von AWS verwaltete Richtlinien bezeichnet. Von AWS verwaltete Richtlinien erleichtern Ihnen die Zuweisung der geeigneten Berechtigungen zu Benutzern, Gruppen und Rollen, so dass Sie die Richtlinien nicht selbst schreiben müssen. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien im IAM Benutzerhandbuch.

Die folgenden AWS verwalteten Richtlinien, die Sie Gruppen und Rollen in Ihrem Konto zuordnen können, sind spezifisch für Amazon Lex:

- AmazonLexReadOnly— Gewährt schreibgeschützten Zugriff auf Amazon Lex Lex-Ressourcen.
- AmazonLexRunBotsNur — Gewährt Zugriff auf die Ausführung von Amazon Lex Lex-Konversationsbots.
- AmazonLexFullAccess— Gewährt vollen Zugriff zum Erstellen, Lesen, Aktualisieren, Löschen und Ausführen aller Amazon Lex Lex-Ressourcen. Gewährt auch die Möglichkeit, Lambda-Funktionen, deren Name `AmazonLex` mit Amazon Lex Lex-Absichten beginnt, zuzuordnen.

> **Note**
>
> Sie können diese Berechtigungsrichtlinien überprüfen, indem Sie sich bei der IAM-Konsole anmelden und nach bestimmten Richtlinien suchen.

Die AmazonLexFullAccessRichtlinie gewährt dem Benutzer nicht die Erlaubnis, die `KendraSearchIntent` Absicht zu verwenden, um einen Amazon Kendra Kendra-Index abzufragen. Um einen Index abzufragen, müssen Sie der Richtlinie zusätzliche Berechtigungen hinzufügen. Die erforderlichen Berechtigungen finden Sie unter IAM-Richtlinie für Amazon Kendra Search.

Sie können auch Ihre eigenen benutzerdefinierten IAM-Richtlinien erstellen, um Berechtigungen für Amazon Lex API-Aktionen zuzulassen. Sie können diese benutzerdefinierten Richtlinien an die IAM-Rollen oder -Gruppen anhängen, für die diese Berechtigungen erforderlich sind.

Einzelheiten zu den von AWS verwalteten Richtlinien für Amazon Lex finden Sie unterAWSverwaltete Richtlinien für Amazon Lex.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder AWS .

```
{
    "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": "*"
        }
    ]
}
```

Alle Amazon Lex Bots löschen

Diese Beispielrichtlinie gewährt einem Benutzer in Ihrem AWS-Konto die Erlaubnis, jeden Bot in Ihrem Konto zu löschen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Erlauben Sie einem Benutzer, einen Bot zu Amazon Lex V2-APIs zu migrieren

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, mit der Migration eines Bots von Amazon Lex zu Amazon Lex V2-APIs zu beginnen und die Liste der Migrationen und deren Fortschritt zu sehen.

```
{
    "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",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "lex:DescribeCustomVocabulary",
                "lex:DescribeCustomVocabularyMetadata"
            ],
            "Resource": [
                "arn:aws:lex:<Region>:<123456789012>:bot/*",
                "arn:aws:lex:<Region>:<123456789012>:bot-alias/*/*"
            ]
        },
        {
            "Sid": "showBots",
            "Effect": "Allow",
            "Action": [
                "lex:CreateUploadUrl",
                "lex:ListBots"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "showMigrations",
            "Effect": "Allow",
            "Action": [
                "lex:GetMigration",
                "lex:GetMigrations"
            ],
            "Resource": "*"
        }
    ]
}
```

Verwenden Sie ein Tag, um auf eine Ressource zuzugreifen

Diese Beispielrichtlinie gewährt einem Benutzer oder einer Rolle in Ihrem AWS Konto die Erlaubnis, den `PostText` Vorgang mit jeder Ressource zu verwenden, die mit dem Schlüssel **`Department`** und dem Wert gekennzeichnet ist**`Support`**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lex:PostText",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "lex:ResourceTag/Department": "Support"
                }
            }
        }
    ]
}
```

AWSverwaltete Richtlinien für Amazon Lex

Eine von AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von AWS erstellt und verwaltet wird. Von AWS verwaltete Richtlinien stellen Berechtigungen für viele häufige

Anwendungsfälle bereit, damit Sie beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS-verwaltete Richtlinien möglicherweise nicht die geringsten Berechtigungen für Ihre spezifischen Anwendungsfälle gewähren, da sie für alle AWS-Kunden verfügbar sind. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Die Berechtigungen, die in den von AWS verwalteten Richtlinien definiert sind, können nicht geändert werden. Wenn AWS Berechtigungen aktualisiert, die in einer von AWS verwalteten Richtlinie definiert werden, wirkt sich das Update auf alle Prinzipalidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWSaktualisiert am wahrscheinlichsten eine von AWS verwaltete Richtlinie, wenn ein neuer AWS-Service gestartet wird oder neue API-Operationen für bestehende Services verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

Von AWS verwaltete Richtlinie:AmazonLexReadOnly

Sie können die `AmazonLexReadOnly`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt Leserechte, die es Benutzern ermöglichen, alle Aktionen im Amazon Lex- und Amazon Lex V2-Modellbauservice einzusehen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `lex`— Lesezugriff auf Amazon Lex- und Amazon Lex V2-Ressourcen im Model Building Service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:GetBot",
                "lex:GetBotAlias",
                "lex:GetBotAliases",
```

```
                "lex:GetBots",
                "lex:GetBotChannelAssociation",
                "lex:GetBotChannelAssociations",
                "lex:GetBotVersions",
                "lex:GetBuiltinIntent",
                "lex:GetBuiltinIntents",
                "lex:GetBuiltinSlotTypes",
                "lex:GetIntent",
                "lex:GetIntents",
                "lex:GetIntentVersions",
                "lex:GetSlotType",
                "lex:GetSlotTypes",
                "lex:GetSlotTypeVersions",
                "lex:GetUtterancesView",
                "lex:DescribeBot",
                "lex:DescribeBotAlias",
                "lex:DescribeBotChannel",
                "lex:DescribeBotLocale",
                "lex:DescribeBotVersion",
                "lex:DescribeExport",
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBots",
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotChannels",
                "lex:ListBotVersions",
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Von AWS verwaltete Richtlinie:AmazonLexRunBotsOnly

Sie können die `AmazonLexRunBotsOnly`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt Leserechte, die den Zugriff auf die Ausführung von Amazon Lex- und Amazon Lex V2-Konversations-Bots ermöglichen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `lex`— Schreibgeschützter Zugriff auf alle Aktionen in der Amazon Lex- und Amazon Lex V2-Runtime.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

Von AWS verwaltete Richtlinie:AmazonLexFullAccess

Sie können die `AmazonLexFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt Administratorberechtigungen, die es dem Benutzer ermöglichen, Amazon Lex- und Amazon Lex V2-Ressourcen zu erstellen, zu lesen, zu aktualisieren und zu löschen sowie Amazon Lex- und Amazon Lex V2-Konversationsbots auszuführen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `lex`— Ermöglicht Prinzipalen Lese- und Schreibzugriff auf alle Aktionen in den Modellbau- und Runtime-Services von Amazon Lex und Amazon Lex V2.
- `cloudwatch`— Ermöglicht es Schulleitern, Amazon anzusehenCloudWatchMetriken und Alarme.
- `iam`— Ermöglicht es Prinzipalen, dienstverknüpfte Rollen zu erstellen und zu löschen, Rollen weiterzugeben und Richtlinien an eine Rolle anzuhängen oder von ihnen zu trennen. Die Berechtigungen sind auf „lex.amazonaws.com“ für Amazon Lex-Operationen und auf „lexv2.amazonaws.com“ für Amazon Lex V2-Operationen beschränkt.
- `kendra`— Ermöglicht es Schulleitern, Amazon Kendra-Indizes aufzulisten.
- `kms`— Ermöglicht es Schulleitern, zu beschreibenAWS KMSSchlüssel und Aliase.
- `lambda`— Ermöglicht Schulleitern das AuflistenAWS LambdaFunktionen und verwalten Sie Berechtigungen, die an jede Lambda-Funktion angehängt sind.
- `polly`— Ermöglicht es Schulleitern, Amazon Polly-Stimmen zu beschreiben und Sprache zu synthetisieren.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
```

```
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Amazon Lex wird aktualisiert aufAWSverwaltete Richtlinien

Details zu Updates für anzeigenAWSverwaltete Richtlinien für Amazon Lex, seit dieser Service damit begonnen hat, diese Änderungen zu verfolgen. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf Amazon LexDokumentverlauf für Amazon LexSeite.

Änderung	Beschreibung	Datum
AmazonLexFullAccess – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf den Amazon Lex V2-Modellbauservice zu ermöglichen.	18. August 2021
AmazonLexReadOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf den Amazon Lex V2-Modellbauservice zu ermöglichen.	18. August 2021
AmazonLexRunBotsOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf Amazon Lex V2-Runtime-Service-Operationen zu ermöglichen.	18. August 2021
Amazon Lex hat begonnen, Änderungen zu verfolgen	Amazon Lex hat begonnen, Änderungen für seineAWSv erwaltete Richtlinien.	18. August 2021

Verwendung von serviceverknüpfte Rollen für Amazon Lex

Amazon Lex verwendetAWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist ein spezieller Typ von IAM-Rolle, der direkt mit Amazon Lex verknüpft ist. Serviceverknüpfte Rollen werden von Amazon Lex vordefiniert und schließen alle Berechtigungen ein, die der Service zum Aufrufen andererAWS -Services in Ihrem Namen erfordert.

Eine serviceverknüpfte Rolle macht die Einrichtung von Amazon Lex einfacher, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Amazon Lex definiert die Berechtigungen seiner mit dem Service verbundenen Rollen, und sofern nicht anders definiert,

kann nur Amazon Lex seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauensrichtlinie und die Berechtigungsrichtlinie, und diese Berechtigungsrichtlinie kann keiner anderen juristischen Stelle von IAM zugeordnet werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem die zugehörigen Ressourcen gelöscht wurden. Dies schützt Ihre Amazon-Lex-Ressourcen, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entfernen können.

Serviceverknüpfte Rollen für Amazon Lex

Amazon Lex verwendet zwei serviceverknüpfte Rollen:

- AWSServiceRoleForLexBots— Amazon Lex verwendet diese serviceverknüpfte Rolle, um Amazon Polly aufzurufen, um Sprachantworten für Ihren Bot zu synthetisieren, um Amazon Comprehend für die Stimmungsanalyse und optional Amazon Kendra für die Suche nach Indizes aufzurufen.
- AWSServiceRoleForLexChannels— Amazon Lex verwendet diese serviceverknüpfte Rolle, um bei der Verwaltung von Kanälen Text an Ihren Bot zu senden.

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine servicegebundene Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigungen im IAM-Benutzerhandbuch.

Erstellen einer serviceverknüpfte Rolle für Amazon Lex

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie einen Bot, einen Bot-Channel oder einen Amazon Kendra erstellenAWS Management Console, erstellt Amazon Lex die serviceverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie einen neuen Bot, eine neue Kanalzuweisung erstellen oder Amazon Kendra suchen, erstellt Amazon Lex die serviceverknüpfte Rolle erneut für Sie.

Sie können auch die verwendenAWS CLI, um eine serviceverknüpfte Rolle mit dem AWSServiceRoleForLexBotsAnwendungsfall zu erstellen. ImAWS CLI Erstellen einer serviceverknüpfte Rolle mit dem Amazon Lex-Service Namen`lex.amazonaws.com`. Weitere Informationen finden Sie unter Schritt 1: Erstellen einer serviceverknüpften Rolle (AWS CLI). Wenn Sie diese servicegebundene Rolle löschen, können Sie mit demselben Verfahren die Rolle erneut erstellen.

Bearbeiten einer serviceverknüpfte Rolle für Amazon Lex

Amazon Lex erlaubt Ihnen nicht, die serviceverknüpfte Rollen Amazon Lex zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Deleting eine serviceverknüpfte Rolle für Amazon Lex

Wenn Sie eine Funktion oder einen Service, die bzw. der eine servicegebundene Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch die Ressourcen für Ihre serviceverknüpften Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

> **Note**
>
> Wenn der Amazon Lex-Service die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt der Löschvorgang möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

So löschen Sie Amazon Lex-Ressourcen, die von servicegebundenen Rollen verwendet werden:

1. Löschen Sie alle von Ihnen verwendeten Bot-Kanäle.
2. Löschen Sie alle Bots in Ihrem Konto.

So löschen Sie die servicegebundene Rolle mit IAM

Verwenden Sie die IAM-KonsoleAWS CLI, um dieAWS serviceverknüpfte Rollen Amazon Lex zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Unterstützte Regionen für Amazon Lex Service-Linked Rollen

Amazon Lex unterstützt die Verwendung von serviceverknüpfte Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter Amazon Lex-Endpunkte und -Kontingente.

Problembehandlung bei Identität und Zugriff auf Amazon Lex

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon Lex und IAM auftreten können.

Themen

Ich bin nicht berechtigt, eine Aktion in Amazon Lex durchzuführen

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `lex:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `lex:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Amazon Lex übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder eine dienstbezogene Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon Lex auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon Lex-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Amazon Lex diese Funktionen unterstützt, finden Sie unterSo funktioniert Amazon Lex mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie im IAM-Benutzerhandbuch unter Kontenübergreifender Ressourcenzugriff in IAM.

Überwachung in Amazon Lex

Die Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer Amazon Lex-Chatbots aufrechtzuerhalten. In diesem Thema wird beschrieben, wie Amazon CloudWatch Logs verwendet und AWS CloudTrail Amazon Lex überwacht wird, und es werden die Amazon Lex-Laufzeit- und Kanalzuordnungsmetriken beschrieben.

Themen

Überwachung von Amazon Lex mit Amazon CloudWatch

Verwenden Sie Amazon, um den Zustand Ihrer Amazon Lex Lex-Bots zu verfolgen CloudWatch. Mit CloudWatch können Sie Metriken für einzelne Amazon Lex Lex-Operationen oder für globale Amazon Lex Lex-Operationen für Ihr Konto abrufen. Sie können auch CloudWatch Alarme einrichten, sodass Sie benachrichtigt werden, wenn eine oder mehrere Metriken einen von Ihnen definierten Schwellenwert überschreiten. Sie können beispielsweise die Anzahl der über einen bestimmten Zeitraum an einen Bot gerichteten Anforderungen überwachen, die Latenz erfolgreicher Anforderungen anzeigen oder einen Alarm auslösen, wenn die Fehlerzahl einen Grenzwert überschreitet.

CloudWatch Metriken für Amazon Lex

Um Metriken für Ihre Amazon Lex Lex-Operationen zu erhalten, müssen Sie die folgenden Informationen angeben:

- Die Metrikdimension. Eine Dimension ist ein Satz von Name-Wert-Paaren, die Sie verwenden, um eine Metrik zu identifizieren. Amazon Lex hat drei Dimensionen:
  - `BotAlias, BotName, Operation`
  - `BotAlias, BotName, InputMode, Operation`
  - `BotName, BotVersion, InputMode, Operation`
- Den Namen der Kennzahl wie `MissedUtteranceCount` oder `RuntimeRequestCount`.

Sie können Metriken für Amazon Lex mit der AWS Management ConsoleAWS CLI, der oder der CloudWatch API abrufen. Sie können die CloudWatch API über eines der Amazon AWS Software

Development Kits (SDKs) oder die CloudWatch API-Tools verwenden. Die Amazon Lex Lex-Konsole zeigt Diagramme an, die auf den Rohdaten der CloudWatch API basieren.

Sie müssen über die entsprechenden CloudWatch Berechtigungen verfügen, um Amazon Lex überwachen zu können CloudWatch . Weitere Informationen finden Sie unter Authentifizierung und Zugriffskontrolle für Amazon CloudWatch im CloudWatch Amazon-Benutzerhandbuch.

Amazon Lex Lex-Metriken anzeigen

Zeigen Sie Amazon Lex Lex-Metriken mit der Amazon Lex Lex-Konsole oder der CloudWatch Konsole an.

So zeigen Sie Metriken an (Amazon Lex Lex-Konsole)

1. Melden Sie sich bei der Amazon Lex Lex-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/lex/.
2. Wählen Sie in der Liste der Bots denjenigen Bot, dessen Kennzahlen angezeigt werden sollen.
3. Wählen Sie Monitoring. Kennzahlen werden in Diagrammen dargestellt.

Um Metriken anzusehen (CloudWatch Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Metrics, All Metrics und dann AWS/Lex.
3. Wählen Sie die Dimension, den Namen einer Metrik und schließlich Add to graph (Dem Diagramm hinzufügen) aus.
4. Wählen Sie einen Wert für den Datumsbereich aus. Die Anzahl der Kennzahlen für den ausgewählten Zeitraum wird im Diagramm angezeigt.

Erstellen eines Alarms

Ein CloudWatch Alarm überwacht eine einzelne Metrik über einen bestimmten Zeitraum und führt eine oder mehrere Aktionen aus: das Senden einer Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS) -Thema oder an eine Auto Scaling Scaling-Richtlinie. Die Aktion oder Aktionen basieren auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über eine von Ihnen angegebene Anzahl von Zeiträumen. CloudWatchkann Ihnen auch eine Amazon SNS SNS-Nachricht senden, wenn sich der Status des Alarms ändert.

CloudWatch Alarme rufen nur dann Aktionen hervor, wenn sich der Status ändert und für den von Ihnen angegebenen Zeitraum andauert.

So legen Sie einen Alarm fest

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Alarms und dann Create Alarm.
3. Wählen Sie AWS/Lex Metrics und dann eine Kennzahl.
4. Wählen Sie für Time Range den zu überwachenden Zeitbereich und dann Next.
5. Geben Sie Name (Name) und Description (Beschreibung) ein.
6. Wählen Sie für Whenever den Wert >= und geben Sie einen Maximalwert ein.
7. Wenn Sie eine E-Mail senden CloudWatch möchten, wenn der Alarmstatus erreicht ist, wählen Sie im Bereich Aktionen für Immer dieser Alarm die Option Status ist ALARM. Wählen Sie für Send notification to (Benachrichtigung senden an) eine Mailingliste oder wählen Sie New List (Neue Liste) und erstellen Sie eine neue Mailingliste.
8. Nutzen Sie die Alarmvorschau im Bereich Alarm Preview. Wenn Sie mit dem Alarm zufrieden sind, wählen Sie Create Alarm.

CloudWatchMetriken für Amazon Lex Runtime

In der folgenden Tabelle werden die Amazon Lex-Laufzeitmetriken beschrieben.

Metrik	Beschreibung
`KendraIndexAccessError`	Die Häufigkeit, mit der Amazon Lex nicht auf Ihren Amazon Kendra Kendra-Index zugreifen konnte. Gültige Dimension für die Operation `PostContent` mit `InputMode` als `Text` oder `Speech`: • `BotName, BotAlias, Operation, InputMode` Gültige Dimension für die Operation `PostText`: • `BotName, BotAlias, Operation`

Metrik	Beschreibung
	Einheit: Anzahl
KendraLatency	Die Zeit, die Amazon Kendra benötigt, um auf eine Anfrage von zu antworten. AMAZON.KendraSearchIntent Gültige Dimensionen für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Gültige Dimensionen für die Operation PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Einheit: Millisekunden
KendraSuccess	Die Anzahl der erfolgreichen Anfragen von Ihrem Amazon Kendra Kendra-Index. AMAZON.KendraSearchIntent Gültige Dimensionen für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Gültige Dimensionen für die Operation PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Einheit: Anzahl

Metrik	Beschreibung
KendraSystemErrors	Die Häufigkeit, mit der Amazon Lex den Amazon Kendra Kendra-Index nicht abfragen konnte. Gültige Dimension für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotAlias, Operation, InputMode Gültige Dimension für die Operation PostText: • BotName, BotAlias, Operation Einheit: Anzahl
KendraThrottledEvents	Die Häufigkeit, mit der Amazon Kendra Anfragen von gedrosselt hat. AMAZON.KendraSearchIntent Gültige Dimension für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotAlias, Operation, InputMode Gültige Dimension für die Operation PostText: • BotName, BotAlias, Operation Einheit: Anzahl

Metrik	Beschreibung
MissedUtteranceCount	Die Anzahl der Äußerungen, die im angegebenen Zeitraum nicht erkannt wurden. Gültige Dimensionen für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Gültige Dimensionen für die Operation PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation
RuntimeConcurrency	Die Anzahl gleichzeitiger Verbindungen im angegebenen Zeitraum. RuntimeConcurrency wird als gemeldet. StatisticSet Gültige Dimensionen für die Operation PostContent mit InputMode als Text oder Speech: • Betrieb BotName, BotVersion, InputMode • Betrieb, BotName, BotAlias, InputMode Gültige Abmessungen für andere Operationen: • Betrieb BotName, BotVersion • Betrieb BotName, BotAlias Einheit: Anzahl

Metrik	Beschreibung
RuntimeInvalidLambdaResponses	Die Anzahl der ungültigen AWS Lambda (Lambda-) Antworten im angegebenen Zeitraum. Gültige Dimension für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotAlias, Operation, InputMode Gültige Dimension für die Operation PostText: • BotName, BotAlias, Operation
RuntimeLambdaErrors	Die Anzahl der Lambda-Laufzeitfehler im angegebenen Zeitraum. Gültige Dimension für die Operation PostContent mit Text als Speech oder InputMode : • BotName, BotAlias, Operation, InputMode Gültige Dimension für die Operation PostText: • BotName, BotAlias, Operation
RuntimePollyErrors	Die Anzahl der ungültigen Amazon Polly Polly-Antworten im angegebenen Zeitraum. Gültige Dimension für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotAlias, Operation, InputMode Gültige Dimension für die Operation PostText: • BotName, BotAlias, Operation

Metrik	Beschreibung
RuntimeRequestCount	Die Anzahl der Laufzeitanforderungen im angegebenen Zeitraum. Gültige Dimensionen für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Gültige Dimensionen für die Operation PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Einheit: Anzahl
RuntimeSucessfulRequestLatency **Important** Diese Metrik ist RuntimeSuccessfulRequestLatency und nichtRuntimeSuccessfulRequestLatency .	Die Latenz für erfolgreiche Anforderungen zwischen dem Zeitpunkt der Anforderung und dem der Antwortrückgabe. Gültige Dimensionen für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Gültige Dimensionen für die Operation PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Einheit: Millisekunden

Metrik	Beschreibung
`RuntimeSystemErrors`	Die Anzahl der Systemfehler im angegebenen Zeitraum. Der Antwortcodebereich für einen Systemfehler lautet 500 bis 599. Gültige Dimension für die Operation `PostContent` mit `InputMode` als `Text` oder `Speech`: • `BotName, BotAlias, Operation, InputMode` Gültige Dimension für die Operation `PostText`: • `BotName, BotAlias, Operation` Einheit: Anzahl
`RuntimeThrottledEvents`	Die Anzahl der gedrosselten Anforderungen. Amazon Lex drosselt eine Anfrage, wenn mehr Anfragen eingehen als das für Ihr Konto festgelegte Limit an Transaktionen pro Sekunde. Wenn der Grenzwert für Ihr Konto häufig überschritten wird, können Sie eine Erweiterung des Limits beantragen. Informationen zum Anfordern einer Erweiterung finden Sie unter AWS Service Limits. Gültige Dimension für die Operation `PostContent` mit `InputMode` als `Text` oder `Speech`: • BotName, Betrieb BotAlias, InputMode Gültige Dimension für die Operation `PostText`: • `BotName, BotAlias, Operation` Einheit: Anzahl

Metrik	Beschreibung
RuntimeUserErrors	Die Anzahl der Benutzerfehler im angegebenen Zeitraum. Der Antwortcode-Bereich für einen Benutzerfehler ist 400 bis 499. Gültige Dimension für die Operation PostContent mit InputMode als Text oder Speech: • BotName, BotAlias, Operation, InputMode Gültige Dimension für die Operation PostText: • BotName, BotAlias, Operation Einheit: Anzahl

Amazon Lex Lex-Laufzeitmetriken verwenden den AWS/Lex Namespace und stellen Metriken in den folgenden Dimensionen bereit. Sie können Metriken in der CloudWatch Konsole nach Dimensionen gruppieren:

Dimension	Beschreibung
BotName, BotAlias, Operation, InputMode	Groups metrics by the bot's alias, the bot's name, the operation (PostContent), and by whether the input was text or speech.
BotName, BotVersion, Operation, InputMode	Gruppiert Metriken nach Bot-Name, Bot-Version, Operation (PostContent) sowie Eingabetyp (Text oder Sprache).
BotName, BotVersion, Operation	Gruppiert Metriken nach Bot-Name, Bot-Version und Operation (PostText).
BotName, BotAlias, Operation	Gruppiert Metriken nach Bot-Name, Bot-Alias und Operation (PostText).

CloudWatch Metriken für Amazon Lex Channel Associations

Eine Kanalzuweisung ist die Verbindung zwischen Amazon Lex und einem Messaging-Kanal wie Facebook. In der folgenden Tabelle werden die Kennzahlen zur Amazon Lex Lex-Channel-Association beschrieben.

Metrik	Beschreibung
`BotChannelAuthErrors`	Die Anzahl der vom Messaging-Kanal im angegebenen Zeitraum zurückgegebenen Authentifizierungsfehler. Ein Authentifizierungsfehler gibt an, dass das geheime Token während der Kanalerstellung ungültig oder abgelaufen ist.
`BotChannelConfigurationErrors`	Die Anzahl der Konfigurationsfehler im angegebenen Zeitraum. Ein Konfigurationsfehler gibt an, dass mindestens ein Konfigurationseintrag für den Kanal ungültig ist.
`BotChannelInboundThrottledEvents`	Gibt an, wie oft Nachrichten, die über den Messaging-Kanal gesendet wurden, im angegebenen Zeitraum von Amazon Lex gedrosselt wurden.
`BotChannelOutboundThrottledEvents`	Gibt an, wie oft ausgehende Ereignisse von Amazon Lex an den Messaging-Kanal im angegebenen Zeitraum gedrosselt wurden.
`BotChannelRequestCount`	Die Anzahl der Anforderungen über einen Kanal im angegebenen Zeitraum.
`BotChannelResponseCardErrors`	Gibt an, wie oft Amazon Lex im angegebenen Zeitraum keine Antwortkarten versenden konnte.
`BotChannelSystemErrors`	Die Anzahl der internen Fehler, die in Amazon Lex für einen Kanal im angegebenen Zeitraum aufgetreten sind.

Amazon Lex Lex-Metriken zur Kanalzuweisung verwenden den `AWS/Lex` Namespace und stellen Metriken für die folgende Dimension bereit. Sie können Metriken in der CloudWatch Konsole nach Dimensionen gruppieren:

Dimension	Beschreibung
`BotAlias, BotChannelName, BotName, Source`	Gruppiert Metriken nach Bot-Alias, Kanalname, Bot-Name und Quelle des Datenverkehrs.

CloudWatch Metriken für Konversationsprotokolle

Amazon Lex verwendet die folgenden Metriken für die Protokollierung von Konversationen:

Metrik	Beschreibung
`ConversationLogsAudioDeliverySuccess`	Die Anzahl der Audioprotokolle, die im angegebenen Zeitraum erfolgreich an den S3-Bucket übermittelt wurden. Einheiten: Anzahl
`ConversationLogsAudioDeliveryFailure`	Die Anzahl der Audioprotokolle, die im angegebenen Zeitraum nicht an den S3-Bucket übermittelt wurden. Ein Zustellungsfehler weist auf einen Fehler mit den für Konversationsprotokolle konfigurierten Ressourcen hin. Zu den Fehlern können unzureichende IAM-Berechtigungen, ein unzugänglicher AWS KMS Schlüssel oder ein unzugänglicher S3-Bucket gehören. Einheiten: Anzahl
`ConversationLogsTextDeliverySuccess`	Die Anzahl der Textprotokolle, die im angegebenen Zeitraum erfolgreich an CloudWatch Logs übermittelt wurden. Einheiten: Anzahl
`ConversationLogsTextDeliveryFailure`	Die Anzahl der Textprotokolle, die im angegebenen Zeitraum nicht an CloudWatc

Metrik	Beschreibung
	h Logs übermittelt werden konnten. Ein Zustellungsfehler weist auf einen Fehler mit den für Konversationsprotokolle konfiguri erten Ressourcen hin. Zu den Fehlern können unzureichende IAM-Berechtigungen, ein AWS KMS Schlüssel, auf den nicht zugegriffen werden kann, oder eine CloudWatch Protokoll gruppe, auf die nicht zugegriffen werden kann, gehören. Einheiten: Anzahl

Die Messwerte für das Konversationsprotokoll von Amazon Lex verwenden den `AWS/Lex` Namespace und stellen Metriken für die folgenden Dimensionen bereit. Sie können Metriken in der CloudWatch Konsole nach Dimension gruppieren.

Dimension	Beschreibung
`BotAlias`	Gruppieren der Metriken nach dem Alias des Bots
`BotName`	Gruppieren der Metriken nach dem Namen des Bots
`BotVersion`	Gruppieren der Metriken nach der Bot-Version

Überwachung von Amazon Lex API-Aufrufen mit AWS CloudTrail Protokollen

Amazon Lex ist in einen Service integriertAWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Amazon Lex ausgeführt wurden. CloudTrail erfasst eine Teilmenge von API-Aufrufen für Amazon Lex als Ereignisse, einschließlich Aufrufe von der Amazon Lex Lex-Konsole und von Codeaufrufen an die Amazon Lex Lex-APIs. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail

Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon Lex. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Amazon Lex gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen darüber CloudTrail, einschließlich der Konfiguration und Aktivierung, finden Sie im AWS CloudTrailBenutzerhandbuch.

Informationen zu Amazon Lex in CloudTrail

CloudTrail ist für Ihr AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn unterstützte Ereignisaktivitäten in Amazon Lex auftreten, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen im CloudTrail Ereignisverlauf in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Amazon Lex, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon Simple Storage Service (Amazon S3) -Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien für den von Ihnen angegebenen S3 Bucket bereit. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Amazon Lex unterstützt die Protokollierung der folgenden Vorgänge als Ereignisse in CloudTrail Protokolldateien:

- CreateBotVersion
- CreateIntentVersion

- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots
- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Mit diesen Informationen können Sie Folgendes bestimmen:

- Ob die Anfrage mit Root- oder -Benutzeranmeldeinformationen ausgeführt wurde.

- Ob die Anfrage mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen föderierten Benutzer ausgeführt wurde
- Ob die Anforderung von einem anderen AWS-Service getätigt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Informationen zu den Amazon Lex-Aktionen, die in CloudTrail Protokollen protokolliert werden, finden Sie unter Amazon Lex Model Building Service. Beispielsweise generieren Aufrufe der DeleteBot Operationen PutBotGetBot, und Einträge im CloudTrail Protokoll. Die unter Amazon Lex Runtime Service dokumentierten Aktionen PostContent und PostText werden nicht protokolliert.

Beispiel: Amazon Lex Lex-Protokolldateieinträge

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Der folgende CloudTrail Beispielprotokolleintrag zeigt das Ergebnis eines Aufrufs der `PutBot` Operation.

```
   {
               "eventVersion": "1.05",
               "userIdentity": {
                   "type": "AssumedRole | FederatedUser | IAMUser | Root | SAMLUser |
WebIdentityUser",
                   "principalId": "principal ID",
                   "arn": "ARN",
                   "accountId": "account ID",
                   "accessKeyId": "access key ID",
                   "userName": "user name"
               },
               "eventTime": "timestamp",
               "eventSource": "lex.amazonaws.com",
               "eventName": "PutBot",
               "awsRegion": "region",
               "sourceIPAddress": "source IP address",
               "userAgent": "user agent",
               "requestParameters": {
```

```
                    "name": "CloudTrailBot",
                    "intents": [
                        {
                            "intentVersion": "11",
                            "intentName": "TestCloudTrail"
                        }
                    ],
                    "voiceId": "Salli",
                    "childDirected": false,
                    "locale": "en-US",
                    "idleSessionTTLInSeconds": 500,
                    "processBehavior": "BUILD",
                    "description": "CloudTrail test bot",
                    "clarificationPrompt": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "I didn't understand you. What would you
like to do?"
                            }
                        ],
                        "maxAttempts": 2
                    },
                    "abortStatement": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "Sorry. I'm not able to assist at this
time."
                            }
                        ]
                    }
                },
                "responseElements": {
                    "voiceId": "Salli",
                    "locale": "en-US",
                    "childDirected": false,
                    "abortStatement": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "Sorry. I'm not able to assist at this
time."
                            }
```

```
                    ]
                },
                "status": "BUILDING",
                "createdDate": "timestamp",
                "lastUpdatedDate": "timestamp",
                "idleSessionTTLInSeconds": 500,
                "intents": [
                    {
                        "intentVersion": "11",
                        "intentName": "TestCloudTrail"
                    }
                ],
                "clarificationPrompt": {
                    "messages": [
                        {
                            "contentType": "PlainText",
                            "content": "I didn't understand you. What would you
like to do?"
                        }
                    ],
                    "maxAttempts": 2
                },
                "version": "$LATEST",
                "description": "CloudTrail test bot",
                "checksum": "checksum",
                "name": "CloudTrailBot"
            },
            "requestID": "request ID",
            "eventID": "event ID",
            "eventType": "AwsApiCall",
            "recipientAccountId": "account ID"
        }
    }
```

Compliance-Validierung für Amazon Lex

Externe Prüfer bewerten im Rahmen verschiedener AWS -Compliance-Programme die Sicherheit und Compliance von Amazon Lex. Amazon Lex ist ein HIPAA-berechtigter Service. Er ist PCI-, SOC- und ISO-konform. Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten in AWS Artifact.

Ihre Compliance-Verantwortung bei der Verwendung von Amazon Lex hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihrer Organisation und den geltenden Gesetzen und Vorschriften ab. Wenn Ihre Nutzung von Amazon Lex der Einhaltung von Standards wie PCI unterliegt, AWS stellt die folgenden Ressourcen zur Unterstützung bereit:

- Schnellstartanleitungen für Sicherheit und Compliance – Bereitstellungsleitfäden, in denen Überlegungen zur Architektur erörtert und Schritte für die Bereitstellung von sicherheits- und Compliance-orientierten Basisumgebungen in dargelegt werden AWS
- Whitepaper zur Erstellung einer Architektur mit HIPAA-konformer Sicherheit und Compliance: Dieses Whitepaper beschreibt, wie Unternehmen mithilfe von AWS HIPAA-konforme Anwendungen erstellen.
- AWS Compliance-Ressourcen – Eine Sammlung von Arbeitsmappen und Leitfäden, die für Ihre Branche und Ihren Standort möglicherweise relevant sind
- AWS Config – Ein Service, der bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen
- AWS Security Hub – Ein umfassender Überblick über Ihren Sicherheitsstatus in AWS , der Ihnen hilft, Ihre Compliance mit den Sicherheitsstandards und bewährten Methoden der Branche zu überprüfen

Eine Liste der - AWS Services, die für bestimmte Compliance-Programme in Frage kommen, finden Sie unter AWS-Services, die nach Compliance-Programm in Frage kommen. Allgemeine Informationen finden Sie unter AWS -Compliance-Programme.

Ausfallsicherheit in Amazon Lex

Die AWS globale -Infrastruktur ist um - AWS Regionen und Availability Zones herum aufgebaut. - AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit niedriger Latenz, hohem Durchsatz und hoher Redundanz verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Neben der AWS globalen -Infrastruktur stellt Amazon Lex verschiedene Funktionen bereit, um Ihren Anforderungen an Ausfallsicherheit und Datensicherung gerecht zu werden.

Infrastruktursicherheit in Amazon Lex

Als verwalteter Service ist Amazon Lex durch die AWS globalen Verfahren zur Gewährleistung der Netzwerksicherheit von geschützt, die im Whitepaper Amazon Web Services: Übersicht über die Sicherheitsprozesse beschrieben sind.

Sie verwenden durch veröffentlichte AWS API-Aufrufe, um über das Netzwerk auf Amazon Lex zuzugreifen. Clients müssen TLS (Transport Layer Security) 1.0 unterstützen. Wir empfehlen TLS 1.2 oder höher. Clients müssen außerdem Cipher Suites mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systeme, z. B. Java 7 und höher, unterstützen diese Modi. Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Sie können diese API-Operationen von jedem Netzwerkstandort aus aufrufen, aber Amazon Lex unterstützt Zugriffsrichtlinien auf Ressourcenebene, die Einschränkungen auf der Grundlage der Quell-IP-Adresse enthalten können. Sie können auch Amazon Lex-Richtlinien verwenden, um den Zugriff von bestimmten Amazon Virtual Private Cloud (Amazon VPC)-Endpunkten oder bestimmten VPCs aus zu steuern. Dadurch wird der Netzwerkzugriff auf eine bestimmte Amazon Lex-Ressource effektiv nur von der spezifischen VPC innerhalb des AWS Netzwerks isoliert.

Richtlinien und Kontingente in Amazon Lex

Die folgenden Abschnitte enthalten Richtlinien und Kontingente für die Verwendung von Amazon Lex.

Themen

Unterstützte Regionen

Eine Liste derAWS Regionen, in denen Amazon Lex verfügbar ist, finden Sie unter Regionen und Endpunkte von AWS in der allgemeinen Amazon-Web-Services-Referenz.

Allgemeine Richtlinien

In diesem Abschnitt werden allgemeine Richtlinien für die Verwendung von Amazon Lex beschrieben.

- Signierungsanfragen — Alle Amazon Lex-Modellerstellungs- und Runtime-API-Operationen in der VersionAPI-Referenz verwenden die Signatur V4 zur Authentifizierung von Anfragen. Weitere Informationen zur Authentifizierung von Anfragen finden Sie unter Signature Version 4-Signaturprozess im Allgemeine Amazon Web Services-Referenz.

  Denn Amazon Lex verwendet die unsignierte Payload-OptionPostContent, die in der Amazon Simple Storage Service (S3) API-Referenz unter Signaturberechnungen für den Autorisierungsheader: Transfering Payload in a Single Chunk (AWS Signature Version 4) beschrieben ist.

  Schließen Sie den Hash der Nutzlast nicht in die kanonische Anforderung ein, wenn Sie die nicht signierte Nutzlast-Option verwenden. Verwenden Sie stattdessen die Literalzeichenfolge "UNSIGNED-PAYLOAD" als Hash der Nutzlast. Fügen Sie zudem einen Header namens `x-amz-content-sha256` und den Wert `UNSIGNED-PAYLOAD` in die `PostContent`-Anforderung ein.

- Beachten Sie Folgendes darüber, wie Amazon Lex Slot-Werte aus Benutzeräußerungen erfasst:

  Amazon Lex verwendet die Aufzählungswerte, die Sie in einer Slot-Typdefinition angeben, um seine Machine-Learning-Modelle zu trainieren. Angenommen, Sie definieren eine Absicht namens `GetPredictionIntent` mit der folgenden Beispieläußerung:

  ```
  "Tell me the prediction for {Sign}"
  ```

  Dabei ist `{Sign}` ein Slot des benutzerdefinierten Typs `ZodiacSign`. Er hat 12 Aufzählungswerte, `Aries` bis `Pisces`. Aus der Benutzeraussage „Sag mir die Vorhersage für...“ Amazon Lex versteht, dass das Folgende ein Sternzeichen ist.

  Wenn das`valueSelectionStrategy` Feld so eingestellt ist, dass derPutSlotType VorgangORIGINAL_VALUE verwendet wird, oder wenn in der Konsole Werte erweitern ausgewählt ist und der Benutzer „Sag mir die Vorhersage für die Erde“ sagt, leitet Amazon Lex ab, dass „Erde“ ein ist,`ZodiacSign` und übergibt sie an Ihre Client-Anwendung oder Lambda-Funktionen. Stellen Sie sicher, dass die Slot-Werte gültig sind, ehe Sie diese in einer Erfüllungsaktivität verwenden.

  Wenn das Feld `valueSelectionStrategy` mithilfe der Operation PutSlotType auf `TOP_RESOLUTION` gesetzt wird oder wenn Restrict to slot values and synonyms in der Konsole ausgewählt wird, werden die zurückgegebenen Werte auf die Werte beschränkt, die Sie als Slot-Typ angegeben haben. Wenn der Benutzer beispielsweise sagen würde, „Gib mir die Vorhersage für Erde“, würde der Wert nicht erkannt werden, weil er nicht zu den Werten gehört, die als Slot-Typ definiert sind. Wenn Sie Synonyme für Slot-Werte definieren, werden sie ebenfalls als Slot-Wert erkannt, jedoch wird der Slot-Wert anstelle des Synonyms zurückgegeben.

  Wenn Amazon Lex eine Lambda-Funktion aufruft oder das Ergebnis einer Sprachinteraktion mit Ihrer Client-Anwendung zurückgibt, kann die Groß- und Kleinschreibung der Slot-Werte nicht garantiert werden. Wenn Sie beispielsweise Werte für den in Amazon.Movie integrierten Slot-Typ

ermitteln möchten und ein Benutzer „Vom Winde verweht“ sagt oder eingibt, gibt Amazon Lex möglicherweise „Vom Winde verweht“, „Vom Winde verweht“ oder „Vom Winde verweht“ zurück. Bei Textinteraktionen entspricht die Groß-/Kleinschreibung der Slot-Werte dem eingegebenen Text oder dem Slot-Wert, abhängig von dem Wert im Feld `valueResolutionStrategy`.

- Verwenden Sie bei der Definition von Slot-Werten, die Akronyme enthalten, die folgenden Muster:
  - Durch Punkte getrennte Großbuchstaben (D.V.D.)
  - Durch Leerzeichen getrennte Großbuchstaben (D V D)

- Amazon Lex unterstützt den integrierten Steckplatztyp AMAZON.LITERAL, den das Alexa Skills Kit unterstützt, nicht. Amazon Lex unterstützt jedoch die Erstellung benutzerdefinierter Slot-Typen, mit denen Sie diese Funktionalität implementieren können. Wie es oben bereits erwähnt ist, können Sie Werte außerhalb der benutzerdefinierten Slot-Typ-Definition erfassen. Sie können weitere, verschiedene Aufzählungswerte hinzufügen, um die Genauigkeit der automatischen Spracherkennung (ASR) und das Verstehen der natürlichen Sprache (NLU) zu verstärken.

- Die integrierten Slot-Typen AMAZON.DATE und AMAZON.TIME erfassen sowohl absolute als auch relative Datums- und Zeitangaben. Relative Daten und Uhrzeiten werden in der Region festgelegt, in der Amazon Lex die Anfrage bearbeitet.

  Wenn der Benutzer beim`AMAZON.TIME` integrierten Slot-Typ nicht angibt, dass eine Uhrzeit vor oder nach Mittag liegt, ist die Uhrzeit mehrdeutig und Amazon Lex fordert den Benutzer erneut dazu auf. Wir empfehlen Aufforderungen, in denen nach der absoluten Zeit gefragt wird. Verwenden Sie beispielsweise eine Aufforderung wie "Wann möchten Sie Ihre Pizza geliefert bekommen? Sie können 6 Uhr morgen oder 6 Uhr abends angeben."

- Wenn Sie in Ihrem Bot verwirrende Trainingsdaten bereitstellen, verringert sich die Fähigkeit von Amazon Lex, Benutzereingaben zu verstehen. Berücksichtigen Sie die folgenden Beispiele:

Angenommen, Sie haben zwei Absichten (`OrderPizza` und `OrderDrink`) in Ihrem Bot und beide wurden mit der Äußerung"Ich möchte bestellen" konfiguriert. Diese Äußerung entspricht keiner bestimmten Absicht, aus der Amazon Lex lernen kann, während das Sprachmodell für den Bot zum Zeitpunkt der Erstellung erstellt wird. Wenn ein Benutzer diese Äußerung zur Laufzeit eingibt, kann Amazon Lex daher keine Absicht mit einem hohen Maß an Sicherheit auswählen.

Erwägen Sie ein weiteres Beispiel, in dem Sie eine benutzerdefinierte Absicht für eine Bestätigung vom Benutzer definieren (beispielsweise `MyCustomConfirmationIntent`), und konfigurieren Sie die Absicht mit den Äußerungen "Ja" und "Nein". Beachten Sie, dass Amazon Lex auch über ein Sprachmodell verfügt, um Benutzerbestätigungen zu verstehen. Dies kann widersprechende Situation schaffen. Wenn der Benutzer mit "Ja" antwortet, bedeutet dies, dass es eine Bestätigung für die laufende Absicht ist, oder dass der Benutzer die benutzerdefinierte Absicht anfordert, die Sie erstellt haben?

Im Allgemeinen sollten die Beispieläußerungen, die Sie bereitstellen, zu einer bestimmten Absicht gehören und, optional, zu bestimmten Slot-Werten.

- Die Laufzeit-API-Operationen PostContent und PostText nehmen eine Benutzer-ID als erforderlichen Parameter. Developer können diese auf jeden Wert stellen, der den in der API beschriebenen Einschränkungen übereinstimmt. Wir empfehlen, dass Sie nicht mit diesem Parameter vertrauliche Informationen wie Benutzerkonten, E-Mails oder Sozialversicherungsnummern senden. Diese ID wird in erster Linie dafür benutzt, eindeutig Konversation mit einem Bot zu identifizieren (Es kann mehrere Benutzer geben, die Pizza bestellen).

- Wenn Ihre Client-Anwendung Amazon Cognito für die Authentifizierung verwendet, können Sie die Amazon Cognito Cognito-Benutzer-ID als Amazon Lex-Benutzer-ID verwenden. Beachten Sie, dass jede für Ihren Bot konfigurierte Lambda-Funktion über einen eigenen Authentifizierungsmechanismus verfügen muss, um den Benutzer zu identifizieren, in dessen Namen Amazon Lex die Lambda-Funktion aufruft.

- Wir empfehlen, dass Sie eine Absicht definieren, die eine Absicht des Benutzers, die Konversation zu beenden, erfasst. Sie können beispielsweise eine Absicht (`NothingIntent`) mit Beispieläußerungen („Ich will nichts“, „exit“, „bye bye“) definieren, ohne Steckplätze und ohne Lambda-Funktion, die als Code-Hook konfiguriert ist. Auf diese Weise können Benutzer einer Konversation elegant beenden.

Kontingente

In diesem Abschnitt werden die aktuellen Kontingente in Amazon Lex beschrieben. Diese Kontingente sind nach Kategorien gruppiert.

Servicekontingente können angepasst oder erhöht werden. Wenden Sie sich an denAWS Kundensupport, um ein Kontingent zu erhöhen. Die Anhebung kann einige Tage dauern. Wenn Sie Ihr Kontingent im Rahmen eines größeren Projekts erhöhen, sollten Sie diese Zeit unbedingt zu Ihrem Plan hinzufügen.

Themen

Laufzeit-Service-Kontingente

Beachten Sie neben den in der API-Referenz angegebenen Kontingente Folgendes:

API-Kontingente

- Die Spracheingabe in der Operation PostContent kann bis zu 15 Sekunden lang sein.

- In beiden Laufzeit-API-Operationen, PostContent und PostText, kann der Eingabetext bis zu 1024 Unicode-Zeichen umfassen.

- Die maximale Größe der `PostContent`-Header beträgt 16 KB. Die maximale Größe der Anforderungs- und Sitzungs-Header zusammen beträgt 12 KB.

- Wenn Sie diePostContentPostText Oder-Operationen im Textmodus verwenden, beträgt die maximale Anzahl gleichzeitiger Konversationen mit einem Bot 2 für den$LATEST Alias und 50 für alle anderen Aliase. Das Kontingent gilt für jede API separat.

- Bei Verwendung derPostContent Operation im Sprachmodus beträgt die maximale Anzahl gleichzeitiger Textmodus-Konversationen mit einem Bot 2 für den$LATEST Alias und 125 für alle anderen Aliase. Das Kontingent gilt für jede API separat.

- Die maximale Anzahl gleichzeitiger Sitzungsverwaltungsaufrufe (PutSessionGetSession, undDeleteSession) beträgt 2 für den$LATEST Alias eines Bots und 50 für alle anderen Aliase.

- Die maximale Eingabegröße für eine Lambda-Funktion beträgt 12 KB. Die maximale Ausgabegröße beträgt 25 KB, wobei der Anteil der Sitzungsattribute 12 KB betragen darf.

Verwendung der **$LATEST**-Version

- Die$LATEST Version Ihres Bots sollte nur für manuelle Tests verwendet werden. Amazon Lex begrenzt die Anzahl der Laufzeitanfragen, die Sie an die$LATEST Version des Bots stellen können.

- Wenn Sie die$LATEST Version des Bots aktualisieren, beendet Amazon Lex alle laufenden Konversationen für jede Client-Anwendung, die die$LATEST Version des Bots verwendet. Im Allgemeinen sollten Sie die $LATEST Version eines Bots während der Produktion nicht verwenden, da die $LATEST Version aktualisiert werden kann. Sie sollten stattdessen eine Version veröffentlichen und diese verwenden.

- Wenn Sie einen Alias aktualisieren, benötigt Amazon Lex einige Minuten, um die Änderung zu übernehmen. Wenn Sie die $LATEST Version des Bots ändern, wird die Änderung sofort übernommen.

Sitzungs-Zeitüberschreitung

- Die bei der Erstellung des Bots eingestellte Zeitüberschreitung der Sitzung legt fest, wie lange der Bot den Kontext der Konversation beibehält, z. B. aktuelle Benutzerabsicht und Slot-Daten.

- Nachdem ein Benutzer die Konversation mit Ihrem Bot begonnen hat und bis die Sitzung abläuft, verwendet Amazon Lex dieselbe Bot-Version, auch wenn Sie den Bot-Alias so aktualisieren, dass er auf eine andere Version verweist.

Kontingente des Modellbaus

Modellbau bezieht sich auf das Erstellen und Verwalten von Bots. Dazu gehört das Erstellen/Verwalten von Bots, Absichten, Slot-Typen, Slots und Bot-Channel-Zuordnungen.

Themen

Bot-Kontingente

- Sie konfigurieren Eingabeaufforderungen und Anweisungen in der Modellbau-API. Jede dieser Eingabeaufforderungen oder Anweisungen kann bis zu fünf Mitteilungen haben und jede Mitteilung kann zwischen 1 und 1000 UTF-8-Zeichen enthalten.

- Bei der Verwendung von Mitteilungsgruppen können Sie bis zu fünf Mitteilungsgruppen für jede Mitteilung definieren. Jede Mitteilungsgruppe kann maximal fünf Mitteilungen enthalten und Sie sind auf 15 Mitteilungen in allen Mitteilungsgruppen beschränkt.

- Sie können Beispiel-Äußerungen für Absichten und Slots definieren. Sie können maximal 200 000 Zeichen für alle Äußerungen verwenden.

- Jeder Slot-Typ kann maximal 10 000 Werte und Synonyme definieren. Jeder Bot kann maximal 50 000 Slot-Typenwerte und Synonyme enthalten.

- Die Namen für Bot, Alias und Bot-Channel-Zuordnung werden zum Zeitpunkt der Erstellung nicht nach Groß- und Kleinschreibung unterschieden. Wenn Sie erst `PizzaBot` erstellen und dann versuchen, `pizzaBot` zu genieren, erhalten Sie eine Fehlermeldung. Beim Zugriff auf eine Ressource allerdings werden die Ressourcennamen nach Groß- und Kleinschreibung unterschieden. Sie müssen `PizzaBot` und nicht `pizzaBot` angeben. Diese Namen müssen zwischen 2 und 50 ASCII-Zeichen umfassen.

- Die maximale Anzahl an Versionen, die Sie für alle Ressourcentypen veröffentlichen können, beträgt 100. Beachten Sie, dass es kein Versioning für Aliasnamen gibt.

- Innerhalb eines Bots müssen Absichtsnamen und Slot-Namen eindeutig sein (d. h., eine Absicht und ein Slot können nicht den gleichen Namen haben).

- Sie können einen Bot erstellen, der dafür konfiguriert ist, mehrere Absichten zu unterstützen. Wenn zwei Absichten einen Slot mit gleichem Namen haben, dann muss der entsprechende Slot-Typ derselbe sein.

  Angenommen, Sie erstellen einen Bot zur Unterstützung zweier Absichten (`OrderPizza` und `OrderDrink`). Wenn beide Absichten den Slot `size` haben, dann muss der Slot-Typ in beiden Fällen derselbe sein.

  Darüber hinaus gelten die Beispieläußerungen, die Sie für einen Slot (in einer der Absichten) bereitstellen, für gleichbenannte Slots in anderen Absichten.

- Sie können einem Bot maximal 250 pro Bot zuordnen.

- Beim Erstellen eines Bots geben Sie einen Wert für die Sitzungs-Zeitüberschreitung an. Die Sitzungs-Zeitüberschreitung kann zwischen einer Minute und einem Tag betragen. Der Standardwert beträgt fünf Minuten.

- Sie können bis zu fünf Aliasse für einen Bot erstellen.

- Sie können bis zu bis zu bis zu bis zu 250 Bots AWS Regionen können.

- Sie können nicht mehrere Absichten erstellen, die Erweiterungen derselben integrierten Absicht sind.

Absicht-Kontingente

- Die Namen für Absichten und Slots werden zum Zeitpunkt der Erstellung nicht nach Groß- und Kleinschreibung unterschieden. Das heißt, wenn Sie die Absicht `0rderPizza` erstellen und dann versuchen, die Absicht `orderPizza` zu generieren, erhalten Sie eine Fehlermeldung. Beim Zugriff auf diese Ressourcen werden die Ressourcennamen allerdings nach Groß- und Kleinschreibung unterschieden (Sie müssen `0rderPizza` anstatt `orderPizza` angeben). Diese Namen müssen zwischen 1 und 100 ASCII-Zeichen umfassen.

- Eine Absicht kann bis zu 1,500 Äußerung-Beispiele haben. Es ist mindestens eine Beispiel-Äußerung erforderlich. Jede Beispiel-Äußerung kann bis zu 200 UTF-8-Zeichen lang sein. Sie können bis zu 200 000 Zeichen für alle Absichts- und Slot-Äußerungen in einem Bot verwenden. Absicht – Beispiel:
  - Kann sich auf Null oder mehr Slot-Namen beziehen.
  - Kann sich auf einen Slot nur einmal beziehen.

Beispiel:

```
I want a pizza
I want a {pizzaSize} pizza
I want a {pizzaSize} {pizzaTopping} pizza
```

- Jede Absicht unterstützt zwar bis zu 1.500 Äußerungen, wenn Sie jedoch weniger Äußerungen verwenden, kann Amazon Lex möglicherweise Eingaben außerhalb Ihres bereitgestellten Satzes besser erkennen.

- Sie können bis zu fünf Mitteilungsgruppen für jede Mitteilung in einer Absicht erstellen. Es können insgesamt 15 Mitteilungen in allen Mitteilungsgruppen für eine Mitteilung vorhanden sein.

- Die Konsole kann nur Mitteilungsgruppen für die Mitteilungen `conclusionStatement` und `followUpPrompt` erstellen. Mit der Amazon Lex-API können Sie Nachrichtengruppen für jede andere Nachricht erstellen.

- Jeder Slot kann bis zu 10 Beispieläußerungen haben. Jede Beispieläußerung muss sich genau einmal auf den Slot-Namen beziehen. Beispiel:

```
{pizzaSize} please
```

- Jeder Bot unterstützt maximal 200.000 Zeichen (Absichts- und Slot-Äußerungen zusammen).

- Sie können keine Äußerungen für Absichten bereitstellen, die Erweiterungen integrierter Absichten sind. Für alle anderen Absichten müssen Sie mindestens eine Beispieläußerung bereitstellen. Absichten enthalten Slots, aber die Beispieläußerungen auf Slot-Ebene sind optional.

- Integrierte Absichten

  - Derzeit unterstützt Amazon Lex die Slot-Erfassung für integrierte Absichten nicht. Sie können keine Lambda-Funktionen erstellen, um die`ElicitSlot` Anweisung in der Antwort mit einer Absicht zurückzugeben, die von integrierten Absichten abgeleitet ist. Weitere Informationen finden Sie unter Reaktion-Format.
  - Der Service unterstützt nicht Hinzufügen von Muster-Äußerungen zu eingebauten Ansichten. Ebenso können Sie nicht Slots zu/von den integrierten Ansichten hinzufügen oder entfernen.

- Sie können bis zu 1000 Absichten pro AWS-Konto erstellen. Sie können bis zu 100 Slots in einer Absicht erstellen.

Slot-Typ-Kontingente

- Die Namen für Slot-Typen werden zum Zeitpunkt der Erstellung nicht nach Groß- und Kleinschreibung unterschieden. Wenn Sie den Slot-Typ `PizzaSize` erstellen und dann versuchen, den Slot-Typ `pizzaSize` zu generieren, erhalten Sie eine Fehlermeldung. Beim Zugriff auf diese Ressourcen allerdings werden die Ressourcennamen nach Groß- und Kleinschreibung unterschieden (Sie müssen `PizzaSize` anstatt `pizzaSize` angeben). Namen müssen zwischen 1 und 100 ASCII-Zeichen umfassen.

- Jeder von Ihnen erstellte benutzerdefinierte Slot-Typ unterstützt maximal 10 000 Aufzählungswerte und Synonyme. Jeder Wert kann aus bis zu 140 UTF-8-Zeichen bestehen. Die Aufzählungswerte und Synonyme dürfen keine Duplikate enthalten.

- Geben Sie einen Wert eines Slot-Typs gegebenenfalls sowohl in Groß- als auch und Kleinschreibung an. Geben Sie z. B. für einen Slot-Typ namens `Procedure` sowohl "MRI" als auch "mri" als Wert an, wenn der Wert `MRI` ist.

- Integrierte Slot-Typen — Derzeit unterstützt Amazon Lex das Hinzufügen von Aufzählungswerten oder Synonymen für die integrierten Slot-Typen nicht.

API-Referenz

Dieser Abschnitt enthält Dokumentation für die Amazon Lex API-Operationen. Eine Liste der AWS-Regionen, in denen Amazon Lex verfügbar ist, finden Sie unter AWS-Regionen und Endpunkte in der Amazon Web Services General Reference.

Topics

Aktionen

Die folgenden Aktionen werden von Amazon Lex Model Building Service unterstützt:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots

- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents
- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

Die folgenden Aktionen werden vom Amazon Lex Runtime Service unterstützt:

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Amazon Lex Modellbau Service

Die folgenden Aktionen werden von Amazon Lex Model Building Service unterstützt:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots
- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents

- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

CreateBotVersion

Service: Amazon Lex Model Building Service

Erstellt eine neue Version des Bots basierend auf der `$LATEST` Version. Wenn sich die `$LATEST` Version dieser Ressource seit der Erstellung der letzten Version nicht geändert hat, erstellt Amazon Lex keine neue Version. Es gibt die zuletzt erstellte Version zurück.

> **Note**
>
> Sie können nur die `$LATEST` Version des Bots aktualisieren. Sie können die nummerierten Versionen, die Sie mit dem `CreateBotVersion` Vorgang erstellen, nicht aktualisieren.

Wenn Sie die erste Version eines Bots erstellen, setzt Amazon Lex die Version auf 1. Nachfolgende Versionen werden um 1 erhöht. Weitere Informationen finden Sie unter Versioning.

Diese Operation setzt eine Berechtigung für die `lex:CreateBotVersion`-Aktion voraus.

Anforderungssyntax

```
POST /bots/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Bots, von dem Sie eine neue Version erstellen möchten. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

checksum

Identifiziert eine bestimmte Version der $LATEST Version des Bots. Wenn Sie eine Prüfsumme angeben und die $LATEST Version des Bots eine andere Prüfsumme hat, wird eine `PreconditionFailedException` Ausnahme zurückgegeben und Amazon Lex veröffentlicht keine neue Version. Wenn Sie keine Prüfsumme angeben, veröffentlicht Amazon Lex die $LATEST Version.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 201
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
```

```
      "responseCard": "string"
   },
   "createdDate": number,
   "description": "string",
   "detectSentiment": boolean,
   "enableModelImprovements": boolean,
   "failureReason": "string",
   "idleSessionTTLInSeconds": number,
   "intents": [
      {
         "intentName": "string",
         "intentVersion": "string"
      }
   ],
   "lastUpdatedDate": number,
   "locale": "string",
   "name": "string",
   "status": "string",
   "version": "string",
   "voiceId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP-201-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

abortStatement

Die Nachricht, die Amazon Lex verwendet, um eine Konversation abzubrechen. Weitere Informationen finden Sie unter PutBot.

Typ: Statement Objekt

checksum

Prüfsumme, die die Version des Bots identifiziert, der erstellt wurde.

Typ: Zeichenfolge

childDirected

Für jeden Amazon Lex-Bot, der mit dem Amazon Lex Model Building Service erstellt wurde, müssen Sie angeben, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm

oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und dem Gesetz zum Schutz der Privatsphäre von Kindern im Internet (Children's Online Privacy Protection Act, COPPA) unterliegt, indem Sie `true` oder `false` im `childDirected` Feld angeben. Durch die Angabe **`true`** in **`childDirected`** diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Durch die Angabe `false` in `childDirected` diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex nicht mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Sie dürfen keinen Standardwert für das `childDirected` Feld angeben, der nicht genau wiedergibt, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt.

Wenn sich Ihre Nutzung von Amazon Lex auf eine Website, ein Programm oder eine andere Anwendung bezieht, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet, müssen Sie die erforderliche nachprüfbare Zustimmung der Eltern gemäß COPPA einholen. Informationen zur Verwendung von Amazon Lex in Verbindung mit Websites, Programmen oder anderen Anwendungen, die sich ganz oder teilweise an Kinder unter 13 Jahren richten oder richten, finden Sie in den häufig gestellten Fragen zu Amazon Lex.

Typ: Boolesch

clarificationPrompt

Die Nachricht, die Amazon Lex verwendet, wenn es die Anfrage des Benutzers nicht versteht. Weitere Informationen finden Sie unter PutBot.

Typ: Prompt Objekt

createdDate

Das Datum, an dem die Bot-Version erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

detectSentiment

Gibt an, ob vom Benutzer eingegebene Äußerungen zur Stimmungsanalyse an Amazon Comprehend gesendet werden sollen.

Typ: Boolesch

enableModelImprovements

Gibt an, ob der Bot Genauigkeitsverbesserungen verwendet. `true`gibt an, dass der Bot die Verbesserungen verwendet, andernfalls`false`.

Typ: Boolesch

failureReason

Falls `status` ja`FAILED`, gibt Amazon Lex den Grund an, warum der Bot nicht erstellt werden konnte.

Typ: Zeichenfolge

idleSessionTTLInSeconds

Die maximale Zeit in Sekunden, für die Amazon Lex die in einer Konversation gesammelten Daten aufbewahrt. Weitere Informationen finden Sie unter PutBot.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 60. Maximaler Wert von 86400.

intents

Ein Array von `Intent`-Objekten. Weitere Informationen finden Sie unter PutBot.

Typ: Array von Intent-Objekten

lastUpdatedDate

Das Datum, an dem die `$LATEST` Version dieses Bots aktualisiert wurde.

Typ: Zeitstempel

locale

Gibt das Zielgebietsschema für den Bot an.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

Der Name des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

status

Wenn Sie eine Anfrage zur Erstellung oder Aktualisierung eines Bots senden, setzt Amazon Lex das `status` Antwortelement auf`BUILDING`. Nachdem Amazon Lex den Bot erstellt hat, wird er `status` auf eingestellt`READY`. Wenn Amazon Lex den Bot nicht erstellen kann, wird er `status` auf gesetzt`FAILED`. Amazon Lex gibt den Grund für den Fehler im `failureReason` Antwortelement zurück.

Typ: Zeichenfolge

Zulässige Werte: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

Die Version des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

voiceId

Die Amazon Polly Polly-Sprach-ID, die Amazon Lex für Sprachinteraktionen mit dem Benutzer verwendet.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

CreateIntentVersion

Service: Amazon Lex Model Building Service

Erstellt eine neue Version einer Absicht, die auf der `$LATEST` Version der Absicht basiert. Wenn sich die `$LATEST` Version dieser Absicht seit der letzten Aktualisierung nicht geändert hat, erstellt Amazon Lex keine neue Version. Es gibt die letzte Version zurück, die Sie erstellt haben.

> **Note**
>
> Sie können nur die `$LATEST` Version der Absicht aktualisieren. Sie können die nummerierten Versionen, die Sie mit dem `CreateIntentVersion` Vorgang erstellen, nicht aktualisieren.

Wenn Sie eine Version einer Absicht erstellen, setzt Amazon Lex die Version auf 1. Nachfolgende Versionen werden um 1 erhöht. Weitere Informationen finden Sie unter Versioning.

Diese Operation erfordert zur Ausführung der `lex:CreateIntentVersion`-Aktion Berechtigungen.

Anforderungssyntax

```
POST /intents/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name der Absicht, von der Sie eine neue Version erstellen möchten. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

checksum

Prüfsumme der $LATEST Version der Absicht, die zur Erstellung der neuen Version verwendet werden soll. Wenn Sie eine Prüfsumme angeben und die $LATEST Version der Absicht eine andere Prüfsumme hat, gibt Amazon Lex eine `PreconditionFailedException` Ausnahme zurück und veröffentlicht keine neue Version. Wenn Sie keine Prüfsumme angeben, veröffentlicht Amazon Lex die $LATEST Version.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 201
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
```

```
    },
    "createdDate": number,
    "description": "string",
    "dialogCodeHook": {
       "messageVersion": "string",
       "uri": "string"
    },
    "followUpPrompt": {
       "prompt": {
          "maxAttempts": number,
          "messages": [
             {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
             }
          ],
          "responseCard": "string"
       },
       "rejectionStatement": {
          "messages": [
             {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
             }
          ],
          "responseCard": "string"
       }
    },
    "fulfillmentActivity": {
       "codeHook": {
          "messageVersion": "string",
          "uri": "string"
       },
       "type": "string"
    },
    "inputContexts": [
       {
          "name": "string"
       }
    ],
    "kendraConfiguration": {
       "kendraIndex": "string",
```

```
      "queryFilterString": "string",
      "role": "string"
   },
   "lastUpdatedDate": number,
   "name": "string",
   "outputContexts": [
      {
         "name": "string",
         "timeToLiveInSeconds": number,
         "turnsToLive": number
      }
   ],
   "parentIntentSignature": "string",
   "rejectionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "sampleUtterances": [ "string" ],
   "slots": [
      {
         "defaultValueSpec": {
            "defaultValueList": [
               {
                  "defaultValue": "string"
               }
            ]
         },
         "description": "string",
         "name": "string",
         "obfuscationSetting": "string",
         "priority": number,
         "responseCard": "string",
         "sampleUtterances": [ "string" ],
         "slotConstraint": "string",
         "slotType": "string",
         "slotTypeVersion": "string",
         "valueElicitationPrompt": {
            "maxAttempts": number,
```

```
            "messages": [
               {
                  "content": "string",
                  "contentType": "string",
                  "groupNumber": number
               }
            ],
            "responseCard": "string"
         }
      }
   ],
   "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP-201-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

checksum

Prüfsumme der erstellten Intent-Version.

Typ: Zeichenfolge

conclusionStatement

Nachdem die im `fulfillmentActivity` Feld angegebene Lambda-Funktion die Absicht erfüllt hat, übermittelt Amazon Lex diese Aussage an den Benutzer.

Typ: Statement Objekt

confirmationPrompt

Falls definiert, die Aufforderung, die Amazon Lex verwendet, um die Absicht des Benutzers zu bestätigen, bevor sie erfüllt wird.

Typ: Prompt Objekt

createdDate

Das Datum, an dem die Absicht erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

dialogCodeHook

Falls definiert, ruft Amazon Lex diese Lambda-Funktion für jede Benutzereingabe auf.

Typ: CodeHook Objekt

followUpPrompt

Falls definiert, verwendet Amazon Lex diese Aufforderung, um zusätzliche Benutzeraktivitäten anzufordern, nachdem die Absicht erfüllt wurde.

Typ: FollowUpPrompt Objekt

fulfillmentActivity

Beschreibt, wie die Absicht erfüllt wird.

Typ: FulfillmentActivity Objekt

inputContexts

Ein Array von `InputContext` Objekten, das die Kontexte auflistet, die aktiv sein müssen, damit Amazon Lex die Absicht in einer Konversation mit dem Benutzer auswählen kann.

Typ: Array von InputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 5 Elemente.

kendraConfiguration

Konfigurationsinformationen, falls vorhanden, für die Verbindung eines Amazon Kendra Kendra-Index mit der `AMAZON.KendraSearchIntent` Absicht.

Typ: KendraConfiguration Objekt

lastUpdatedDate

Das Datum, an dem die Absicht aktualisiert wurde.

Typ: Zeitstempel

name

Der Name der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

outputContexts

Ein Array von `OutputContext` Objekten, das die Kontexte auflistet, die durch die Absicht aktiviert werden, wenn die Absicht erfüllt ist.

Typ: Array von OutputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

parentIntentSignature

Ein eindeutiger Bezeichner für eine integrierte Absicht.

Typ: Zeichenfolge

rejectionStatement

Wenn der Benutzer die in definierte Frage mit „Nein“ beantwortet`confirmationPrompt`, antwortet Amazon Lex mit dieser Erklärung, um zu bestätigen, dass die Absicht storniert wurde.

Typ: Statement Objekt

sampleUtterances

Eine Reihe von Beispieläußerungen, die für die Absicht konfiguriert wurden.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 1500 Elementen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 200 Zeichen.

slots

Eine Reihe von Slot-Typen, die die Informationen definieren, die zur Erfüllung der Absicht erforderlich sind.

Typ: Array von Slot-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 100 Elemente.

version

Die Versionsnummer, die der neuen Version der Absicht zugewiesen wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

CreateSlotTypeVersion

Service: Amazon Lex Model Building Service

Erstellt eine neue Version eines Slot-Typs basierend auf der `$LATEST` Version des angegebenen Slot-Typs. Wenn sich die `$LATEST` Version dieser Ressource seit der letzten Version, die Sie erstellt haben, nicht geändert hat, erstellt Amazon Lex keine neue Version. Es gibt die letzte Version zurück, die Sie erstellt haben.

> **Note**
>
> Sie können nur die `$LATEST` Version eines Slot-Typs aktualisieren. Sie können die nummerierten Versionen, die Sie mit dem `CreateSlotTypeVersion` Vorgang erstellen, nicht aktualisieren.

Wenn Sie eine Version eines Slot-Typs erstellen, setzt Amazon Lex die Version auf 1. Nachfolgende Versionen werden um 1 erhöht. Weitere Informationen finden Sie unter Versioning.

Diese Operation erfordert Berechtigungen für die Aktion `lex:CreateSlotTypeVersion`.

Anforderungssyntax

```
POST /slottypes/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Slot-Typs, für den Sie eine neue Version erstellen möchten. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

checksum

Prüfsumme für die $LATEST Version des Slot-Typs, den Sie veröffentlichen möchten. Wenn Sie eine Prüfsumme angeben und die $LATEST Version des Slot-Typs eine andere Prüfsumme hat, gibt Amazon Lex eine PreconditionFailedException Ausnahme zurück und veröffentlicht die neue Version nicht. Wenn Sie keine Prüfsumme angeben, veröffentlicht Amazon Lex die $LATEST Version.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 201
Content-type: application/json

{
   "checksum": "string",
   "createdDate": number,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
```

```
    ],
    "valueSelectionStrategy": "string",
    "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP-201-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

checksum

Prüfsumme der `$LATEST` Version des Slot-Typs.

Typ: Zeichenfolge

createdDate

Das Datum, an dem der Slot-Typ erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

enumerationValues

Eine Liste von `EnumerationValue` Objekten, die die Werte definiert, die der Slot-Typ annehmen kann.

Typ: Array von EnumerationValue-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 10000 Elementen.

lastUpdatedDate

Das Datum, an dem der Slot-Typ aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Der integrierte Steckplatztyp, der als übergeordnetes Element des Steckplatztyps verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Konfigurationsinformationen, die den Typ des übergeordneten integrierten Steckplatzes erweitern.

Typ: Array von SlotTypeConfiguration-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

valueSelectionStrategy

Die Strategie, mit der Amazon Lex den Wert des Slots bestimmt. Weitere Informationen finden Sie unter PutSlotType.

Typ: Zeichenfolge

Zulässige Werte: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

Die Version, die der neuen Slot-Typ-Version zugewiesen wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteBot

Service: Amazon Lex Model Building Service

Löscht alle Versionen des Bots, einschließlich der `$LATEST` Version. Verwenden Sie den DeleteBotVersion Vorgang, um eine bestimmte Version des Bots zu löschen. Durch den `DeleteBot` Vorgang wird das Bot-Schema nicht sofort entfernt. Stattdessen wird es zum Löschen markiert und später entfernt.

Amazon Lex speichert Äußerungen auf unbestimmte Zeit, um die Fähigkeit Ihres Bots zu verbessern, auf Benutzereingaben zu reagieren. Diese Äußerungen werden nicht entfernt, wenn der Bot gelöscht wird. Verwenden Sie die Operation, um die Äußerungen zu entfernen. DeleteUtterances

Wenn ein Bot einen Alias hat, können Sie ihn nicht löschen. Stattdessen gibt die `DeleteBot` Operation eine `ResourceInUseException` Ausnahme zurück, die einen Verweis auf den Alias enthält, der auf den Bot verweist. Um den Verweis auf den Bot zu entfernen, löschen Sie den Alias. Wenn Sie dieselbe Ausnahme erneut erhalten, löschen Sie den verweisenden Alias, bis der `DeleteBot` Vorgang erfolgreich ist.

Diese Operation erfordert Berechtigungen für die Aktion `lex:DeleteBot`.

Anforderungssyntax

```
DELETE /bots/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Bots. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie zu löschen versuchen, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteBotAlias

Service: Amazon Lex Model Building Service

Löscht einen Alias für den angegebenen Bot.

Sie können keinen Alias löschen, der in der Verbindung zwischen einem Bot und einem Messaging-Kanal verwendet wird. Wenn ein Alias in einer Kanalzuordnung verwendet wird, gibt der `DeleteBot` Vorgang eine `ResourceInUseException` Ausnahme zurück, die einen Verweis auf die Kanalzuweisung enthält, die sich auf den Bot bezieht. Sie können den Verweis auf den Alias entfernen, indem Sie die Kanalzuordnung löschen. Wenn Sie dieselbe Ausnahme erneut erhalten, löschen Sie die verweisende Zuordnung, bis der `DeleteBotAlias` Vorgang erfolgreich ist.

Anforderungssyntax

```
DELETE /bots/botName/aliases/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botName

Der Name des Bots, auf den der Alias zeigt.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

name

Der Name des Aliasses, der gelöscht werden soll. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie zu löschen versuchen, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteBotChannelAssociation

Service: Amazon Lex Model Building Service

Löscht die Verknüpfung zwischen einem Amazon Lex Lex-Bot und einer Messaging-Plattform.

Diese Operation setzt eine Berechtigung für die `lex:DeleteBotChannelAssociation`-Aktion voraus.

Anforderungssyntax

```
DELETE /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

aliasName

Ein Alias, der auf die spezifische Version des Amazon Lex Lex-Bot verweist, zu dem diese Zuordnung hergestellt wird.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

botName

Der Name des Amazon Lex Lex-Bot.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

name

Der Name der Zuordnung. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteBotVersion

Service: Amazon Lex Model Building Service

Löscht die angegebene Version eines Bots Verwenden Sie den DeleteBot Vorgang, um alle Versionen eines Bots zu löschen.

Diese Operation erfordert Berechtigungen für die Aktion `lex:DeleteBotVersion`.

Anforderungssyntax

```
DELETE /bots/name/versions/version HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Bots.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

version

Die Version des zu löschenden Bots. Sie können die `$LATEST` Version des Bots nicht löschen. Verwenden Sie den DeleteBot Vorgang, um die `$LATEST` Version zu löschen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht gut formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie zu löschen versuchen, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteIntent

Service: Amazon Lex Model Building Service

Löscht alle Versionen der Absicht, einschließlich der `$LATEST` Version. Verwenden Sie den DeleteIntentVersion Vorgang, um eine bestimmte Version der Absicht zu löschen.

Sie können eine Version einer Absicht nur löschen, wenn nicht auf sie verwiesen wird. Um eine Absicht zu löschen, auf die in einem oder mehreren Bots verwiesen wird (sieheAmazon Lex — Funktionsweise), müssen Sie zuerst diese Verweise entfernen.

> **Note**
>
> Wenn Sie die `ResourceInUseException` Ausnahme erhalten, erhalten Sie eine Beispielreferenz, die zeigt, wo auf die Absicht verwiesen wird. Um den Verweis auf die Absicht zu entfernen, aktualisieren Sie entweder den Bot oder löschen Sie ihn. Wenn Sie beim erneuten Versuch, die Absicht zu löschen, dieselbe Ausnahme erhalten, wiederholen Sie den Vorgang, bis die Absicht keine Referenzen mehr hat und der Aufruf von erfolgreich `DeleteIntent` ist.

Diese Operation setzt eine Berechtigung für die `lex:DeleteIntent`-Aktion voraus.

Anforderungssyntax

```
DELETE /intents/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name der Absicht. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie zu löschen versuchen, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteIntentVersion

Service: Amazon Lex Model Building Service

Löscht die angegebene Version einer Absicht Verwenden Sie den DeleteIntent Vorgang, um alle Versionen einer Absicht zu löschen.

Diese Operation erfordert Berechtigungen für die Aktion `lex:DeleteIntentVersion`.

Anforderungssyntax

```
DELETE /intents/name/versions/version HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name der Absicht.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

version

Die Version der Absicht, die gelöscht werden soll. Sie können die `$LATEST` Version der Absicht nicht löschen. Verwenden Sie den DeleteIntent Vorgang, um die `$LATEST` Version zu löschen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht gut formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie löschen möchten, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteSlotType

Service: Amazon Lex Model Building Service

Löscht alle Versionen des Slot-Typs, einschließlich der `$LATEST` Version. Verwenden Sie den DeleteSlotTypeVersion Vorgang, um eine bestimmte Version des Slot-Typs zu löschen.

Sie können eine Version eines Slot-Typs nur löschen, wenn nicht darauf verwiesen wird. Um einen Slot-Typ zu löschen, auf den in einer oder mehreren Absichten verwiesen wird, müssen Sie zuerst diese Verweise entfernen.

> **Note**
>
> Wenn Sie die `ResourceInUseException` Ausnahme erhalten, enthält die Ausnahme eine Beispielreferenz, aus der hervorgeht, in welcher Absicht auf den Slot-Typ verwiesen wird. Um den Verweis auf den Slot-Typ zu entfernen, aktualisieren Sie entweder die Absicht oder löschen Sie sie. Wenn Sie beim erneuten Versuch, den Slot-Typ zu löschen, dieselbe Ausnahme erhalten, wiederholen Sie den Vorgang, bis der Slot-Typ keine Referenzen mehr hat und der `DeleteSlotType` Aufruf erfolgreich ist.

Diese Operation setzt eine Berechtigung für die `lex:DeleteSlotType`-Aktion voraus.

Anforderungssyntax

```
DELETE /slottypes/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Slot-Typs. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie zu löschen versuchen, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteSlotTypeVersion

Service: Amazon Lex Model Building Service

Löscht die angegebene Version eines Slot-Typs Verwenden Sie den DeleteSlotType Vorgang, um alle Versionen eines Slot-Typs zu löschen.

Diese Operation erfordert Berechtigungen für die Aktion `lex:DeleteSlotTypeVersion`.

Anforderungssyntax

```
DELETE /slottypes/name/version/version HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Slot-Typs.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

version

Die Version des zu löschenden Slot-Typs. Sie können die `$LATEST` Version des Slot-Typs nicht löschen. Verwenden Sie den DeleteSlotType Vorgang, um die `$LATEST` Version zu löschen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht gut formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

ResourceInUseException

Auf die Ressource, die Sie zu löschen versuchen, wird von einer anderen Ressource verwiesen. Verwenden Sie diese Informationen, um Verweise auf die Ressource zu entfernen, die Sie löschen möchten.

Der Hauptteil der Ausnahme enthält ein JSON-Objekt, das die Ressource beschreibt.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP Status Code: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

DeleteUtterances

Service: Amazon Lex Model Building Service

Löscht gespeicherte Äußerungen.

Amazon Lex speichert die Äußerungen, die Benutzer an Ihren Bot senden. Äußerungen werden 15 Tage lang gespeichert, damit sie für den GetUtterancesView Vorgang verwendet werden können. Anschließend werden sie auf unbestimmte Zeit gespeichert, um die Fähigkeit Ihres Bots zu verbessern, auf Benutzereingaben zu reagieren.

Verwenden Sie diesen `DeleteUtterances` Vorgang, um gespeicherte Äußerungen für einen bestimmten Benutzer manuell zu löschen. Wenn Sie den `DeleteUtterances` Vorgang verwenden, werden Äußerungen, die gespeichert wurden, um die Fähigkeit Ihres Bots zu verbessern, auf Benutzereingaben zu reagieren, sofort gelöscht. Äußerungen, die für den `GetUtterancesView` Vorgang gespeichert wurden, werden nach 15 Tagen gelöscht.

Diese Operation erfordert Berechtigungen für die Aktion `lex:DeleteUtterances`.

Anforderungssyntax

```
DELETE /bots/botName/utterances/userId HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botName

Der Name des Bots, der die Äußerungen gespeichert hat.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

userId

Die eindeutige Kennung für den Benutzer, der die Äußerungen gemacht hat. Dies ist die Benutzer-ID, die in der PostTextOperationsanforderung gesendet wurde, die PostContentdie Äußerung enthielt.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBot

Service: Amazon Lex Model Building Service

Gibt Metadateninformationen für einen bestimmten Bot zurück. Sie müssen den Bot-Namen und die Bot-Version oder den Alias angeben.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetBot`.

Anforderungssyntax

```
GET /bots/name/versions/versionoralias HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Bots. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

versionoralias

Die Version oder der Alias des Bots.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "abortStatement": {
      "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "checksum": "string",
      "childDirected": boolean,
      "clarificationPrompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "createdDate": number,
      "description": "string",
      "detectSentiment": boolean,
      "enableModelImprovements": boolean,
      "failureReason": "string",
      "idleSessionTTLInSeconds": number,
      "intents": [
         {
            "intentName": "string",
            "intentVersion": "string"
         }
      ],
      "lastUpdatedDate": number,
      "locale": "string",
      "name": "string",
      "nluIntentConfidenceThreshold": number,
      "status": "string",
      "version": "string",
      "voiceId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

abortStatement

Die Nachricht, die Amazon Lex zurückgibt, wenn der Benutzer beschließt, die Konversation zu beenden, ohne sie abzuschließen. Weitere Informationen finden Sie unter PutBot.

Typ: Statement Objekt

checksum

Prüfsumme des Bots, anhand derer eine bestimmte Version der Bot-Version identifiziert wurde. `$LATEST`

Typ: Zeichenfolge

childDirected

Für jeden Amazon Lex-Bot, der mit dem Amazon Lex Model Building Service erstellt wurde, müssen Sie angeben, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und dem Gesetz zum Schutz der Privatsphäre von Kindern im Internet (Children's Online Privacy Protection Act, COPPA) unterliegt, indem Sie `true` oder `false` im `childDirected` Feld angeben. Durch die Angabe **`true`** in **`childDirected`** diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Durch die Angabe `false` in `childDirected` diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex nicht mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Sie dürfen keinen Standardwert für das `childDirected` Feld angeben, der nicht genau wiedergibt, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt.

Wenn sich Ihre Nutzung von Amazon Lex auf eine Website, ein Programm oder eine andere Anwendung bezieht, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet, müssen Sie

die erforderliche nachprüfbare Zustimmung der Eltern gemäß COPPA einholen. Informationen zur Verwendung von Amazon Lex in Verbindung mit Websites, Programmen oder anderen Anwendungen, die sich ganz oder teilweise an Kinder unter 13 Jahren richten oder richten, finden Sie in den häufig gestellten Fragen zu Amazon Lex.

Typ: Boolesch

clarificationPrompt

Die Nachricht, die Amazon Lex verwendet, wenn es die Anfrage des Benutzers nicht versteht. Weitere Informationen finden Sie unter PutBot.

Typ: Prompt Objekt

createdDate

Das Datum, an dem der Bot erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

detectSentiment

Gibt an, ob Benutzeräußerungen zur Stimmungsanalyse an Amazon Comprehend gesendet werden sollen.

Typ: Boolesch

enableModelImprovements

Gibt an, ob der Bot Genauigkeitsverbesserungen verwendet. `true`gibt an, dass der Bot die Verbesserungen verwendet, andernfalls`false`.

Typ: Boolesch

failureReason

Falls `status` ja`FAILED`, erklärt Amazon Lex, warum der Bot nicht erstellt werden konnte.

Typ: Zeichenfolge

idleSessionTTLInSeconds

Die maximale Zeit in Sekunden, für die Amazon Lex die in einer Konversation gesammelten Daten aufbewahrt. Weitere Informationen finden Sie unter PutBot.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 60. Maximaler Wert von 86400.

intents

Ein Array von `intent`-Objekten. Weitere Informationen finden Sie unter PutBot.

Typ: Array von Intent-Objekten

lastUpdatedDate

Das Datum, an dem der Bot aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

locale

Das Zielgebietsschema für den Bot.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

Der Name des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

Die Punktzahl, die bestimmt, wo Amazon Lex die oder beide einfügt `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`, wenn alternative

Absichten in einer PostContentPostTextOder-Antwort zurückgegeben werden. `AMAZON.FallbackIntent`wird eingefügt, wenn der Konfidenzwert für alle Absichten unter diesem Wert liegt. `AMAZON.KendraSearchIntent`wird nur eingefügt, wenn es für den Bot konfiguriert ist.

Type: Double

Gültiger Bereich: Mindestwert 0. Maximalwert von 1.

status

Der Status des Bots.

Wenn der Status lautet, `BUILDING` erstellt Amazon Lex den Bot zum Testen und Verwenden.

Wenn der Status des Bots lautet`READY_BASIC_TESTING`, können Sie den Bot anhand der genauen Äußerungen testen, die in den Absichten des Bots angegeben sind. Wenn der Bot für vollständige Tests oder zur Ausführung bereit ist, lautet der Status. `READY`

Wenn beim Erstellen des Bots ein Problem aufgetreten ist, lautet der Status `FAILED` und das `failureReason` Feld erklärt, warum der Bot nicht gebaut wurde.

Wenn der Bot gespeichert, aber nicht gebaut wurde, lautet der Status`NOT_BUILT`.

Typ: Zeichenfolge

Zulässige Werte: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

Die Version des Bots. Für einen neuen Bot ist die Version immer gültig`$LATEST`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

voiceId

Die Amazon Polly Polly-Sprach-ID, die Amazon Lex für die Sprachinteraktion mit dem Benutzer verwendet. Weitere Informationen finden Sie unter PutBot.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python

- AWS SDK for Ruby V3

GetBotAlias

Service: Amazon Lex Model Building Service

Gibt Informationen über einen Amazon Lex-Bot-Alias zurück. Weitere Informationen zu Aliassen finden Sie unter Versioning und Aliasnamen.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetBotAlias`.

Anforderungssyntax

```
GET /bots/botName/aliases/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botName

Der Name des Bots.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

name

Der Name des Bot-Alias. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "botName": "string",
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string",
            "resourcePrefix": "string"
         }
      ]
   },
   "createdDate": number,
   "description": "string",
   "lastUpdatedDate": number,
   "name": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

botName

Der Name des Bots, auf den der Alias zeigt.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

botVersion

Die Version des Bots, auf den der Alias verweist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

checksum

Prüfsumme des Bot-Alias.

Typ: Zeichenfolge

conversationLogs

Die Einstellungen, die bestimmen, wie Amazon Lex Konversationsprotokolle für den Alias verwendet.

Typ: ConversationLogsResponse Objekt

createdDate

Das Datum, an dem der Bot-Alias erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung des Bot-Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

lastUpdatedDate

Das Datum, an dem der Bot-Alias aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name des Bot-Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python

- AWS SDK for Ruby V3

GetBotAliases

Service: Amazon Lex Model Building Service

Gibt eine Liste von Aliasen für einen bestimmten Amazon Lex Lex-Bot zurück.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetBotAliases`.

Anforderungssyntax

```
GET /bots/botName/aliases/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botName

Der Name des Bots.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

maxResults

Die maximale Anzahl von Aliasen, die in der Antwort zurückgegeben werden sollen. Die Standardeinstellung ist 50.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nameContains

Teilzeichenfolge, der in Bot-Aliasnamen zugeordnet werden soll. Ein Alias wird zurückgegeben, wenn ein Teil seines Namens mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Aliasnamen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Aliasen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "BotAliases": [
      {
         "botName": "string",
         "botVersion": "string",
         "checksum": "string",
         "conversationLogs": {
            "iamRoleArn": "string",
            "logSettings": [
               {
                  "destination": "string",
                  "kmsKeyArn": "string",
                  "logType": "string",
                  "resourceArn": "string",
                  "resourcePrefix": "string"
               }
            ]
         },
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string"
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

BotAliases

Eine Reihe von `BotAliasMetadata` Objekten, von denen jedes einen Bot-Alias beschreibt.

Typ: Array von BotAliasMetadata-Objekten

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Aliasen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Aliasen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBotChannelAssociation

Service: Amazon Lex Model Building Service

Gibt Informationen über die Verbindung zwischen einem Amazon Lex Lex-Bot und einer Messaging-Plattform zurück.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetBotChannelAssociation`.

Anforderungssyntax

```
GET /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

aliasName

Ein Alias, der auf die spezifische Version des Amazon Lex Lex-Bot verweist, zu dem diese Zuordnung hergestellt wird.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

botName

Der Name des Amazon Lex Lex-Bot.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

name

Der Name der Assoziation zwischen dem Bot und dem Kanal. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: ^([A-Za-z]_?)+$

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "botAlias": "string",
   "botConfiguration": {
      "string" : "string"
   },
   "botName": "string",
   "createdDate": number,
   "description": "string",
   "failureReason": "string",
   "name": "string",
   "status": "string",
   "type": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

botAlias

Ein Alias, der auf die spezifische Version des Amazon Lex Lex-Bot verweist, zu dem diese Zuordnung hergestellt wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: ^([A-Za-z]_?)+$

botConfiguration

Stellt Informationen bereit, die die Messaging-Plattform für die Kommunikation mit dem Amazon Lex Lex-Bot benötigt.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Maximale Anzahl von 10 Elementen.

botName

Der Name des Amazon Lex Lex-Bot.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

createdDate

Das Datum, an dem die Verbindung zwischen dem Bot und dem Kanal erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung der Verbindung zwischen dem Bot und dem Kanal.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

failureReason

Falls `status` ja`FAILED`, gibt Amazon Lex den Grund an, warum die Zuordnung nicht erstellt werden konnte.

Typ: Zeichenfolge

name

Der Name der Assoziation zwischen dem Bot und dem Kanal.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

status

Der Status des Bot-Kanals.

- `CREATED`- Der Kanal wurde erstellt und ist einsatzbereit.
- `IN_PROGRESS`- Die Erstellung des Kanals ist im Gange.
- `FAILED`- Beim Erstellen des Kanals ist ein Fehler aufgetreten. Informationen zur Ursache des Fehlers finden Sie in dem `failureReason` Feld.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | CREATED | FAILED`

type

Der Typ der Messaging-Plattform.

Typ: Zeichenfolge

Zulässige Werte: `Facebook | Slack | Twilio-Sms | Kik`

Fehler

BadRequestException

Die Anfrage ist nicht wohlformuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBotChannelAssociations

Service: Amazon Lex Model Building Service

Gibt eine Liste aller Kanäle zurück, die dem angegebenen Bot zugeordnet sind.

Für den `GetBotChannelAssociations` Vorgang sind Berechtigungen für die `lex:GetBotChannelAssociations` Aktion erforderlich.

Anforderungssyntax

```
GET /bots/botName/aliases/aliasName/channels/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

aliasName

Ein Alias, der auf die spezifische Version des Amazon Lex Lex-Bot verweist, zu dem diese Zuordnung hergestellt wird.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^(-|^([A-Za-z]_?)+$)$`

Erforderlich: Ja

botName

Der Name des Amazon Lex-Bots in der Assoziation.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

maxResults

Die maximale Anzahl von Assoziationen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 50.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nameContains

Teilzeichenfolge, die in den Namen der Kanalzuordnungen übereinstimmen soll. Eine Assoziation wird zurückgegeben, wenn ein Teil ihres Namens mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“. Um alle Bot-Kanalzuordnungen zurückzugeben, verwenden Sie einen Bindestrich („-“) als Parameter. `nameContains`

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Assoziationen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Verknüpfungen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "botChannelAssociations": [
      {
         "botAlias": "string",
         "botConfiguration": {
            "string" : "string"
         },
         "botName": "string",
         "createdDate": number,
         "description": "string",
         "failureReason": "string",
         "name": "string",
         "status": "string",
         "type": "string"
      }
```

```
    ],
    "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

botChannelAssociations

Eine Reihe von Objekten, eines für jede Assoziation, die Informationen über den Amazon Lex Lex-Bot und seine Zuordnung zum Kanal bereitstellt.

Typ: Array von BotChannelAssociation-Objekten

nextToken

Ein Paginierungstoken, das die nächste Seite mit Assoziationen abruft. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Verknüpfungen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBots

Service: Amazon Lex Model Building Service

Gibt Bot-Informationen wie folgt zurück:

- Wenn Sie das `nameContains` Feld angeben, enthält die Antwort Informationen zur `$LATEST` Version aller Bots, deren Name die angegebene Zeichenfolge enthält.
- Wenn Sie das `nameContains` Feld nicht angeben, gibt der Vorgang Informationen über die `$LATEST` Version all Ihrer Bots zurück.

Diese Operation setzt eine Berechtigung für die `lex:GetBots`-Aktion voraus.

Anforderungssyntax

```
GET /bots/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Bots, die in der Antwort zurückgegeben werden sollen, die die Anfrage zurückgibt. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nameContains

Teilzeichenfolge, der in Bot-Namen zugeordnet werden soll. Ein Bot wird zurückgegeben, wenn ein Teil seines Namens mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nextToken

Ein Paginierungstoken, das die nächste Seite mit Bots abruft. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Bots abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

bots

Eine Reihe von `botMetadata` Objekten mit einem Eintrag für jeden Bot.

Typ: Array von BotMetadata-Objekten

nextToken

Wenn die Antwort gekürzt ist, enthält sie ein Paginierungstoken, das Sie in Ihrer nächsten Anfrage angeben können, um die nächste Seite mit Bots abzurufen.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht wohlformuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python

- AWS SDK for Ruby V3

GetBotVersions

Service: Amazon Lex Model Building Service

Ruft Informationen über alle Versionen eines Bots ab.

Die `GetBotVersions` Operation gibt ein `BotMetadata` Objekt für jede Version eines Bots zurück. Wenn ein Bot beispielsweise über drei nummerierte Versionen verfügt, gibt die `GetBotVersions` Operation in der Antwort vier `BotMetadata` Objekte zurück, eines für jede nummerierte Version und eines für die `$LATEST` Version.

Die `GetBotVersions` Operation gibt immer mindestens eine Version zurück, die `$LATEST` Version.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetBotVersions`.

Anforderungssyntax

```
GET /bots/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Bot-Versionen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

name

Der Name der Bots, für den Versionen zurückgegeben werden soll.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Bot-Versionen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um

die nächste Seite mit Versionen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

bots

Ein Array von `BotMetadata` Objekten, eines für jede nummerierte Version des Bots plus eines für die `$LATEST` Version.

Typ: Array von BotMetadata-Objekten

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Bot-Versionen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um

die nächste Seite mit Versionen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBuiltinIntent

Service: Amazon Lex Model Building Service

Gibt Informationen zu einer integrierten Absicht zurück.

Diese Operation setzt eine Berechtigung für die `lex:GetBuiltinIntent`-Aktion voraus.

Anforderungssyntax

```
GET /builtins/intents/signature HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

signature

Die eindeutige Kennung für eine integrierte Absicht. Informationen zur Signatur für eine Absicht finden Sie unter Integrierte Standardabsichten im Alexa Skills Kit.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "signature": "string",
   "slots": [
      {
         "name": "string"
      }
   ],
   "supportedLocales": [ "string" ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

signature

Die eindeutige Kennung für eine integrierte Absicht.

Typ: Zeichenfolge

slots

Eine Reihe von `BuiltinIntentSlot` Objekten, ein Eintrag für jeden Slot-Typ in der Absicht.

Typ: Array von BuiltinIntentSlot-Objekten

supportedLocales

Eine Liste von Gebietsschemas, die von der Intent unterstützt werden.

Typ: Zeichenfolgen-Array

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBuiltinIntents

Service: Amazon Lex Model Building Service

Ruft eine Liste der integrierten Absichten ab, die den angegebenen Kriterien entsprechen

Diese Operation setzt eine Berechtigung für die `lex:GetBuiltinIntents`-Aktion voraus.

Anforderungssyntax

```
GET /builtins/intents/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

locale

Eine Liste von Gebietsschemas, die der Intent unterstützt.

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

maxResults

Die maximale Anzahl von Absichten, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nextToken

Ein Paginierungstoken, das die nächste Seite mit Absichten abruft. Wenn dieser API-Aufruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Intents abzurufen, verwenden Sie das Paginierungstoken in der nächsten Anfrage.

signatureContains

Teilzeichenfolge, der in den integrierten Absichtssignaturen zugeordnet werden soll. Eine Absicht wird zurückgegeben, wenn ein Teil ihrer Signatur mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“. Informationen zur Signatur für eine Absicht finden Sie unter Integrierte Standardabsichten im Alexa Skills Kit.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

intents

Eine Reihe von `builtinIntentMetadata` Objekten, eines für jede Absicht in der Antwort.

Typ: Array von BuiltinIntentMetadata-Objekten

nextToken

Ein Paginierungstoken, das die nächste Seite mit Absichten abruft. Wenn die Antwort auf diesen API-Aufruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Absichten abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetBuiltinSlotTypes

Service: Amazon Lex Model Building Service

Ruft eine Liste der integrierten Slot-Typen ab, die den angegebenen Kriterien entsprechen

Eine Liste der integrierten Slot-Typen finden Sie unter Slot-Typ-Referenz im Alexa Skills Kit.

Diese Operation setzt eine Berechtigung für die `lex:GetBuiltInSlotTypes`-Aktion voraus.

Anforderungssyntax

```
GET /builtins/slottypes/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

locale

Eine Liste der Gebietsschemas, die der Slot-Typ unterstützt.

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

maxResults

Die maximale Anzahl von Slot-Typen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nextToken

Ein Paginierungstoken, das die nächste Seite mit Slot-Typen abruft. Wenn die Antwort auf diesen API-Aufruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Slot-Typen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

signatureContains

Teilzeichenfolge, der in den integrierten Slot-Typ-Signaturen zugeordnet werden soll. Ein Slot-Typ wird zurückgegeben, wenn ein Teil seiner Signatur mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

nextToken

Wenn die Antwort gekürzt ist, enthält die Antwort ein Paginierungstoken, das Sie in Ihrer nächsten Anfrage verwenden können, um die nächste Seite mit Slot-Typen abzurufen.

Typ: Zeichenfolge

slotTypes

Eine Reihe von `BuiltInSlotTypeMetadata` Objekten, ein Eintrag für jeden zurückgegebenen Slot-Typ.

Typ: Array von BuiltinSlotTypeMetadata-Objekten

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetExport

Service: Amazon Lex Model Building Service

Exportiert den Inhalt einer Amazon Lex Lex-Ressource in einem angegebenen Format.

Anforderungssyntax

```
GET /exports/?exportType=exportType&name=name&resourceType=resourceType&version=version
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

exportType

Das Format der exportierten Daten.

Zulässige Werte: `ALEXA_SKILLS_KIT | LEX`

Erforderlich: Ja

name

Der Name des zu exportierenden Bots.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `[a-zA-Z_]+`

Erforderlich: Ja

resourceType

Der Typ der zu exportierenden Ressource.

Zulässige Werte: `BOT | INTENT | SLOT_TYPE`

Erforderlich: Ja

version

Die Version des zu exportierenden Bots.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "exportStatus": "string",
   "exportType": "string",
   "failureReason": "string",
   "name": "string",
   "resourceType": "string",
   "url": "string",
   "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

exportStatus

Der Status des Exports.

- `IN_PROGRESS`- Der Export ist im Gange.
- `READY`- Der Export ist abgeschlossen.
- `FAILED`- Der Export konnte nicht abgeschlossen werden.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | READY | FAILED`

exportType

Das Format der exportierten Daten.

Typ: Zeichenfolge

Zulässige Werte: `ALEXA_SKILLS_KIT | LEX`

failureReason

Falls `status` ja`FAILED`, gibt Amazon Lex den Grund an, warum die Ressource nicht exportiert werden konnte.

Typ: Zeichenfolge

name

Der Name des Bots, der exportiert wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `[a-zA-Z_]+`

resourceType

Der Typ der exportierten Ressource.

Typ: Zeichenfolge

Zulässige Werte: `BOT | INTENT | SLOT_TYPE`

url

Eine vorsignierte S3-URL, die den Speicherort der exportierten Ressource angibt. Die exportierte Ressource ist ein ZIP-Archiv, das die exportierte Ressource im JSON-Format enthält. Die Struktur des Archivs kann sich ändern. Ihr Code sollte sich nicht auf die Archivstruktur verlassen.

Typ: Zeichenfolge

version

Die Version des Bots, der exportiert wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht gut formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetImport

Service: Amazon Lex Model Building Service

Ruft Informationen über einen Importauftrag ab, der mit dem `StartImport` Vorgang gestartet wurde.

Anforderungssyntax

```
GET /imports/importId HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

importId

Die ID der zurückzugebenden Importauftragsinformationen.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "createdDate": number,
   "failureReason": [ "string" ],
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

createdDate

Ein Zeitstempel für das Datum und die Uhrzeit der Erstellung des Importauftrags.

Typ: Zeitstempel

failureReason

Eine Zeichenfolge, die beschreibt, warum ein Importauftrag nicht abgeschlossen werden konnte.

Typ: Zeichenfolgen-Array

importId

Der Bezeichner für den spezifischen Importauftrag.

Typ: Zeichenfolge

importStatus

Der Status des Importauftrags. Wenn der Status lautet`FAILED`, können Sie den Grund für den Fehler aus dem `failureReason` Feld abrufen.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

Die Aktion, die ausgeführt wurde, als ein Konflikt zwischen einer vorhandenen Ressource und einer Ressource in der Importdatei auftrat.

Typ: Zeichenfolge

Zulässige Werte: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

Der Name, der dem Importauftrag gegeben wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `[a-zA-Z_]+`

resourceType

Der Typ der importierten Ressource.

Typ: Zeichenfolge

Zulässige Werte: `BOT | INTENT | SLOT_TYPE`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetIntent

Service: Amazon Lex Model Building Service

Gibt Informationen über eine Absicht zurück. Zusätzlich zum Namen der Absicht müssen Sie die Version der Absicht angeben.

Diese Operation erfordert zur Ausführung der `lex:GetIntent`-Aktion Berechtigungen.

Anforderungssyntax

```
GET /intents/name/versions/version HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name der Absicht. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

version

Die Version der Absicht.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "rejectionStatement": {
         "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      }
   },
   "fulfillmentActivity": {
      "codeHook": {
         "messageVersion": "string",
         "uri": "string"
      },
      "type": "string"
   },
   "inputContexts": [
      {
         "name": "string"
      }
   ],
   "kendraConfiguration": {
      "kendraIndex": "string",
      "queryFilterString": "string",
      "role": "string"
   },
   "lastUpdatedDate": number,
   "name": "string",
   "outputContexts": [
      {
         "name": "string",
         "timeToLiveInSeconds": number,
         "turnsToLive": number
      }
   ],
   "parentIntentSignature": "string",
   "rejectionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
```

```
      "responseCard": "string"
   },
   "sampleUtterances": [ "string" ],
   "slots": [
      {
         "defaultValueSpec": {
            "defaultValueList": [
               {
                  "defaultValue": "string"
               }
            ]
         },
         "description": "string",
         "name": "string",
         "obfuscationSetting": "string",
         "priority": number,
         "responseCard": "string",
         "sampleUtterances": [ "string" ],
         "slotConstraint": "string",
         "slotType": "string",
         "slotTypeVersion": "string",
         "valueElicitationPrompt": {
            "maxAttempts": number,
            "messages": [
               {
                  "content": "string",
                  "contentType": "string",
                  "groupNumber": number
               }
            ],
            "responseCard": "string"
         }
      }
   ],
   "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

checksum

Prüfsumme der Absicht.

Typ: Zeichenfolge

conclusionStatement

Nachdem die im `fulfillmentActivity` Element angegebene Lambda-Funktion die Absicht erfüllt hat, übermittelt Amazon Lex diese Aussage an den Benutzer.

Typ: Statement Objekt

confirmationPrompt

Falls im Bot definiert, verwendet Amazon Lex die Aufforderung, um die Absicht zu bestätigen, bevor die Anfrage des Benutzers bearbeitet wird. Weitere Informationen finden Sie unter PutIntent.

Typ: Prompt Objekt

createdDate

Das Datum, an dem die Absicht erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

dialogCodeHook

Falls im Bot definiert, ruft Amazon Amazon Lex diese Lambda-Funktion für jede Benutzereingabe auf. Weitere Informationen finden Sie unter PutIntent.

Typ: CodeHook Objekt

followUpPrompt

Falls im Bot definiert, verwendet Amazon Lex diese Aufforderung, um zusätzliche Benutzeraktivitäten anzufordern, nachdem die Absicht erfüllt wurde. Weitere Informationen finden Sie unter PutIntent.

Typ: FollowUpPrompt Objekt

fulfillmentActivity

Beschreibt, wie die Absicht erfüllt wird. Weitere Informationen finden Sie unter PutIntent.

Typ: FulfillmentActivity Objekt

inputContexts

Ein Array von `InputContext` Objekten, das die Kontexte auflistet, die aktiv sein müssen, damit Amazon Lex die Absicht in einer Konversation mit dem Benutzer auswählen kann.

Typ: Array von InputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 5 Elemente.

kendraConfiguration

Konfigurationsinformationen, falls vorhanden, um mit der `AMAZON.KendraSearchIntent` Absicht eine Verbindung zu Amazon Kendra Kendra-Index herzustellen.

Typ: KendraConfiguration Objekt

lastUpdatedDate

Das Datum, an dem die Absicht aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

outputContexts

Eine Reihe von `OutputContext` Objekten, die die Kontexte auflistet, die die Absicht aktiviert, wenn die Absicht erfüllt ist.

Typ: Array von OutputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

parentIntentSignature

Ein eindeutiger Bezeichner für eine integrierte Absicht.

Typ: Zeichenfolge

rejectionStatement

Wenn der Benutzer die in definierte Frage mit „Nein“ beantwortet`confirmationPrompt`, antwortet Amazon Lex mit dieser Erklärung, um zu bestätigen, dass die Absicht storniert wurde.

Typ: Statement Objekt

sampleUtterances

Eine Reihe von Beispieläußerungen, die für die Absicht konfiguriert wurden.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 1500 Elementen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 200 Zeichen.

slots

Eine Reihe von Intent-Slots, die für die Absicht konfiguriert sind.

Typ: Array von Slot-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 100 Elemente.

version

Die Version der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht wohlformuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python

- AWS SDK for Ruby V3

GetIntents

Service: Amazon Lex Model Building Service

Gibt Informationen zur Absicht wie folgt zurück:

- Wenn Sie das `nameContains` Feld angeben, wird die `$LATEST` Version aller Absichten zurückgegeben, die die angegebene Zeichenfolge enthalten.
- Wenn Sie das `nameContains` Feld nicht angeben, werden Informationen über die `$LATEST` Version aller Absichten zurückgegeben.

Für den Vorgang ist eine Genehmigung für die `lex:GetIntents` Aktion erforderlich.

Anforderungssyntax

```
GET /intents/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Absichten, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nameContains

Teilzeichenfolge, der in den Absichtsnamen zugeordnet werden soll. Eine Absicht wird zurückgegeben, wenn ein Teil ihres Namens mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nextToken

Ein Paginierungstoken, das die nächste Seite mit Absichten abruft. Wenn die Antwort auf diesen API-Aufruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die

nächste Seite mit Absichten abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

intents

Ein Array von `Intent`-Objekten. Weitere Informationen finden Sie unter PutBot.

Typ: Array von IntentMetadata-Objekten

nextToken

Wenn die Antwort gekürzt ist, enthält die Antwort ein Paginierungstoken, das Sie in Ihrer nächsten Anfrage angeben können, um die nächste Seite mit Absichten abzurufen.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python

- AWS SDK for Ruby V3

GetIntentVersions

Service: Amazon Lex Model Building Service

Ruft Informationen über alle Versionen einer Absicht ab.

Die `GetIntentVersions` Operation gibt ein `IntentMetadata` Objekt für jede Version einer Absicht zurück. Wenn eine Absicht beispielsweise drei nummerierte Versionen hat, gibt die `GetIntentVersions` Operation vier `IntentMetadata` Objekte in der Antwort zurück, eines für jede nummerierte Version und eines für die `$LATEST` Version.

Die `GetIntentVersions` Operation gibt immer mindestens eine Version zurück, die `$LATEST` Version.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetIntentVersions`.

Anforderungssyntax

```
GET /intents/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Intent-Versionen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

name

Der Name der Absicht, für die Versionen zurückgegeben werden sollen.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Intent-Versionen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um

die nächste Seite mit Versionen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

intents

Ein Array von `IntentMetadata` Objekten, eines für jede nummerierte Version der Absicht und eines für die `$LATEST` Version.

Typ: Array von IntentMetadata-Objekten

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Intent-Versionen. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um

die nächste Seite mit Versionen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2

- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetMigration

Service: Amazon Lex Model Building Service

Enthält Details zu einer laufenden oder vollständigen Migration von einem Amazon Lex V1-Bot zu einem Amazon Lex V2-Bot. Verwenden Sie diesen Vorgang, um die Migrationswarnungen und -warnungen im Zusammenhang mit der Migration anzuzeigen.

Anforderungssyntax

```
GET /migrations/migrationId HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

migrationId

Die eindeutige Kennung der Migration, die angezeigt werden soll. Der `migrationID` wird von der StartMigration Operation zurückgegeben.

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "alerts": [
      {
         "details": [ "string" ],
         "message": "string",
```

```
            "referenceURLs": [ "string" ],
            "type": "string"
        }
    ],
    "migrationId": "string",
    "migrationStatus": "string",
    "migrationStrategy": "string",
    "migrationTimestamp": number,
    "v1BotLocale": "string",
    "v1BotName": "string",
    "v1BotVersion": "string",
    "v2BotId": "string",
    "v2BotRole": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

alerts

Eine Liste von Warnungen und Warnungen, die auf Probleme bei der Migration des Amazon Lex V1-Bot zu Amazon Lex V2 hinweisen. Sie erhalten eine Warnung, wenn eine Amazon Lex V1-Funktion in Amazon Lex V2 anders implementiert ist.

Weitere Informationen finden Sie unter Einen Bot migrieren im Amazon Lex V2-Entwicklerhandbuch.

Typ: Array von MigrationAlert-Objekten

migrationId

Die eindeutige Kennung der Migration. Dies ist derselbe wie der Bezeichner, der beim Aufrufen der `GetMigration` Operation verwendet wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

migrationStatus

Zeigt den Status der Migration an. Wenn der Status lautet, ist `COMPLETE` die Migration abgeschlossen und der Bot ist in Amazon Lex V2 verfügbar. Möglicherweise gibt es Warnungen und Warnungen, die behoben werden müssen, um die Migration abzuschließen.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | COMPLETED | FAILED`

migrationStrategy

Die Strategie, mit der die Migration durchgeführt wurde.

- `CREATE_NEW`- Erstellt einen neuen Amazon Lex V2-Bot und migriert den Amazon Lex V1-Bot auf den neuen Bot.
- `UPDATE_EXISTING`- Überschreibt die vorhandenen Amazon Lex V2-Bot-Metadaten und das zu migrierende Gebietsschema. Es ändert keine anderen Gebietsschemas im Amazon Lex V2-Bot. Wenn das Gebietsschema nicht existiert, wird ein neues Gebietsschema im Amazon Lex V2-Bot erstellt.

Typ: Zeichenfolge

Zulässige Werte: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

Datum und Uhrzeit des Beginns der Migration.

Typ: Zeitstempel

v1BotLocale

Das Gebietsschema des Amazon Lex V1-Bots wurde auf Amazon Lex V2 migriert.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

v1BotName

Der Name des Amazon Lex V1-Bots, der auf Amazon Lex V2 migriert wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

v1BotVersion

Die Version des Amazon Lex V1-Bots wurde auf Amazon Lex V2 migriert.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

v2BotId

Die eindeutige Kennung des Amazon Lex V2-Bots, auf den Amazon Lex V1 migriert wird.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

v2BotRole

Die IAM-Rolle, die Amazon Lex verwendet, um den Amazon Lex V2-Bot auszuführen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetMigrations

Service: Amazon Lex Model Building Service

Ruft eine Liste der Migrationen zwischen Amazon Lex V1 und Amazon Lex V2 ab.

Anforderungssyntax

```
GET /migrations?
maxResults=maxResults&migrationStatusEquals=migrationStatusEquals&nextToken=nextToken&sortByAtt
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Migrationen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

migrationStatusEquals

Filtert die Liste so, dass sie nur Migrationen im angegebenen Status enthält.

Zulässige Werte: `IN_PROGRESS | COMPLETED | FAILED`

nextToken

Ein Paginierungstoken, das die nächste Seite mit Migrationen abruft. Wenn die Antwort auf diesen Vorgang gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Migrationen abzurufen, geben Sie das Paginierungstoken in der Anfrage an.

sortByAttribute

Das Feld, nach dem die Liste der Migrationen sortiert werden soll. Sie können nach dem Namen des Amazon Lex V1-Bots oder dem Datum und der Uhrzeit des Starts der Migration sortieren.

Zulässige Werte: `V1_BOT_NAME | MIGRATION_DATE_TIME`

sortByOrder

Die Reihenfolge, in der die Liste sortiert wird.

Zulässige Werte: `ASCENDING | DESCENDING`

v1BotNameContains

Filtert die Liste so, dass sie nur Bots enthält, deren Name die angegebene Zeichenfolge enthält. Die Zeichenfolge entspricht einer beliebigen Stelle im Bot-Namen.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "migrationSummaries": [
      {
         "migrationId": "string",
         "migrationStatus": "string",
         "migrationStrategy": "string",
         "migrationTimestamp": number,
         "v1BotLocale": "string",
         "v1BotName": "string",
         "v1BotVersion": "string",
         "v2BotId": "string",
         "v2BotRole": "string"
      }
   ],
   "nextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

migrationSummaries

Eine Reihe von Zusammenfassungen für Migrationen von Amazon Lex V1 zu Amazon Lex V2. Einzelheiten der Migration finden Sie in der `migrationId` Zusammenfassung in einem Aufruf des Vorgangs. GetMigration

Typ: Array von MigrationSummary-Objekten

nextToken

Wenn die Antwort gekürzt ist, enthält sie ein Paginierungstoken, das Sie in Ihrer nächsten Anfrage angeben können, um die nächste Seite mit Migrationen abzurufen.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetSlotType

Service: Amazon Lex Model Building Service

Gibt Informationen hinsichtlich einer bestimmten Version eines Slot-Typs zurück. Zusätzlich zur Angabe des Slot-Typ-Namens müssen Sie die Slot-Typ-Version angeben.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetSlotType`.

Anforderungssyntax

```
GET /slottypes/name/versions/version HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Slot-Typs. Der Name berücksichtigt Groß- und Kleinschreibung.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

version

Die Version des Steckplatztyps.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "checksum": "string",
   "createdDate": number,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

checksum

Prüfsumme der $LATEST Version des Slot-Typs.

Typ: Zeichenfolge

createdDate

Das Datum, an dem der Slot-Typ erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

enumerationValues

Eine Liste von `EnumerationValue` Objekten, die die Werte definiert, die der Slot-Typ annehmen kann.

Typ: Array von EnumerationValue-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 10000 Elementen.

lastUpdatedDate

Das Datum, an dem der Slot-Typ aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Der integrierte Steckplatztyp, der als übergeordnetes Element für den Steckplatztyp verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Konfigurationsinformationen, die den Typ des übergeordneten integrierten Steckplatzes erweitern.

Typ: Array von SlotTypeConfiguration-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

valueSelectionStrategy

Die Strategie, mit der Amazon Lex den Wert des Slots bestimmt. Weitere Informationen finden Sie unter PutSlotType.

Typ: Zeichenfolge

Zulässige Werte: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

Die Version des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetSlotTypes

Service: Amazon Lex Model Building Service

Gibt Informationen zum Slot-Typ wie folgt zurück:

- Wenn Sie das `nameContains` Feld angeben, wird die `$LATEST` Version aller Slot-Typen zurückgegeben, die die angegebene Zeichenfolge enthalten.
- Wenn Sie das `nameContains` Feld nicht angeben, werden Informationen über die `$LATEST` Version aller Slot-Typen zurückgegeben.

Für den Vorgang ist eine Genehmigung für die `lex:GetSlotTypes` Aktion erforderlich.

Anforderungssyntax

```
GET /slottypes/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Slot-Typen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

nameContains

Teilzeichenfolge, die in den Namen der Slot-Typen übereinstimmen soll. Ein Slot-Typ wird zurückgegeben, wenn ein Teil seines Namens mit der Teilzeichenfolge übereinstimmt. Zum Beispiel entspricht „xyz“ sowohl „xyzabc“ als auch „abcxyz“.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nextToken

Ein Paginierungstoken, das die nächste Seite mit Slot-Typen abruft. Wenn die Antwort auf diesen API-Aufruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um

die nächste Seite mit Slot-Typen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

nextToken

Wenn die Antwort gekürzt ist, enthält sie ein Paginierungstoken, das Sie in Ihrer nächsten Anfrage angeben können, um die nächste Seite mit Slot-Typen abzurufen.

Typ: Zeichenfolge

slotTypes

Eine Reihe von Objekten, eines für jeden Slot-Typ, das Informationen wie den Namen des Slot-Typs, die Version und eine Beschreibung bereitstellt.

Typ: Array von SlotTypeMetadata-Objekten

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3

- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetSlotTypeVersions

Service: Amazon Lex Model Building Service

Ruft Informationen über alle Versionen eines Slot-Typs ab.

Die `GetSlotTypeVersions` Operation gibt ein `SlotTypeMetadata` Objekt für jede Version eines Slot-Typs zurück. Wenn ein Slot-Typ beispielsweise drei nummerierte Versionen hat, gibt die `GetSlotTypeVersions` Operation vier `SlotTypeMetadata` Objekte in der Antwort zurück, eines für jede nummerierte Version und eines für die `$LATEST` Version.

Die `GetSlotTypeVersions` Operation gibt immer mindestens eine Version zurück, die `$LATEST` Version.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetSlotTypeVersions`.

Anforderungssyntax

```
GET /slottypes/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

maxResults

Die maximale Anzahl von Slot-Typ-Versionen, die in der Antwort zurückgegeben werden sollen. Der Standardwert ist 10.

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 50.

name

Der Name des Slot-Typs, für den Versionen zurückgegeben werden sollen.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Versionen vom Typ Slot. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken

zurück. Um die nächste Seite mit Versionen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

nextToken

Ein Paginierungstoken zum Abrufen der nächsten Seite mit Versionen vom Typ Slot. Wenn die Antwort auf diesen Anruf gekürzt wird, gibt Amazon Lex in der Antwort ein Paginierungstoken zurück. Um die nächste Seite mit Versionen abzurufen, geben Sie das Paginierungstoken in der nächsten Anfrage an.

Typ: Zeichenfolge

slotTypes

Eine Reihe von `SlotTypeMetadata` Objekten, eines für jede nummerierte Version des Slot-Typs plus eines für die `$LATEST` Version.

Typ: Array von SlotTypeMetadata-Objekten

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetUtterancesView

Service: Amazon Lex Model Building Service

Verwenden Sie die `GetUtterancesView` Operation, um Informationen über die Äußerungen zu erhalten, die Ihre Benutzer gegenüber Ihrem Bot gemacht haben. Sie können diese Liste verwenden, um die Äußerungen zu optimieren, auf die Ihr Bot reagiert.

Nehmen wir zum Beispiel an, dass Sie einen Bot erstellt haben, um Blumen zu bestellen. Nachdem Ihre Benutzer Ihren Bot eine Weile verwendet haben, können Sie mithilfe des `GetUtterancesView` Vorgangs überprüfen, welche Anfragen sie gestellt haben und ob sie erfolgreich waren. Möglicherweise stellen Sie fest, dass die Äußerung „Ich möchte Blumen“ nicht erkannt wird. Sie könnten diese Äußerung zur `OrderFlowers` Absicht hinzufügen, sodass Ihr Bot diese Äußerung erkennt.

Nachdem Sie eine neue Version eines Bots veröffentlicht haben, können Sie Informationen über die alte und die neue Version abrufen, sodass Sie die Leistung der beiden Versionen vergleichen können.

Statistiken zu den Äußerungen werden einmal täglich erstellt. Daten sind für die letzten 15 Tage verfügbar. Sie können in jeder Anfrage Informationen für bis zu 5 Versionen Ihres Bots anfordern. Amazon Lex gibt die häufigsten Äußerungen zurück, die der Bot in den letzten 15 Tagen erhalten hat. Die Antwort enthält Informationen über maximal 100 Äußerungen für jede Version.

Unter den folgenden Bedingungen werden keine Statistiken zu Äußerungen generiert:

- Das `childDirected` Feld wurde auf „true“ gesetzt, als der Bot erstellt wurde.
- Sie verwenden die Slot-Obfuscation mit einem oder mehreren Slots.
- Sie haben sich von der Teilnahme an der Verbesserung von Amazon Lex abgemeldet.

Diese Operation erfordert Berechtigungen für die Aktion `lex:GetUtterancesView`.

Anforderungssyntax

```
GET /bots/botname/utterances?
view=aggregation&bot_versions=botVersions&status_type=statusType HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botname

Der Name des Bots, für den Informationen zu Äußerungen zurückgegeben werden sollen.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

botVersions

Eine Reihe von Bot-Versionen, für die Informationen zu Äußerungen zurückgegeben werden sollen. Das Limit liegt bei 5 Versionen pro Anfrage.

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 5 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Ja

statusType

Um Äußerungen zurückzugeben, die erkannt und verarbeitet wurden, verwenden Sie`Detected`. Um Äußerungen zurückzugeben, die nicht erkannt wurden, verwenden Sie. `Missed`

Zulässige Werte: `Detected | Missed`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "botName": "string",
   "utterances": [
      {
         "botVersion": "string",
```

```
         "utterances": [
            {
               "count": number,
               "distinctUsers": number,
               "firstUtteredDate": number,
               "lastUtteredDate": number,
               "utteranceString": "string"
            }
         ]
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

botName

Der Name des Bots, für den Informationen zu Äußerungen zurückgegeben wurden.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

utterances

Eine Reihe von UtteranceList Objekten, von denen jedes eine Liste von UtteranceData Objekten enthält, die die Äußerungen beschreiben, die von Ihrem Bot verarbeitet wurden. Die Antwort enthält maximal 100 `UtteranceData` Objekte für jede Version. Amazon Lex gibt die häufigsten Äußerungen zurück, die der Bot in den letzten 15 Tagen erhalten hat.

Typ: Array von UtteranceList-Objekten

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

ListTagsForResource

Service: Amazon Lex Model Building Service

Ruft eine Liste von Tags ab, die der angegebenen Ressource zugeordnet sind. Nur Bots, Bot-Aliasnamen und Bot-Kanälen können Tags zugeordnet werden.

Anforderungssyntax

```
GET /tags/resourceArn HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

resourceArn

Der Amazon-Ressourcenname (ARN) der Ressource, für die eine Liste von Tags abgerufen werden soll.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1011.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

tags

Die mit einer Ressource verknüpften Tags.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PutBot

Service: Amazon Lex Model Building Service

Erzeugt einen Amazon Lex Lex-Konversationsbot oder ersetzt einen vorhandenen Bot. Wenn Sie einen Bot erstellen oder aktualisieren, müssen Sie nur einen Namen und ein Gebietsschema angeben und angeben, ob sich der Bot an Kinder unter 13 Jahren richtet. Sie können dies verwenden, um später Absichten hinzuzufügen oder um Absichten aus einem vorhandenen Bot zu entfernen. Wenn Sie einen Bot mit den Mindestinformationen erstellen, wird der Bot erstellt oder aktualisiert, aber Amazon Lex gibt die Antwort zurück`FAILED`. Sie können den Bot erstellen, nachdem Sie eine oder mehrere Absichten hinzugefügt haben. Weitere Informationen zu Amazon Lex-Bots finden Sie unterAmazon Lex — Funktionsweise.

Wenn Sie den Namen eines vorhandenen Bots angeben, ersetzen die Felder in der Anfrage die vorhandenen Werte in der `$LATEST` Version des Bots. Amazon Lex entfernt alle Felder, für die Sie in der Anfrage keine Werte angeben, mit Ausnahme der `privacySettings` Felder `idleTTLInSeconds` und, die auf ihre Standardwerte festgelegt sind. Wenn Sie keine Werte für Pflichtfelder angeben, löst Amazon Lex eine Ausnahme aus.

Diese Operation erfordert Berechtigungen für die Aktion `lex:PutBot`. Weitere Informationen finden Sie unter Identity and Access Management für Amazon Lex.

Anforderungssyntax

```
PUT /bots/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
```

```
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createVersion": boolean,
   "description": "string",
   "detectSentiment": boolean,
   "enableModelImprovements": boolean,
   "idleSessionTTLInSeconds": number,
   "intents": [
      {
         "intentName": "string",
         "intentVersion": "string"
      }
   ],
   "locale": "string",
   "nluIntentConfidenceThreshold": number,
   "processBehavior": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ],
   "voiceId": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Bots. Beim Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

abortStatement

Wenn Amazon Lex die Eingabe des Benutzers im Kontext nicht verstehen kann, versucht es einige Male, die Informationen abzurufen. Danach sendet Amazon Lex die in definierte Nachricht `abortStatement` an den Benutzer und bricht dann die Konversation ab. Verwenden Sie das `valueElicitationPrompt` Feld für den Slot-Typ, um die Anzahl der Wiederholungen festzulegen.

In einem Pizza-Bestellbot könnte Amazon Lex einen Benutzer beispielsweise fragen: „Welche Art von Kruste hätten Sie gerne?“ Wenn die Antwort des Benutzers nicht zu den erwarteten Antworten gehört (z. B. „dünne Kruste“, „tiefe Schale“ usw.), versucht Amazon Lex noch einige Male, eine korrekte Antwort auszulösen.

In einer Anwendung zur Bestellung von Pizza `OrderPizza` könnte dies beispielsweise eine der Absichten sein. Für diese Absicht ist möglicherweise der `CrustType` Slot erforderlich. Sie geben das `valueElicitationPrompt` Feld an, wenn Sie den `CrustType` Slot erstellen.

Wenn Sie eine Fallback-Absicht definiert haben, wird die Stornierungsanweisung nicht an den Benutzer gesendet, sondern die Fallback-Absicht wird verwendet. Weitere Informationen finden Sie unter AMAZON. FallbackIntent.

Typ: Statement Objekt

Erforderlich: Nein

checksum

Identifiziert eine bestimmte Version der `$LATEST` Version.

Wenn Sie einen neuen Bot erstellen, lassen Sie das `checksum` Feld leer. Wenn Sie eine Prüfsumme angeben, erhalten Sie eine `BadRequestException` Ausnahme.

Wenn Sie einen Bot aktualisieren möchten, setzen Sie das `checksum` Feld auf die Prüfsumme der letzten Version der `$LATEST` Version. Wenn Sie das `checksum` Feld nicht angeben

oder wenn die Prüfsumme nicht mit der `$LATEST` Version übereinstimmt, erhalten Sie eine `PreconditionFailedException` Ausnahme.

Typ: Zeichenfolge

Erforderlich: Nein

childDirected

Für jeden Amazon Lex-Bot, der mit dem Amazon Lex Model Building Service erstellt wurde, müssen Sie angeben, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und dem Gesetz zum Schutz der Privatsphäre von Kindern im Internet (Children's Online Privacy Protection Act, COPPA) unterliegt, indem Sie `true` oder `false` im `childDirected` Feld angeben. Durch die Angabe **`true`** in **`childDirected`** diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Durch die Angabe `false` in `childDirected` diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex nicht mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Sie dürfen keinen Standardwert für das `childDirected` Feld angeben, der nicht genau wiedergibt, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt.

Wenn sich Ihre Nutzung von Amazon Lex auf eine Website, ein Programm oder eine andere Anwendung bezieht, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet, müssen Sie die erforderliche nachprüfbare Zustimmung der Eltern gemäß COPPA einholen. Informationen zur Verwendung von Amazon Lex in Verbindung mit Websites, Programmen oder anderen Anwendungen, die sich ganz oder teilweise an Kinder unter 13 Jahren richten oder richten, finden Sie in den häufig gestellten Fragen zu Amazon Lex.

Typ: Boolesch

Erforderlich: Ja

clarificationPrompt

Wenn Amazon Lex die Absicht des Benutzers nicht versteht, verwendet es diese Nachricht, um eine Klarstellung zu erhalten. Verwenden Sie das `maxAttempts` Feld, um anzugeben, wie oft

Amazon Lex die Aufforderung zur Klärung wiederholen soll. Wenn Amazon Lex es immer noch nicht versteht, sendet es die Nachricht vor `abortStatement` Ort.

Wenn Sie eine Klarstellungsaufforderung erstellen, stellen Sie sicher, dass sie die richtige Antwort des Benutzers vorschlägt. Beispielsweise könnten Sie für einen Bot, der Pizza und Getränke bestellt, diese Klarstellungsaufforderung erstellen: „Was möchten Sie tun? Du kannst „Pizza bestellen“ oder „Ein Getränk bestellen“ sagen.“

Wenn Sie eine Fallback-Absicht definiert haben, wird diese aufgerufen, wenn die Klarstellungsaufforderung so oft wiederholt wird, wie im Feld angegeben. `maxAttempts` Weitere Informationen finden Sie unter AMAZON. FallbackIntent.

Wenn Sie keine Klarstellungsaufforderung definieren, gibt Amazon Lex zur Laufzeit in drei Fällen eine 400 Bad Request-Ausnahme zurück:

- Aufforderung zur Nachverfolgung — Wenn der Benutzer auf eine Folgeaufforderung reagiert, aber keine Absicht angibt. Zum Beispiel als Antwort auf eine Folgeaufforderung mit der Aufschrift „Möchten Sie heute noch etwas anderes?“ der Benutzer sagt „Ja“. Amazon Lex gibt die Ausnahme 400 Bad Request zurück, da es keine Klarstellungsaufforderung gibt, die an den Benutzer gesendet werden muss, um eine Absicht zu erhalten.
- Lambda-Funktion — Wenn Sie eine Lambda-Funktion verwenden, geben Sie einen `ElicitIntent` Dialogtyp zurück. Da Amazon Lex keine Klarstellungsaufforderung hat, um eine Absicht des Benutzers zu erhalten, gibt es eine 400 Bad Request-Ausnahme zurück.
- PutSession Operation — Wenn Sie die `PutSession` Operation verwenden, senden Sie einen `ElicitIntent` Dialogtyp. Da Amazon Lex keine Klarstellungsaufforderung hat, um eine Absicht des Benutzers zu erhalten, gibt es eine 400 Bad Request-Ausnahme zurück.

Typ: Prompt Objekt

Erforderlich: Nein

createVersion

Bei Einstellung auf `true` eine neue nummerierte Version wird der Bot erstellt. Dies entspricht dem Aufrufen der `CreateBotVersion` Operation. Wenn Sie nichts angeben`createVersion`, ist die Standardeinstellung`false`.

Typ: Boolesch

Erforderlich: Nein

description

Eine Beschreibung des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

detectSentiment

Wenn diese Option aktiviert ist, werden `true` Benutzeräußerungen zur Stimmungsanalyse an Amazon Comprehend gesendet. Wenn Sie nichts angeben`detectSentiment`, ist die Standardeinstellung. `false`

Typ: Boolesch

Erforderlich: Nein

enableModelImprovements

Wird auf gesetzt, `true` um den Zugriff auf Verbesserungen des Verständnisses natürlicher Sprache zu ermöglichen.

Wenn Sie den `enableModelImprovements` Parameter auf einstellen, können `true` Sie den `nluIntentConfidenceThreshold` Parameter verwenden, um Konfidenzwerte zu konfigurieren. Weitere Informationen finden Sie unter Konfidenzwerte.

Sie können den `enableModelImprovements` Parameter nur in bestimmten Regionen festlegen. Wenn Sie den Parameter auf setzen`true`, hat Ihr Bot Zugriff auf Genauigkeitsverbesserungen.

Die Regionen, auf die Sie den `enableModelImprovements` Parameter `false` für das Gebietsschema en-US setzen können, sind:

- USA Ost (Nord-Virginia): (us-east-1)
- USA West (Oregon): (us-west-2)
- Asien-Pazifik (Sydney): (ap-southeast-2)
- EU (Irland) (eu-west-1)

In anderen Regionen und Gebietsschemas ist der `enableModelImprovements` Parameter standardmäßig auf `true` eingestellt. In diesen Regionen und Gebietsschemas `false` löst die Einstellung des Parameters auf eine Ausnahme aus. `ValidationException`

Typ: Boolesch

Erforderlich: Nein

idleSessionTTLInSeconds

Die maximale Zeit in Sekunden, für die Amazon Lex die in einer Konversation gesammelten Daten aufbewahrt.

Eine Benutzerinteraktionssitzung bleibt für den angegebenen Zeitraum aktiv. Wenn während dieser Zeit kein Gespräch stattfindet, läuft die Sitzung ab und Amazon Lex löscht alle Daten, die vor dem Timeout bereitgestellt wurden.

Nehmen wir zum Beispiel an, ein Benutzer wählt die OrderPizza Absicht, wird aber nach der Hälfte der Bestellung abgelenkt. Wenn der Benutzer die Bestellung nicht innerhalb der angegebenen Zeit abschließt, verwirft Amazon Lex die gesammelten Slot-Informationen und der Benutzer muss von vorne beginnen.

Wenn Sie das `idleSessionTTLInSeconds` Element nicht in eine `PutBot` Betriebsanforderung aufnehmen, verwendet Amazon Lex den Standardwert. Dies gilt auch, wenn die Anfrage einen vorhandenen Bot ersetzt.

Der Standardwert ist 300 Sekunden (5 Minuten).

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 60. Maximaler Wert von 86400.

Erforderlich: Nein

intents

Ein Array von `Intent`-Objekten. Jede Absicht steht für einen Befehl, den ein Benutzer ausdrücken kann. Beispielsweise könnte ein Bot für die Pizzabestellung eine OrderPizza Absicht unterstützen. Weitere Informationen finden Sie unter Amazon Lex — Funktionsweise.

Typ: Array von Intent-Objekten

Erforderlich: Nein

locale

Gibt das Zielgebietsschema für den Bot an. Jede im Bot verwendete Absicht muss mit dem Gebietsschema des Bots kompatibel sein.

Der Standardwert ist `en-US`.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Erforderlich: Ja

nluIntentConfidenceThreshold

Legt den Schwellenwert fest`AMAZON.FallbackIntent`, `AMAZON.KendraSearchIntent` an dem Amazon Lex bei der Rückgabe alternativer Absichten in einer PostTextOder-Antwort die Werte PostContentoder beides einfügt. `AMAZON.FallbackIntent`und `AMAZON.KendraSearchIntent` werden nur eingefügt, wenn sie für den Bot konfiguriert sind.

Sie müssen den `enableModelImprovements` Parameter auf einstellen, `true` um Konfidenzwerte in den folgenden Regionen verwenden zu können.

- USA Ost (Nord-Virginia): (us-east-1)
- USA West (Oregon): (us-west-2)
- Asien-Pazifik (Sydney): (ap-southeast-2)
- EU (Irland) (eu-west-1)

In anderen Regionen ist der `enableModelImprovements` Parameter `true` standardmäßig auf eingestellt.

Nehmen wir zum Beispiel an, ein Bot ist mit einem Konfidenzschwellenwert von 0,80 konfiguriert und der`AMAZON.FallbackIntent`. Amazon Lex gibt drei alternative Absichten mit den folgenden Konfidenzwerten zurück: IntentA (0,70), IntentB (0,60), IntentC (0,50). Die Antwort des Vorgangs wäre: `PostText`

- AMAZON. FallbackIntent
- In Tenta
- Absicht B
- Absicht C

Type: Double

Gültiger Bereich: Mindestwert 0. Maximalwert von 1.

Erforderlich: Nein

processBehavior

Wenn Sie das `processBehavior` Element auf setzen`BUILD`, erstellt Amazon Lex den Bot so, dass er ausgeführt werden kann. Wenn Sie das Element auf `SAVE` Amazon Lex setzen, speichert Lex den Bot, erstellt ihn aber nicht.

Wenn Sie diesen Wert nicht angeben, ist der Standardwert`BUILD`.

Typ: Zeichenfolge

Zulässige Werte: `SAVE | BUILD`

Erforderlich: Nein

tags

Eine Liste der Tags, die dem Bot hinzugefügt werden sollen. Sie können Tags nur hinzufügen, wenn Sie einen Bot erstellen. Sie können den `PutBot` Vorgang nicht verwenden, um die Tags auf einem Bot zu aktualisieren. Um Tags zu aktualisieren, verwenden Sie den `TagResource`-Vorgang.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

voiceId

Die Amazon Polly Polly-Sprach-ID, die Amazon Lex für Sprachinteraktionen mit dem Benutzer verwenden soll. Das für die Stimme konfigurierte Gebietsschema muss mit dem Gebietsschema des Bots übereinstimmen. Weitere Informationen finden Sie unter Voices in Amazon Polly im Amazon Polly Developer Guide.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "createVersion": boolean,
   "description": "string",
   "detectSentiment": boolean,
   "enableModelImprovements": boolean,
   "failureReason": "string",
   "idleSessionTTLInSeconds": number,
   "intents": [
      {
         "intentName": "string",
         "intentVersion": "string"
      }
   ],
   "lastUpdatedDate": number,
   "locale": "string",
   "name": "string",
   "nluIntentConfidenceThreshold": number,
   "status": "string",
   "tags": [
      {
```

```
            "key": "string",
            "value": "string"
        }
    ],
    "version": "string",
    "voiceId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

abortStatement

Die Nachricht, die Amazon Lex verwendet, um eine Konversation abzubrechen. Weitere Informationen finden Sie unter PutBot.

Typ: Statement Objekt

checksum

Prüfsumme des Bots, den Sie erstellt haben.

Typ: Zeichenfolge

childDirected

Für jeden Amazon Lex-Bot, der mit dem Amazon Lex Model Building Service erstellt wurde, müssen Sie angeben, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und dem Gesetz zum Schutz der Privatsphäre von Kindern im Internet (Children's Online Privacy Protection Act, COPPA) unterliegt, indem Sie `true` oder `false` im `childDirected` Feld angeben. Durch die Angabe **`true`** in **`childDirected`** diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Durch die Angabe `false` in `childDirected` diesem Feld bestätigen Sie, dass Ihre Nutzung von Amazon Lex nicht mit einer Website, einem Programm oder einer anderen Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt. Sie dürfen keinen Standardwert für das `childDirected` Feld angeben, der nicht genau wiedergibt, ob Ihre Nutzung von Amazon Lex mit einer Website, einem Programm oder einer anderen

Anwendung zusammenhängt, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet oder darauf abzielt und der COPPA unterliegt.

Wenn sich Ihre Nutzung von Amazon Lex auf eine Website, ein Programm oder eine andere Anwendung bezieht, die sich ganz oder teilweise an Kinder unter 13 Jahren richtet, müssen Sie die erforderliche nachprüfbare Zustimmung der Eltern gemäß COPPA einholen. Informationen zur Verwendung von Amazon Lex in Verbindung mit Websites, Programmen oder anderen Anwendungen, die sich ganz oder teilweise an Kinder unter 13 Jahren richten oder richten, finden Sie in den häufig gestellten Fragen zu Amazon Lex.

Typ: Boolesch

clarificationPrompt

Die Eingabeaufforderungen, die Amazon Lex verwendet, wenn es die Absicht des Benutzers nicht versteht. Weitere Informationen finden Sie unter PutBot.

Typ: Prompt Objekt

createdDate

Das Datum, an dem der Bot erstellt wurde.

Typ: Zeitstempel

createVersion

`True`ob eine neue Version des Bots erstellt wurde. Wenn das `createVersion` Feld in der Anfrage nicht angegeben wurde, wird das `createVersion` Feld in der Antwort auf falsch gesetzt.

Typ: Boolesch

description

Eine Beschreibung des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

detectSentiment

`true`wenn der Bot so konfiguriert ist, dass er Benutzeräußerungen zur Stimmungsanalyse an Amazon Comprehend sendet. Wenn das `detectSentiment` Feld in der Anfrage nicht angegeben wurde, ist `false` das `detectSentiment` Feld in der Antwort enthalten.

Typ: Boolesch

enableModelImprovements

Gibt an, ob der Bot Genauigkeitsverbesserungen verwendet. `true`gibt an, dass der Bot die Verbesserungen verwendet, andernfalls`false`.

Typ: Boolesch

failureReason

Falls `status` ja`FAILED`, gibt Amazon Lex den Grund an, warum der Bot nicht erstellt werden konnte.

Typ: Zeichenfolge

idleSessionTTLInSeconds

Die maximale Dauer, für die Amazon Lex die in einer Konversation gesammelten Daten aufbewahrt. Weitere Informationen finden Sie unter PutBot.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 60. Maximaler Wert von 86400.

intents

Ein Array von `Intent`-Objekten. Weitere Informationen finden Sie unter PutBot.

Typ: Array von Intent-Objekten

lastUpdatedDate

Das Datum, an dem der Bot aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

locale

Das Zielgebietsschema für den Bot.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

Der Name des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

Die Punktzahl, die bestimmt, wo Amazon Lex die oder beide einfügt `AMAZON.FallbackIntent`AMAZON.KendraSearchIntent, wenn alternative Absichten in einer PostContentPostTextOder-Antwort zurückgegeben werden. `AMAZON.FallbackIntent`wird eingefügt, wenn der Konfidenzwert für alle Absichten unter diesem Wert liegt. `AMAZON.KendraSearchIntent`wird nur eingefügt, wenn es für den Bot konfiguriert ist.

Type: Double

Gültiger Bereich: Mindestwert 0. Maximalwert von 1.

status

Wenn Sie eine Anfrage zur Erstellung eines Bots mit der `processBehavior` Einstellung auf sendenBUILD, setzt Amazon Lex das `status` Antwortelement aufBUILDING.

In diesem `READY_BASIC_TESTING` Status können Sie den Bot mit Benutzereingaben testen, die genau den Äußerungen entsprechen, die für die Absichten und Werte des Bots in den Slot-Typen konfiguriert sind.

Wenn Amazon Lex den Bot nicht erstellen kann, setzt Amazon Lex `status` aufFAILED. Amazon Lex gibt den Grund für den Fehler im `failureReason` Antwortelement zurück.

Wenn Sie `processBehavior` auf setzenSAVE, setzt Amazon Lex den Statuscode aufNOT BUILT.

Wenn sich der Bot im `READY` Status befindet, können Sie den Bot testen und veröffentlichen.

Typ: Zeichenfolge

Zulässige Werte: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

tags

Eine Liste von Tags, die dem Bot zugeordnet sind.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

version

Die Version des Bots. Für einen neuen Bot ist die Version immer gültig`$LATEST`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

voiceId

Die Amazon Polly Polly-Sprach-ID, die Amazon Lex für die Sprachinteraktion mit dem Benutzer verwendet. Weitere Informationen finden Sie unter PutBot.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PutBotAlias

Service: Amazon Lex Model Building Service

Erzeugt einen Alias für die angegebene Version des Bots oder ersetzt einen Alias für den angegebenen Bot. Um die Version des Bots zu ändern, auf die der Alias verweist, ersetzen Sie den Alias. Weitere Informationen zu Aliassen finden Sie unter Versioning und Aliasnamen.

Diese Operation erfordert Berechtigungen für die Aktion `lex:PutBotAlias`.

Anforderungssyntax

```
PUT /bots/botName/aliases/name HTTP/1.1
Content-type: application/json

{
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string"
         }
      ]
   },
   "description": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botName

Der Name des Bots.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

name

Der Name des Alias. Beim Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

botVersion

Die Version des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Ja

checksum

Identifiziert eine bestimmte Version der `$LATEST` Version.

Wenn Sie einen neuen Bot-Alias erstellen, lassen Sie das `checksum` Feld leer. Wenn Sie eine Prüfsumme angeben, erhalten Sie eine `BadRequestException` Ausnahme.

Wenn Sie einen Bot-Alias aktualisieren möchten, setzen Sie das `checksum` Feld auf die Prüfsumme der letzten Version der `$LATEST` Version. Wenn Sie das `checksum` Feld nicht

angeben oder wenn die Prüfsumme nicht mit der `$LATEST` Version übereinstimmt, erhalten Sie eine `PreconditionFailedException` Ausnahme.

Typ: Zeichenfolge

Erforderlich: Nein

conversationLogs

Einstellungen für Konversationsprotokolle für den Alias.

Typ: ConversationLogsRequest Objekt

Erforderlich: Nein

description

Eine Beschreibung des Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

tags

Eine Liste von Tags, die dem Bot-Alias hinzugefügt werden sollen. Sie können Tags nur hinzufügen, wenn Sie einen Alias erstellen. Sie können den `PutBotAlias` Vorgang nicht verwenden, um die Tags eines Bot-Alias zu aktualisieren. Um Tags zu aktualisieren, verwenden Sie den `TagResource`-Vorgang.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "botName": "string",
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string",
            "resourcePrefix": "string"
         }
      ]
   },
   "createdDate": number,
   "description": "string",
   "lastUpdatedDate": number,
   "name": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

botName

Der Name des Bots, auf den der Alias zeigt.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

botVersion

Die Version des Bots, auf den der Alias verweist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

checksum

Die Prüfsumme für die aktuelle Version des Alias.

Typ: Zeichenfolge

conversationLogs

Die Einstellungen, die bestimmen, wie Amazon Lex Konversationsprotokolle für den Alias verwendet.

Typ: ConversationLogsResponse Objekt

createdDate

Das Datum, an dem der Bot-Alias erstellt wurde.

Typ: Zeitstempel

description

Eine Beschreibung des Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

lastUpdatedDate

Das Datum, an dem der Bot-Alias aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name des Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

tags

Eine Liste von Tags, die einem Bot zugeordnet sind.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Fehler

BadRequestException

Die Anfrage ist nicht wohlformuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PutIntent

Service: Amazon Lex Model Building Service

Erstellt eine Absicht oder ersetzt eine vorhandene Absicht.

Um die Interaktion zwischen dem Benutzer und Ihrem Bot zu definieren, verwenden Sie eine oder mehrere Absichten. Für einen Pizza-Bestell-Bot würdest du beispielsweise eine `OrderPizza` Absicht erstellen.

Um eine Absicht zu erstellen oder eine bestehende Absicht zu ersetzen, müssen Sie Folgendes angeben:

- Name der Absicht. z. B. `OrderPizza`.
- Beispiele für Äußerungen. Zum Beispiel: „Kann ich bitte eine Pizza bestellen?“ und „Ich möchte eine Pizza bestellen.“
- Zu sammelnde Informationen. Sie geben Slot-Typen für die Informationen an, die Ihr Bot vom Benutzer anfordert. Sie können Standard-Slottypen wie ein Datum oder eine Uhrzeit oder benutzerdefinierte Slot-Typen wie Größe und Kruste einer Pizza angeben.
- Wie die Absicht erfüllt wird. Sie können eine Lambda-Funktion bereitstellen oder die Absicht so konfigurieren, dass die Absichtsinformationen an die Client-Anwendung zurückgegeben werden. Wenn Sie eine Lambda-Funktion verwenden und alle Informationen zur Absicht verfügbar sind, ruft Amazon Lex Ihre Lambda-Funktion auf. Wenn Sie Ihre Absicht so konfigurieren, dass die Informationen zur Absicht an die Client-Anwendung zurückgegeben werden.

Sie können in der Anfrage weitere optionale Informationen angeben, z. B.:

- Eine Bestätigungsaufforderung, in der der Benutzer aufgefordert wird, eine Absicht zu bestätigen. Zum Beispiel: „Soll ich deine Pizza bestellen?“
- Eine Abschlusserklärung, die an den Benutzer gesendet wird, nachdem die Absicht erfüllt wurde. Zum Beispiel: „Ich habe Ihre Pizzabestellung aufgegeben.“
- Eine Folgeaufforderung, in der der Benutzer um zusätzliche Aktivitäten gebeten wird. Fragen Sie beispielsweise: „Möchten Sie zu Ihrer Pizza ein Getränk bestellen?“

Wenn Sie einen vorhandenen Absichtsnamen angeben, um die Absicht zu aktualisieren, ersetzt Amazon Lex die Werte in der `$LATEST` Version der Absicht durch die Werte in der Anfrage. Amazon Lex entfernt Felder, die Sie in der Anfrage nicht angeben. Wenn Sie die erforderlichen Felder nicht angeben, löst Amazon Lex eine Ausnahme aus. Wenn Sie die `$LATEST` Version einer Absicht

aktualisieren, wird das `status` Feld jedes Bots, der die `$LATEST` Version der Absicht verwendet, auf `NOT_BUILT` gesetzt.

Weitere Informationen finden Sie unter Amazon Lex — Funktionsweise.

Diese Operation erfordert Berechtigungen für die Aktion `lex:PutIntent`.

Anforderungssyntax

```
PUT /intents/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createVersion": boolean,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
```

```
        "messages": [
          {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
          }
        ],
        "responseCard": "string"
      },
      "rejectionStatement": {
        "messages": [
          {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
          }
        ],
        "responseCard": "string"
      }
    },
    "fulfillmentActivity": {
      "codeHook": {
        "messageVersion": "string",
        "uri": "string"
      },
      "type": "string"
    },
    "inputContexts": [
      {
        "name": "string"
      }
    ],
    "kendraConfiguration": {
      "kendraIndex": "string",
      "queryFilterString": "string",
      "role": "string"
    },
    "outputContexts": [
      {
        "name": "string",
        "timeToLiveInSeconds": number,
        "turnsToLive": number
      }
    ],
```

```
    "parentIntentSignature": "string",
    "rejectionStatement": {
       "messages": [
          {
             "content": "string",
             "contentType": "string",
             "groupNumber": number
          }
       ],
       "responseCard": "string"
    },
    "sampleUtterances": [ "string" ],
    "slots": [
       {
          "defaultValueSpec": {
             "defaultValueList": [
                {
                   "defaultValue": "string"
                }
             ]
          },
          "description": "string",
          "name": "string",
          "obfuscationSetting": "string",
          "priority": number,
          "responseCard": "string",
          "sampleUtterances": [ "string" ],
          "slotConstraint": "string",
          "slotType": "string",
          "slotTypeVersion": "string",
          "valueElicitationPrompt": {
             "maxAttempts": number,
             "messages": [
                {
                   "content": "string",
                   "contentType": "string",
                   "groupNumber": number
                }
             ],
             "responseCard": "string"
          }
       }
    ]
```

```
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name der Absicht. Beim Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Der Name darf nicht mit einem integrierten Absichtsnamen oder einem integrierten Absichtsnamen mit „AMAZON“ übereinstimmen. entfernt. Da es beispielsweise eine integrierte Absicht namens gibt`AMAZON.HelpIntent`, können Sie keine benutzerdefinierte Absicht namens erstellen`HelpIntent`.

Die Liste der integrierten Absichten finden Sie unter Standard Built-in Intents (Integrierte Standardabsichten) im Alexa Skills Kit.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

checksum

Identifiziert eine bestimmte Version der `$LATEST` Version.

Wenn Sie eine neue Absicht erstellen, lassen Sie das `checksum` Feld leer. Wenn Sie eine Prüfsumme angeben, erhalten Sie eine `BadRequestException` Ausnahme.

Wenn Sie eine Absicht aktualisieren möchten, setzen Sie das `checksum` Feld auf die Prüfsumme der letzten Version der `$LATEST` Version. Wenn Sie das `checksum` Feld nicht angeben oder wenn die Prüfsumme nicht mit der `$LATEST` Version übereinstimmt, erhalten Sie eine `PreconditionFailedException` Ausnahme.

Typ: Zeichenfolge

Erforderlich: Nein

conclusionStatement

Die Aussage, die Amazon Lex dem Benutzer übermitteln soll, nachdem die Absicht von der Lambda-Funktion erfolgreich erfüllt wurde.

Dieses Element ist nur relevant, wenn Sie eine Lambda-Funktion in der `fulfillmentActivity` angeben. Wenn Sie die Absicht an die Client-Anwendung zurückgeben, können Sie dieses Element nicht angeben.

> **Note**
>
> Die `followUpPrompt` beiden schließen `conclusionStatement` sich gegenseitig aus. Sie können nur einen angeben.

Typ: Statement Objekt

Erforderlich: Nein

confirmationPrompt

Fordert den Benutzer auf, die Absicht zu bestätigen. Diese Frage sollte eine Ja- oder Nein-Antwort haben.

Amazon Lex verwendet diese Aufforderung, um sicherzustellen, dass der Benutzer bestätigt, dass die Absicht zur Erfüllung bereit ist. Mit dieser `OrderPizza` Absicht möchten Sie beispielsweise überprüfen, ob die Bestellung korrekt ist, bevor Sie sie aufgeben. Bei anderen Absichten, z. B. bei Absichten, die lediglich auf Benutzerfragen antworten, müssen Sie den Benutzer möglicherweise nicht um Bestätigung bitten, bevor Sie die Informationen bereitstellen.

> **Note**
>
> Sie müssen sowohl das als auch das oder `rejectionStatement` keines von beiden angeben. `confirmationPrompt`

Typ: Prompt Objekt

Erforderlich: Nein

createVersion

Bei Auswahl dieser Option wird `true` eine neue nummerierte Version der Absicht erstellt. Dies entspricht dem Aufrufen der `CreateIntentVersion` Operation. Wenn Sie nichts angeben`createVersion`, ist die Standardeinstellung`false`.

Typ: Boolesch

Erforderlich: Nein

description

Eine Beschreibung der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

dialogCodeHook

Spezifiziert eine Lambda-Funktion, die für jede Benutzereingabe aufgerufen werden soll. Sie können diese Lambda-Funktion aufrufen, um die Benutzerinteraktion zu personalisieren.

Nehmen wir zum Beispiel an, Ihr Bot stellt fest, dass der Benutzer John ist. Ihre Lambda-Funktion ruft möglicherweise Johns Informationen aus einer Backend-Datenbank ab und füllt einige der Werte vorab aus. Wenn Sie beispielsweise feststellen, dass John eine Glutenunverträglichkeit hat, können Sie den entsprechenden Intent-Slot,, auf true setzen. `GlutenIntolerant` Möglicherweise finden Sie Johns Telefonnummer und legen das entsprechende Sitzungsattribut fest.

Typ: CodeHook Objekt

Erforderlich: Nein

followUpPrompt

Amazon Lex verwendet diese Aufforderung, um nach Erfüllung einer Absicht weitere Aktivitäten anzufordern. Wenn die `OrderPizza` Absicht erfüllt wurde, könnten Sie den Benutzer beispielsweise auffordern, ein Getränk zu bestellen.

Die Aktion, die Amazon Lex ergreift, hängt wie folgt von der Antwort des Benutzers ab:

- Wenn der Benutzer „Ja“ sagt, antwortet er mit der Klarstellungsaufforderung, die für den Bot konfiguriert ist.

- Wenn der Benutzer „Ja“ sagt und mit einer Äußerung fortfährt, die eine Absicht auslöst, wird eine Konversation über diese Absicht gestartet.
- Wenn der Benutzer „Nein“ sagt, antwortet er mit der Ablehnungserklärung, die für die Folgeaufforderung konfiguriert ist.
- Wenn er die Äußerung nicht erkennt, wiederholt er die Folgeaufforderung erneut.

Das `followUpPrompt` Feld und das `conclusionStatement` Feld schließen sich gegenseitig aus. Sie können nur eines angeben.

Typ: FollowUpPrompt Objekt

Erforderlich: Nein

fulfillmentActivity

Erforderlich Beschreibt, wie die Absicht erfüllt wird. Wenn ein Benutzer beispielsweise alle Informationen für eine Pizzabestellung eingegeben hat, wird `fulfillmentActivity` definiert, wie der Bot eine Bestellung bei einer lokalen Pizzeria aufgibt.

Sie können Amazon Lex so konfigurieren, dass alle Informationen zur Absicht an die Client-Anwendung zurückgegeben werden, oder sie anweisen, eine Lambda-Funktion aufzurufen, die die Absicht verarbeiten kann (z. B. eine Bestellung bei einer Pizzeria aufgeben).

Typ: FulfillmentActivity Objekt

Erforderlich: Nein

inputContexts

Ein Array von `InputContext` Objekten, das die Kontexte auflistet, die aktiv sein müssen, damit Amazon Lex die Absicht in einer Konversation mit dem Benutzer auswählen kann.

Typ: Array von InputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 5 Elemente.

Erforderlich: Nein

kendraConfiguration

Konfigurationsinformationen, die erforderlich sind, um die `AMAZON.KendraSearchIntent` Absicht zu verwenden, eine Verbindung zu einem Amazon Kendra Kendra-Index herzustellen. Weitere Informationen finden Sie unter AMAZON. KendraSearchIntent.

Typ: KendraConfiguration Objekt

Erforderlich: Nein

outputContexts

Ein Array von `OutputContext` Objekten, das die Kontexte auflistet, die die Absicht aktiviert, wenn die Absicht erfüllt ist.

Typ: Array von OutputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

Erforderlich: Nein

parentIntentSignature

Eine Eindeutige Kennung für die integrierte Absicht, auf die diese Absicht basieren soll. Informationen zur Suche nach der Signatur für eine Absicht finden Sie unter Integrierte Standardabsichten im Alexa Skills Kit.

Typ: Zeichenfolge

Erforderlich: Nein

rejectionStatement

Wenn der Benutzer die in definierte Frage mit „Nein“ beantwortet`confirmationPrompt`, antwortet Amazon Lex mit dieser Erklärung, um zu bestätigen, dass die Absicht storniert wurde.

> **Note**
>
> Sie müssen `rejectionStatement` sowohl das als auch das oder keines von beiden angeben. `confirmationPrompt`

Typ: Statement Objekt

Erforderlich: Nein

sampleUtterances

Eine Reihe von Äußerungen (Zeichenketten), die ein Benutzer sagen könnte, um die Absicht zu signalisieren. Zum Beispiel „Ich möchte {PizzaSize} Pizza“, „Bestellung {Menge} {PizzaSize} Pizzen“.

In jeder Äußerung wird ein Slot-Name in geschweiften Klammern eingeschlossen.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 1500 Elementen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

slots

Eine Reihe von Intent-Slots. Zur Laufzeit ruft Amazon Lex mithilfe der in den Slots definierten Eingabeaufforderungen die erforderlichen Slot-Werte vom Benutzer ab. Weitere Informationen finden Sie unter Amazon Lex — Funktionsweise.

Typ: Array von Slot-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 100 Elemente.

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
```

```
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "createVersion": boolean,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "rejectionStatement": {
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      }
   },
   "fulfillmentActivity": {
      "codeHook": {
         "messageVersion": "string",
         "uri": "string"
      },
      "type": "string"
   },
```

```
    "inputContexts": [
       {
          "name": "string"
       }
    ],
    "kendraConfiguration": {
       "kendraIndex": "string",
       "queryFilterString": "string",
       "role": "string"
    },
    "lastUpdatedDate": number,
    "name": "string",
    "outputContexts": [
       {
          "name": "string",
          "timeToLiveInSeconds": number,
          "turnsToLive": number
       }
    ],
    "parentIntentSignature": "string",
    "rejectionStatement": {
       "messages": [
          {
             "content": "string",
             "contentType": "string",
             "groupNumber": number
          }
       ],
       "responseCard": "string"
    },
    "sampleUtterances": [ "string" ],
    "slots": [
       {
          "defaultValueSpec": {
             "defaultValueList": [
                {
                   "defaultValue": "string"
                }
             ]
          },
          "description": "string",
          "name": "string",
          "obfuscationSetting": "string",
          "priority": number,
```

```
         "responseCard": "string",
         "sampleUtterances": [ "string" ],
         "slotConstraint": "string",
         "slotType": "string",
         "slotTypeVersion": "string",
         "valueElicitationPrompt": {
            "maxAttempts": number,
            "messages": [
               {
                  "content": "string",
                  "contentType": "string",
                  "groupNumber": number
               }
            ],
            "responseCard": "string"
         }
      }
   ],
   "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

checksum

Prüfsumme der `$LATEST` Version der erstellten oder aktualisierten Absicht.

Typ: Zeichenfolge

conclusionStatement

Nachdem die in der Absicht angegebene Lambda-Funktion die `fulfillmentActivity` Absicht erfüllt hat, übermittelt Amazon Lex diese Aussage an den Benutzer.

Typ: Statement Objekt

confirmationPrompt

Falls in der Absicht definiert, fordert Amazon Lex den Benutzer auf, die Absicht zu bestätigen, bevor sie erfüllt wird.

Typ: Prompt Objekt

createdDate

Das Datum, an dem die Absicht erstellt wurde.

Typ: Zeitstempel

createVersion

`True`ob eine neue Version der Absicht erstellt wurde. Wenn das `createVersion` Feld in der Anfrage nicht angegeben wurde, wird das `createVersion` Feld in der Antwort auf „Falsch“ gesetzt.

Typ: Boolesch

description

Eine Beschreibung der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

dialogCodeHook

Falls in der Absicht definiert, ruft Amazon Lex diese Lambda-Funktion für jede Benutzereingabe auf.

Typ: CodeHook Objekt

followUpPrompt

Falls in der Absicht definiert, verwendet Amazon Lex diese Aufforderung, um zusätzliche Benutzeraktivitäten anzufordern, nachdem die Absicht erfüllt wurde.

Typ: FollowUpPrompt Objekt

fulfillmentActivity

Falls in der Absicht definiert, ruft Amazon Lex diese Lambda-Funktion auf, um die Absicht zu erfüllen, nachdem der Benutzer alle für die Absicht erforderlichen Informationen bereitgestellt hat.

Typ: FulfillmentActivity Objekt

inputContexts

Ein Array von `InputContext` Objekten, das die Kontexte auflistet, die aktiv sein müssen, damit Amazon Lex die Absicht in einer Konversation mit dem Benutzer auswählen kann.

Typ: Array von InputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 5 Elemente.

kendraConfiguration

Konfigurationsinformationen, falls vorhanden, erforderlich, um eine Verbindung zu einem Amazon Kendra Kendra-Index herzustellen und die `AMAZON.KendraSearchIntent` Absicht zu verwenden.

Typ: KendraConfiguration Objekt

lastUpdatedDate

Das Datum, an dem die Absicht aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

outputContexts

Eine Reihe von `OutputContext` Objekten, die die Kontexte auflistet, die die Absicht aktiviert, wenn die Absicht erfüllt ist.

Typ: Array von OutputContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

parentIntentSignature

Eine eindeutige Kennung für die integrierte Absicht, auf der diese Absicht basiert.

Typ: Zeichenfolge

rejectionStatement

Wenn der Benutzer die in `confirmationPrompt` Amazon Lex definierte Frage mit „Nein“ beantwortet, antwortet Lex mit dieser Erklärung, um zu bestätigen, dass die Absicht storniert wurde.

Typ: Statement Objekt

sampleUtterances

Eine Reihe von Beispieläußerungen, die für die Absicht konfiguriert sind.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 1500 Elementen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 200 Zeichen.

slots

Eine Reihe von Intent-Slots, die für den Intent konfiguriert sind.

Typ: Array von Slot-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 100 Elemente.

version

Die Version der Absicht. Für eine neue Absicht gilt immer die Version`$LATEST`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht wohlformuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PutSlotType

Service: Amazon Lex Model Building Service

Erstellt einen benutzerdefinierten Slot-Typ oder ersetzt einen vorhandenen, benutzerdefinierten Slot-Typ.

Um einen benutzerdefinierten Slot-Typ zu erstellen, geben Sie einen Namen für den Slot-Typ und eine Reihe von Aufzählungswerten an. Dies sind die Werte, die ein Slot dieses Typs annehmen kann. Weitere Informationen finden Sie unter Amazon Lex — Funktionsweise.

Wenn Sie den Namen eines vorhandenen Slot-Typs angeben, ersetzen die Felder in der Anfrage die vorhandenen Werte in der `$LATEST` Version des Slot-Typs. Amazon Lex entfernt die Felder, die Sie in der Anfrage nicht angeben. Wenn Sie keine Pflichtfelder angeben, löst Amazon Lex eine Ausnahme aus. Wenn Sie die `$LATEST` Version eines Slot-Typs aktualisieren und ein Bot die `$LATEST` Version einer Intent verwendet, die den Slot-Typ enthält, wird das `status` Feld des Bots auf `NOT_BUILT` gesetzt.

Diese Operation erfordert Berechtigungen für die Aktion `lex:PutSlotType`.

Anforderungssyntax

```
PUT /slottypes/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "checksum": "string",
   "createVersion": boolean,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
```

```
    "valueSelectionStrategy": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

name

Der Name des Slot-Typs. Beim Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Der Name darf nicht mit einem Namen eines integrierten Steckplatztyps oder einem Namen eines integrierten Steckplatztyps mit „AMAZON“ übereinstimmen. entfernt. Da es beispielsweise einen integrierten Slot-Typ namens gibt`AMAZON.DATE`, können Sie keinen benutzerdefinierten Slot-Typ namens erstellen`DATE`.

Eine Liste der integrierten Slot-Typen finden Sie unter Slot-Typ-Referenz im Alexa Skills Kit.

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

checksum

Identifiziert eine bestimmte Version der `$LATEST` Version.

Wenn Sie einen neuen Slot-Typ erstellen, lassen Sie das `checksum` Feld leer. Wenn Sie eine Prüfsumme angeben, erhalten Sie eine `BadRequestException` Ausnahme.

Wenn Sie einen Slot-Typ aktualisieren möchten, setzen Sie das `checksum` Feld auf die Prüfsumme der letzten Version der `$LATEST` Version. Wenn Sie das `checksum` Feld nicht angeben oder wenn die Prüfsumme nicht mit der `$LATEST` Version übereinstimmt, erhalten Sie eine `PreconditionFailedException` Ausnahme.

Typ: Zeichenfolge

Erforderlich: Nein

createVersion

Wenn diese Option aktiviert ist, wird `true` eine neue nummerierte Version des Slot-Typs erstellt. Dies entspricht dem Aufrufen der `CreateSlotTypeVersion` Operation. Wenn Sie nichts angeben`createVersion`, ist die Standardeinstellung`false`.

Typ: Boolesch

Erforderlich: Nein

description

Eine Beschreibung des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

enumerationValues

Eine Liste von `EnumerationValue` Objekten, die die Werte definiert, die der Slot-Typ annehmen kann. Jeder Wert kann eine Liste mit zusätzlichen Werten enthalten`synonyms`, mit deren Hilfe das Modell des maschinellen Lernens anhand der Werte trainiert werden kann, die es für einen Slot auflöst.

Ein Slot-Typ mit regulären Ausdrücken erfordert keine Aufzählungswerte. Alle anderen Slot-Typen erfordern eine Liste von Aufzählungswerten.

Wenn Amazon Lex einen Slot-Wert auflöst, generiert es eine Auflösungsliste, die bis zu fünf mögliche Werte für den Slot enthält. Wenn Sie eine Lambda-Funktion verwenden, wird diese Auflösungsliste an die Funktion übergeben. Wenn Sie keine Lambda-Funktion verwenden, können Sie wählen, ob Sie den vom Benutzer eingegebenen Wert oder den ersten Wert in der Auflösungsliste als Slot-Wert zurückgeben möchten. Das `valueSelectionStrategy` Feld gibt die zu verwendende Option an.

Typ: Array von EnumerationValue-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 10000 Artikeln.

Erforderlich: Nein

parentSlotTypeSignature

Der integrierte Steckplatztyp, der als übergeordneter Steckplatztyp verwendet wird. Wenn Sie einen übergeordneten Steckplatztyp definieren, hat der neue Steckplatztyp dieselbe Konfiguration wie der übergeordnete Steckplatztyp.

Nur `AMAZON.AlphaNumeric` wird unterstützt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Erforderlich: Nein

slotTypeConfigurations

Konfigurationsinformationen, die den Typ des übergeordneten integrierten Steckplatzes erweitern. Die Konfiguration wird zu den Einstellungen für den Typ des übergeordneten Steckplatzes hinzugefügt.

Typ: Array von SlotTypeConfiguration-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

Erforderlich: Nein

valueSelectionStrategy

Bestimmt die Strategie zur Steckplatzauflösung, die Amazon Lex verwendet, um Slot-Typ-Werte zurückzugeben. Das Feld kann auf einen der folgenden Werte gestellt werden:

- `ORIGINAL_VALUE`- Gibt den vom Benutzer eingegebenen Wert zurück, wenn der Benutzerwert dem Slot-Wert ähnlich ist.
- `TOP_RESOLUTION`- Wenn es eine Auflösungsliste für den Steckplatz gibt, geben Sie den ersten Wert in der Auflösungsliste als Slot-Typwert zurück. Wenn keine Auflösungsliste vorhanden ist, wird null zurückgegeben.

Wenn Sie den nicht angeben`valueSelectionStrategy`, ist der Standardwert`ORIGINAL_VALUE`.

Typ: Zeichenfolge

Zulässige Werte: `ORIGINAL_VALUE | TOP_RESOLUTION`

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "checksum": "string",
   "createdDate": number,
   "createVersion": boolean,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

checksum

Prüfsumme der `$LATEST` Version des Slot-Typs.

Typ: Zeichenfolge

createdDate

Das Datum, an dem der Slot-Typ erstellt wurde.

Typ: Zeitstempel

createVersion

`True`wenn eine neue Version des Slot-Typs erstellt wurde. Wenn das `createVersion` Feld in der Anfrage nicht angegeben wurde, wird das `createVersion` Feld in der Antwort auf „Falsch“ gesetzt.

Typ: Boolesch

description

Eine Beschreibung des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

enumerationValues

Eine Liste von `EnumerationValue` Objekten, die die Werte definiert, die der Slot-Typ annehmen kann.

Typ: Array von EnumerationValue-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 10000 Elementen.

lastUpdatedDate

Das Datum, an dem der Slot-Typ aktualisiert wurde. Wenn Sie einen Slot-Typ erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

name

Der Name des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Der integrierte Slot-Typ, der dem Slot-Typ als übergeordnetes Objekt verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Konfigurationsinformationen, die den Typ des übergeordneten integrierten Steckplatzes erweitern.

Typ: Array von SlotTypeConfiguration-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

valueSelectionStrategy

Die Strategie zur Steckplatzauflösung, die Amazon Lex verwendet, um den Wert des Steckplatzes zu bestimmen. Weitere Informationen finden Sie unter PutSlotType.

Typ: Zeichenfolge

Zulässige Werte: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

Die Version des Steckplatztyps. Für einen neuen Slot-Typ ist die Version immer`$LATEST`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

PreconditionFailedException

Die Prüfsumme der Ressource, die Sie ändern möchten, stimmt nicht mit der Prüfsumme in der Anfrage überein. Überprüfen Sie die Prüfsumme der Ressource und versuchen Sie es erneut.

HTTP-Statuscode: 412

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

StartImport

Service: Amazon Lex Model Building Service

Startet einen Auftrag zum Importieren einer Ressource in Amazon Lex.

Anforderungssyntax

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
   "mergeStrategy": "string",
   "payload": blob,
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

URI-Anfrageparameter

Die Anforderung verwendet keine URI-Parameter.

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

mergeStrategy

Gibt die Aktion an, die der `StartImport` Vorgang ausführen soll, wenn eine Ressource mit demselben Namen vorhanden ist.

- FAIL_ON_CONFLICT — Der Importvorgang wird beim ersten Konflikt zwischen einer Ressource in der Importdatei und einer vorhandenen Ressource gestoppt. Der Name der Ressource, die den Konflikt verursacht hat, steht im `failureReason` Feld der Antwort auf den Vorgang. `GetImport`

  OVERWRITE_LATEST — Der Importvorgang wird auch dann fortgesetzt, wenn ein Konflikt mit einer vorhandenen Ressource besteht. Die $LASTST-Version der vorhandenen Ressource wird mit den Daten aus der Importdatei überschrieben.

Typ: Zeichenfolge

Zulässige Werte: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

Erforderlich: Ja

payload

Ein ZIP-Archiv im Binärformat. Das Archiv sollte eine Datei enthalten, eine JSON-Datei, die die zu importierende Ressource enthält. Die Ressource sollte dem im `resourceType` Feld angegebenen Typ entsprechen.

Typ: Base64-kodiertes Binärdatenobjekt

Erforderlich: Ja

resourceType

Gibt den Typ der zu exportierenden Ressource an. Jede Ressource exportiert auch alle Ressourcen, von denen sie abhängig ist.

- Ein Bot exportiert abhängige Absichten.
- Ein Intent exportiert abhängige Slot-Typen.

Typ: Zeichenfolge

Zulässige Werte: `BOT | INTENT | SLOT_TYPE`

Erforderlich: Ja

tags

Eine Liste von Tags, die dem importierten Bot hinzugefügt werden sollen. Sie können Tags nur hinzufügen, wenn Sie einen Bot importieren. Sie können keine Tags zu einer Absicht oder einem Slot-Typ hinzufügen.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 201
Content-type: application/json

{
   "createdDate": number,
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP-201-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

createdDate

Ein Zeitstempel für das Datum und die Uhrzeit, zu der der Importjob angefordert wurde.

Typ: Zeitstempel

importId

Der Bezeichner für den spezifischen Importjob.

Typ: Zeichenfolge

importStatus

Der Status des Importauftrags. Wenn der Status lautet`FAILED`, können Sie den Grund für den Fehler mithilfe der `GetImport` Operation ermitteln.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

Die Aktion, die ergriffen werden soll, wenn ein Zusammenführungskonflikt vorliegt.

Typ: Zeichenfolge

Zulässige Werte: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

Der Name, der dem Importauftrag gegeben wurde.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `[a-zA-Z_]+`

resourceType

Der Typ der zu importierenden Ressource.

Typ: Zeichenfolge

Zulässige Werte: `BOT | INTENT | SLOT_TYPE`

tags

Eine Liste von Tags, die dem importierten Bot hinzugefügt wurden.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Fehler

BadRequestException

Die Anfrage ist nicht wohlformuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

Weitere Informationen finden Sie auch unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

StartMigration

Service: Amazon Lex Model Building Service

Startet die Migration eines Bots von Amazon Lex V1 zu Amazon Lex V2. Migrieren Sie Ihren Bot, wenn Sie die neuen Funktionen von Amazon Lex V2 nutzen möchten.

Weitere Informationen finden Sie unter Einen Bot migrieren im Amazon Lex Developer Guide.

Anforderungssyntax

```
POST /migrations HTTP/1.1
Content-type: application/json

{
   "migrationStrategy": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotName": "string",
   "v2BotRole": "string"
}
```

URI-Anfrageparameter

Die Anforderung verwendet keine URI-Parameter.

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

migrationStrategy

Die Strategie, mit der die Migration durchgeführt wurde.

- `CREATE_NEW`- Erstellt einen neuen Amazon Lex V2-Bot und migriert den Amazon Lex V1-Bot auf den neuen Bot.
- `UPDATE_EXISTING`- Überschreibt die vorhandenen Amazon Lex V2-Bot-Metadaten und das zu migrierende Gebietsschema. Es ändert keine anderen Gebietsschemas im Amazon Lex V2-Bot. Wenn das Gebietsschema nicht existiert, wird ein neues Gebietsschema im Amazon Lex V2-Bot erstellt.

Typ: Zeichenfolge

Zulässige Werte: `CREATE_NEW | UPDATE_EXISTING`

Erforderlich: Ja

v1BotName

Der Name des Amazon Lex V1-Bots, den Sie zu Amazon Lex V2 migrieren.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

v1BotVersion

Die Version des Bots, der auf Amazon Lex V2 migriert werden soll. Sie können die `$LATEST` Version sowie jede nummerierte Version migrieren.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Ja

v2BotName

Der Name des Amazon Lex V2-Bots, zu dem Sie den Amazon Lex V1-Bot migrieren.

- Wenn der Amazon Lex V2-Bot nicht existiert, müssen Sie die `CREATE_NEW` Migrationsstrategie verwenden.
- Wenn der Amazon Lex V2-Bot vorhanden ist, müssen Sie die `UPDATE_EXISTING` Migrationsstrategie verwenden, um den Inhalt des Amazon Lex V2-Bots zu ändern.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([0-9a-zA-Z][_-]?)+$`

Erforderlich: Ja

v2BotRole

Die IAM-Rolle, die Amazon Lex verwendet, um den Amazon Lex V2-Bot auszuführen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 202
Content-type: application/json

{
   "migrationId": "string",
   "migrationStrategy": "string",
   "migrationTimestamp": number,
   "v1BotLocale": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotId": "string",
   "v2BotRole": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 202-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

migrationId

Die eindeutige Kennung, die Amazon Lex der Migration zugewiesen hat.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

migrationStrategy

Die Strategie, mit der die Migration durchgeführt wurde.

Typ: Zeichenfolge

Zulässige Werte: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

Datum und Uhrzeit des Beginns der Migration.

Typ: Zeitstempel

v1BotLocale

Das für den Amazon Lex V1-Bot verwendete Gebietsschema.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

v1BotName

Der Name des Amazon Lex V1-Bots, den Sie zu Amazon Lex V2 migrieren.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

v1BotVersion

Die Version des Bots, der auf Amazon Lex V2 migriert werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

v2BotId

Die eindeutige Kennung für den Amazon Lex V2-Bot.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

v2BotRole

Die IAM-Rolle, die Amazon Lex verwendet, um den Amazon Lex V2-Bot auszuführen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Fehler

AccessDeniedException

Ihr IAM-Benutzer oder Ihre IAM-Rolle ist nicht berechtigt, die Amazon Lex V2-APIs aufzurufen, die für die Migration Ihres Bots erforderlich sind.

HTTP Status Code: 403

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

TagResource

Service: Amazon Lex Model Building Service

Fügt der angegebenen Ressource die angegebenen Tags hinzu. Wenn ein Tag-Schlüssel bereits vorhanden ist, wird der vorhandene Wert durch den neuen Wert ersetzt.

Anforderungssyntax

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

resourceArn

Der Amazon-Ressourcenname (ARN) des Bots, Bot-Alias oder Bot-Kanals, der markiert werden soll.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1011.

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

tags

Eine Liste von Tag-Schlüsseln, die der Ressource hinzugefügt werden sollen. Wenn ein Tag-Schlüssel bereits vorhanden ist, wird der vorhandene Wert durch den neuen Wert ersetzt.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

UntagResource

Service: Amazon Lex Model Building Service

Entfernt Tags von einem Bot, Bot-Alias oder Bot-Kanal.

Anforderungssyntax

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

resourceArn

Der Amazon-Ressourcenname (ARN) der Ressource, aus der die Tags entfernt werden sollen.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1011.

Erforderlich: Ja

tagKeys

Eine Liste von Tag-Schlüsseln, die aus der Ressource entfernt werden sollen. Wenn ein Tag-Schlüssel auf der Ressource nicht vorhanden ist, wird er ignoriert.

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 204
```

Antwortelemente

Wenn die Aktion erfolgreich ist, gibt der Dienst eine HTTP-204-Antwort mit leerem HTTP-Textinhalt zurück.

Fehler

BadRequestException

Die Anfrage ist nicht korrekt formuliert. Beispielsweise ist ein Wert ungültig oder ein erforderliches Feld fehlt. Überprüfen Sie die Feldwerte und versuchen Sie es erneut.

HTTP Status Code: 400

ConflictException

Bei der Verarbeitung der Anfrage ist ein Konflikt aufgetreten. Versuchen Sie es erneut.

HTTP-Statuscode: 409

InternalFailureException

Ein interner Amazon Lex Lex-Fehler ist aufgetreten. Versuchen Sie es erneut.

HTTP Status Code: 500

LimitExceededException

Die Anfrage hat ein Limit überschritten. Versuchen Sie es erneut.

HTTP-Statuscode: 429

NotFoundException

Die in der Anfrage angegebene Ressource wurde nicht gefunden. Überprüfen Sie die Ressource und versuchen Sie es erneut.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

Amazon Lex Laufzeit-Service

Die folgenden Aktionen werden vom Amazon Lex Runtime Service unterstützt:

- DeleteSession
- GetSession
- PostContent
- PostText
- PutSession

DeleteSession

Service: Amazon Lex Runtime Service

Entfernt Sitzungsinformationen für einen angegebenen Bot, Alias und Benutzer-ID.

Anforderungssyntax

```
DELETE /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botAlias

Der Alias, der für den Bot verwendet wird, der die Sitzungsdaten enthält.

Erforderlich: Ja

botName

Der Name des Bots, der die Sitzungsdaten enthält.

Erforderlich: Ja

userId

Die Kennung des Benutzers, der mit den Sitzungsdaten verknüpft ist.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Pattern: `[0-9a-zA-Z._:-]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "botAlias": "string",
   "botName": "string",
   "sessionId": "string",
   "userId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

botAlias

Der Alias, der für den Bot verwendet wird, der mit den Sitzungsdaten verknüpft ist.

Typ: Zeichenfolge

botName

Der Name des Bots, der mit den Sitzungsdaten verknüpft ist.

Typ: Zeichenfolge

sessionId

Die eindeutige Kennung für die Sitzung.

Typ: Zeichenfolge

userId

Die ID des Benutzers der Client-Anwendung.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Pattern: `[0-9a-zA-Z._:-]+`

Fehler

BadRequestException

Die Überprüfung der Anfrage ist fehlgeschlagen, es gibt keine brauchbare Nachricht im Kontext, oder der Bot-Build ist fehlgeschlagen, ist noch in Bearbeitung oder enthält noch nicht erstellte Änderungen.

HTTP Status Code: 400

ConflictException

Zwei Kunden verwenden dasselbe AWS-Konto, denselben Amazon Lex Lex-Bot und dieselbe Benutzer-ID.

HTTP-Statuscode: 409

InternalFailureException

Interner Servicefehler. Versuchen Sie den Anruf erneut.

HTTP Status Code: 500

LimitExceededException

Ein Limit wurde überschritten.

HTTP-Statuscode: 429

NotFoundException

Die Ressource (z. B. der Amazon Lex Lex-Bot oder ein Alias), auf die verwiesen wird, wurde nicht gefunden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

GetSession

Service: Amazon Lex Runtime Service

Gibt Sitzungsinformationen für einen angegebenen Bot, Alias und Benutzer-ID zurück.

Anforderungssyntax

```
GET /bot/botName/alias/botAlias/user/userId/session/?
checkpointLabelFilter=checkpointLabelFilter HTTP/1.1
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botAlias

Der Alias, der für den Bot verwendet wird, der die Sitzungsdaten enthält.

Erforderlich: Ja

botName

Der Name des Bots, der die Sitzungsdaten enthält.

Erforderlich: Ja

checkpointLabelFilter

Eine Zeichenfolge, mit der die in der `recentIntentSummaryView` Struktur zurückgegebenen Absichten gefiltert werden.

Wenn Sie einen Filter angeben, werden nur Absichten zurückgegeben, deren `checkpointLabel` Feld auf diese Zeichenfolge gesetzt ist.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 255 Zeichen.

Pattern: `[a-zA-Z0-9-]+`

userId

Die ID des Benutzers der Client-Anwendung. Amazon Lex verwendet dies, um die Konversation eines Benutzers mit Ihrem Bot zu identifizieren.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Pattern: `[0-9a-zA-Z._:-]+`

Erforderlich: Ja

Anforderungstext

Der Anforderung besitzt keinen Anforderungstext.

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "dialogAction": {
      "fulfillmentState": "string",
      "intentName": "string",
      "message": "string",
      "messageFormat": "string",
      "slots": {
         "string" : "string"
      },
      "slotToElicit": "string",
      "type": "string"
   },
   "recentIntentSummaryView": [
      {
         "checkpointLabel": "string",
         "confirmationStatus": "string",
         "dialogActionType": "string",
         "fulfillmentState": "string",
```

```
            "intentName": "string",
            "slots": {
               "string" : "string"
            },
            "slotToElicit": "string"
         }
      ],
      "sessionAttributes": {
         "string" : "string"
      },
      "sessionId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

activeContexts

Eine Liste der aktiven Kontexte für die Sitzung. Ein Kontext kann festgelegt werden, wenn eine Absicht erfüllt ist, oder durch Aufrufen der `PutSession` Operation `PostContentPostText`, oder.

Sie können einen Kontext verwenden, um die Absichten zu steuern, die einer Absicht folgen können, oder um den Betrieb Ihrer Anwendung zu ändern.

Typ: Array von ActiveContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 50 Elemente.

dialogAction

Beschreibt den aktuellen Status des Bots.

Typ: DialogAction Objekt

recentIntentSummaryView

Eine Reihe von Informationen über die in der Sitzung verwendeten Absichten. Das Array kann maximal drei Zusammenfassungen enthalten. Wenn in der Sitzung mehr als drei Absichten

verwendet werden, enthält der `recentIntentSummaryView` Vorgang Informationen über die letzten drei verwendeten Absichten.

Wenn Sie den `checkpointLabelFilter` Parameter in der Anforderung festlegen, enthält das Array nur die Absichten mit der angegebenen Bezeichnung.

Typ: Array von IntentSummary-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 3 Elementen.

sessionAttributes

Karte von Schlüssel/Wert-Paaren, die die sitzungsspezifischen Kontextinformationen darstellen. Es enthält Anwendungsinformationen, die zwischen Amazon Lex und einer Client-Anwendung ausgetauscht werden.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

sessionId

Eindeutiger Bezeichner für die Sitzung.

Typ: Zeichenfolge

Fehler

BadRequestException

Die Überprüfung der Anfrage ist fehlgeschlagen, es gibt keine brauchbare Nachricht im Kontext, oder der Bot-Build ist fehlgeschlagen, ist noch in Bearbeitung oder enthält noch nicht erstellte Änderungen.

HTTP Status Code: 400

InternalFailureException

Interner Dienstfehler. Versuchen Sie den Anruf erneut.

HTTP Status Code: 500

LimitExceededException

Ein Limit wurde überschritten.

HTTP-Statuscode: 429

NotFoundException

Die Ressource (z. B. der Amazon Lex Lex-Bot oder ein Alias), auf die verwiesen wird, wurde nicht gefunden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PostContent

Service: Amazon Lex Runtime Service

Sendet eine Benutzereingabe (Text oder Sprache) an Amazon Lex Kunden verwenden diese API, um zur Laufzeit Text- und Audioanfragen an Amazon Lex zu senden. Amazon Lex interpretiert die Benutzereingaben mithilfe des Machine-Learning-Modells, das es für den Bot entwickelt hat.

Der `PostContent` Vorgang unterstützt Audioeingänge bei 8 kHz und 16 kHz. Sie können 8-kHz-Audio verwenden, um eine höhere Spracherkennungsgenauigkeit in Telefonaudioanwendungen zu erreichen.

Als Antwort sendet Amazon Lex die nächste Nachricht zurück, die dem Benutzer übermittelt werden soll. Betrachten Sie die folgenden Beispielnachrichten:

- Bei einer Benutzereingabe „Ich hätte gerne eine Pizza“ gibt Amazon Lex möglicherweise eine Antwort mit einer Nachricht zurück, die Slot-Daten ausgibt (z. B.`PizzaSize`): „Welche Pizzagröße möchten Sie?“.
- Nachdem der Benutzer alle Informationen zur Pizzabestellung eingegeben hat, gibt Amazon Lex möglicherweise eine Antwort mit der Nachricht zurück, dass der Benutzer eine Bestätigung erhält: „Pizza bestellen?“.
- Nachdem der Benutzer die Bestätigungsaufforderung mit „Ja“ beantwortet hat, gibt Amazon Lex möglicherweise eine Schlussfolgerung zurück: „Danke, Ihre Käsepizza wurde bestellt.“.

Nicht alle Amazon Lex Lex-Nachrichten erfordern eine Antwort vom Benutzer. Zum Beispiel erfordern Schlußfolgerungen keine Antwort. Einige Nachrichten erfordern nur eine Ja- oder Nein-Antwort. Darüber hinaus bietet Amazon Lex zusätzlichen Kontext zu der `message` Nachricht in der Antwort, den Sie verwenden können, um das Kundenverhalten zu verbessern, z. B. die Anzeige der entsprechenden Client-Benutzeroberfläche. Betrachten Sie die folgenden Beispiele:

- Wenn die Nachricht Slot-Daten abrufen soll, gibt Amazon Lex die folgenden Kontextinformationen zurück:
  - `x-amz-lex-dialog-state`Header gesetzt auf `ElicitSlot`
  - `x-amz-lex-intent-name`Header, der auf den Namen der Absicht im aktuellen Kontext gesetzt ist
  - `x-amz-lex-slot-to-elicit`Header, der auf den Slot-Namen gesetzt `message` ist, für den Informationen abgerufen werden

- `x-amz-lex-slots`Header, der auf eine Zuordnung von Slots gesetzt ist, die für den Intent konfiguriert sind, mit ihren aktuellen Werten

- Handelt es sich bei der Nachricht um eine Bestätigungsaufforderung, wird der `x-amz-lex-dialog-state` Header auf gesetzt `Confirmation` und der `x-amz-lex-slot-to-elicit` Header weggelassen.
- Handelt es sich bei der Nachricht um eine für die Absicht konfigurierte Klarstellungsaufforderung, die darauf hinweist, dass die Benutzerabsicht nicht verstanden wurde, wird der `x-amz-dialog-state` Header auf gesetzt `ElicitIntent` und der `x-amz-slot-to-elicit` Header weggelassen.

Darüber hinaus sendet Amazon Lex auch Ihre `sessionAttributes` anwendungsspezifischen Daten zurück. Weitere Informationen finden Sie unter Konversationskontext verwalten.

Anforderungssyntax

```
POST /bot/botName/alias/botAlias/user/userId/content HTTP/1.1
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-request-attributes: requestAttributes
Content-Type: contentType
Accept: accept
x-amz-lex-active-contexts: activeContexts

inputStream
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

accept

Sie übergeben diesen Wert als `Accept` HTTP-Header.

Die Nachricht, die Amazon Lex in der Antwort zurückgibt, kann entweder Text oder Sprache sein, basierend auf dem `Accept` HTTP-Header-Wert in der Anfrage.

- Wenn der Wert ist`text/plain; charset=utf-8`, gibt Amazon Lex Text in der Antwort zurück.
- Beginnt der Wert mit`audio/`, gibt Amazon Lex in der Antwort Sprache zurück. Amazon Lex verwendet Amazon Polly, um die Sprache zu generieren (mit der Konfiguration, die Sie in

der `Accept` Kopfzeile angegeben haben). Wenn Sie beispielsweise `audio/mpeg` als Wert angeben, gibt Amazon Lex Sprache im MPEG-Format zurück.

- Wenn der Wert „ist“`audio/pcm`, wird die zurückgegebene Sprache `audio/pcm` im 16-Bit-Little-Endian-Format zurückgegeben.
- Die folgenden Werte sind zulässig:
  - Audio/mpeg
  - Audio/OGG
  - Audio/PCM
  - Text/Einfach; Zeichensatz = UTF-8
  - audio/* (standardmäßig mpeg)

activeContexts

Eine Liste von Kontexten, die für die Anfrage aktiv sind. Ein Kontext kann aktiviert werden, wenn eine vorherige Absicht erfüllt ist, oder indem der Kontext in die Anfrage aufgenommen wird.

Wenn Sie keine Liste von Kontexten angeben, verwendet Amazon Lex die aktuelle Liste der Kontexte für die Sitzung. Wenn Sie eine leere Liste angeben, werden alle Kontexte für die Sitzung gelöscht.

botAlias

Alias des Amazon Lex Lex-Bot.

Erforderlich: Ja

botName

Name des Amazon Lex Lex-Bots.

Erforderlich: Ja

contentType

Sie übergeben diesen Wert als `Content-Type` HTTP-Header.

Gibt das Audioformat oder den Text an. Der Header-Wert muss mit einem der folgenden Präfixe beginnen:

- PCM-Format, Audiodaten müssen in Little-Endian-Byte-Reihenfolge vorliegen.
  - audio/l16; Rate = 16000; Kanäle = 1
  - audio/x-l16; Abtastrate = 16000; Kanalanzahl = 1

- audio/lpcm; Abtastrate=8000; =16; Kanalzahl=1; sample-size-bits =falsch is-big-endian
- Opus-Format
  - audio/ x-cbr-opus-with -präambel; Größe der Präambel = 0; Bitrate = 256000; =4 frame-size-milliseconds
- Textformat
  - Text/Einfach; Zeichensatz=UTF-8

Erforderlich: Ja

requestAttributes

Sie übergeben diesen Wert als HTTP-Header. `x-amz-lex-request-attributes`

Anforderungsspezifische Informationen, die zwischen Amazon Lex und einer Client-Anwendung übertragen werden. Bei dem Wert muss es sich um eine serialisierte und Base64-kodierte JSON-Map mit String-Schlüsseln und Werten handeln. Die Gesamtgröße der `requestAttributes` `sessionAttributes` AND-Header ist auf 12 KB begrenzt.

Der Namespace `x-amz-lex:` ist für spezielle Attribute reserviert. Erstellen Sie keine Anforderungsattribute mit dem Präfix`x-amz-lex:`.

Weitere Informationen finden Sie unter Anforderungsattribute festlegen.

sessionAttributes

Sie übergeben diesen Wert als `x-amz-lex-session-attributes` HTTP-Header.

Anwendungsspezifische Informationen, die zwischen Amazon Lex und einer Client-Anwendung ausgetauscht werden. Bei dem Wert muss es sich um eine serialisierte und Base64-kodierte JSON-Map mit String-Schlüsseln und Werten handeln. Die Gesamtgröße der `sessionAttributes` `requestAttributes` AND-Header ist auf 12 KB begrenzt.

Weitere Informationen finden Sie unter Sitzungsattribute festlegen.

userId

Die ID des Benutzers der Client-Anwendung. Amazon Lex verwendet dies, um die Konversation eines Benutzers mit Ihrem Bot zu identifizieren. Zur Laufzeit muss jede Anfrage das `userID` Feld enthalten.

Bei der Entscheidung, welche Benutzer-ID für Ihre Anwendung verwendet werden soll, sollten Sie die folgenden Faktoren berücksichtigen.

- Das `userID` Feld darf keine persönlich identifizierbaren Informationen des Benutzers enthalten, z. B. Namen, persönliche Identifikationsnummern oder andere persönliche Daten des Endbenutzers.
- Wenn Sie möchten, dass ein Benutzer eine Konversation auf einem Gerät beginnt und auf einem anderen Gerät fortsetzt, verwenden Sie eine benutzerspezifische Kennung.
- Wenn Sie möchten, dass derselbe Benutzer zwei unabhängige Konversationen auf zwei verschiedenen Geräten führen kann, wählen Sie eine gerätespezifische Kennung.
- Ein Benutzer kann nicht zwei unabhängige Konversationen mit zwei verschiedenen Versionen desselben Bots führen. Beispielsweise kann ein Benutzer keine Konversation mit den PROD- und BETA-Versionen desselben Bots führen. Wenn Sie davon ausgehen, dass ein Benutzer beispielsweise beim Testen eine Konversation mit zwei verschiedenen Versionen führen muss, fügen Sie den Bot-Alias in die Benutzer-ID ein, um die beiden Konversationen voneinander zu trennen.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Pattern: `[0-9a-zA-Z._:-]+`

Erforderlich: Ja

Anforderungstext

Die Anfrage akzeptiert die folgenden Binärdaten.

inputStream

Benutzereingaben im PCM- oder Opus-Audioformat oder im Textformat, wie im `Content-Type` HTTP-Header beschrieben.

Sie können Audiodaten zu Amazon Lex streamen oder einen lokalen Puffer erstellen, der alle Audiodaten vor dem Senden aufzeichnet. Im Allgemeinen erzielen Sie eine bessere Leistung, wenn Sie Audiodaten streamen, anstatt die Daten lokal zu puffern.

Erforderlich: Ja

Antwortsyntax

```
HTTP/1.1 200
```

```
Content-Type: contentType
x-amz-lex-intent-name: intentName
x-amz-lex-nlu-intent-confidence: nluIntentConfidence
x-amz-lex-alternative-intents: alternativeIntents
x-amz-lex-slots: slots
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-sentiment: sentimentResponse
x-amz-lex-message: message
x-amz-lex-encoded-message: encodedMessage
x-amz-lex-message-format: messageFormat
x-amz-lex-dialog-state: dialogState
x-amz-lex-slot-to-elicit: slotToElicit
x-amz-lex-input-transcript: inputTranscript
x-amz-lex-encoded-input-transcript: encodedInputTranscript
x-amz-lex-bot-version: botVersion
x-amz-lex-session-id: sessionId
x-amz-lex-active-contexts: activeContexts

audioStream
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die Antwort gibt die folgenden HTTP-Header zurück.

activeContexts

Eine Liste der aktiven Kontexte für die Sitzung. Ein Kontext kann festgelegt werden, wenn eine Absicht erfüllt ist, oder durch Aufrufen der `PutSession` Operation `PostContentPostText`, oder.

Sie können einen Kontext verwenden, um die Absichten zu steuern, die einer Absicht folgen können, oder um den Betrieb Ihrer Anwendung zu ändern.

alternativeIntents

Ein bis vier alternative Absichten, die auf die Absicht des Benutzers zutreffen können.

Jede Alternative beinhaltet eine Bewertung, die angibt, wie sicher Amazon Lex ist, dass die Absicht mit der Absicht des Benutzers übereinstimmt. Die Absichten sind nach dem Konfidenzwert sortiert.

botVersion

Die Version des Bots, der auf die Konversation geantwortet hat. Anhand dieser Informationen können Sie feststellen, ob eine Version eines Bots besser abschneidet als eine andere Version.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+|\$LATEST`

contentType

Inhaltstyp, wie im `Accept` HTTP-Header der Anfrage angegeben.

dialogState

Identifiziert den aktuellen Status der Benutzerinteraktion. Amazon Lex gibt einen der folgenden Werte als zurück`dialogState`. Der Kunde kann diese Informationen optional verwenden, um die Benutzeroberfläche anzupassen.

- `ElicitIntent`- Amazon Lex möchte die Absicht des Benutzers ermitteln. Betrachten Sie die folgenden Beispiele:

  Beispielsweise könnte ein Benutzer eine Absicht äußern („Ich möchte eine Pizza bestellen“). Wenn Amazon Lex die Absicht des Benutzers aus dieser Äußerung nicht ableiten kann, gibt es diesen Dialogstatus zurück.

- `ConfirmIntent`- Amazon Lex erwartet eine Antwort mit „Ja“ oder „Nein“.

  Amazon Lex möchte beispielsweise eine Bestätigung durch den Benutzer, bevor eine Absicht erfüllt wird. Statt einer einfachen Antwort mit „Ja“ oder „Nein“ könnte ein Benutzer mit zusätzlichen Informationen antworten. Zum Beispiel „Ja, aber mach eine Pizza mit dicker Kruste daraus“ oder „Nein, ich möchte ein Getränk bestellen“. Amazon Lex kann solche zusätzlichen Informationen verarbeiten (in diesen Beispielen aktualisieren Sie das Feld für den Krustentyp oder ändern Sie die Absicht von OrderPizza zu OrderDrink).

- `ElicitSlot`- Amazon Lex erwartet den Wert eines Slots für die aktuelle Absicht.

  Nehmen wir zum Beispiel an, dass Amazon Lex in der Antwort die folgende Nachricht sendet: „Welche Pizzagröße hätten Sie gerne?“. Ein Benutzer könnte mit dem Slot-Wert antworten (z. B. „mittel“). Der Benutzer kann in der Antwort auch zusätzliche Informationen angeben (z. B. „Pizza mit mitteldicker Kruste“). Amazon Lex kann solche zusätzlichen Informationen angemessen verarbeiten.

- `Fulfilled`- Übermittelt, dass die Lambda-Funktion die Absicht erfolgreich erfüllt hat.

- `ReadyForFulfillment`- Vermittelt, dass der Kunde die Anfrage erfüllen muss.
- `Failed`- Übermittelt, dass die Konversation mit dem Benutzer fehlgeschlagen ist.

  Dies kann verschiedene Gründe haben, z. B. weil der Benutzer nicht angemessen auf Anfragen des Service reagiert (Sie können konfigurieren, wie oft Amazon Lex einen Benutzer zur Eingabe bestimmter Informationen auffordern kann), oder wenn die Lambda-Funktion die Absicht nicht erfüllt.

Zulässige Werte: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedInputTranscript

Der Text, der zur Bearbeitung der Anfrage verwendet wurde.

Wurde ein Audio-Stream eingegeben, enthält das Feld `encodedInputTranscript` den aus dem Audio-Stream extrahierten Text. Dies ist der Text, der tatsächlich verarbeitet wurde, um Absichten und Slot-Werte zu erkennen. Anhand dieser Informationen können Sie feststellen, ob Amazon Lex das von Ihnen gesendete Audio korrekt verarbeitet.

Das `encodedInputTranscript` Feld ist Base-64-codiert. Sie müssen das Feld dekodieren, bevor Sie den Wert verwenden können.

encodedMessage

Die Nachricht, die dem Benutzer übermittelt werden soll. Die Nachricht kann aus der Konfiguration des Bots oder aus einer Lambda-Funktion stammen.

Wenn die Absicht nicht mit einer Lambda-Funktion konfiguriert ist oder wenn die Lambda-Funktion `Delegate` als Antwort zurückgegeben wurde, entscheidet Amazon Lex über die nächste Vorgehensweise und wählt auf der Grundlage des aktuellen Interaktionskontextes eine entsprechende Nachricht aus der Konfiguration des Bots aus. `dialogAction.type` Wenn Amazon Lex beispielsweise Benutzereingaben nicht verstehen kann, verwendet es eine Klarstellungsaufforderung.

Wenn Sie eine Absicht erstellen, können Sie Nachrichten Gruppen zuweisen. Wenn Nachrichten Gruppen zugewiesen werden, gibt Amazon Lex in der Antwort eine Nachricht von jeder Gruppe zurück. Das Nachrichtenfeld ist eine maskierte JSON-Zeichenfolge, die die Nachrichten enthält. Weitere Hinweise zur Struktur der zurückgegebenen JSON-Zeichenfolge finden Sie unterUnterstützte Mitteilungsformate.

Wenn die Lambda-Funktion eine Nachricht zurückgibt, leitet Amazon Lex sie in seiner Antwort an den Client weiter.

Das `encodedMessage` Feld ist Base-64-codiert. Sie müssen das Feld dekodieren, bevor Sie den Wert verwenden können.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1366.

inputTranscript

Dieser Header ist veraltet.

Der Text, der zur Bearbeitung der Anfrage verwendet wurde.

Sie können dieses Feld nur in den Gebietsschemas de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR und it-IT verwenden. `inputTranscript`In allen anderen Gebietsschemas ist das Feld Null. Sie sollten stattdessen das `encodedInputTranscript` Feld verwenden.

Wurde ein Audio-Stream eingegeben, enthält das Feld `inputTranscript` den aus dem Audio-Stream extrahierten Text. Dies ist der Text, der tatsächlich verarbeitet wurde, um Absichten und Slot-Werte zu erkennen. Anhand dieser Informationen können Sie feststellen, ob Amazon Lex das von Ihnen gesendete Audio korrekt verarbeitet.

intentName

Aktuelle Benutzerabsicht, die Amazon Lex bekannt ist.

message

Dieser Header ist veraltet.

Sie können dieses Feld nur in den Gebietsschemas de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR und it-IT verwenden. `message`In allen anderen Gebietsschemas ist das Feld Null. Sie sollten stattdessen das `encodedMessage` Feld verwenden.

Die Nachricht, die dem Benutzer übermittelt werden soll. Die Nachricht kann aus der Konfiguration des Bots oder aus einer Lambda-Funktion stammen.

Wenn die Absicht nicht mit einer Lambda-Funktion konfiguriert ist oder wenn die Lambda-Funktion `Delegate` als Antwort zurückgegeben wurde, entscheidet Amazon Lex über die nächste Vorgehensweise und wählt auf der Grundlage des aktuellen Interaktionskontextes eine entsprechende Nachricht aus der Konfiguration des Bots aus. `dialogAction.type` Wenn Amazon Lex beispielsweise Benutzereingaben nicht verstehen kann, verwendet es eine Klarstellungsaufforderung.

Wenn Sie eine Absicht erstellen, können Sie Nachrichten Gruppen zuweisen. Wenn Nachrichten Gruppen zugewiesen werden, gibt Amazon Lex in der Antwort eine Nachricht von jeder Gruppe zurück. Das Nachrichtenfeld ist eine maskierte JSON-Zeichenfolge, die die Nachrichten enthält. Weitere Hinweise zur Struktur der zurückgegebenen JSON-Zeichenfolge finden Sie unterUnterstützte Mitteilungsformate.

Wenn die Lambda-Funktion eine Nachricht zurückgibt, leitet Amazon Lex sie in seiner Antwort an den Client weiter.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 1024 Zeichen.

messageFormat

Das Format der Antwortnachricht. Einer der folgenden Werte:

- `PlainText`- Die Nachricht enthält einfachen UTF-8-Text.
- `CustomPayload`- Die Nachricht ist ein benutzerdefiniertes Format für den Client.
- `SSML`- Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `Composite`- Die Nachricht enthält ein maskiertes JSON-Objekt, das eine oder mehrere Nachrichten aus den Gruppen enthält, denen die Nachrichten bei der Erstellung der Absicht zugewiesen wurden.

Zulässige Werte: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Stellt eine Bewertung bereit, die angibt, wie sicher Amazon Lex ist, dass die zurückgegebene Absicht der Absicht des Benutzers entspricht. Die Punktzahl liegt zwischen 0,0 und 1,0.

Die Punktzahl ist eine relative Punktzahl, keine absolute Punktzahl. Die Punktzahl kann sich aufgrund von Verbesserungen an Amazon Lex ändern.

sentimentResponse

Das in einer Äußerung zum Ausdruck gebrachte Gefühl.

Wenn der Bot so konfiguriert ist, dass er Äußerungen zur Stimmungsanalyse an Amazon Comprehend sendet, enthält dieses Feld das Ergebnis der Analyse.

sessionAttributes

Karte von Schlüssel/Wert-Paaren, die die sitzungsspezifischen Kontextinformationen darstellen.

sessionId

Die eindeutige Kennung für die Sitzung.

slots

Zuordnung von null oder mehr Intent-Slots (Name/Wert-Paare), die Amazon Lex anhand der Benutzereingabe während der Konversation erkannt hat. Das Feld ist Base-64-codiert.

Amazon Lex erstellt eine Auflösungsliste mit wahrscheinlichen Werten für einen Slot. Der zurückgegebene Wert wird durch den `valueSelectionStrategy` ausgewählten Slot-Typ bestimmt, der bei der Erstellung oder Aktualisierung des Slot-Typs ausgewählt wurde. Wenn auf gesetzt `valueSelectionStrategy` ist`ORIGINAL_VALUE`, wird der vom Benutzer angegebene Wert zurückgegeben, wenn der Benutzerwert den Slot-Werten ähnelt. Wenn auf `TOP_RESOLUTION` Amazon Lex gesetzt `valueSelectionStrategy` ist, wird der erste Wert in der Auflösungsliste zurückgegeben oder, falls es keine Auflösungsliste gibt, Null. Wenn Sie kein a angeben`valueSelectionStrategy`, ist die Standardeinstellung`ORIGINAL_VALUE`.

slotToElicit

Wenn der `dialogState` Wert ist`ElicitSlot`, wird der Name des Slots zurückgegeben, für den Amazon Lex einen Wert ermittelt.

Die Antwort gibt folgendes als HTTP-Hauptteil zurück.

audioStream

Die Aufforderung (oder Aussage), die dem Benutzer übermittelt werden soll. Dies basiert auf der Konfiguration und dem Kontext des Bots. Wenn Amazon Lex beispielsweise die Benutzerabsicht nicht verstanden hat, sendet es die für den Bot `clarificationPrompt` konfigurierten. Wenn die Absicht bestätigt werden muss, bevor die Erfüllungsaktion ausgeführt werden kann, sendet es die`confirmationPrompt`. Ein anderes Beispiel: Angenommen, die Lambda-Funktion hat die Absicht erfolgreich erfüllt und eine Nachricht zur Übermittlung an den Benutzer gesendet. Dann sendet Amazon Lex diese Nachricht in der Antwort.

Fehler

BadGatewayException

Entweder wird der Amazon Lex Lex-Bot noch erstellt, oder einer der abhängigen Dienste (Amazon Polly, AWS Lambda) ist mit einem internen Servicefehler ausgefallen.

HTTP-Statuscode: 502

BadRequestException

Die Überprüfung der Anfrage ist fehlgeschlagen, es gibt keine brauchbare Nachricht im Kontext, oder der Bot-Build ist fehlgeschlagen, ist noch in Bearbeitung oder enthält noch nicht erstellte Änderungen.

HTTP Status Code: 400

ConflictException

Zwei Kunden verwenden dasselbe AWS-Konto, denselben Amazon Lex Lex-Bot und dieselbe Benutzer-ID.

HTTP-Statuscode: 409

DependencyFailedException

Eine der Abhängigkeiten, wie AWS Lambda oder Amazon Polly, hat eine Ausnahme ausgelöst. Zum Beispiel

- Wenn Amazon Lex nicht über ausreichende Berechtigungen verfügt, um eine Lambda-Funktion aufzurufen.
- Wenn die Ausführung einer Lambda-Funktion länger als 30 Sekunden dauert.
- Wenn eine Fulfillment-Lambda-Funktion eine `Delegate` Dialogaktion zurückgibt, ohne Slot-Werte zu entfernen.

HTTP-Statuscode: 424

InternalFailureException

Interner Dienstfehler. Versuchen Sie den Anruf erneut.

HTTP Status Code: 500

LimitExceededException

Ein Limit wurde überschritten.

HTTP-Statuscode: 429

LoopDetectedException

Diese Ausnahme wird nicht verwendet.

HTTP-Statuscode: 508

NotAcceptableException

Der Accept-Header in der Anfrage hat keinen gültigen Wert.

HTTP-Statuscode: 406

NotFoundException

Die Ressource (z. B. der Amazon Lex Lex-Bot oder ein Alias), auf die verwiesen wird, wurde nicht gefunden.

HTTP Status Code: 404

RequestTimeoutException

Die Eingabesprache ist zu lang.

HTTP-Statuscode: 408

UnsupportedMediaTypeException

Der Content-Type-Header (`PostContent`API) hat einen ungültigen Wert.

HTTP-Statuscode: 415

Beispiele

Beispiel 1

In dieser Anfrage identifiziert der URI einen Bot (Traffic), eine Bot-Version ($LATEST) und einen Endbenutzernamen (someuser). Der `Content-Type` Header identifiziert das Format des Audios im Hauptteil. Amazon Lex unterstützt auch andere Formate. Um Audio bei Bedarf von einem Format in ein anderes zu konvertieren, können Sie die Open-Source-Software SoX verwenden. Sie geben das Format an, in dem Sie die Antwort erhalten möchten, indem Sie den `Accept` HTTP-Header hinzufügen.

In der Antwort zeigt der `x-amz-lex-message` Header die Antwort, die Amazon Lex zurückgegeben hat. Der Client kann diese Antwort dann an den Benutzer senden. Dieselbe Nachricht wird im Audio/MPEG-Format durch Blockcodierung (wie gewünscht) gesendet.

Beispielanforderung

```
"POST /bot/Traffic/alias/$LATEST/user/someuser/content HTTP/1.1[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
```

```
"Content-Type: audio/x-l16; channel-count=1; sample-rate=16000f[\r][\n]"
"Accept: audio/mpeg[\r][\n]"
"Host: runtime.lex.us-east-1.amazonaws.com[\r][\n]"
"Authorization: AWS4-HMAC-SHA256 Credential=BLANKED_OUT/20161230/us-east-1/lex/
aws4_request,
SignedHeaders=accept;content-type;host;x-amz-content-sha256;x-amz-date;x-amz-lex-
session-attributes,
 Signature=78ca5b54ea3f64a17ff7522de02cd90a9acd2365b45a9ce9b96ea105bb1c7ec2[\r][\n]"
"X-Amz-Date: 20161230T181426Z[\r][\n]"
"X-Amz-Content-Sha256:
 e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855[\r][\n]"
"Transfer-Encoding: chunked[\r][\n]"
"Connection: Keep-Alive[\r][\n]"
"User-Agent: Apache-HttpClient/4.5.x (Java/1.8.0_112)[\r][\n]"
"Accept-Encoding: gzip,deflate[\r][\n]"
"[\r][\n]"
"1000[\r][\n]"
"[0x7][0x0][0x7][0x0][\n]"
"[0x0][0x7][0x0][0xfc][0xff][\n]"
"[0x0][\n]"
…
```

Beispielantwort

```
"HTTP/1.1 200 OK[\r][\n]"
"x-amzn-RequestId: cc8b34af-cebb-11e6-a35c-55f3a992f28d[\r][\n]"
"x-amz-lex-message: Sorry, can you repeat that?[\r][\n]"
"x-amz-lex-dialog-state: ElicitIntent[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
"Content-Type: audio/mpeg[\r][\n]"
"Transfer-Encoding: chunked[\r][\n]"
"Date: Fri, 30 Dec 2016 18:14:28 GMT[\r][\n]"
"[\r][\n]"
"2000[\r][\n]"
"ID3[0x4][0x0][0x0][0x0][0x0][0x0]#TSSE[0x0][0x0][0x0][0xf][0x0][0x0]
[0x3]Lavf57.41.100[0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0xff]
[0xf3]`[0xc4][0x0][0x1b]{[0x8d][0xe8][0x1]C[0x18][0x1][0x0]J[0xe0]`b[0xdd][0xd1]
[0xb][0xfd][0x11][0xdf][0xfe]";[0xbb][0xbb][0x9f][0xee][0xee][0xee][0xee]|DDD/[0xff]
[0xff][0xff][0xff]www?D[0xf7]w^?[0xff][0xfa]h[0x88][0x85][0xfe][0x88][0x88][0x88]
[[0xa2]'[0xff][0xfa]"{[0x9f][0xe8][0x88]]D[0xeb][0xbb][0xbb][0xa2]!u[0xfd][0xdd][0xdf]
[0x88][0x94][0x0]F[0xef][0xa1]8[0x0][0x82]w[0x88]N[0x0][0x0][0x9b][0xbb][0xe8][0xe
…
```

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der AWS sprachspezifischen SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PostText

Service: Amazon Lex Runtime Service

Sendet Benutzereingaben an Amazon Lex. Client-Anwendungen können diese API verwenden, um zur Laufzeit Anfragen an Amazon Lex zu senden. Amazon Lex interpretiert dann die Benutzereingaben mithilfe des Machine-Learning-Modells, das es für den Bot erstellt hat.

Als Antwort gibt Amazon Lex den nächsten zurück`message`, um dem Benutzer eine Option `responseCard` zur Anzeige mitzuteilen. Betrachten Sie die folgenden Beispielnachrichten:

- Bei einer Benutzereingabe „Ich hätte gerne eine Pizza“ gibt Amazon Lex möglicherweise eine Antwort mit einer Nachricht zurück, die Slot-Daten ausgibt (z. B. PizzaSize): „Welche Pizzagröße hätten Sie gerne?“
- Nachdem der Benutzer alle Informationen zur Pizzabestellung eingegeben hat, sendet Amazon Lex möglicherweise eine Antwort mit der Nachricht zurück, dass der Benutzer die Bestätigung „Mit der Pizzabestellung fortfahren?“ erhält.
- Nachdem der Benutzer auf eine Bestätigungsaufforderung mit „Ja“ geantwortet hat, gibt Amazon Lex möglicherweise eine Schlussfolgerung zurück: „Danke, Ihre Käsepizza wurde bestellt.“.

Nicht alle Amazon Lex Lex-Nachrichten erfordern eine Benutzerantwort. Für eine Schlußfolgerung ist beispielsweise keine Antwort erforderlich. Für einige Nachrichten ist nur eine Benutzerantwort mit „Ja“ oder „Nein“ erforderlich. Darüber hinaus bietet Amazon Lex zusätzlichen Kontext zu der `message` Nachricht in der Antwort, den Sie verwenden können, um das Kundenverhalten zu verbessern, z. B. um die entsprechende Client-Benutzeroberfläche anzuzeigen. Dies sind die `slots` Felder `slotToElicitdialogState`,`intentName`, und in der Antwort. Betrachten Sie die folgenden Beispiele:

- Wenn die Nachricht Slot-Daten abrufen soll, gibt Amazon Lex die folgenden Kontextinformationen zurück:
  - `dialogState`eingestellt auf ElicitSlot
  - `intentName`auf den Namen der Absicht im aktuellen Kontext gesetzt
  - `slotToElicit`auf den Slot-Namen gesetzt, für den `message` Informationen abgerufen werden
  - `slots`wird auf eine Karte von Slots gesetzt, die für den Intent konfiguriert sind und deren Werte aktuell bekannt sind
- Handelt es sich bei der Meldung um eine Bestätigungsaufforderung, `dialogState` ist der Wert auf Null gesetzt ConfirmIntent und `SlotToElicit` wird auf Null gesetzt.

- Handelt es sich bei der Meldung um eine Klarstellungsaufforderung (für die Absicht konfiguriert), die darauf hinweist, dass die Benutzerabsicht nicht verstanden wurde, `dialogState` wird die auf Null gesetzt ElicitIntent und `slotToElicit` auf Null gesetzt.

Darüber hinaus sendet Amazon Lex auch Ihre anwendungsspezifische `sessionAttributes` zurück. Weitere Informationen finden Sie unter Konversationskontext verwalten.

Anforderungssyntax

```
POST /bot/botName/alias/botAlias/user/userId/text HTTP/1.1
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "inputText": "string",
   "requestAttributes": {
      "string" : "string"
   },
   "sessionAttributes": {
      "string" : "string"
   }
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

botAlias

Der Alias des Amazon Lex Lex-Bot.

Erforderlich: Ja

botName

Der Name des Amazon Lex-Bots.

Erforderlich: Ja

userId

Die ID des Benutzers der Client-Anwendung. Amazon Lex verwendet dies, um die Konversation eines Benutzers mit Ihrem Bot zu identifizieren. Zur Laufzeit muss jede Anfrage das `userID` Feld enthalten.

Bei der Entscheidung, welche Benutzer-ID für Ihre Anwendung verwendet werden soll, sollten Sie die folgenden Faktoren berücksichtigen.

- Das `userID` Feld darf keine persönlich identifizierbaren Informationen des Benutzers enthalten, z. B. Namen, persönliche Identifikationsnummern oder andere persönliche Daten des Endbenutzers.
- Wenn Sie möchten, dass ein Benutzer eine Konversation auf einem Gerät beginnt und auf einem anderen Gerät fortsetzt, verwenden Sie eine benutzerspezifische Kennung.
- Wenn Sie möchten, dass derselbe Benutzer zwei unabhängige Konversationen auf zwei verschiedenen Geräten führen kann, wählen Sie eine gerätespezifische Kennung.
- Ein Benutzer kann nicht zwei unabhängige Konversationen mit zwei verschiedenen Versionen desselben Bots führen. Beispielsweise kann ein Benutzer keine Konversation mit den PROD- und BETA-Versionen desselben Bots führen. Wenn Sie davon ausgehen, dass ein Benutzer beispielsweise beim Testen eine Konversation mit zwei verschiedenen Versionen führen muss, fügen Sie den Bot-Alias in die Benutzer-ID ein, um die beiden Konversationen voneinander zu trennen.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Pattern: `[0-9a-zA-Z._:-]+`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

activeContexts

Eine Liste der für die Anfrage aktiven Kontexte. Ein Kontext kann aktiviert werden, wenn eine vorherige Absicht erfüllt ist, oder indem der Kontext in die Anfrage aufgenommen wird.

Wenn Sie keine Liste von Kontexten angeben, verwendet Amazon Lex die aktuelle Liste der Kontexte für die Sitzung. Wenn Sie eine leere Liste angeben, werden alle Kontexte für die Sitzung gelöscht.

Typ: Array von ActiveContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 50 Elemente.

Erforderlich: Nein

inputText

Der Text, den der Benutzer eingegeben hat (Amazon Lex interpretiert diesen Text).

Wenn Sie die AWS-CLI verwenden, können Sie im `--input-text` Parameter keine URL übergeben. Übergeben Sie die URL stattdessen mithilfe des `--cli-input-json` Parameters.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge von 1 024.

Erforderlich: Ja

requestAttributes

Anforderungsspezifische Informationen, die zwischen Amazon Lex und einer Client-Anwendung übertragen werden.

Der Namespace `x-amz-lex:` ist für spezielle Attribute reserviert. Erstellen Sie keine Anforderungsattribute mit dem Präfix`x-amz-lex:`.

Weitere Informationen finden Sie unter Anforderungsattribute festlegen.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Erforderlich: Nein

sessionAttributes

Anwendungsspezifische Informationen, die zwischen Amazon Lex und einer Client-Anwendung ausgetauscht werden.

Weitere Informationen finden Sie unter Sitzungsattribute einrichten.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "alternativeIntents": [
      {
         "intentName": "string",
         "nluIntentConfidence": {
            "score": number
         },
         "slots": {
            "string" : "string"
         }
      }
   ],
   "botVersion": "string",
   "dialogState": "string",
   "intentName": "string",
```

```
   "message": "string",
   "messageFormat": "string",
   "nluIntentConfidence": {
      "score": number
   },
   "responseCard": {
      "contentType": "string",
      "genericAttachments": [
         {
            "attachmentLinkUrl": "string",
            "buttons": [
               {
                  "text": "string",
                  "value": "string"
               }
            ],
            "imageUrl": "string",
            "subTitle": "string",
            "title": "string"
         }
      ],
      "version": "string"
   },
   "sentimentResponse": {
      "sentimentLabel": "string",
      "sentimentScore": "string"
   },
   "sessionAttributes": {
      "string" : "string"
   },
   "sessionId": "string",
   "slots": {
      "string" : "string"
   },
   "slotToElicit": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Service im JSON-Format zurückgegeben.

activeContexts

Eine Liste der aktiven Kontexte für die Sitzung. Ein Kontext kann festgelegt werden, wenn eine Absicht erfüllt ist, oder durch Aufrufen der `PutSession` Operation `PostContentPostText`, oder.

Sie können einen Kontext verwenden, um die Absichten zu steuern, die einer Absicht folgen können, oder um den Betrieb Ihrer Anwendung zu ändern.

Typ: Array von ActiveContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 50 Elemente.

alternativeIntents

Ein bis vier alternative Absichten, die auf die Absicht des Benutzers zutreffen können.

Jede Alternative beinhaltet eine Bewertung, die angibt, wie sicher Amazon Lex ist, dass die Absicht mit der Absicht des Benutzers übereinstimmt. Die Absichten sind nach dem Konfidenzwert sortiert.

Typ: Array von PredictedIntent-Objekten

Array-Mitglieder: Maximale Anzahl von 4 Elementen.

botVersion

Die Version des Bots, der auf die Konversation geantwortet hat. Anhand dieser Informationen können Sie feststellen, ob eine Version eines Bots besser abschneidet als eine andere Version.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[0-9]+|\$LATEST`

dialogState

Identifiziert den aktuellen Status der Benutzerinteraktion. Amazon Lex gibt einen der folgenden Werte als zurück`dialogState`. Der Kunde kann diese Informationen optional verwenden, um die Benutzeroberfläche anzupassen.

- `ElicitIntent`- Amazon Lex möchte Benutzerabsichten wecken.

Beispielsweise könnte ein Benutzer eine Absicht äußern („Ich möchte eine Pizza bestellen“). Wenn Amazon Lex die Benutzerabsicht aus dieser Äußerung nicht ableiten kann, gibt es diesen DialogState zurück.

- `ConfirmIntent`- Amazon Lex erwartet eine Antwort mit „Ja“ oder „Nein“.

  Amazon Lex möchte beispielsweise eine Bestätigung durch den Benutzer, bevor eine Absicht erfüllt wird.

  Statt eines einfachen „Ja“ oder „Nein“ könnte ein Benutzer mit zusätzlichen Informationen antworten. Zum Beispiel „Ja, aber mach Pizza mit dicker Kruste daraus“ oder „Nein, ich möchte ein Getränk bestellen“. Amazon Lex kann solche zusätzlichen Informationen verarbeiten (in diesen Beispielen den Slot-Wert für den Krustentyp aktualisieren oder die Absicht von OrderPizza zu ändern OrderDrink).
- `ElicitSlot`- Amazon Lex erwartet einen Slot-Wert für die aktuelle Absicht.

  Nehmen wir zum Beispiel an, dass Amazon Lex in der Antwort die folgende Nachricht sendet: „Welche Pizzagröße hätten Sie gerne?“. Ein Benutzer könnte mit dem Slot-Wert antworten (z. B. „mittel“). Der Benutzer kann in der Antwort auch zusätzliche Informationen angeben (z. B. „Pizza mit mitteldicker Kruste“). Amazon Lex kann solche zusätzlichen Informationen angemessen verarbeiten.
- `Fulfilled`- Übermittelt, dass die für die Absicht konfigurierte Lambda-Funktion die Absicht erfolgreich erfüllt hat.
- `ReadyForFulfillment`- Vermittelt, dass der Kunde die Absicht erfüllen muss.
- `Failed`- Übermittelt, dass die Konversation mit dem Benutzer fehlgeschlagen ist.

  Dies kann verschiedene Gründe haben, z. B. weil der Benutzer nicht angemessen auf Anfragen des Service geantwortet hat (Sie können konfigurieren, wie oft Amazon Lex einen Benutzer zur Eingabe bestimmter Informationen auffordern kann) oder dass die Lambda-Funktion die Absicht nicht erfüllt hat.

Typ: Zeichenfolge

Zulässige Werte: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

intentName

Die aktuelle Benutzerabsicht, die Amazon Lex bekannt ist.

Typ: Zeichenfolge

message

Die Nachricht, die dem Benutzer übermittelt werden soll. Die Nachricht kann aus der Konfiguration des Bots oder aus einer Lambda-Funktion stammen.

Wenn die Absicht nicht mit einer Lambda-Funktion konfiguriert ist oder wenn die Lambda-Funktion `Delegate` als Antwort zurückgegeben wird, entscheidet Amazon Lex über die nächste Vorgehensweise und wählt auf der Grundlage des aktuellen Interaktionskontextes eine entsprechende Nachricht aus der Konfiguration des Bots aus. `dialogAction.type` Wenn Amazon Lex beispielsweise Benutzereingaben nicht verstehen kann, verwendet es eine Klarstellungsaufforderung.

Wenn Sie eine Absicht erstellen, können Sie Nachrichten Gruppen zuweisen. Wenn Nachrichten Gruppen zugewiesen werden, gibt Amazon Lex in der Antwort eine Nachricht von jeder Gruppe zurück. Das Nachrichtenfeld ist eine maskierte JSON-Zeichenfolge, die die Nachrichten enthält. Weitere Hinweise zur Struktur der zurückgegebenen JSON-Zeichenfolge finden Sie unterUnterstützte Mitteilungsformate.

Wenn die Lambda-Funktion eine Nachricht zurückgibt, leitet Amazon Lex sie in seiner Antwort an den Client weiter.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 1024 Zeichen.

messageFormat

Das Format der Antwortnachricht. Einer der folgenden Werte:

- `PlainText`- Die Nachricht enthält einfachen UTF-8-Text.
- `CustomPayload`- Die Nachricht ist ein benutzerdefiniertes Format, das von der Lambda-Funktion definiert wird.
- `SSML`- Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `Composite`- Die Nachricht enthält ein maskiertes JSON-Objekt, das eine oder mehrere Nachrichten aus den Gruppen enthält, denen die Nachrichten bei der Erstellung der Absicht zugewiesen wurden.

Typ: Zeichenfolge

Zulässige Werte: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Stellt eine Bewertung bereit, die angibt, wie sicher Amazon Lex ist, dass die zurückgegebene Absicht der Absicht des Benutzers entspricht. Die Punktzahl liegt zwischen 0,0 und 1,0. Weitere Informationen finden Sie unter Konfidenzwerte.

Der Wert ist ein relativer Wert, kein absoluter Wert. Die Punktzahl kann sich aufgrund von Verbesserungen an Amazon Lex ändern.

Typ: IntentConfidence Objekt

responseCard

Stellt die Optionen dar, die der Benutzer hat, um auf die aktuelle Aufforderung zu antworten. Die Antwortkarte kann aus der Bot-Konfiguration (wählen Sie in der Amazon Lex Lex-Konsole die Einstellungsschaltfläche neben einem Steckplatz) oder aus einem Code-Hook (Lambda-Funktion) stammen.

Typ: ResponseCard Objekt

sentimentResponse

Das in einer Äußerung zum Ausdruck gebrachte Gefühl.

Wenn der Bot so konfiguriert ist, dass er Äußerungen zur Stimmungsanalyse an Amazon Comprehend sendet, enthält dieses Feld das Ergebnis der Analyse.

Typ: SentimentResponse Objekt

sessionAttributes

Eine Zuordnung von Schlüssel-Wert-Paaren, die die sitzungsspezifischen Kontextinformationen darstellen.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

sessionId

Eindeutiger Bezeichner für die Sitzung.

Typ: Zeichenfolge

slots

Die Intent-Slots, die Amazon Lex anhand der Benutzereingaben in der Konversation erkannt hat.

Amazon Lex erstellt eine Auflösungsliste mit wahrscheinlichen Werten für einen Slot. Der zurückgegebene Wert wird durch den `valueSelectionStrategy` ausgewählten Slot-Typ bestimmt, der bei der Erstellung oder Aktualisierung des Slot-Typs ausgewählt wurde. Wenn auf gesetzt `valueSelectionStrategy` ist`ORIGINAL_VALUE`, wird der vom Benutzer angegebene Wert zurückgegeben, wenn der Benutzerwert den Slot-Werten ähnelt. Wenn auf `TOP_RESOLUTION` Amazon Lex gesetzt `valueSelectionStrategy` ist, wird der erste Wert in der Auflösungsliste zurückgegeben oder, falls es keine Auflösungsliste gibt, Null. Wenn Sie kein a angeben`valueSelectionStrategy`, ist die Standardeinstellung`ORIGINAL_VALUE`.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

slotToElicit

Wenn der `dialogState` Wert ist`ElicitSlot`, wird der Name des Slots zurückgegeben, für den Amazon Lex einen Wert ermittelt.

Typ: Zeichenfolge

Fehler

BadGatewayException

Entweder wird der Amazon Lex Lex-Bot noch erstellt, oder einer der abhängigen Dienste (Amazon Polly, AWS Lambda) ist mit einem internen Servicefehler ausgefallen.

HTTP-Statuscode: 502

BadRequestException

Die Überprüfung der Anfrage ist fehlgeschlagen, es gibt keine brauchbare Nachricht im Kontext, oder der Bot-Build ist fehlgeschlagen, ist noch in Bearbeitung oder enthält noch nicht erstellte Änderungen.

HTTP Status Code: 400

ConflictException

Zwei Kunden verwenden dasselbe AWS-Konto, denselben Amazon Lex Lex-Bot und dieselbe Benutzer-ID.

HTTP-Statuscode: 409

DependencyFailedException

Eine der Abhängigkeiten, wie AWS Lambda oder Amazon Polly, hat eine Ausnahme ausgelöst. Zum Beispiel

- Wenn Amazon Lex nicht über ausreichende Berechtigungen verfügt, um eine Lambda-Funktion aufzurufen.
- Wenn die Ausführung einer Lambda-Funktion länger als 30 Sekunden dauert.
- Wenn eine Fulfillment-Lambda-Funktion eine `Delegate` Dialogaktion zurückgibt, ohne Slot-Werte zu entfernen.

HTTP-Statuscode: 424

InternalFailureException

Interner Dienstfehler. Versuchen Sie den Anruf erneut.

HTTP Status Code: 500

LimitExceededException

Ein Limit wurde überschritten.

HTTP-Statuscode: 429

LoopDetectedException

Diese Ausnahme wird nicht verwendet.

HTTP-Statuscode: 508

NotFoundException

Die Ressource (z. B. der Amazon Lex Lex-Bot oder ein Alias), auf die verwiesen wird, wurde nicht gefunden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

PutSession

Service: Amazon Lex Runtime Service

Erstellt eine neue Sitzung oder ändert eine bestehende Sitzung mit einem Amazon Lex Bot. Verwenden Sie diesen Vorgang, damit Ihre Anwendung den Status des Bots festlegen kann.

Weitere Informationen finden Sie im Artikel zum Verwalten von Sitzungen.

Anforderungssyntax

```
POST /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
Accept: accept
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "dialogAction": {
      "fulfillmentState": "string",
      "intentName": "string",
      "message": "string",
      "messageFormat": "string",
      "slots": {
         "string" : "string"
      },
      "slotToElicit": "string",
      "type": "string"
   },
   "recentIntentSummaryView": [
      {
         "checkpointLabel": "string",
         "confirmationStatus": "string",
         "dialogActionType": "string",
```

```
        "fulfillmentState": "string",
        "intentName": "string",
        "slots": {
           "string" : "string"
        },
        "slotToElicit": "string"
     }
  ],
  "sessionAttributes": {
     "string" : "string"
  }
}
```

URI-Anfrageparameter

Die Anforderung verwendet die folgenden URI-Parameter.

accept

Die Nachricht, die Amazon Lex in der Antwort zurückgibt, kann je nach Wert dieses Felds entweder text- oder sprachbasiert sein.

- Wenn der Wert ist`text/plain; charset=utf-8`, gibt Amazon Lex Text in der Antwort zurück.
- Beginnt der Wert mit`audio/`, gibt Amazon Lex in der Antwort Sprache zurück. Amazon Lex verwendet Amazon Polly, um die Sprache in der von Ihnen angegebenen Konfiguration zu generieren. Wenn Sie beispielsweise `audio/mpeg` als Wert angeben, gibt Amazon Lex Sprache im MPEG-Format zurück.
- Wenn der Wert „ist“`audio/pcm`, wird die Sprache `audio/pcm` im 16-Bit-Little-Endian-Format zurückgegeben.
- Die folgenden Werte werden akzeptiert:
  - `audio/mpeg`
  - `audio/ogg`
  - `audio/pcm`
  - `audio/*`(standardmäßig mpeg)
  - `text/plain; charset=utf-8`

botAlias

Der Alias, der für den Bot verwendet wird, der die Sitzungsdaten enthält.

Erforderlich: Ja

botName

Der Name des Bots, der die Sitzungsdaten enthält.

Erforderlich: Ja

userId

Die ID des Benutzers der Client-Anwendung. Amazon Lex verwendet dies, um die Konversation eines Benutzers mit Ihrem Bot zu identifizieren.

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge beträgt 100 Zeichen.

Pattern: `[0-9a-zA-Z._:-]+`

Erforderlich: Ja

Anforderungstext

Die Anforderung akzeptiert die folgenden Daten im JSON-Format.

activeContexts

Eine Liste der für die Anfrage aktiven Kontexte. Ein Kontext kann aktiviert werden, wenn eine vorherige Absicht erfüllt ist, oder indem der Kontext in die Anfrage aufgenommen wird.

Wenn Sie keine Liste von Kontexten angeben, verwendet Amazon Lex die aktuelle Liste der Kontexte für die Sitzung. Wenn Sie eine leere Liste angeben, werden alle Kontexte für die Sitzung gelöscht.

Typ: Array von ActiveContext-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 50 Elemente.

Erforderlich: Nein

dialogAction

Legt die nächste Aktion fest, die der Bot ergreifen soll, um die Konversation abzuwickeln.

Typ: DialogAction Objekt

Erforderlich: Nein

recentIntentSummaryView

Eine Zusammenfassung der jüngsten Absichten für den Bot. Sie können die Ansicht mit der Zusammenfassung der Absichten verwenden, um eine Checkpoint-Bezeichnung für eine Absicht festzulegen und die Attribute von Absichten zu ändern. Sie können sie auch verwenden, um Objekte mit einer Zusammenfassung von Absichten zu entfernen oder der Liste hinzuzufügen.

Eine Absicht, die Sie ändern oder der Liste hinzufügen, muss für den Bot sinnvoll sein. Beispielsweise muss der Name der Absicht für den Bot gültig sein. Sie müssen gültige Werte angeben für:

- `intentName`
- Steckplatznamen
- `slotToElict`

Wenn Sie den `recentIntentSummaryView` Parameter in einer `PutSession` Anfrage senden, ersetzt der Inhalt der neuen Übersichtsansicht die alte Übersichtsansicht. Wenn eine `GetSession` Anfrage beispielsweise drei Absichten in der Übersichtsansicht zurückgibt und Sie `PutSession` mit einer Absicht in der Übersichtsansicht aufrufen, gibt der nächste Aufruf von nur eine Absicht zurück. `GetSession`

Typ: Array von IntentSummary-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Maximale Anzahl von 3 Elementen.

Erforderlich: Nein

sessionAttributes

Karte von Schlüssel/Wert-Paaren, die die sitzungsspezifischen Kontextinformationen darstellen. Es enthält Anwendungsinformationen, die zwischen Amazon Lex und einer Client-Anwendung ausgetauscht werden.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
HTTP/1.1 200
```

```
Content-Type: contentType
x-amz-lex-intent-name: intentName
x-amz-lex-slots: slots
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-message: message
x-amz-lex-encoded-message: encodedMessage
x-amz-lex-message-format: messageFormat
x-amz-lex-dialog-state: dialogState
x-amz-lex-slot-to-elicit: slotToElicit
x-amz-lex-session-id: sessionId
x-amz-lex-active-contexts: activeContexts

audioStream
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die Antwort gibt die folgenden HTTP-Header zurück.

activeContexts

Eine Liste der aktiven Kontexte für die Sitzung.

contentType

Inhaltstyp, wie im `Accept` HTTP-Header der Anfrage angegeben.

dialogState

- `ConfirmIntent`- Amazon Lex erwartet eine Antwort mit „Ja“ oder „Nein“ zur Bestätigung der Absicht, bevor eine Absicht erfüllt wird.
- `ElicitIntent`- Amazon Lex möchte die Absicht des Benutzers ermitteln.
- `ElicitSlot`- Amazon Lex erwartet den Wert eines Slots für die aktuelle Absicht.
- `Failed`- Übermittelt, dass die Konversation mit dem Benutzer fehlgeschlagen ist. Dies kann aus verschiedenen Gründen geschehen, z. B. wenn der Benutzer nicht angemessen auf Anfragen des Dienstes reagiert oder wenn die Lambda-Funktion die Absicht nicht erfüllt.
- `Fulfilled`- Übermittelt, dass die Lambda-Funktion die Absicht erfolgreich erfüllt hat.
- `ReadyForFulfillment`- Vermittelt, dass der Kunde die Absicht erfüllen muss.

Zulässige Werte: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedMessage

Die nächste Nachricht, die dem Benutzer angezeigt werden soll.

Das `encodedMessage` Feld ist Base-64-codiert. Sie müssen das Feld dekodieren, bevor Sie den Wert verwenden können.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1366.

intentName

Der Name der aktuellen Absicht.

message

Dieser Header ist veraltet.

Die nächste Nachricht, die dem Benutzer angezeigt werden sollte.

Sie können dieses Feld nur in den Gebietsschemas de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR und it-IT verwenden. `message`In allen anderen Gebietsschemas ist das Feld Null. Sie sollten stattdessen das `encodedMessage` Feld verwenden.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 1024 Zeichen.

messageFormat

Das Format der Antwortnachricht. Einer der folgenden Werte:

- `PlainText`- Die Nachricht enthält einfachen UTF-8-Text.
- `CustomPayload`- Die Nachricht ist ein benutzerdefiniertes Format für den Client.
- `SSML`- Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `Composite`- Die Nachricht enthält ein maskiertes JSON-Objekt, das eine oder mehrere Nachrichten aus den Gruppen enthält, denen die Nachrichten bei der Erstellung der Absicht zugewiesen wurden.

Zulässige Werte: `PlainText | CustomPayload | SSML | Composite`

sessionAttributes

Karte von Schlüssel/Wert-Paaren, die sitzungsspezifische Kontextinformationen darstellen.

sessionId

Eindeutiger Bezeichner für die Sitzung.

slots

Zuordnung von null oder mehr Intent-Slots, die Amazon Lex anhand der Benutzereingabe während der Konversation erkannt hat.

Amazon Lex erstellt eine Auflösungsliste mit wahrscheinlichen Werten für einen Slot. Der zurückgegebene Wert wird durch den `valueSelectionStrategy` ausgewählten Slot-Typ bestimmt, der bei der Erstellung oder Aktualisierung des Slot-Typs ausgewählt wurde. Wenn auf gesetzt `valueSelectionStrategy` ist`ORIGINAL_VALUE`, wird der vom Benutzer angegebene Wert zurückgegeben, wenn der Benutzerwert den Slot-Werten ähnelt. Wenn auf `TOP_RESOLUTION` Amazon Lex gesetzt `valueSelectionStrategy` ist, wird der erste Wert in der Auflösungsliste zurückgegeben oder, falls es keine Auflösungsliste gibt, Null. Wenn Sie kein a angeben, ist `valueSelectionStrategy` die Standardeinstellung`ORIGINAL_VALUE`.

slotToElicit

Wenn ja`ElicitSlot`, `dialogState` wird der Name des Slots zurückgegeben, für den Amazon Lex einen Wert ermittelt.

Die Antwort gibt folgendes als HTTP-Hauptteil zurück.

audioStream

Die Audioversion der Nachricht, die dem Benutzer übermittelt werden soll.

Fehler

BadGatewayException

Entweder wird der Amazon Lex Lex-Bot noch erstellt, oder einer der abhängigen Dienste (Amazon Polly, AWS Lambda) ist mit einem internen Servicefehler ausgefallen.

HTTP-Statuscode: 502

BadRequestException

Die Überprüfung der Anfrage ist fehlgeschlagen, es gibt keine brauchbare Nachricht im Kontext, oder der Bot-Build ist fehlgeschlagen, ist noch in Bearbeitung oder enthält noch nicht erstellte Änderungen.

HTTP Status Code: 400

ConflictException

Zwei Kunden verwenden dasselbe AWS-Konto, denselben Amazon Lex Lex-Bot und dieselbe Benutzer-ID.

HTTP-Statuscode: 409

DependencyFailedException

Eine der Abhängigkeiten, wie AWS Lambda oder Amazon Polly, hat eine Ausnahme ausgelöst. Zum Beispiel

- Wenn Amazon Lex nicht über ausreichende Berechtigungen verfügt, um eine Lambda-Funktion aufzurufen.
- Wenn die Ausführung einer Lambda-Funktion länger als 30 Sekunden dauert.
- Wenn eine Fulfillment-Lambda-Funktion eine `Delegate` Dialogaktion zurückgibt, ohne Slot-Werte zu entfernen.

HTTP-Statuscode: 424

InternalFailureException

Interner Dienstfehler. Versuchen Sie den Anruf erneut.

HTTP Status Code: 500

LimitExceededException

Ein Limit wurde überschritten.

HTTP-Statuscode: 429

NotAcceptableException

Der Accept-Header in der Anfrage hat keinen gültigen Wert.

HTTP-Statuscode: 406

NotFoundException

Die Ressource (z. B. der Amazon Lex Lex-Bot oder ein Alias), auf die verwiesen wird, wurde nicht gefunden.

HTTP Status Code: 404

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK für JavaScript V3
- AWS SDK for PHP V3
- AWS SDK für Python
- AWS SDK for Ruby V3

Datentypen

Die folgenden Datentypen werden von Amazon Lex Model Building Service unterstützt:

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata
- BuiltinIntentMetadata
- BuiltinIntentSlot
- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent

- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message
- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt
- ResourceReference
- Slot
- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration
- Statement
- Tag
- UtteranceData
- UtteranceList

Die folgenden Datentypen werden von Amazon Lex Runtime Service unterstützt:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent

- ResponseCard
- SentimentResponse

Amazon Lex Modellbau Service

Die folgenden Datentypen werden von Amazon Lex Model Building Service unterstützt:

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata
- BuiltinIntentMetadata
- BuiltinIntentSlot
- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent
- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message
- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt
- ResourceReference
- Slot

- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration
- Statement
- Tag
- UtteranceData
- UtteranceList

BotAliasMetadata

Service: Amazon Lex Model Building Service

Stellt Informationen über einen Bot-Alias bereit.

Inhalt

botName

Der Name des Bots, auf den der Alias zeigt.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

botVersion

Die Version des Amazon Lex Lex-Bot, auf den der Alias verweist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

checksum

Prüfsumme des Bot-Alias.

Typ: Zeichenfolge

Erforderlich: Nein

conversationLogs

Einstellungen, die bestimmen, wie Amazon Lex Konversationsprotokolle für den Alias verwendet.

Typ: ConversationLogsResponse Objekt

Erforderlich: Nein

createdDate

Das Datum, an dem der Bot-Alias erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

description

Eine Beschreibung des Bot-Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

lastUpdatedDate

Das Datum, an dem der Bot-Alias aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

Erforderlich: Nein

name

Der Name des Bot-Alias.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

BotChannelAssociation

Service: Amazon Lex Model Building Service

Stellt eine Verbindung zwischen einem Amazon Lex Lex-Bot und einer externen Messaging-Plattform dar.

Inhalt

botAlias

Ein Alias, der auf die spezifische Version des Amazon Lex Lex-Bot verweist, zu dem diese Zuordnung hergestellt wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

botConfiguration

Stellt Informationen bereit, die für die Kommunikation mit der Messaging-Plattform erforderlich sind.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Maximale Anzahl von 10 Elementen.

Erforderlich: Nein

botName

Der Name des Amazon Lex Lex-Bots, zu dem diese Zuordnung hergestellt wird.

> **Note**
>
> Derzeit unterstützt Amazon Lex Verbindungen mit Facebook und Slack sowie Twilio.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

createdDate

Das Datum, an dem die Verbindung zwischen dem Amazon Lex Lex-Bot und dem Kanal erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

description

Eine Textbeschreibung der Assoziation, die Sie erstellen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

failureReason

Falls `status` ja`FAILED`, gibt Amazon Lex den Grund an, warum die Zuordnung nicht erstellt werden konnte.

Typ: Zeichenfolge

Erforderlich: Nein

name

Der Name der Assoziation zwischen dem Bot und dem Kanal.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

status

Der Status des Bot-Kanals.

- `CREATED`- Der Kanal wurde erstellt und ist einsatzbereit.
- `IN_PROGRESS`- Die Erstellung des Kanals ist im Gange.
- `FAILED`- Beim Erstellen des Kanals ist ein Fehler aufgetreten. Informationen zur Ursache des Fehlers finden Sie in dem `failureReason` Feld.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | CREATED | FAILED`

Erforderlich: Nein

type

Gibt die Art der Verbindung an, indem die Art des Kanals angegeben wird, der zwischen dem Amazon Lex Lex-Bot und der externen Messaging-Plattform eingerichtet wird.

Typ: Zeichenfolge

Zulässige Werte: `Facebook | Slack | Twilio-Sms | Kik`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

BotMetadata

Service: Amazon Lex Model Building Service

Stellt Informationen über einen Bot bereit.

Inhalt

createdDate

Das Datum, an dem der Bot erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

description

Eine Beschreibung des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

lastUpdatedDate

Das Datum, an dem der Bot aktualisiert wurde. Wenn Sie einen Bot erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

Erforderlich: Nein

name

Der Name des Bots.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

status

Der Status des Bots.

Typ: Zeichenfolge

Zulässige Werte: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

Erforderlich: Nein

version

Die Version des Bots. Für einen neuen Bot ist die Version immer gültig`$LATEST`.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

BuiltinIntentMetadata

Service: Amazon Lex Model Building Service

Stellt Metadaten für eine integrierte Absicht bereit.

Inhalt

signature

Eine eindeutige Kennung für die integrierte Absicht. Informationen zur Signatur für eine Absicht finden Sie unter Integrierte Standardabsichten im Alexa Skills Kit.

Typ: Zeichenfolge

Erforderlich: Nein

supportedLocales

Eine Liste von Kennungen für die Gebietsschemas, die von der Absicht unterstützt werden.

Typ: Zeichenfolgen-Array

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

BuiltinIntentSlot

Service: Amazon Lex Model Building Service

Stellt Informationen zu einem Slot bereit, der in einer integrierten Absicht verwendet wird.

Inhalt

name

Eine Liste der für den Intent definierten Slots.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

BuiltinSlotTypeMetadata

Service: Amazon Lex Model Building Service

Stellt Informationen über einen eingebauten Steckplatztyp bereit.

Inhalt

signature

Eine eindeutige Kennung für den integrierten Steckplatztyp. Die Signatur für einen Slot-Typ finden Sie unter Slot-Typ-Referenz im Alexa Skills Kit.

Typ: Zeichenfolge

Erforderlich: Nein

supportedLocales

Eine Liste der Zielgebietsschemas für den Slot.

Typ: Zeichenfolgen-Array

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CodeHook

Service: Amazon Lex Model Building Service

Spezifiziert eine Lambda-Funktion, die Anfragen an einen Bot verifiziert oder die Anfrage des Benutzers an einen Bot erfüllt.

Inhalt

messageVersion

Die Version der Anfrage-Antwort, die Amazon Lex verwenden soll, um Ihre Lambda-Funktion aufzurufen. Weitere Informationen finden Sie unter Verwenden von Lambda-Funktionen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 5.

Erforderlich: Ja

uri

Der Amazon-Ressourcenname (ARN) der -Lambda-Funktion.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:aws[a-zA-Z-]*:lambda:[a-z]+-[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-_]+)?`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ConversationLogsRequest

Service: Amazon Lex Model Building Service

Stellt die Einstellungen bereit, die für Konversationsprotokolle erforderlich sind.

Inhalt

iamRoleArn

Der Amazon-Ressourcenname (ARN) einer IAM-Rolle mit der Berechtigung, in Ihre CloudWatch Logs für Textprotokolle und in Ihren S3-Bucket für Audio-Logs zu schreiben. Wenn die Audioverschlüsselung aktiviert ist, bietet diese Rolle auch die Zugriffsberechtigung für den AWS-KMS-Schlüssel, der für die Verschlüsselung von Audioprotokollen verwendet wird. Weitere Informationen finden Sie unter Erstellen einer IAM-Rolle und Richtlinie für Konversationsprotokolle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Erforderlich: Ja

logSettings

Die Einstellungen für Ihre Konversationsprotokolle. Sie können den Konversationstext, das Gesprächsaudio oder beides protokollieren.

Typ: Array von LogSettingsRequest-Objekten

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ConversationLogsResponse

Service: Amazon Lex Model Building Service

Enthält Informationen zu den Einstellungen für das Konversationsprotokoll.

Inhalt

iamRoleArn

Der Amazon-Ressourcenname (ARN) der IAM-Rolle, die verwendet wurde, um Ihre Protokolle in Logs oder einen S3-Bucket zu CloudWatch schreiben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Erforderlich: Nein

logSettings

Die Einstellungen für Ihre Konversationsprotokolle. Sie können Text, Audio oder beides protokollieren.

Typ: Array von LogSettingsResponse-Objekten

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

EnumerationValue

Service: Amazon Lex Model Building Service

Jeder Slot-Typ kann eine Reihe von Werten haben. Jeder Aufzählungswert steht für einen Wert, den der Slot-Typ annehmen kann.

Ein Bot für die Pizzabestellung könnte beispielsweise einen Slot-Typ haben, der die Art der Kruste angibt, die die Pizza haben soll. Der Slot-Typ könnte die folgenden Werte enthalten

- stark
- thin
- gestopft

Inhalt

value

Der Wert des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 140 Zeichen.

Erforderlich: Ja

synonyms

Zusätzliche Werte, die sich auf den Wert des Slot-Typs beziehen.

Typ: Zeichenfolgen-Array

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 140 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

FollowUpPrompt

Service: Amazon Lex Model Building Service

Eine Aufforderung zu weiteren Aktivitäten, nachdem eine Absicht erfüllt wurde. Wenn die `OrderPizza` Absicht erfüllt wurde, könnten Sie den Benutzer beispielsweise auffordern, herauszufinden, ob der Benutzer Getränke bestellen möchte.

Inhalt

prompt

Fordert den Benutzer zur Eingabe von Informationen auf.

Typ: Prompt Objekt

Erforderlich: Ja

rejectionStatement

Wenn der Benutzer die in dem `prompt` Feld definierte Frage mit „Nein“ beantwortet, antwortet Amazon Lex mit dieser Erklärung, um zu bestätigen, dass die Absicht storniert wurde.

Typ: Statement Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

FulfillmentActivity

Service: Amazon Lex Model Building Service

Beschreibt, wie die Absicht erfüllt wird, nachdem der Benutzer alle für die Absicht erforderlichen Informationen bereitgestellt hat. Sie können eine Lambda-Funktion bereitstellen, um die Absicht zu verarbeiten, oder Sie können die Absichtsinformationen an die Client-Anwendung zurückgeben. Wir empfehlen Ihnen, eine Lambda-Funktion zu verwenden, damit die relevante Logik in der Cloud gespeichert ist, und den clientseitigen Code hauptsächlich auf die Präsentation zu beschränken. Wenn Sie die Logik aktualisieren müssen, aktualisieren Sie nur die Lambda-Funktion. Sie müssen Ihre Client-Anwendung nicht aktualisieren.

Betrachten Sie die folgenden Beispiele:

- In einer Pizza-Bestellanwendung verwenden Sie, nachdem der Benutzer alle Informationen für die Bestellung eingegeben hat, eine Lambda-Funktion, um eine Bestellung bei einer Pizzeria aufzugeben.
- Wenn ein Benutzer in einer Spieleanwendung sagt „Heb einen Stein auf“, müssen diese Informationen an die Client-Anwendung zurückgesendet werden, damit diese den Vorgang ausführen und die Grafik aktualisieren kann. In diesem Fall möchten Sie, dass Amazon Lex die Absichtsdaten an den Client zurückgibt.

Inhalt

type

Wie die Absicht erfüllt werden soll, entweder durch Ausführen einer Lambda-Funktion oder durch Rückgabe der Slot-Daten an die Client-Anwendung.

Typ: Zeichenfolge

Zulässige Werte: `ReturnIntent | CodeHook`

Erforderlich: Ja

codeHook

Eine Beschreibung der Lambda-Funktion, die ausgeführt wird, um die Absicht zu erfüllen.

Typ: CodeHook Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

InputContext

Service: Amazon Lex Model Building Service

Der Name eines Kontextes, der aktiv sein muss, damit eine Absicht von Amazon Lex ausgewählt wird.

Inhalt

name

Der Name des Kontexts.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Intent

Service: Amazon Lex Model Building Service

Identifiziert die spezifische Version einer Absicht.

Inhalt

intentName

Der Name der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

intentVersion

Die Version der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

IntentMetadata

Service: Amazon Lex Model Building Service

Stellt Informationen über eine Absicht bereit.

Inhalt

createdDate

Das Datum, an dem die Absicht erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

description

Eine Beschreibung der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

lastUpdatedDate

Das Datum, an dem die Absicht aktualisiert wurde. Wenn Sie eine Absicht erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

Erforderlich: Nein

name

Der Name der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

version

Die Version der Absicht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

KendraConfiguration

Service: Amazon Lex Model Building Service

Stellt Konfigurationsinformationen für den AMAZON bereit. KendraSearchIntentAbsicht. Wenn Sie diese Absicht verwenden, durchsucht Amazon Lex den angegebenen Amazon Kendra Kendra-Index und gibt Dokumente aus dem Index zurück, die der Äußerung des Benutzers entsprechen. Weitere Informationen finden Sie unter AMAZON. KendraSearchIntent.

Inhalt

kendraIndex

Der Amazon-Ressourcenname (ARN) des Amazon Kendra-Indexes, den Sie als AMAZON verwenden möchten. KendraSearchIntent Absicht zu suchen. Der Index muss sich im selben Konto und in derselben Region wie der Amazon Lex Lex-Bot befinden. Wenn der Amazon Kendra Kendra-Index nicht existiert, erhalten Sie eine Ausnahme, wenn Sie den `PutIntent` Vorgang aufrufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9][a-zA-Z0-9_-]*`

Erforderlich: Ja

role

Der Amazon-Ressourcenname (ARN) einer IAM-Rolle, die berechtigt ist, den Amazon Kendra-Index zu durchsuchen. Die Rolle muss sich im selben Konto und in derselben Region befinden wie der Amazon Lex Lex-Bot. Wenn die Rolle nicht existiert, erhalten Sie eine Ausnahme, wenn Sie den `PutIntent` Vorgang aufrufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `arn:aws:iam::[0-9]{12}:role/.*`

Erforderlich: Ja

queryFilterString

Ein Abfragefilter, den Amazon Lex an Amazon Kendra sendet, um die Antwort aus der Abfrage zu filtern. Der Filter hat das von Amazon Kendra definierte Format. Weitere Informationen finden Sie unter Abfragen filtern.

Sie können diese Filterzeichenfolge zur Laufzeit durch eine neue Filterzeichenfolge überschreiben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LogSettingsRequest

Service: Amazon Lex Model Building Service

Einstellungen, die zur Konfiguration des Übermittlungsmodus und des Ziels für Konversationsprotokolle verwendet werden.

Inhalt

destination

Wohin die Protokolle geliefert werden. Textprotokolle werden an eine Protokollgruppe „ CloudWatch Protokolle“ übermittelt. Audioprotokolle werden an einen S3-Bucket übermittelt.

Typ: Zeichenfolge

Zulässige Werte: `CLOUDWATCH_LOGS | S3`

Erforderlich: Ja

logType

Die Art der Protokollierung, die aktiviert werden soll. Textprotokolle werden an eine Protokollgruppe „ CloudWatchProtokolle“ übermittelt. Audioprotokolle werden an einen S3-Bucket übermittelt.

Typ: Zeichenfolge

Zulässige Werte: `AUDIO | TEXT`

Erforderlich: Ja

resourceArn

Der Amazon-Ressourcenname (ARN) der CloudWatch Logs-Protokollgruppe oder des S3-Buckets, in den die Protokolle geliefert werden sollen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:(?:logs:[\w\-]+:[\d]{12}:log-group:[\.\-_/#A-Za-z0-9]{1,512}(?::\*)?|s3:::[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9])$`

Erforderlich: Ja

kmsKeyArn

Der Amazon-Ressourcenname (ARN) des vom Kunden verwalteten AWS KMS KMS-Schlüssels zur Verschlüsselung von Audioprotokollen, die an einen S3-Bucket gesendet werden. Der Schlüssel gilt nicht für CloudWatch Logs und ist für S3-Buckets optional.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LogSettingsResponse

Service: Amazon Lex Model Building Service

Die Einstellungen für Konversationsprotokolle.

Inhalt

destination

Das Ziel, an das die Protokolle übermittelt werden.

Typ: Zeichenfolge

Zulässige Werte: `CLOUDWATCH_LOGS | S3`

Erforderlich: Nein

kmsKeyArn

Der Amazon-Ressourcenname (ARN) des Schlüssels, der zum Verschlüsseln von Audioprotokollen in einem S3-Bucket verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Erforderlich: Nein

logType

Die Art der Protokollierung, die aktiviert ist.

Typ: Zeichenfolge

Zulässige Werte: `AUDIO | TEXT`

Erforderlich: Nein

resourceArn

Der Amazon-Ressourcenname (ARN) der CloudWatch Logs-Protokollgruppe oder des S3-Buckets, in den die Protokolle geliefert werden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:(?:logs:[\w\-]+:[\d]{12}:log-group:[\.\-_/#A-Za-z0-9]{1,512}(?::\*)?|s3:::[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9])$`

Erforderlich: Nein

resourcePrefix

Das Ressourcenpräfix ist der erste Teil des S3-Objektschlüssels innerhalb des S3-Buckets, den Sie für die Aufnahme von Audioprotokollen angegeben haben. Bei CloudWatch Logs ist es das Präfix des Log-Stream-Namens innerhalb der von Ihnen angegebenen Protokollgruppe.

Typ: Zeichenfolge

Längenbeschränkungen: Maximale Länge von 1 024.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Message

Service: Amazon Lex Model Building Service

Das Nachrichtenobjekt, das den Nachrichtentext und seinen Typ bereitstellt.

Inhalt

content

Der Text der Nachricht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 1 000 Zeichen.

Erforderlich: Ja

contentType

Der Inhaltstyp der Nachrichtenzeichenfolge.

Typ: Zeichenfolge

Zulässige Werte: `PlainText | SSML | CustomPayload`

Erforderlich: Ja

groupNumber

Identifiziert die Nachrichtengruppe, zu der die Nachricht gehört. Wenn einer Nachricht eine Gruppe zugewiesen wird, gibt Amazon Lex in der Antwort eine Nachricht von jeder Gruppe zurück.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximalwert von 5.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MigrationAlert

Service: Amazon Lex Model Building Service

Stellt Informationen zu Alarmen und Warnungen bereit, die Amazon Lex während einer Migration sendet. Die Benachrichtigungen enthalten Informationen darüber, wie das Problem gelöst werden kann.

Inhalt

details

Zusätzliche Details zur Warnung.

Typ: Zeichenfolgen-Array

Erforderlich: Nein

message

Eine Nachricht, die beschreibt, warum die Warnung ausgegeben wurde.

Typ: Zeichenfolge

Erforderlich: Nein

referenceURLs

Ein Link zur Amazon Lex-Dokumentation, in der beschrieben wird, wie die Warnung behoben werden kann.

Typ: Zeichenfolgen-Array

Erforderlich: Nein

type

Die Art der Warnung. Es gibt zwei Arten von Benachrichtigungen:

- `ERROR`- Bei der Migration ist ein Problem aufgetreten, das nicht gelöst werden kann. Die Migration wird gestoppt.
- `WARN`— Bei der Migration ist ein Problem aufgetreten, das manuelle Änderungen am neuen Amazon Lex V2-Bot erfordert. Die Migration wird fortgesetzt.

Typ: Zeichenfolge

Zulässige Werte: `ERROR | WARN`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MigrationSummary

Service: Amazon Lex Model Building Service

Enthält Informationen zur Migration eines Bots von Amazon Lex V1 zu Amazon Lex V2.

Inhalt

migrationId

Die eindeutige Kennung, die Amazon Lex der Migration zugewiesen hat.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

Erforderlich: Nein

migrationStatus

Der Status der aktuellen Operation. Wenn der Status lautet, ist `COMPLETE` der Bot in Amazon Lex V2 verfügbar. Möglicherweise gibt es Warnungen und Warnungen, die behoben werden müssen, um die Migration abzuschließen.

Typ: Zeichenfolge

Zulässige Werte: `IN_PROGRESS | COMPLETED | FAILED`

Erforderlich: Nein

migrationStrategy

Die Strategie, mit der die Migration durchgeführt wurde.

Typ: Zeichenfolge

Zulässige Werte: `CREATE_NEW | UPDATE_EXISTING`

Erforderlich: Nein

migrationTimestamp

Datum und Uhrzeit des Beginns der Migration.

Typ: Zeitstempel

Erforderlich: Nein

v1BotLocale

Das Gebietsschema des Amazon Lex V1-Bots, der die Quelle der Migration ist.

Typ: Zeichenfolge

Zulässige Werte: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Erforderlich: Nein

v1BotName

Der Name des Amazon Lex V1-Bots, der die Quelle der Migration ist.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 2. Maximale Länge = 50 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

v1BotVersion

Die Version des Amazon Lex V1-Bots, der die Quelle der Migration ist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

v2BotId

Die eindeutige Kennung von Amazon Lex V2, die das Ziel der Migration ist.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 10.

Pattern: `^[0-9a-zA-Z]+$`

Erforderlich: Nein

v2BotRole

Die IAM-Rolle, die Amazon Lex verwendet, um den Amazon Lex V2-Bot auszuführen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 20. Maximale Länge beträgt 2048 Zeichen.

Pattern: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

OutputContext

Service: Amazon Lex Model Building Service

Die Spezifikation eines Ausgabekontextes, der festgelegt wird, wenn eine Absicht erfüllt ist.

Inhalt

name

Der Name des Kontexts.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

timeToLiveInSeconds

Die Anzahl der Sekunden, für die der Kontext aktiv sein soll, nachdem er zum ersten Mal in einer `PostContent` `PostText` OR-Antwort gesendet wurde. Sie können den Wert zwischen 5 und 86.400 Sekunden (24 Stunden) festlegen.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert von 5. Maximaler Wert von 86400.

Erforderlich: Ja

turnsToLive

Die Anzahl der Konversationsrunden, bei denen der Kontext aktiv sein sollte. Eine Konversation ist eine `PostContent` `PostText` Oder-Anfrage und die entsprechende Antwort von Amazon Lex.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximalwert von 20.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Prompt

Service: Amazon Lex Model Building Service

Ruft Informationen vom Benutzer ab. Um eine Aufforderung zu definieren, geben Sie eine oder mehrere Nachrichten ein und geben Sie die Anzahl der Versuche an, Informationen vom Benutzer abzurufen. Wenn Sie mehr als eine Nachricht angeben, wählt Amazon Lex eine der Nachrichten aus, anhand derer der Benutzer aufgefordert wird. Weitere Informationen finden Sie unter Amazon Lex — Funktionsweise.

Inhalt

maxAttempts

Die Häufigkeit, mit der der Benutzer zur Eingabe von Informationen aufgefordert wird.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximalwert von 5.

Erforderlich: Ja

messages

Ein Array von Objekten, von denen jedes eine Nachrichtenzeichenfolge und deren Typ bereitstellt. Sie können die Nachrichtenzeichenfolge im Klartext oder in Speech Synthesis Markup Language (SSML) angeben.

Typ: Array von Message-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Maximale Anzahl von 15 Elementen.

Erforderlich: Ja

responseCard

Eine Antwortkarte. Amazon Lex verwendet diese Aufforderung zur Laufzeit in der `PostText` API-Antwort. Es ersetzt Platzhalter auf der Antwortkarte durch Sitzungsattribute und Slot-Werte. Weitere Informationen finden Sie unter Eine Antwortkarte verwenden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 50000.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResourceReference

Service: Amazon Lex Model Building Service

Beschreibt die Ressource, die auf die Ressource verweist, die Sie zu löschen versuchen. Dieses Objekt wird als Teil der `ResourceInUseException` Ausnahme zurückgegeben.

Inhalt

name

Der Name der Ressource, die die Ressource verwendet, die Sie löschen möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `[a-zA-Z_]+`

Erforderlich: Nein

version

Die Version der Ressource, die die Ressource verwendet, die Sie löschen möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Slot

Service: Amazon Lex Model Building Service

Identifiziert die Version eines bestimmten Steckplatzes.

Inhalt

name

Der Name des Slots.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z](-|_|.)?)+$`

Erforderlich: Ja

slotConstraint

Gibt an, ob der Slot erforderlich oder optional ist.

Typ: Zeichenfolge

Zulässige Werte: `Required | Optional`

Erforderlich: Ja

defaultValueSpec

Eine Liste von Standardwerten für den Steckplatz. Standardwerte werden verwendet, wenn Amazon Lex keinen Wert für einen Slot ermittelt hat. Sie können Standardwerte aus Kontextvariablen, Sitzungsattributen und definierten Werten angeben.

Typ: SlotDefaultValueSpec Objekt

Erforderlich: Nein

description

Eine Beschreibung des Steckplatzes.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

obfuscationSetting

Bestimmt, ob ein Slot in Gesprächsprotokollen und gespeicherten Äußerungen verschleiert ist. Wenn Sie einen Slot verschleiern, wird der Wert durch den Slot-Namen in geschweiften Klammern ({}) ersetzt. Lautet der Slot-Name beispielsweise „full_name“, werden verschleierte Werte durch „{full_name}“ ersetzt. Weitere Informationen finden Sie unter Slot-Verschleierung.

Typ: Zeichenfolge

Zulässige Werte: `NONE | DEFAULT_OBFUSCATION`

Erforderlich: Nein

priority

Weist Amazon Lex die Reihenfolge an, in der dieser Slot-Wert vom Benutzer abgerufen werden soll. Wenn die Absicht beispielsweise zwei Slots mit den Prioritäten 1 und 2 hat, ruft AWS Amazon Lex zunächst einen Wert für den Slot mit Priorität 1 ab.

Wenn mehrere Slots dieselbe Priorität haben, ist die Reihenfolge, in der Amazon Lex Werte abruft, willkürlich.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 0. Maximalwert 100.

Erforderlich: Nein

responseCard

Eine Reihe möglicher Antworten für den Slot-Typ, der von textbasierten Clients verwendet wird. Ein Benutzer wählt eine Option auf der Antwortkarte aus, anstatt Text für die Antwort zu verwenden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 50000.

Erforderlich: Nein

sampleUtterances

Wenn Sie ein bestimmtes Muster kennen, nach dem Benutzer auf eine Anfrage von Amazon Lex nach einem Slot-Wert antworten könnten, können Sie diese Äußerungen angeben, um die Genauigkeit zu verbessern. Dieser Schritt ist optional. In den meisten Fällen ist Amazon Lex in der Lage, Benutzeräußerungen zu verstehen.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

slotType

Der Typ des Steckplatzes, entweder ein benutzerdefinierter Steckplatztyp, den Sie definiert haben, oder einer der integrierten Steckplatztypen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Erforderlich: Nein

slotTypeVersion

Die Version des Steckplatztyps.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

valueElicitationPrompt

Die Aufforderung, die Amazon Lex verwendet, um dem Benutzer den Slot-Wert zu entlocken.

Typ: Prompt Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SlotDefaultValue

Service: Amazon Lex Model Building Service

Ein Standardwert für einen Steckplatz.

Inhalt

defaultValue

Der Standardwert für den Steckplatz. Sie können einen der folgenden Werte angeben:

- `#context-name.slot-name`- Der Slot-Wert „Slot-Name“ im Kontext „Context-Name“.
- `{attribute}`- Der Slot-Wert des Sitzungsattributs „Attribut“.
- `'value'`- Der diskrete Wert „Wert“.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 202.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SlotDefaultValueSpec

Service: Amazon Lex Model Building Service

Enthält die Standardwerte für einen Slot. Standardwerte werden verwendet, wenn Amazon Lex keinen Wert für einen Slot ermittelt hat.

Inhalt

defaultValueList

Die Standardwerte für einen Slot. Sie können mehr als einen Standardwert angeben. Sie können beispielsweise einen Standardwert angeben, der aus einer passenden Kontextvariablen, einem Sitzungsattribut oder einem festen Wert verwendet werden soll.

Der gewählte Standardwert wird auf der Grundlage der Reihenfolge ausgewählt, in der Sie sie in der Liste angegeben haben. Wenn Sie beispielsweise eine Kontextvariable und einen festen Wert in dieser Reihenfolge angeben, verwendet Amazon Lex die Kontextvariable, sofern sie verfügbar ist, andernfalls verwendet es den festen Wert.

Typ: Array von SlotDefaultValue-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SlotTypeConfiguration

Service: Amazon Lex Model Building Service

Stellt Konfigurationsinformationen für einen Steckplatztyp bereit.

Inhalt

regexConfiguration

Ein regulärer Ausdruck, der verwendet wird, um den Wert eines Slots zu überprüfen.

Typ: SlotTypeRegexConfiguration Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SlotTypeMetadata

Service: Amazon Lex Model Building Service

Stellt Informationen über einen Steckplatztyp bereit..

Inhalt

createdDate

Das Datum, an dem der Slot-Typ erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

description

Eine Beschreibung des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Höchstlänge = 200 Zeichen.

Erforderlich: Nein

lastUpdatedDate

Das Datum, an dem der Slot-Typ aktualisiert wurde. Wenn Sie eine Ressource erstellen, stimmen das Erstellungsdatum und das Datum der letzten Aktualisierung überein.

Typ: Zeitstempel

Erforderlich: Nein

name

Der Name des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Nein

version

Die Version des Slot-Typs.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SlotTypeRegexConfiguration

Service: Amazon Lex Model Building Service

Liefert einen regulärern Ausdruck, der verwendet wird, um den Wert eines Slots zu überprüfen.

Inhalt

pattern

Ein regulärer Ausdruck, der verwendet wird, um den Wert eines Slots zu überprüfen.

Verwenden Sie einen regulären Standardausdruck. Amazon Lex unterstützt die folgenden Zeichen im regulären Ausdruck:

- A-Z, a-z
- 0-9
- Unicode-Zeichen („\ u <Unicode>„)

Stellt Unicode-Zeichen mit vier Ziffern dar, zum Beispiel „\ u0041" oder „\ u005A“.

Die folgenden Operatoren für reguläre Ausdrücke werden nicht unterstützt:

- Endlose Wiederholer: *, +, oder {x,} ohne Obergrenze.
- Platzhalter (.)

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der AWS sprachspezifischen SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Statement

Service: Amazon Lex Model Building Service

Eine Sammlung von Nachrichten, die dem Benutzer Informationen vermitteln. Zur Laufzeit wählt Amazon Lex die zu übermittelnde Nachricht aus.

Inhalt

messages

Eine Sammlung von Nachrichtenobjekten.

Typ: Array von Message-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Maximale Anzahl von 15 Elementen.

Erforderlich: Ja

responseCard

Wenn der Client zur Laufzeit die PostTextAPI verwendet, nimmt Amazon Lex die Antwortkarte in die Antwort auf. Es ersetzt alle Sitzungsattribute und Slot-Werte durch Platzhalter auf der Antwortkarte.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 50000.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Tag

Service: Amazon Lex Model Building Service

Eine Liste von Schlüssel/Wert-Paaren, die einen Bot, einen Bot-Alias oder einen Bot-Kanal identifizieren. Tag-Schlüssel und -Werte können aus Unicode-Buchstaben, Ziffern, Leerzeichen und einem der folgenden Symbole bestehen: _.:/= + - @.

Inhalt

key

Der Schlüssel für das Tag. Bei den Schlüsseln wird nicht zwischen Groß- und Kleinschreibung unterschieden und sie müssen eindeutig sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

value

Der einem Schlüssel zugeordnete Wert. Der Wert kann eine leere Zeichenfolge sein, er darf jedoch nicht Null sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UtteranceData

Service: Amazon Lex Model Building Service

Stellt Informationen zu einer einzelnen Äußerung bereit, die an Ihren Bot gerichtet wurde.

Inhalt

count

Gibt an, wie oft die Äußerung verarbeitet wurde.

Typ: Ganzzahl

Erforderlich: Nein

distinctUsers

Die Gesamtzahl der Personen, die die Äußerung verwendet haben.

Typ: Ganzzahl

Erforderlich: Nein

firstUtteredDate

Das Datum, an dem die Äußerung zum ersten Mal aufgezeichnet wurde.

Typ: Zeitstempel

Erforderlich: Nein

lastUtteredDate

Das Datum, an dem die Äußerung zuletzt aufgezeichnet wurde.

Typ: Zeitstempel

Erforderlich: Nein

utteranceString

Der Text, der vom Benutzer eingegeben wurde, oder die Textdarstellung eines Audioclips.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 2000.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UtteranceList

Service: Amazon Lex Model Building Service

Bietet eine Liste von Äußerungen, die zu einer bestimmten Version Ihres Bots gemacht wurden. Die Liste enthält maximal 100 Äußerungen.

Inhalt

botVersion

Die Version des Bots, der die Liste verarbeitet hat.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `\$LATEST|[0-9]+`

Erforderlich: Nein

utterances

Ein oder mehrere UtteranceData Objekte, die Informationen über die Äußerungen enthalten, die gegenüber einem Bot gemacht wurden. Die maximale Anzahl von Objekten ist 100.

Typ: Array von UtteranceData-Objekten

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Amazon Lex Laufzeit-Service

Die folgenden Datentypen werden von Amazon Lex Runtime Service unterstützt:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

ActiveContext

Service: Amazon Lex Runtime Service

Ein Kontext ist eine Variable, die Informationen über den aktuellen Status der Konversation zwischen einem Benutzer und Amazon Lex enthält. Der Kontext kann automatisch von Amazon Lex festgelegt werden, wenn eine Absicht erfüllt ist, oder er kann zur Laufzeit mithilfe der `PutSession` Operation`PutContent`,`PutText`, oder festgelegt werden.

Inhalt

name

Der Name des Kontexts.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Pattern: `^([A-Za-z]_?)+$`

Erforderlich: Ja

parameters

Statusvariablen für den aktuellen Kontext. Sie können diese Werte als Standardwerte für Slots in nachfolgenden Ereignissen verwenden.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Karteneinträge: Mindestanzahl von 0 Elementen. Die maximale Anzahl beträgt 10 Elemente.

Schlüssel-Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 100 Zeichen.

Längenbeschränkungen des Wertes: Minimale Länge von 1. Maximale Länge von 1 024.

Erforderlich: Ja

timeToLive

Die Dauer oder Anzahl der Runden, in denen ein Kontext aktiv bleibt.

Typ: ActiveContextTimeToLive Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ActiveContextTimeToLive

Service: Amazon Lex Runtime Service

Die Dauer oder Anzahl der Runden, in denen ein Kontext aktiv bleibt.

Inhalt

timeToLiveInSeconds

Die Anzahl der Sekunden, für die der Kontext aktiv sein sollte, nachdem er zum ersten Mal in einer `PostContent` `PostText` OR-Antwort gesendet wurde. Sie können den Wert zwischen 5 und 86.400 Sekunden (24 Stunden) festlegen.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert von 5. Maximaler Wert von 86400.

Erforderlich: Nein

turnsToLive

Die Anzahl der Konversationsrunden, bei denen der Kontext aktiv sein sollte. Eine Konversation ist eine `PostContent` `PostText` Oder-Anfrage und die entsprechende Antwort von Amazon Lex.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximalwert von 20.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Button

Service: Amazon Lex Runtime Service

Stellt eine Option dar, die auf der Client-Plattform (Facebook, Slack usw.) angezeigt werden soll.

Inhalt

text

Text, der für den Benutzer auf der Schaltfläche sichtbar ist.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 15 Zeichen.

Erforderlich: Ja

value

Der Wert, der an Amazon Lex gesendet wird, wenn ein Benutzer die Schaltfläche auswählt. Betrachten Sie zum Beispiel den Button-Text „NYC“. Wenn der Benutzer die Schaltfläche auswählt, kann der gesendete Wert „New York City“ lauten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Höchstlänge = 1 000 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DialogAction

Service: Amazon Lex Runtime Service

Beschreibt die nächste Aktion, die der Bot in seiner Interaktion mit dem Benutzer ausführen sollte, und liefert Informationen über den Kontext, in dem die Aktion stattfindet. Verwenden Sie den `DialogAction` Datentyp, um die Interaktion auf einen bestimmten Status zu setzen oder um die Interaktion in einen vorherigen Zustand zurückzuversetzen.

Inhalt

type

Die nächste Aktion, die der Bot bei seiner Interaktion mit dem Benutzer ausführen soll. Die möglichen Werte sind:

- `ConfirmIntent`- Die nächste Aktion besteht darin, den Benutzer zu fragen, ob die Absicht vollständig und bereit ist, erfüllt zu werden. Das ist eine Ja/Nein-Frage wie „Bestellung aufgeben?“
- `Close`- Zeigt an, dass es keine Antwort vom Benutzer geben wird. Zum Beispiel erfordert die Aussage „Ihre Bestellung wurde aufgegeben“ keine Antwort.
- `Delegate`- Die nächste Aktion wird von Amazon Lex festgelegt.
- `ElicitIntent`- Die nächste Aktion besteht darin, die Absicht zu bestimmen, die der Benutzer erfüllen möchte.
- `ElicitSlot`- Die nächste Aktion besteht darin, dem Benutzer einen Slot-Wert zu entlocken.

Typ: Zeichenfolge

Zulässige Werte: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Erforderlich: Ja

fulfillmentState

Der Erfüllungsstatus der Absicht. Die möglichen Werte sind:

- `Failed`- Die mit der Absicht verknüpfte Lambda-Funktion konnte die Absicht nicht erfüllen.
- `Fulfilled`- Die Absicht wurde durch die mit der Absicht verknüpfte Lambda-Funktion erfüllt.
- `ReadyForFulfillment`- Alle für die Absicht erforderlichen Informationen sind vorhanden und die Absicht kann von der Client-Anwendung erfüllt werden.

Typ: Zeichenfolge

Zulässige Werte: `Fulfilled | Failed | ReadyForFulfillment`

Erforderlich: Nein

intentName

Der Name der Absicht.

Typ: Zeichenfolge

Erforderlich: Nein

message

Die Nachricht, die dem Benutzer angezeigt werden soll. Wenn Sie keine Nachricht angeben, verwendet Amazon Lex die für die Absicht konfigurierte Nachricht.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 1024 Zeichen.

Erforderlich: Nein

messageFormat

- `PlainText`- Die Nachricht enthält einfachen UTF-8-Text.
- `CustomPayload`- Die Nachricht ist ein benutzerdefiniertes Format für den Client.
- `SSML`- Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `Composite`- Die Nachricht enthält ein maskiertes JSON-Objekt, das eine oder mehrere Nachrichten enthält. Weitere Informationen finden Sie unter Nachrichtengruppen.

Typ: Zeichenfolge

Zulässige Werte: `PlainText | CustomPayload | SSML | Composite`

Erforderlich: Nein

slots

Karte der gesammelten Slots und ihrer Werte.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Erforderlich: Nein

slotToElicit

Der Name des Slots, der dem Benutzer ermittelt werden soll.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

GenericAttachment

Service: Amazon Lex Runtime Service

Stellt eine Option dar, die dem Benutzer angezeigt wird, wenn eine Eingabeaufforderung angezeigt wird. Dabei kann es sich um ein Bild, eine Schaltfläche, einen Link oder Text handeln.

Inhalt

attachmentLinkUrl

Die URL eines Anhangs zur Antwortkarte.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

buttons

Die Liste der Optionen, die dem Benutzer angezeigt werden sollen.

Typ: Array von Button-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 5 Elemente.

Erforderlich: Nein

imageUrl

Die URL eines Bildes, das dem Benutzer angezeigt wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

subTitle

Der Untertitel wird unter dem Titel angezeigt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 80 Zeichen.

Erforderlich: Nein

title

Der Titel der Option.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 80 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

IntentConfidence

Service: Amazon Lex Runtime Service

Stellt eine Bewertung bereit, die angibt, dass Amazon Lex davon überzeugt ist, dass eine Absicht die Absicht des Benutzers erfüllt.

Inhalt

score

Ein Wert, der angibt, wie sicher Amazon Lex ist, dass eine Absicht die Absicht des Benutzers erfüllt. Der Bereich liegt zwischen 0,00 und 1,00. Höhere Werte deuten auf ein höheres Selbstvertrauen hin.

Type: Double

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

IntentSummary

Service: Amazon Lex Runtime Service

Stellt Informationen über den Status einer Absicht bereit. Sie können diese Informationen verwenden, um den aktuellen Status einer Absicht abzurufen, sodass Sie die Absicht verarbeiten können, oder um die Absicht auf ihren vorherigen Status zurückzusetzen.

Inhalt

dialogActionType

Die nächste Aktion, die der Bot bei seiner Interaktion mit dem Benutzer ausführen sollte. Die möglichen Werte sind:

- `ConfirmIntent`- Die nächste Aktion besteht darin, den Benutzer zu fragen, ob die Absicht vollständig und bereit ist, erfüllt zu werden. Das ist eine Ja/Nein-Frage wie „Bestellung aufgeben?“
- `Close`- Zeigt an, dass es keine Antwort vom Benutzer geben wird. Zum Beispiel erfordert die Aussage „Ihre Bestellung wurde aufgegeben“ keine Antwort.
- `ElicitIntent`- Die nächste Aktion besteht darin, die Absicht zu bestimmen, die der Benutzer erfüllen möchte.
- `ElicitSlot`- Die nächste Aktion besteht darin, dem Benutzer einen Slot-Wert zu entlocken.

Typ: Zeichenfolge

Zulässige Werte: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Erforderlich: Ja

checkpointLabel

Eine benutzerdefinierte Bezeichnung, die eine bestimmte Absicht identifiziert. Sie können dieses Label verwenden, um zu einer vorherigen Absicht zurückzukehren.

Verwenden Sie den `checkpointLabelFilter` Parameter der `GetSessionRequest` Operation, um die von der Operation zurückgegebenen Absichten nach solchen zu filtern, die nur die angegebene Bezeichnung haben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 255 Zeichen.

Pattern: `[a-zA-Z0-9-]+`

Erforderlich: Nein

confirmationStatus

Der Status der Absicht, nachdem der Benutzer auf die Bestätigungsaufforderung geantwortet hat. Wenn der Benutzer die Absicht bestätigt, setzt Amazon Lex dieses Feld auf`Confirmed`. Wenn der Benutzer die Absicht ablehnt, setzt Amazon Lex diesen Wert auf`Denied`. Die möglichen Werte sind:

- `Confirmed`- Der Benutzer hat auf die Bestätigungsaufforderung mit „Ja“ geantwortet und bestätigt, dass die Absicht abgeschlossen ist und zur Ausführung bereit ist.
- `Denied`- Der Benutzer hat auf die Bestätigungsaufforderung mit „Nein“ geantwortet.
- `None`- Der Benutzer wurde nie zur Bestätigung aufgefordert; oder der Benutzer wurde aufgefordert, hat die Aufforderung aber nicht bestätigt oder abgelehnt.

Typ: Zeichenfolge

Zulässige Werte: `None | Confirmed | Denied`

Erforderlich: Nein

fulfillmentState

Der Erfüllungsstatus der Absicht. Die möglichen Werte sind:

- `Failed`- Die mit der Absicht verknüpfte Lambda-Funktion konnte die Absicht nicht erfüllen.
- `Fulfilled`- Die Absicht wurde durch die mit der Absicht verknüpfte Lambda-Funktion erfüllt.
- `ReadyForFulfillment`- Alle für die Absicht erforderlichen Informationen sind vorhanden und die Absicht kann von der Client-Anwendung erfüllt werden.

Typ: Zeichenfolge

Zulässige Werte: `Fulfilled | Failed | ReadyForFulfillment`

Erforderlich: Nein

intentName

Der Name der Absicht.

Typ: Zeichenfolge

Erforderlich: Nein

slots

Karte der gesammelten Slots und ihrer Werte.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Erforderlich: Nein

slotToElicit

Der nächste Slot, der dem Benutzer entlockt werden soll. Wenn es keinen Slot gibt, der abgerufen werden kann, ist das Feld leer.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

PredictedIntent

Service: Amazon Lex Runtime Service

Eine Absicht, die Amazon Lex vorschlägt, erfüllt die Absicht des Benutzers. Enthält den Namen der Absicht, die Gewissheit von Amazon Lex, dass die Absicht des Benutzers erfüllt ist, und die für die Absicht definierten Slots.

Inhalt

intentName

Der Name der Absicht, die Amazon Lex vorschlägt, entspricht der Absicht des Benutzers.

Typ: Zeichenfolge

Erforderlich: Nein

nluIntentConfidence

Zeigt an, wie sicher Amazon Lex ist, dass eine Absicht die Absicht des Benutzers erfüllt.

Typ: IntentConfidence Objekt

Erforderlich: Nein

slots

Der Slot und die Slot-Werte, die mit der vorhergesagten Absicht verknüpft sind.

Typ: Abbildung einer Zeichenfolge auf eine Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResponseCard

Service: Amazon Lex Runtime Service

Wenn Sie bei der Erstellung Ihrer Bots eine Antwortkarte konfigurieren, ersetzt Amazon Lex die verfügbaren Sitzungsattribute und Slot-Werte und gibt sie dann zurück. Die Antwortkarte kann auch von einer Lambda-Funktion (`dialogCodeHook`und `fulfillmentActivity` mit Absicht) stammen.

Inhalt

contentType

Der Inhaltstyp der Antwort.

Typ: Zeichenfolge

Zulässige Werte: `application/vnd.amazonaws.card.generic`

Erforderlich: Nein

genericAttachments

Eine Reihe von Anhangsobjekten, die Optionen darstellen.

Typ: Array von GenericAttachment-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 10 Elemente.

Erforderlich: Nein

version

Die Version des Antwortkartenformats.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

SentimentResponse

Service: Amazon Lex Runtime Service

Das in einer Äußerung zum Ausdruck gebrachte Gefühl.

Wenn der Bot so konfiguriert ist, dass er Äußerungen zur Stimmungsanalyse an Amazon Comprehend sendet, enthält diese Feldstruktur das Ergebnis der Analyse.

Inhalt

sentimentLabel

Die abgeleitete Stimmung, der Amazon Comprehend am meisten vertraut.

Typ: Zeichenfolge

Erforderlich: Nein

sentimentScore

Die Wahrscheinlichkeit, dass die Stimmung korrekt abgeleitet wurde.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser API in einem der sprachspezifischen AWS SDKs finden Sie im Folgenden:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Dokumentverlauf für Amazon Lex

- Letzte Aktualisierung der Dokumentation: 9. September 2021

In der folgenden Tabelle werden wichtige Änderungen in den einzelnen Versionen von Amazon Lex beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Neue Funktion	Amazon Lex unterstützt jetzt das koreanische Gebietssc hema (ko-KR). Weitere Informationen finden Sie unter Von Amazon Lex unterstützte Sprachen.	9. September 2021
Neue Funktion	Amazon Lex unterstützt jetzt das englische (indische) Gebietsschema. Weitere Informationen finden Sie unter In Amazon Lex unterstützte Sprachen.	15. Juli 2021
Neue Funktion	Amazon Lex bietet jetzt ein Tool zur Migration eines Bots zur Amazon Lex V2-API. Weitere Informationen finden Sie unter Migrieren eines Bots.	13. Juli 2021
Neue Funktion	Amazon Lex unterstützt jetzt das japanische Gebietssc hema (Japan). Weitere Informationen finden Sie unter Von Amazon Lex unterstützte Sprachen.	01. April 2021

Neue Funktion	Amazon Lex unterstützt jetzt die Sprachversionen Deutsch (Deutsch) (de-DE) und Spanisch (Lateinamerika) (es-419). Weitere Informationen finden Sie unter Von Amazon Lex unterstützte Sprachen.	23. November 2020
Neue Funktion	Amazon Lex unterstützt jetzt die Verwendung von Kontexten zur Verwaltung von Aktivierungsabsichten. Weitere Informationen finden Sie unter Festlegen des Absichtskontextes.	19. November 2020
Neue Funktion	Amazon Lex unterstützt jetzt die Sprachversionen Französisch (fr-FR), Französisch-Kanadisch (fr-CA), Italienisch (it-IT) und Spanisch (es-ES). Eine vollständige Liste der unterstützten Gebietsschemata finden Sie unter Von Amazon Lex unterstützte Sprachen.	11. November 2020
Neue Funktion	Amazon Lex unterstützt jetzt das Gebietsschema Spanisch (USA) (es-US). Weitere Informationen finden Sie unter Von Amazon Lex unterstützte Sprachen.	22. September 2020

Neue Funktion	Amazon Lex unterstützt jetzt das englische (britische) (en-GB) Gebietsschema. Weitere Informationen finden Sie unter Von Amazon Lex unterstützte Sprachen.	15. September 2020
Neue Funktion	Amazon Lex unterstützt jetzt das englische (australische) (en-AU) Gebietsschema. Weitere Informationen finden Sie unter Von Amazon Lex unterstützte Sprachen.	8. September 2020
Neue Funktion	Amazon Lex verfügt jetzt über 7 neue integrierte Intents und 9 neue integrierte Slot-Typen. Weitere Informationen finden Sie unter Integrierte Absichten und Slot-Typen.	8. September 2020
Neues Beispiel	Erfahren Sie, wie Sie einen Amazon Lex Lex-Bot erstellen , mit dem Kundenbetreuer Kundenfragen beantwort en können, indem Sie mit Amazon Kendra nach Antworten suchen. Weitere Informationen finden Sie unter Beispiel: Call Center Agent Assistant.	10. August 2020

Neue Funktion	Amazon Lex kann jetzt bis zu vier alternative Absichten auf der Grundlage von Konfidenz werten ermitteln. Weitere Informationen finden Sie unter Verwenden von Konfidenz werten.	6. August 2020
Regionale Regionale Erweiterung	Amazon Lex ist jetzt im Asien-Pazifik (Tokio) (ap-north east-1) (ap-northeast-1) (ap-northeast-1) (Asien-Pazifik	30. Juni 2020
Neue Funktion	Amazon Lex unterstützt jetzt die Suche in Amazon Kendra Kendra-Indizes nach Antworten auf häufig gestellte Fragen. Weitere Informationen finden Sie unter AMAZON. KendraSearchIntent.	11. Juni 2020
Neue Funktion	Amazon Lex gibt jetzt mehr Informationen in Konversat ionsprotokollen zurück. Weitere Informationen finden Sie unter Textprotokolle in Amazon CloudWatch Logs anzeigen.	9. Juni 2020

Regionale Regionale Erweiterung	Amazon Lex ist jetzt in den Asien-Pazifik (Singapur) (ap-southeast-1), Europa (London) (eu-west-2) und Europa (London) (eu-west-2) und Europa (London) (eu-west-2) und Europa (London) (eu-west-2) und Europa (London) (eu-west-2) und Europa	23. April 2020
Neue Funktion	Amazon Lex unterstützt jetzt Tagging. Mithilfe von Tagging können Sie Ressourcen identifizieren, Kosten zuweisen und den Zugriff steuern. Weitere Informationen finden Sie unter Markieren Ihrer Amazon Lex Lex-Ressourcen.	12. März 2020
Neue Funktion	Amazon Lex unterstützt jetzt reguläre Ausdrücke für den AMAZON. AlphaNumeric eingebauter Slot-Typ. Weitere Informationen finden Sie unter AMAZON. AlphaNumeric.	6. Februar 2020
Neue Funktion	Amazon Lex kann jetzt Konversationsinformationen protokollieren und Slot-Werte in diesen Protokollen verschleiern. Weitere Informationen finden Sie unter Erstellen von Konversationsprotokollen und Slot-Verschleierung.	19. Dezember 2019

Regionale Regionale Erweiterung	Amazon Lex ist jetzt im Asien-Pazifik (Sydney) (ap-south east-2) (ap-southeast-2) (ap-southeast-2) (Asien-Pazifik	17. Dezember 2019
Neue Funktion	Amazon Lex ist jetzt HIPAA-konform. Weitere Informationen finden Sie unter Compliance-Überprüfung für Amazon Lex.	10. Dezember 2019
Neue Funktion	Amazon Lex kann jetzt Benutzeräußerungen an Amazon Comprehend senden, um die Stimmung der Äußerung zu analysieren. Weitere Informationen finden Sie unter Stimmungsanalyse.	21. November 2019
Neue Funktion	Amazon Lex ist jetzt SOC-konform. Weitere Informationen finden Sie unter Compliance-Überprüfung für Amazon Lex.	19. November 2019
Neue Funktion	Amazon Lex ist jetzt PCI-konform. Weitere Informationen finden Sie unter Compliance-Überprüfung für Amazon Lex.	17. Oktober 2019

Neue Funktion	Es wurde Unterstützung für das Hinzufügen eines Prüfpunkts zu einer Absicht hinzugefügt, damit Sie während einer Konversat ion problemlos zur Absicht zurückkehren können. Weitere Informationen finden Sie im Artikel zum Verwalten von Sitzungen.	10. Oktober 2019
Neue Funktion	Unterstützung für den `AMAZON.FallbackInt ent` hinzugefügt, sodass Ihr Bot Situationen verarbeit en kann, in denen Benutzere ingaben nicht wie erwartet sind. Weitere Informationen finden Sie unter AMAZON. FallbackIntent.	3. Oktober 2019
Neue Funktion	Mit Amazon Lex können Sie Sitzungsinformationen für Ihre Bots verwalten. Weitere Informationen finden Sie unter Sitzungen mit der Amazon Lex-API verwalten.	8. August 2019
Regionale Regionale Erweiterung	Amazon Lex ist jetzt in den USA West (Oregon) (us-west-2) (us-west-2) (us-west-2) (us-west-2) verfügbar.	8. Mai 2018

Neue Funktion	Unterstützung für Export und Import im Amazon Lex-Format hinzugefügt. Weitere Informationen finden Sie unter Amazon Lex Lex-Bots, Intents und Slot-Typen importieren und exportieren.	13. Februar 2018
Neue Funktion	Amazon Lex unterstützt jetzt zusätzliche Antwortnachrichten für Bots. Weitere Informationen finden Sie unter Antworten.	8. Februar 2018
Regionale Regionale Erweiterung	Amazon Lex ist jetzt in Europa (Irland) (eu-west-1) (eu-west-1) (Europa (Irland) (eu-west-1) (Europa	21. November 2017
Neue Funktion	Unterstützung für die Bereitstellung von Amazon Lex Lex-Bots auf Kik hinzugefügt. Weitere Informationen finden Sie unter Integration eines Amazon Lex Lex-Bot mit Kik.	20. November 2017
Neue Funktion	Unterstützung für neue integrierte Slot-Typen und Anforderungsattribute wurde hinzugefügt. Weitere Informationen finden Sie unter Integrierte Slot-Typen und Festlegen von Anforderungsattributen .	3. November 2017

Neue Funktion	Es wurde eine Exportmöglichkeit zur Alexa Skills Kit-Funktionalität hinzugefügt. Weitere Informationen finden Sie unter Exportieren in einen Alexa-Skill.	7. September 2017
Neue Funktion	Es wurde ein Synonym-Support für Slot-Typ-Werte hinzugefügt. Weitere Informationen finden Sie unter Benutzerdefinierte Slot-Typen.	31. August 2017
Neue Funktion	Integration in AWS CloudTrail wurde hinzugefügt. Weitere Informationen finden Sie unter Überwachen von Amazon Lex-API-Aufrufen mithilfe vonAWS CloudTrail Protokollen.	15. August 2017
Erweiterte Dokumentation	Erste Schritte-Beispiele für die AWS CLI wurden hinzugefügt. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/lex/latest/dg/gs-cli.html	22. Mai 2017
Neues Handbuch	Dies ist die erste Version des Amazon Lex Benutzerhandbuch.	19. April 2017

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.