aws

Guida per gli sviluppatori

Amazon Lex versione 1

Amazon Lex versione 1: Guida per gli sviluppatori

Table of Contents

Se utilizzi Amazon Lex V2, consulta invece la guida Amazon Lex V2.

Se utilizzi Amazon Lex V1, ti consigliamo di aggiornare i bot ad Amazon Lex V2. Non stiamo più aggiungendo nuove funzionalità alla V1 e consigliamo vivamente di utilizzare la V2 per tutti i nuovi bot.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Che cos'è Amazon Lex?

Amazon Lex è un servizio AWS per la creazione di interfacce di comunicazione tramite voce e testo in qualsiasi applicazione. Con Amazon Lex, lo stesso motore di comunicazione su cui si basa Amazon Alexa è ora a disposizione di tutti gli sviluppatori, permettendoti così di creare chatbot sofisticati e naturali nelle applicazioni nuove ed esistenti. Amazon Lex offre funzionalità e flessibilità avanzate per la comprensione del linguaggio naturale (NLU) e il riconoscimento vocale automatico (ASR), consentendoti di creare esperienze utente coinvolgenti con interazioni di comunicazione naturali, oltre che di creare nuove categorie di prodotti.

Amazon Lex consente a qualsiasi sviluppatore di creare rapidamente chatbot di conversazione. Con Amazon Lex, non è necessaria alcuna esperienza nell'apprendimento profondo: per creare un bot è sufficiente che specifichi il flusso di conversazione di base nella console di Amazon Lex. Amazon Lex gestisce il dialogo e regola in modo dinamico le risposte nella conversazione. Tramite la console, puoi creare, testare e pubblicare chatbot di testo o vocali. Quindi puoi aggiungere ai bot le interfacce di comunicazione sui dispositivi mobili, le applicazioni Web e le piattaforme di chat (ad esempio, Facebook Messenger).

Amazon Lex fornisce un'integrazione preconfigurata conAWS Lambdae puoi eseguire facilmente l'integrazione con molti altri servizi sulla piattaforma AWS, tra cui Amazon Cognito,AWS Mobile Hub, Amazon CloudWatch e Amazon DynamoDB. L'integrazione con Lambda fornisce ai bot l'accesso ai connettori aziendali serverless predefiniti per effettuare il collegamento ai dati nelle applicazioni SaaS, come Salesforce, HubSpot o Marketo.

Alcuni dei vantaggi derivanti dall'utilizzo di Amazon Lex sono:

- Semplicità- Amazon Lex ti assiste nell'utilizzo della console per creare in pochi minuti il tuo chatbot. Tu fornisci solo alcune frasi di esempio e Amazon Lex crea un modello completo di linguaggio naturale tramite cui il bot può interagire utilizzando voce e testo per porre domande, ottenere risposte e completare operazioni sofisticate.

- Tecnologie di apprendimento profondo democratizzateBasato sulla stessa tecnologia di Alexa, Amazon Lex fornisce tecnologie ASR e NLU per creare un sistema di comprensione del linguaggio parlato (SLU). Tramite il sistema SLU, Amazon Lex utilizza l'input di testo e vocale del linguaggio naturale, comprende l'intento alla base dell'input e realizza l'intento dell'utente invocando la funzione aziendale appropriata.

In ambito informatico, il riconoscimento vocale e la comprensione del linguaggio naturale sono tra i problemi più complicati affrontati finora, tanto che necessitano di algoritmi di apprendimento approfondito molto complessi da addestrare con enormi quantità di dati e infrastrutture apposite. Amazon Lex mette a disposizione di tutti gli sviluppatori le tecnologie di apprendimento profondo, basato sulla stessa tecnologia di Alexa. I chatbot Amazon Lex convertono in testo le parti parlate in entrata e comprendono l'intento dell'utente per generare una risposta intelligente, permettendoti di concentrarti sulla creazione dei tuoi bot con un valore aggiunto diversificato per i tuoi clienti per definire categorie di prodotti completamente nuove grazie alle interfacce di comunicazione.

- Installazione e scalabilità senza interruzioni- Con Amazon Lex puoi creare, testare e distribuire i tuoi chatbot direttamente dalla console di Amazon Lex. Amazon Lex ti permette di pubblicare facilmente i tuoi chatbot vocali e di testo da utilizzare sui dispositivi mobili, le applicazioni Web e i servizi di chat (ad esempio, Facebook Messenger). Amazon Lex ricalibra automaticamente le risorse, pertanto non devi pensare al provisioning di hardware e alla gestione dell'infrastruttura per sviluppare la tua esperienza con i bot.

- Integrazione predefinita con la piattaforma AWS- Amazon Lex dispone dell'interoperabilità nativa con altri servizi AWS, come Amazon Cognito,AWS Lambda, Amazon CloudWatch eAWS Mobile Hub. Puoi trarre vantaggio dalla potenza della piattaforma AWS per la sicurezza, il monitoraggio, l'autenticazione utente, la logica di business, lo storage e lo sviluppo di applicazioni mobili.

- Convenienza- Con Amazon Lex, non vi sono costi anticipati o tariffe minime. Ti vengono addebitate solo le richieste vocali o di testo che effettui. I prezzi in base al consumo e i costi ridotti a richiesta rendono il servizio un modo conveniente di creare interfacce di comunicazione. Con il piano gratuito di Amazon Lex, puoi provare Amazon Lex facilmente, senza alcun investimento iniziale.

Sei un primo utente di Amazon Lex?

Se utilizzi Amazon Lex per la prima volta, ti consigliamo di leggere le sezioni seguenti nell'ordine indicato:

1. Guida introduttiva ad Amazon Lex- in questa sezione puoi configurare il tuo account e testare Amazon Lex.
2. Documentazione di riferimento delle API : in questa sezione vengono forniti ulteriori esempi che puoi utilizzare per esplorare Amazon Lex.

Amazon Lex: come funziona: come funziona

Amazon Lex ti consente di creare applicazioni utilizzando un'interfaccia vocale o testuale basata sulla stessa tecnologia che alimenta Amazon Alexa. Di seguito sono riportati i passaggi tipici da eseguire quando si lavora con Amazon Lex:

1. Crea un bot e configuralo con uno o più intenti che desideri supportare. Configura il bot in modo che comprenda l'obiettivo dell'utente (intento), inizi una conversazione per ottenere le informazioni e realizzi l'intento dell'utente.
2. Esegui il test del bot. Puoi utilizzare il client della finestra di test fornito dalla console Amazon Lex.
3. Pubblica una versione e crea un alias.
4. Distribuisci il bot. Puoi distribuire il bot sulle piattaforme come le applicazioni mobili oppure su piattaforme di messaggistica come Facebook Messenger.

Prima di iniziare, acquisisci familiarità con i concetti principali e la terminologia di i concetti principali e la terminologia di la terminologia di Amazon Lex, riportati di di riferimento, riportati di di

- Bot: un bot esegue attività automatizzate come ordinare una pizza, prenotare un hotel, ordinare fiori e così via. Un bot Amazon Lex è basato sulle funzionalità di riconoscimento vocale automatico (ASR) e comprensione del linguaggio naturale (NLU). Ogni bot deve avere deve avere un nome univoco deve avere deve avere un nome univoco.

  I bot Amazon Lex sono in grado di comprendere gli input degli utenti forniti tramite testo o voce e conversare in linguaggio naturale. È possibile creare funzioni Lambda e aggiungerle come hook di codice nella configurazione degli intenti per eseguire attività di convalida e gestione dei dati utente.

- Un intento: un intento rappresenta un'operazione che l'utente vuole eseguire. Si crea un bot per supportare uno o più intenti correlati. Ad esempio, potresti creare un bot che ordina pizze e bevande. Per ogni intento, fornisci le informazioni obbligatorie seguenti:

  - Nome dell'intento: un nome descrittivo per l'intento. Ad esempio, **`OrderPizza`**. I nomi di intenti devono essere univoci devono essere univoci devono essere univoci devono essere univoci

- Esempi di enunciati: come un utente potrebbe esprimere l'intento. Ad esempio, un utente potrebbe dire "Posso ordinare una pizza per favore?" oppure "Vorrei ordinare una pizza".
- Come soddisfare l'intento: come desideri soddisfare l'intento dopo che l'utente ha fornito le informazioni necessarie (ad esempio, ha effettuato un ordine presso una pizzeria locale). Consigliamo di creare una funzione Lambda per soddisfare l'intento.

  Facoltativamente, puoi configurare l'intento in modo che Amazon Lex restituisca semplicemente le informazioni all'applicazione client per eseguire l'adempimento necessario.

Oltre agli scopi personalizzati, come ordinare una pizza, Amazon Lex offre anche funzionalità integrate per configurare rapidamente il bot. Per ulteriori informazioni, consulta Intenti incorporati e tipi di slot.

- Slot: un intento può richiedere zero o più slot o parametri. Si aggiungono slot come parte della configurazione dell'intento. In fase di esecuzione, Amazon Lex richiede all'utente valori di slot specifici. L'utente deve fornire i valori per tutti gli slot richiesti prima che Amazon Lex possa soddisfare l'intento.

  Ad esempio, l'intento `OrderPizza` richiede slot come le dimensioni della pizza, il tipo di crosta e il numero di pizze. Tali slot devono essere aggiunti nella configurazione dell'intento. Per ogni slot, fornisci il tipo di slot e una richiesta che Amazon Lex deve inviare al client per ottenere dati dall'utente. Un utente può rispondere con un valore di slot che includa parole aggiuntive, come «pizza grande per favore» o «limitiamoci a quella piccola». Amazon Lex è ancora in grado di comprendere il valore previsto dello slot.

- Tipo di slot: ogni slot ha un tipo. Puoi creare i tuoi tipi di slot personalizzati o utilizzare i tipi di slot incorporati. Ogni tipo di slot deve avere deve avere deve avere un nome univoco deve avere deve avere deve avere un nome univoco. Ad esempio, puoi creare e utilizzare i tipi di slot riportati sotto per l'intento `OrderPizza`:

- Dimensioni: con valori di enumerazione `Small` `Medium` e `Large`.
- Crosta: con valori di enumerazione `Thick` e `Thin`.

Amazon Lex fornisce inoltre fornisce inoltre fornisce fornisce fornisce fornisce fornisce fornisce fornisce fornisce inoltre di Ad esempio, `AMAZON.NUMBER` è un tipo di slot incorporato che puoi utilizzare per il numero di pizze ordinate. Per ulteriori informazioni, consulta Intenti incorporati e tipi di slot.

Per un elenco delle regioni AWS in cui Amazon Lex è disponibile, consulta Endpoint del servizio AWS nella Guida di riferimento generale di Amazon Web Services.

Negli argomenti seguenti vengono fornite informazioni aggiuntive. Consigliamo di leggerle in ordine e di esaminare gli esercizi Guida introduttiva ad Amazon Lex.

Argomenti

Lingue supportate in Amazon Lex

Amazon Lex V1 supporta una varietà di lingue e impostazioni locali. Le lingue supportate e le funzionalità che le supportano sono elencate nelle tabelle seguenti.

Amazon Lex V2 supporta lingue aggiuntive, vedi Lingue supportate in Amazon Lex V2

Lingue e impostazioni locali supportate

Amazon Lex V1 supporta le seguenti lingue e impostazioni locali.

Codice	Lingue e locali
de-DE	Tedesco (tedesco)
en-AU	Inglese (Australia)
en-GB	Inglese (Regno Unito)
en-IN	Inglese (India)
it-IT	Inglese (Stati Uniti)
it-419	Spagnolo (America Latina)
es-ES	Spagnolo (Spagna)
es-US	Spagnolo (Stati Uniti)
fr-CA	Francese (Canada)
fr-FR	Francese (Francia)
it-IT	Italiano (Italia)
ja-JP	Giapponese (Giappone)
ko-KR	Coreano (Corea)

Lingue e impostazioni locali supportate dalle funzionalità di Amazon Lex

Tutte le funzionalità di Amazon Lex sono supportate in tutte le lingue e in tutte le lingue, ad eccezione di quelle elencate in questa tabella.

Caratteristica	Lingue e impostazioni locali supportate
Impostazione del contesto dell'intento	Inglese (Stati Uniti) (en-US)

Modello di programmazione

Un bot è il tipo di risorsa principale in Amazon Lex. Gli altri tipi di risorse in Amazon Lex sono l'intento, il tipo di slot, l'alias e l'associazione dei canali del bot.

Puoi creare un bot utilizzando la console Amazon Lex o l'API per la creazione di modelli. La console fornisce un'interfaccia utente grafica che puoi usare per creare un bot pronto per la produzione per la tua applicazione. Se preferisci, puoi utilizzare l'API di costruzione del modello tramite la AWS CLI o il tuo programma personalizzato per creare un bot.

Dopo aver creato un bot, devi distribuirlo in una delle piattaforme supportate oppure devi integrarlo nella tua applicazione. Quando un utente interagisce con il bot, l'applicazione client invia richieste al bot utilizzando l'API di runtime di Amazon Lex. Ad esempio, quando un utente dice «Voglio ordinare una pizza», il cliente invia questo input ad Amazon Lex utilizzando una delle operazioni API di runtime. Gli utenti possono fornire input vocali o di testo.

È inoltre possibile creare funzioni Lambda e utilizzarle con un intento. Utilizza questi hook di codice delle funzioni Lambda per eseguire attività di runtime come l'inizializzazione, la convalida dell'input dell'utente e l'adempimento degli intenti. Nelle sezioni seguenti vengono fornite informazioni aggiuntive.

Argomenti

- Operazioni API di costruzione dei modelli
- Operazioni API di runtime
- Funzioni Lambda come ganci di codice

Operazioni API di costruzione dei modelli

Per creare in modo programmatico bot, intenti e tipi di slot, puoi utilizzare le operazioni API di costruzione dei modelli. Puoi anche utilizzare l'API di costruzione dei modelli per gestire, aggiornare ed eliminare risorse per il tuo bot. Le operazioni API di costruzione dei modelli includono:

- PutBot, PutBotAlias, PutIntent e PutSlotType per creare e aggiornare rispettivamente bot, alias dei bot, intenti e tipi di slot.
- CreateBotVersion, CreateIntentVersion e CreateSlotTypeVersion per creare e pubblicare rispettivamente versioni dei tuoi bot, intenti e tipi di slot.
- GetBot e GetBots per ottenere rispettivamente un bot specifico o un elenco di bot che hai creato.
- GetIntent e GetIntents per ottenere rispettivamente un intento specifico o elenco di intenti che hai creato.
- GetSlotType e GetSlotTypes per ottenere rispettivamente uno tipo di slot specifico o elenco di tipi di slot che hai creato.
- GetBuiltinIntenteGetBuiltinSlotTypes per ottenere un intento integrato in Amazon Lex, un elenco di intenti integrati in Amazon Lex o un elenco di tipi di slot integrati che puoi utilizzare nel tuo bot, rispettivamente.GetBuiltinIntents
- GetBotChannelAssociation e GetBotChannelAssociations per ottenere rispettivamente un'associazione tra il tuo bot e una piattaforma di messaggistica o un elenco di associazioni tra il tuo bot e piattaforme di messaggistica.
- DeleteBot, DeleteBotAlias, DeleteBotChannelAssociation, DeleteIntent e DeleteSlotType per rimuovere risorse non necessarie dal tuo account.

Puoi utilizzare l'API di creazione di modelli per creare strumenti personalizzati per gestire le tue risorse Amazon Lex. Ad esempio, esiste un limite di 100 versioni per ogni bot, intento e tipo di slot. Puoi utilizzare l'API di costruzione dei modelli per creare uno strumento che elimini automaticamente le vecchie versioni quando il tuo bot si avvicina al limite.

Per assicurarsi che una sola operazione aggiorni una risorsa alla volta, Amazon Lex utilizza i checksum. Quando si utilizza un'operazione`Put` API—PutBotPutBotAliasPutIntent, oPutSlotType —per aggiornare una risorsa, è necessario passare il checksum corrente della risorsa nella richiesta. Se due strumenti tentano di aggiornare una risorsa contemporaneamente, entrambi forniscono lo stesso checksum corrente. La prima richiesta per raggiungere Amazon Lex corrisponde al checksum corrente della risorsa. Nel momento in cui arriva la seconda richiesta, il checksum è differente. Il

secondo strumento riceve un'eccezione `PreconditionFailedException` e l'aggiornamento termina.

Le`Get` operazioniGetBot, eGetIntent, allaGetSlotType fine sono coerenti. Se utilizzi un'operazione `Get` immediatamente dopo aver creato o modificato una risorsa con una delle operazioni `Put`, le modifiche potrebbero non essere restituite. Dopo che un'operazione `Get` restituisce l'aggiornamento più recente, restituisce sempre quella risorsa aggiornata fino a quando la risorsa non viene nuovamente modificata. È possibile determinare se una risorsa aggiornata è stata restituita osservando il checksum.

Operazioni API di runtime

Le applicazioni client utilizzano le seguenti operazioni API di runtime per comunicare con Amazon Lex:

- PostContent— Accetta l'input vocale o testuale e restituisce informazioni sull'intento e un messaggio di testo o vocale da trasmettere all'utente. Attualmente, Amazon Lex supporta i formati audio seguenti:

  Formati audio di input: LPCM e Opus

  Formati audio di output: MPEG, OGG e PCM

  L'operazione `PostContent` supporta input audio a 8 kHz e 16 kHz. Le applicazioni in cui l'utente finale parla con Amazon Lex al telefono, come un call center automatizzato, possono trasmettere direttamente un audio a 8 kHz.

- PostText: prende il testo come input e restituisce informazioni sull'intento e un messaggio di testo da comunicare all'utente.

L'applicazione client utilizza l'API di runtime per chiamare un bot Amazon Lex specifico per elaborare le espressioni: testo o input vocale dell'utente. Ad esempio, supponiamo che un utente dica "Voglio una pizza". Il client invia questo input utente a un bot utilizzando una delle operazioni API di runtime di Amazon Lex. Dall'input dell'utente, Amazon Lex riconosce che la richiesta dell'utente riguarda

l'`OrderPizza`intento definito nel bot. Amazon Lex coinvolge l'utente in una conversazione per raccogliere le informazioni richieste o i dati relativi agli slot, come le dimensioni della pizza, i condimenti e il numero di pizze. Dopo che l'utente ha fornito tutti i dati di slot necessari, Amazon Lex richiama l'hook del codice della funzione Lambda per soddisfare l'intento o restituisce i dati dell'intento al client, a seconda di come è configurato l'intento.

Usa l'operazione PostContent quando il tuo bot utilizza l'input vocale. Ad esempio, un'applicazione automatizzata per call center può inviare messaggi vocali a un bot Amazon Lex anziché a un agente per rispondere alle richieste dei clienti. Puoi utilizzare il formato audio a 8 kHz per inviare l'audio direttamente dal telefono ad Amazon Lex.

La finestra di test nella console Amazon Lex utilizza l'PostContentAPI per inviare richieste vocali e di testo ad Amazon Lex. Puoi usare questa finestra di prova negli esercizi Guida introduttiva ad Amazon Lex.

Funzioni Lambda come ganci di codice

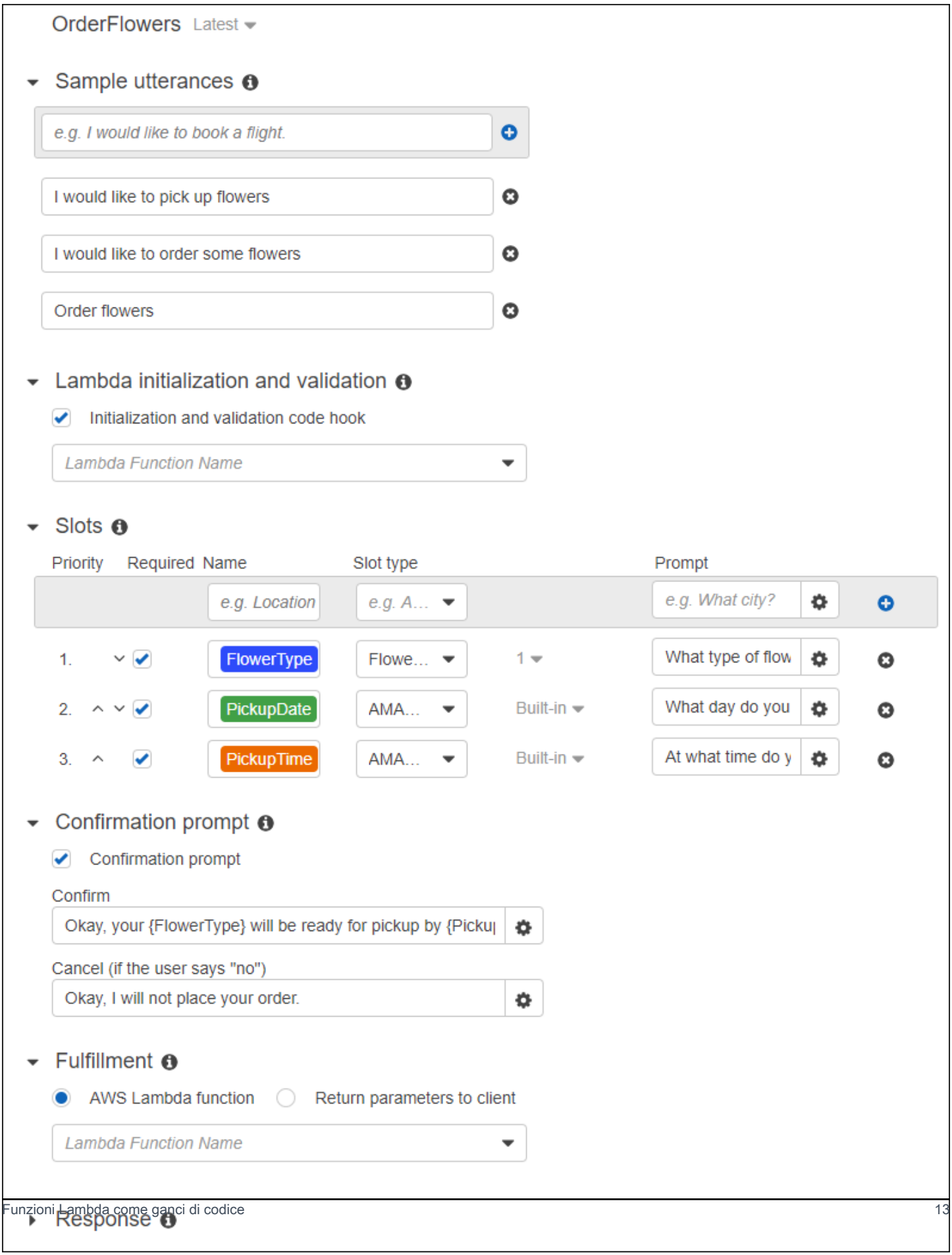
Puoi configurare il tuo bot Amazon Lex per richiamare una funzione Lambda come hook di codice. L'hook di codice può servire a più scopi:

- Personalizza l'interazione con l'utente: ad esempio, quando Joe chiede i condimenti disponibili per la pizza, puoi utilizzare una conoscenza preliminare delle scelte di Joe per visualizzare un sottoinsieme di condimenti.
- Convalida l'input dell'utente, supponiamo che Jen voglia raccogliere fiori fuori orario. È possibile convalidare l'ora che Jen inserisce e inviare una risposta appropriata.
- Soddisfa l'intenzione dell'utente: dopo che Joe ha fornito tutte le informazioni per il suo ordine di pizza, Amazon Lex può richiamare una funzione Lambda per effettuare l'ordine presso una pizzeria locale.

Quando si configura un intento, si specificano le funzioni Lambda come hook di codice nei seguenti punti:

- Hook di codice di dialogo per l'inizializzazione e la convalida: questa funzione Lambda viene richiamata su ogni input dell'utente, supponendo che Amazon Lex abbia compreso l'intento dell'utente.
- Codice di adempimento Hook: questa funzione Lambda viene richiamata dopo che l'utente ha fornito tutti i dati dello slot necessari per soddisfare l'intento.

Scegli l'intento e imposta gli hook di codice nella console Amazon Lex, come mostrato nella schermata seguente:


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

È anche possibile impostare gli hook dei codici utilizzando i campi `dialogCodeHook` e `fulfillmentActivity` nell'operazione PutIntent.

Una funzione Lambda può eseguire l'inizializzazione, la convalida e l'adempimento. I dati sugli eventi ricevuti dalla funzione Lambda contengono un campo che identifica il chiamante come finestra di dialogo o codice di evasione. È possibile utilizzare queste informazioni per eseguire la parte di codice appropriata.

È possibile utilizzare una funzione Lambda per creare un bot in cui è possibile utilizzare una funzione Lambda per creare un bot in cui è possibile navigare in una finestra di dialogo di più. Utilizzi il`dialogAction` campo nella risposta della funzione Lambda per indirizzare Amazon Lex a intraprendere azioni specifiche. Ad esempio, puoi utilizzare l'azione`ElicitSlot` di dialogo per dire ad Amazon Lex di chiedere all'utente un valore di slot non obbligatorio. Se è stato definito un prompt di chiarimento, è possibile utilizzare l'operazione della finestra di dialogo `ElicitIntent` per ottenere un nuovo intento quando l'utente ha terminato con quello precedente.

Per ulteriori informazioni, consulta Utilizzo delle funzioni Lambda.

Gestione dei messaggi

Argomenti

- Tipi di messaggi
- Contesti per la configurazione di messaggi
- Formati di messaggio supportati
- Gruppi di messaggi
- Schede di risposta

Quando crei un bot, puoi configurare messaggi illustrativi o informativi da inviare al client. Considerare i seguenti esempi:

- Puoi configurare il tuo bot con il prompt illustrativo seguente:

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

  Amazon Lex invia questo messaggio al cliente se non comprende le intenzioni dell'utente.

- Supponiamo che venga creato un bot per supportare un intento denominato `OrderPizza`. Per un ordine di pizza, gli utenti devono fornire informazioni quali le dimensioni della pizza, i contorni e il tipo di crosta. È possibile configurare i seguenti prompt:

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

  Dopo aver determinato l'intenzione dell'utente di ordinare una pizza, Amazon Lex invia questi messaggi al cliente per ottenere informazioni dall'utente.

In questa sezione viene descritta la progettazione delle interazioni dell'utente nella configurazione del tuo bot.

Tipi di messaggi

Un messaggio può essere un prompt o un'istruzione.

- Un prompt è normalmente una domanda che prevede la risposta dell'utente.
- Un'istruzione è informativa. Non prevede una risposta.

Un messaggio può includere riferimenti a slot, attributi di sessione e attributi di richiesta. In fase di esecuzione, Amazon Lex sostituisce questi riferimenti con valori effettivi.

Per fare riferimento ai valori slot che sono stati impostati, utilizza la sintassi seguente:

```
{SlotName}
```

Per fare riferimento agli attributi di sessione, utilizza la sintassi seguente:

```
[SessionAttributeName]
```

Per fare riferimento agli attributi di richiesta, utilizza la sintassi seguente:

```
((RequestAttributeName))
```

I messaggi possono includere valori di slot, attributi di sessione e attributi di richiesta.

Supponiamo di aver configurato il seguente seguente, riportati di di di di di di di di di, riportati di di di di di di di OrderPizza di di di di di

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

Questo messaggio fa riferimento sia ad attributi slot (`PizzaTopping`) sia ad attributi di sessione (`FirstName` e `DeliveryTime`). In fase di esecuzione, Amazon Lex sostituisce questi segnaposti con valori e restituisce al client il seguente messaggio:

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

Per includere parentesi quadre ([]) o graffe ({}) in un messaggio, utilizza il carattere escape della barra rovesciata (\). Ad esempio, il messaggio seguente include parentesi graffe e parentesi quadre:

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

Il testo restituito all'applicazione del client ha questo aspetto:

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

Per informazioni sugli attributi di sessione, consulta le operazioni API di runtime PostText e PostContent. Per un esempio, consulta Prenota viaggio.

Le funzioni Lambda possono anche generare messaggi e restituirli ad Amazon Lex per inviarli all'utente. Se si aggiungono funzioni Lambda quando si configura l'intento, è possibile creare messaggi in modo dinamico. Fornendo i messaggi durante la configurazione del bot, è possibile eliminare la necessità di creare un prompt nella funzione Lambda.

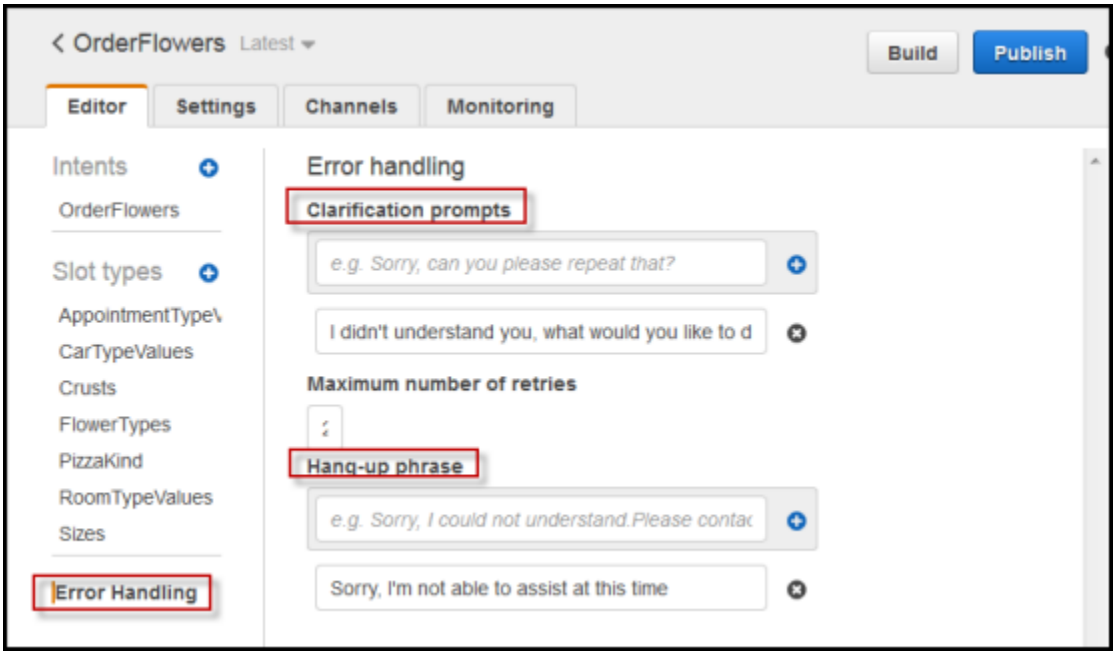
Contesti per la configurazione di messaggi

Durante la creazione del bot, è possibile creare messaggi in diversi contesti, ad esempio richieste di chiarimento nel bot, richieste per i valori degli slot e messaggi relativi agli intenti. Amazon Lex sceglie un messaggio appropriato in ogni contesto da restituire all'utente. Puoi fornire un gruppo di messaggi per ogni contesto. In tal caso, Amazon Lex sceglie a caso un messaggio dal gruppo. Puoi specificare anche il formato del messaggio oppure raggruppare i messaggi. Per ulteriori informazioni, consulta Formati di messaggio supportati.

Se hai una funzione Lambda associata a un intento, puoi sovrascrivere qualsiasi messaggio configurato in fase di compilazione. Tuttavia, non è necessaria una funzione Lambda per utilizzare nessuno di questi messaggi.

Messaggi bot

Puoi configurare il tuo bot con richieste di chiarimento e messaggi di fine sessione. In fase di esecuzione, Amazon Lex utilizza la richiesta di chiarimenti se non comprende l'intento dell'utente. Puoi configurare il numero di volte in cui Amazon Lex richiede chiarimenti prima di inviare il messaggio di fine sessione. Puoi configurare i messaggi a livello di bot nella sezione Gestione degli errori della console Amazon Lex, come nell'immagine seguente:



Con l'API, si possono configurare messaggi impostando i campi `clarificationPrompt` e `abortStatement` nell'operazione PutBot.

Se utilizzi una funzione Lambda con un intento, la funzione Lambda potrebbe restituire una risposta che indirizza Amazon Lex a chiedere l'intento di un utente. Se la funzione Lambda non fornisce tale messaggio, Amazon Lex utilizza la richiesta di chiarimenti.

Prompt di slot

È necessario specificare almeno un messaggio di prompt per ognuno degli slot obbligatori in un intento. In fase di esecuzione, Amazon Lex utilizza uno di questi messaggi per richiedere all'utente di fornire un valore per lo slot. Ad esempio, per uno slot `cityName`, il seguente è un prompt valido:

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

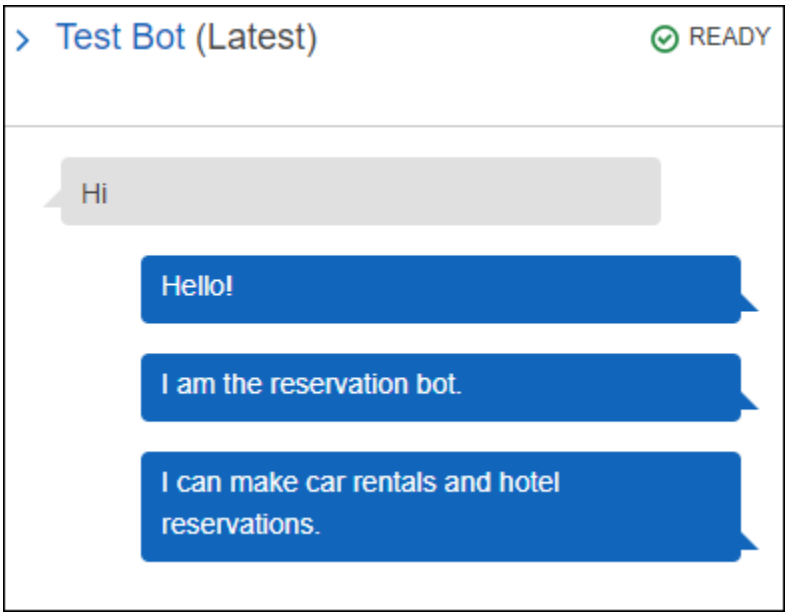
Puoi impostare uno o più prompt per ogni slot utilizzando la console. Puoi anche creare gruppi di prompt utilizzando l'operazione PutIntent. Per ulteriori informazioni, consulta Gruppi di messaggi.

Risposte

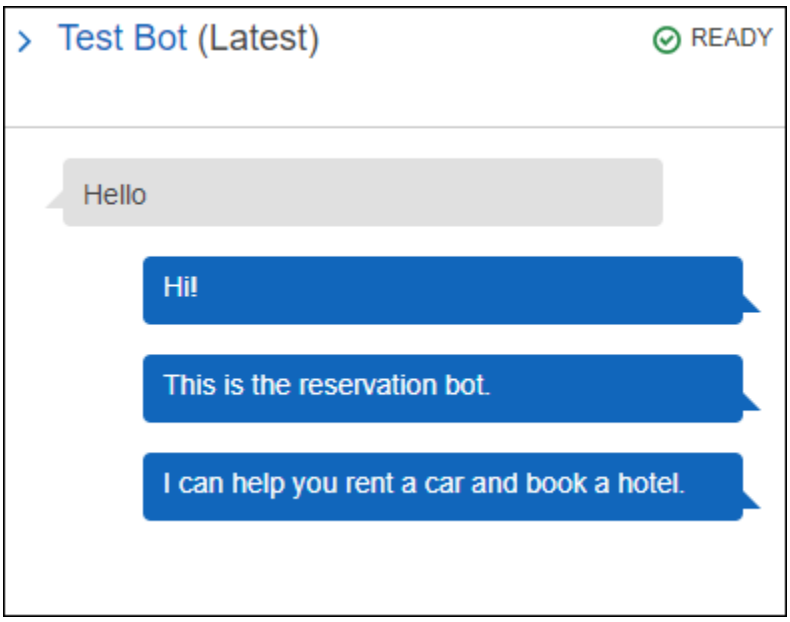
Nella console, utilizza la sezione delle risposte per creare conversazioni dinamiche e coinvolgenti per il tuo bot. Puoi creare uno o più gruppi di messaggi per una risposta. In fase di esecuzione, Amazon Lex crea una risposta selezionando un messaggio da ciascun gruppo di messaggi. Per ulteriori informazioni sui gruppi di messaggi, consulta Gruppi di messaggi.

Ad esempio, il primo gruppo di messaggi può contenere diversi saluti: "Salve", "Ciao" e "Saluti". Il secondo gruppo di messaggi può contenere diverse forme di presentazioni: "Io sono il bot per le prenotazioni" e "Questo è il bot per le prenotazioni". Un terzo gruppo di messaggi può comunicare le funzionalità del bot: "Posso noleggiare un'auto o prenotare un albergo", "Puoi noleggiare un'auto o prenotare un albergo" e "Posso aiutarti a noleggiare un'auto e prenotare un albergo".

Lex utilizza un messaggio da ogni gruppo di messaggi e crea risposte dinamiche dando origine a una conversazione. Ad esempio, un'interazione potrebbe essere la seguente:

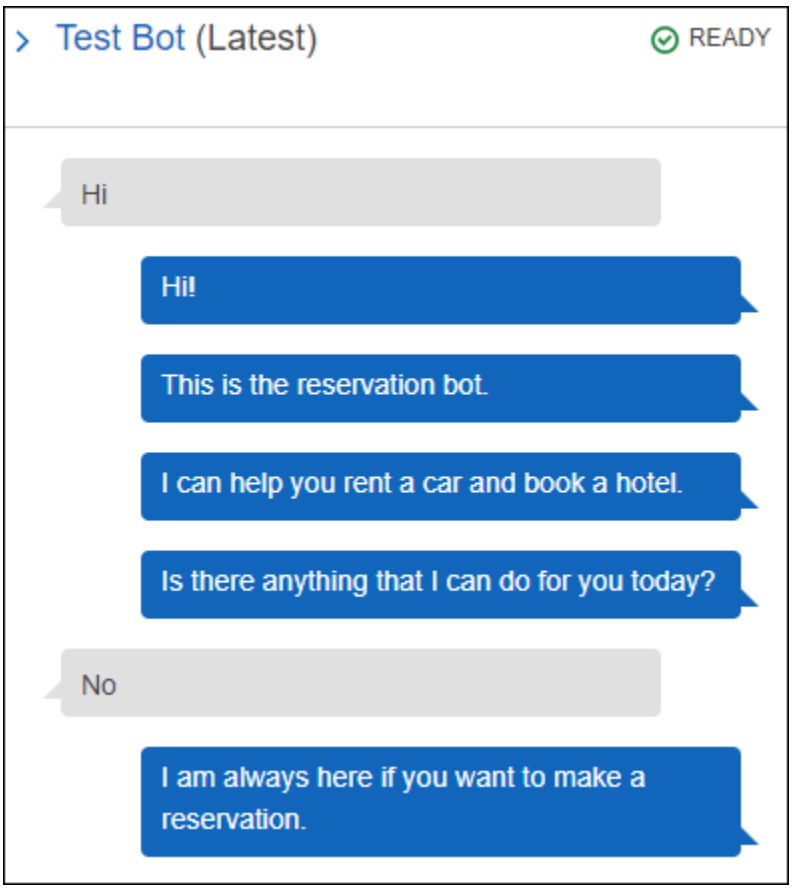


Un'altra potrebbe essere la seguente:

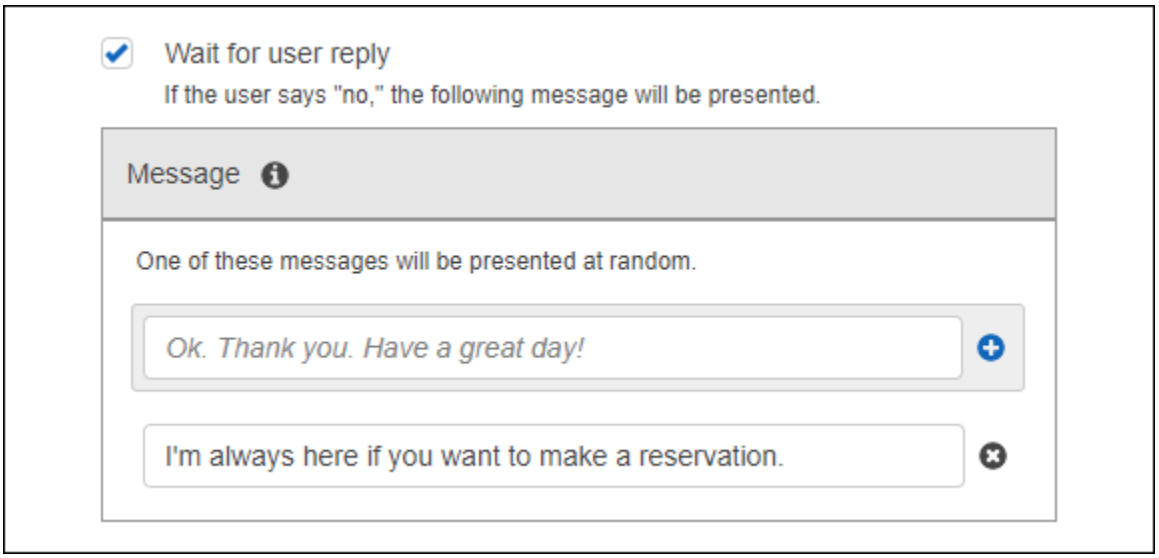


In entrambi i casi, l'utente può rispondere con un nuovo intento, ad esempio l'intento `BookCar` o `BookHotel`.

È possibile impostare il bot in modo che ponga una domanda di follow-up nella risposta. Ad esempio, per l'interazione precedente, puoi creare un quarto gruppo di messaggi con le domande seguenti: "Posso essere d'aiuto con un'auto o un albergo?", "Desideri effettuare una prenotazione ora?" e "C'è qualcosa che posso fare per te?". Per messaggi che includono "No" come risposta, puoi creare un prompt di follow-up. Il seguente seguente seguente seguente seguente seguente seguente seguente seguente seguente seguente seguente seguente seguente seguente



Per creare un prompt di follow-up, scegli Attendi risposta utente. Successivamente digita il messaggio o i messaggi che desideri inviare quando un utente dice "No". Quando crei una risposta da utilizzare come prompt di follow-up, è necessario specificare anche un'istruzione se la risposta all'istruzione è "No". Vedi l'immagine seguente per un esempio:



Per aggiungere risposte a un intento con l'API, utilizza l'operazione `PutIntent`. Per specificare una risposta, imposta il campo `conclusionStatement` nella richiesta `PutIntent`. Per impostare un prompt di follow-up, imposta il campo `followUpPrompt` e includi l'istruzione da inviare se l'utente dice "No". Non puoi impostare entrambi i campi `conclusionStatement` e `followUpPrompt` nello stesso intento.

Formati di messaggio supportati

Quando usi l'PostTextoperazione o quando usi l'PostContentoperazione con l'`Accept`intestazione impostata su`text/plain;charset=utf8`, Amazon Lex supporta i messaggi nei seguenti formati:

- `PlainText`—Il messaggio contiene testo UTF-8 semplice.
- `SSML`—Il messaggio contiene testo formattato per l'output vocale.
- `CustomPayload`—Il messaggio contiene un formato personalizzato che hai creato per il tuo cliente. Puoi definire il payload in base alle esigenze della tua applicazione.
- `Composite`—Il messaggio è una raccolta di messaggi, uno per ogni gruppo di messaggi. Per ulteriori informazioni sui gruppi di messaggi, consulta Gruppi di messaggi.

Per impostazione predefinita, Amazon Lex restituisce uno qualsiasi dei messaggi definiti per un determinato prompt. Ad esempio, se definisci cinque messaggi per ottenere un valore di slot, Amazon Lex sceglie uno dei messaggi a caso e lo restituisce al client.

Se desideri che Amazon Lex restituisca un tipo specifico di messaggio al client in una richiesta in fase di esecuzione, imposta il parametro della`x-amzn-lex:accept-content-types` richiesta. La risposta si limita al tipo o ai tipi richiesti. Se è presente più di un messaggio del tipo specificato, Amazon Lex ne restituisce uno a caso. Per ulteriori informazioni sull'intestazione `x-amz-lex:accept-content-types`, consulta Impostazione del tipo di risposta.

Gruppi di messaggi

Un gruppo di messaggi è un insieme di risposte adatte a un particolare prompt. Utilizza gruppi di messaggi quando desideri che il tuo bot crei risposte dinamiche dando origine a una conversazione. Quando Amazon Lex restituisce una risposta all'applicazione client, sceglie casualmente un messaggio da ciascun gruppo. Puoi creare un massimo di cinque gruppi di messaggi per ogni risposta. Ogni gruppo può contenere un massimo di cinque messaggi. Per esempi su come creare gruppi di messaggi nella console, consulta Risposte.

Per creare un gruppo di messaggi, puoi utilizzare la console o le operazioni PutBot, PutIntent o PutSlotType per assegnare il numero di gruppo a un messaggio. Se non crei un gruppo di messaggi o se crei un solo gruppo di messaggi, Amazon Lex invia un singolo messaggio nel`Message` campo. Le applicazioni dei client ricevono più messaggi in una risposta solo quando hai creato più di un gruppo di messaggi nella console oppure quando crei più di un gruppo di messaggi durante la creazione o l'aggiornamento di un intento con l'operazione PutIntent.

Quando Amazon Lex invia un messaggio da un gruppo, il`Message` campo della risposta contiene un oggetto JSON escape che contiene i messaggi. L'esempio seguente mostra i contenuti del campo `Message` quando include più messaggi.

> **Note**
>
> L'esempio è stato formattato per la leggibilità. In una risposta non sono contenuti ritorni a capo (CR).

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

È possibile impostare il formato dei messaggi. Il formato può essere uno dei seguenti:

- PlainText—Il messaggio è in semplice testo UTF-8.
- SSML: il messaggio è SSML (Speech Synthesis Markup Language).
- CustomPayload—Il messaggio è in un formato personalizzato specificato dall'utente.

Per controllare il formato dei messaggi che le operazioni `PostContent` e `PostText` restituiscono nel campo `Message`, imposta l'attributo della richiesta `x-amz-lex:accept-content-types`. Ad esempio, se si imposta l'intestazione a quanto segue, si ricevono solo messaggi di solo testo e SSML nella risposta:

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

Se si richiede un formato di messaggio specifico e un gruppo di messaggi non contiene quel messaggio con quel formato, ottieni un'eccezione `NoUsableMessageException`. Quando usi

un gruppo di messaggi per raggruppare i messaggi per tipo, non utilizzare l'intestazione `x-amz-lex:accept-content-types`.

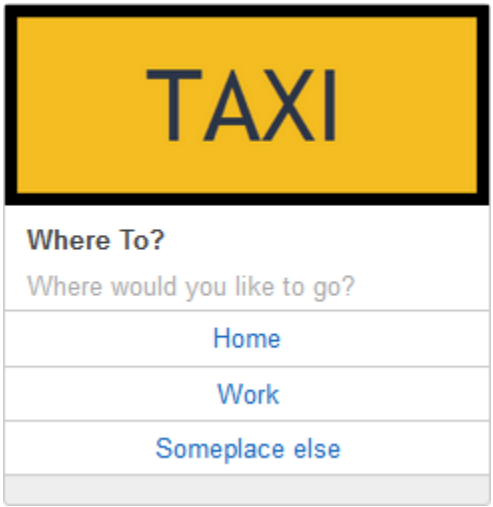
Per ulteriori informazioni sull'intestazione `x-amz-lex:accept-content-types`, consulta Impostazione del tipo di risposta.

Schede di risposta

> **Note**
>
> Le schede di risposta non funzionano con la chat di Amazon Connect. Tuttavia, vedi Aggiungere messaggi interattivi alla chat per funzionalità simili.

Una scheda di risposta contiene un insieme di risposte appropriate per un prompt. Usa le schede di risposta per semplificare le interazioni per i tuoi utenti e aumentare la precisione dei tuoi bot riducendo errori tipografici nelle interazioni di testo. Puoi inviare una scheda di risposta per ogni prompt inviato da Amazon Lex all'applicazione client. Le schede di risposta possono essere utilizzate con Facebook Messenger, Slack, Twilio e le applicazioni del tuo client.

Ad esempio, in un'applicazione per taxi, è possibile configurare un'opzione nella scheda di risposta per "Casa" e impostare il valore sull'indirizzo di casa dell'utente. Quando l'utente seleziona questa opzione, Amazon Lex riceve l'intero indirizzo come testo di input. Vedi l'immagine seguente:



Puoi definire una scheda di risposta per i prompt seguenti:

- Istruzione di conclusione

- Prompt di conferma
- Prompt di follow-up
- Istruzione di rifiuto
- Enunciazioni tipo slot

Puoi definire solo una scheda di risposta per ogni prompt.

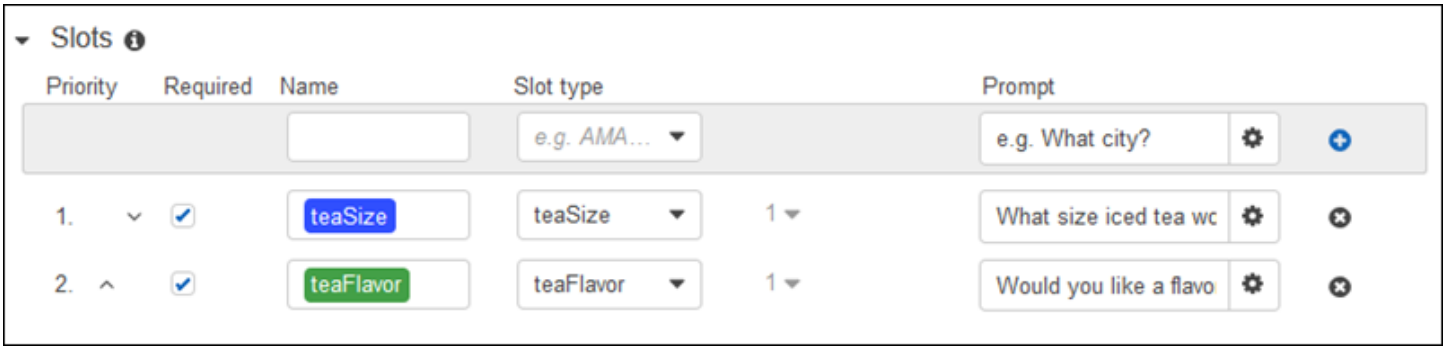
Puoi configurare le schede di risposta quando crei un intento. Puoi definire una scheda di risposta statica in fase di compilazione utilizzando la console o l'operazione PutIntent. Oppure è possibile definire una scheda di risposta dinamica in fase di esecuzione in una funzione Lambda. Se definisci entrambe le schede di risposta statica e dinamica, la scheda di risposta dinamica ha la precedenza.

Amazon Lex invia le schede di risposta nel formato comprensibile al cliente. Questo trasforma le schede di risposta per Facebook Messenger, Slack e Twilio. Per gli altri clienti, Amazon Lex invia una struttura JSON nellaPostText risposta. Ad esempio, se il client è Facebook Messenger, Amazon Lex trasforma la scheda di risposta in un modello generico. Per ulteriori informazioni sui modelli generici di Facebook Messenger, consulta Modello generico sul sito Web di Facebook. Per un esempio della struttura JSON, consulta Generazione dinamica di schede di risposta.

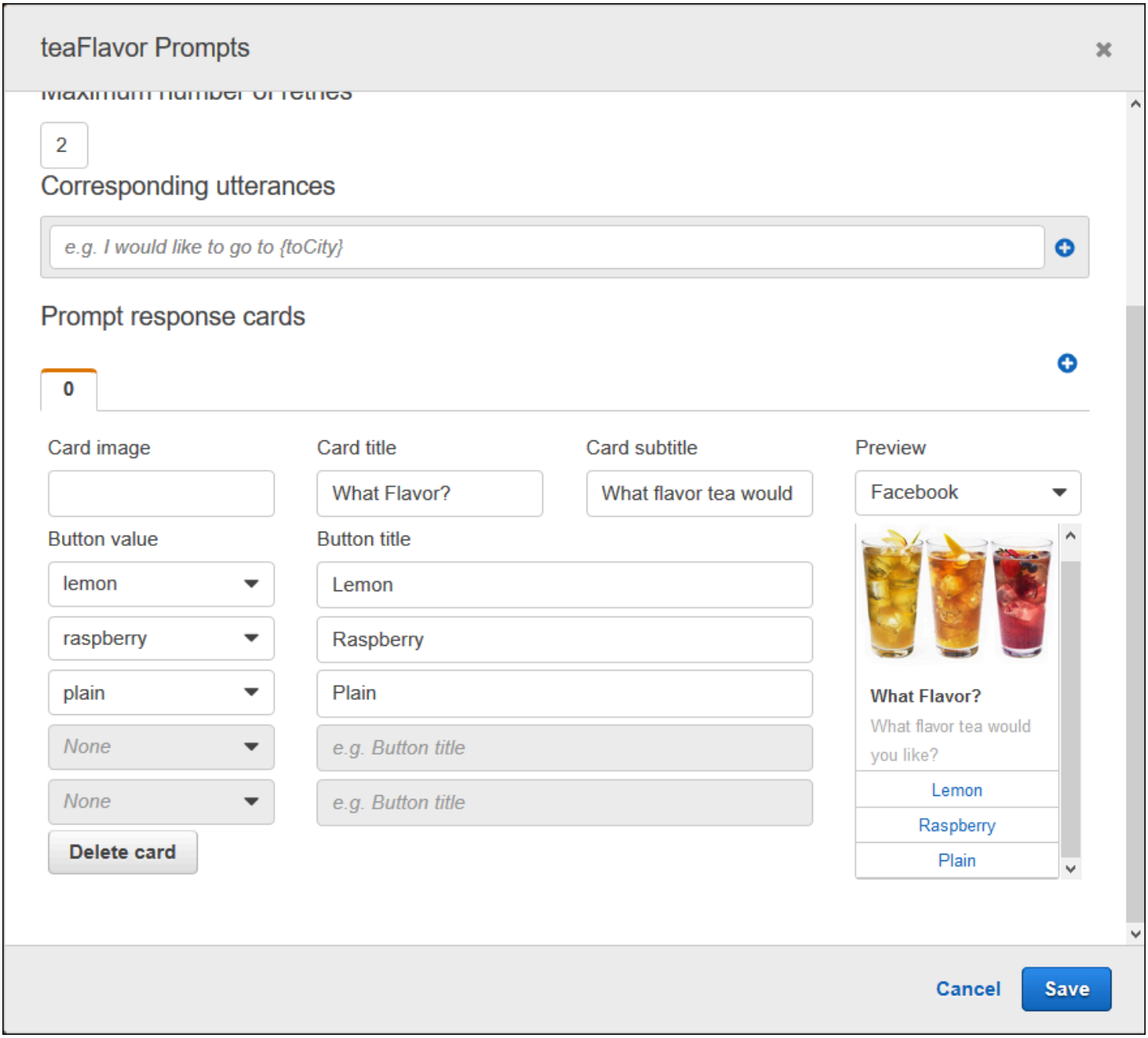
Puoi utilizzare le schede di risposta solo con l'operazione PostText. Non puoi utilizzare le schede di risposta con l'operazione PostContent.

Definizione di Schede di risposta statica

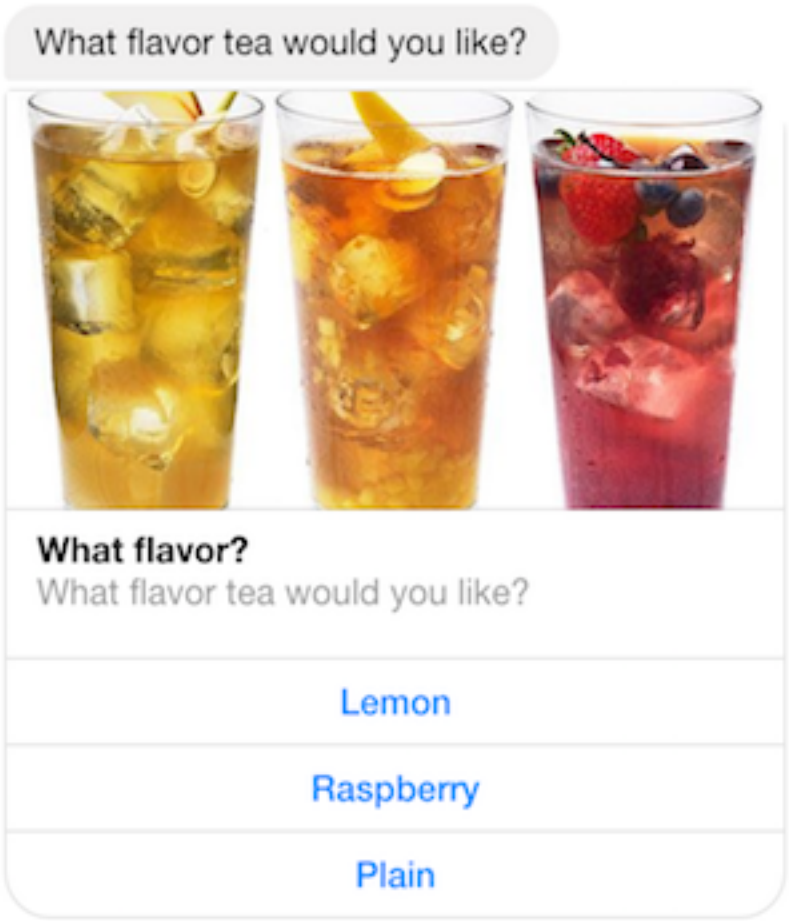
Definisci le schede di risposta statiche con l'PutBotoperazione o la console Amazon Lex quando crei un intento. Una risposta statica viene definita contemporaneamente all'intento. Utilizza una scheda di risposta statica quando le risposte sono fisse. Supponiamo che stai creando un bot con un intento che ha uno slot per il gusto. Quando definisci lo slot del gusto, devi specificare i prompt, come illustrato nella schermata della console seguente:



Quando si definiscono i prompt, è possibile associare facoltativamente una scheda di risposta e definire i dettagli all'PutBotoperazione oppure, nella console Amazon Lex, come mostrato nell'esempio seguente:



A questo punto, supponiamo che il tuo bot sia stato integrato con Facebook Messenger. L'utente può fare clic sui pulsanti per scegliere un gusto, come mostrato nell'illustrazione seguente:



Per personalizzare il contenuto di una scheda di risposta, si può fare riferimento agli attributi della sessione. In fase di esecuzione, Amazon Lex sostituisce questi riferimenti con valori appropriati tratti dagli attributi della sessione. Per ulteriori informazioni, consulta Impostazione degli attributi di sessione. Per un esempio, consulta Utilizzo di una scheda di risposta.

Generazione dinamica di schede di risposta

Per generare schede di risposta in modo dinamico in fase di esecuzione, utilizza la funzione Lambda di inizializzazione e convalida per l'intento. Usa una scheda di risposta dinamica quando le risposte vengono determinate in fase di esecuzione nella funzione Lambda. In risposta all'input dell'utente, la funzione Lambda genera una scheda di risposta e la restituisce nella`dialogAction` sezione della risposta. Per ulteriori informazioni, consulta Formato della risposta.

Quella che segue è una risposta parziale di una funzione Lambda che mostra l'`responseCard`elemento. Questo genera un'esperienza utente simile a quella illustrata nella sezione precedente.

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

Per un esempio, consulta Pianifica un appuntamento.

Gestione del contesto di una conversazione

Il contesto di conversazione è l'informazione che un utente, un'applicazione o una funzione Lambda fornisce a un bot Amazon Lex per soddisfare un intento. Il contesto di conversazione include i dati degli slot forniti dall'utente, gli attributi della richiesta impostati dall'applicazione client e gli attributi di sessione creati dall'applicazione client e dalle funzioni Lambda.

Argomenti

- Impostazione del contesto dell'intento
- Utilizzo dei valori di slot predefiniti

- Impostazione degli attributi di sessione
- Impostazione degli attributi di richiesta
- Impostazione del timeout di sessione
- Condivisione di informazioni tra intenti diversi
- Impostazione di attributi complessi

Impostazione del contesto dell'intento

Puoi impostare gli intenti di attivazione di Amazon Lex in base al contesto. Un contesto è una variabile di stato che può essere associata a un intento quando si definisce un bot.

I contesti per un intento si configurano quando si crea l'intento utilizzando la console o utilizzando l'PutIntentoperazione. Puoi usare solo i contesti nella lingua inglese (USA) (en-US) e solo se imposti il`enableModelImprovements` parametro su`true` quando hai creato il bot con l'PutBotoperazione.

Esistono due tipi di relazioni per i contesti, i contesti di output e i contesti di input. Un contesto di output diventa attivo quando viene soddisfatto un intento associato. Un contesto di output viene restituito all'applicazione nella risposta dell'PostContentoperazionePostText or ed è impostato per la sessione corrente. Dopo l'attivazione, un contesto rimane attivo per il numero di turni o il limite di tempo configurato al momento della definizione del contesto.

Un contesto di input specifica le condizioni in base alle quali un intento può essere riconosciuto. Un'intenzione può essere riconosciuta durante una conversazione solo quando tutti i suoi contesti di input sono attivi. Un'intenzione senza contesti di input è sempre idonea al riconoscimento.

Amazon Lex gestisce automaticamente il ciclo di vita dei contesti attivati soddisfacendo gli intenti con contesti di output. È inoltre possibile impostare contesti attivi in una chiamata all'`PostText`operazione`PostContent` o.

È inoltre possibile impostare il contesto di una conversazione utilizzando la funzione Lambda per l'intento. Il contesto di output di Amazon Lex viene inviato all'evento di input della funzione Lambda. La funzione Lambda può inviare contesti nella sua risposta. Per ulteriori informazioni, consulta Formato di evento di input e di risposta della funzione Lambda.

Ad esempio, supponiamo che tu abbia intenzione di prenotare un'auto a noleggio configurata per restituire un contesto di output chiamato «book_car_fulfilled». Quando l'intento è soddisfatto, Amazon Lex imposta la variabile di contesto di output «book_car_fulfilled». Poiché «book_car_fulfilled» è un

contesto attivo, un intento con il contesto «book_car_fulfilled» impostato come contesto di input viene ora considerato per il riconoscimento, a condizione che un'espressione dell'utente venga riconosciuta come un tentativo di suscitare tale intento. Puoi usarlo per scopi che hanno senso solo dopo aver prenotato un'auto, come inviare una ricevuta via e-mail o modificare una prenotazione.

Contesto di output

Amazon Lex rende attivi i contesti di output di un intento quando l'intento viene soddisfatto. È possibile utilizzare il contesto di output per controllare gli intenti idonei a dare seguito all'intento corrente.

Ogni contesto ha un elenco di parametri che vengono mantenuti nella sessione. I parametri sono i valori dello slot per l'intento raggiunto. È possibile utilizzare questi parametri per precompilare i valori degli slot per altri scopi. Per ulteriori informazioni, consulta Utilizzo dei valori di slot predefiniti.

Il contesto di output viene configurato quando si crea un intento con la console o con l'PutIntentoperazione. È possibile configurare un intento con più di un contesto di output. Quando l'intento è soddisfatto, tutti i contesti di output vengono attivati e restituiti nellaPostContent rispostaPostText or.

Quanto segue mostra l'assegnazione di un contesto di output a un intento utilizzando la console.

Quando definisci un contesto di output, definisci anche la sua durata di vita, la durata o il numero di turni in cui il contesto è incluso nelle risposte di Amazon Lex. Un turno è una richiesta della tua applicazione ad Amazon Lex. Una volta scaduto il numero di turni o il tempo, il contesto non è più attivo.

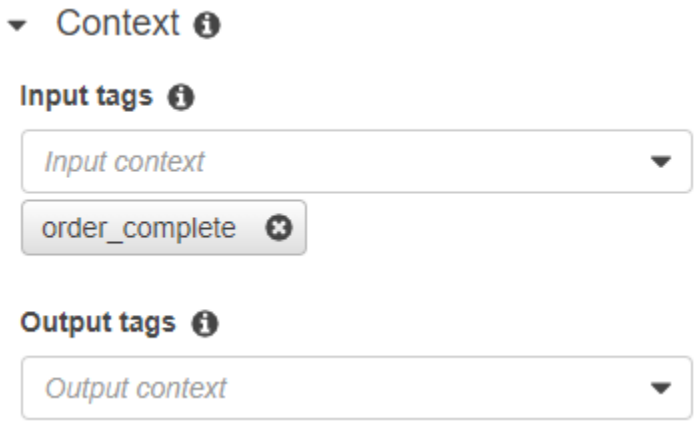
L'applicazione può utilizzare il contesto di output secondo necessità. Ad esempio, l'applicazione può utilizzare il contesto di output per:

- Modifica il comportamento dell'applicazione in base al contesto. Ad esempio, un'applicazione di viaggio potrebbe avere un'azione diversa per il contesto «book_car_fulfilled» rispetto a «rental_hotel_fulfilled».
- Restituisci il contesto di output ad Amazon Lex come contesto di input per l'enunciato successivo. Se Amazon Lex riconosce l'enunciato come un tentativo di suscitare un intento, utilizza il contesto per limitare gli intenti che possono essere restituiti a quelli con il contesto specificato.

Contesto di input

È possibile impostare un contesto di input per limitare i punti della conversazione in cui viene riconosciuto l'intento. Gli intenti senza un contesto di input possono sempre essere riconosciuti.

È possibile impostare i contesti di input a cui risponde un intento utilizzando la console o l'`PutIntent`operazione. Un intento può avere più di un contesto di input. Di seguito viene illustrata l'assegnazione di un contesto di input a un intento utilizzando la console.



Per un intento con più di un contesto di input, tutti i contesti devono essere attivi per attivare l'intento. È possibile impostare un contesto di input quando si chiama l'PutSessionoperazionePostTextPostContent, o.

È possibile configurare gli slot con l'intento di prendere i valori predefiniti dal contesto attivo corrente. I valori predefiniti vengono utilizzati quando Amazon Lex riconosce un nuovo intento ma non riceve un valore di slot. Il nome del contesto e il nome dello slot vengono specificati nel modulo`#context-name.parameter-name` quando si definisce lo slot. Per ulteriori informazioni, consulta Utilizzo dei valori di slot predefiniti.

Utilizzo dei valori di slot predefiniti

Quando si utilizza un valore predefinito, si specifica una fonte per il valore di uno slot da riempire per nuovi scopi quando l'input dell'utente non fornisce alcuno slot. Questa fonte può essere un precedente attributo della finestra di dialogo, della richiesta o della sessione o un valore fisso impostato in fase di compilazione.

È possibile utilizzare quanto segue come origine per i valori di default.

- Finestra di dialogo precedente (contesti) — #context -name.parameter-name
- Attributi della sessione — [nome-attributo]
- Attributi della richiesta — <attribute-name>
- Valore fisso: qualsiasi valore che non corrisponde a quello precedente

Quando si utilizza l'PutIntentoperazione per aggiungere slot a un intento, è possibile aggiungere un elenco di valori predefiniti. I valori predefiniti vengono utilizzati nell'ordine in cui sono elencati. Ad esempio, si supponga di avere un'intenzione con uno slot con la seguente definizione:

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

Quando l'intento viene riconosciuto, lo slot denominato "reservation-start-date" ha il suo valore impostato su uno dei seguenti.

1. Se il contestobook-car-fulfilled "" è attivo, il valore del parametro «startDate» viene utilizzato come valore predefinito.

2. Se il contestobook-car-fulfilled "" non è attivo o se il parametro «startDate» non è impostato, il valore dell'attributo di sessionereservationStartDate "" viene utilizzato come valore predefinito.
3. Se non viene utilizzato nessuno dei primi due valori predefiniti, lo slot non ha un valore predefinito e Amazon Lex genererà un valore come al solito.

Se viene utilizzato un valore predefinito per lo slot, lo slot non viene generato anche se è richiesto.

Impostazione degli attributi di sessione

Gli attributi di sessione contengono informazioni specifiche dell'applicazione che vengono trasmesse tra un bot e un'applicazione client durante una sessione. Amazon Lex passa gli attributi di sessione a tutte le funzioni Lambda configurate per un bot. Se una funzione Lambda aggiunge o aggiorna gli attributi di sessione, Amazon Lex ritrasmette le nuove informazioni all'applicazione client. Ad esempio:

- Nell'Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console), il bot dell'esempio utilizza l'attributo di sessione `price` per gestire il prezzo dei fiori. La funzione Lambda imposta questo attributo in base al tipo di fiori ordinati. Per ulteriori informazioni, consulta Fase 5 (facoltativo): Revisione dei dettagli del flusso di informazioni (console).
- Nell'Prenota viaggio, il bot dell'esempio utilizza l'attributo di sessione `currentReservation` per mantenere una copia dei dati del tipo di slot durante la conversazione per prenotare un hotel o noleggiare un'automobile. Per ulteriori informazioni, consulta Dettagli del flusso di informazioni.

Usa gli attributi di sessione nelle tue funzioni Lambda per inizializzare un bot e personalizzare i prompt e le schede di risposta. Ad esempio:

- Inizializzazione: in un bot per ordinare pizze, l'applicazione client passa la posizione dell'utente come attributo di sessione nella prima chiamata all'PostTextoperazionePostContent or. Ad esempio, `"Location": "111 Maple Street"`. La funzione Lambda utilizza queste informazioni per trovare la pizzeria più vicina a cui effettuare l'ordine.
- Personalize i prompt: configura i prompt e le schede di risposta in modo che facciano riferimento agli attributi della sessione. Ad esempio, «Ehi [FirstName], quali condimenti vorresti?» Se passi il nome dell'utente come attributo di sessione (`{"FirstName": "Jo"}`), Amazon Lex sostituisce il nome con il segnaposto. e invia quindi un prompt personalizzato all'utente: "Salve Giovanni, quali ingredienti aggiuntivi desideri?"

Gli attributi della sessione vengono mantenuti per tutta la durata della sessione. Amazon Lex li archivia in un archivio dati crittografato fino al termine della sessione. Il client può creare gli attributi di sessione in una richiesta chiamando l'operazione PostContent o PostText con il campo `sessionAttributes` impostato su un valore. Una funzione Lambda può creare un attributo di sessione in una risposta. Dopo che il client o una funzione Lambda hanno creato un attributo di sessione, il valore dell'attributo memorizzato viene utilizzato ogni volta che l'applicazione client non include un`sessionAttribute` campo in una richiesta ad Amazon Lex.

Presupponi, ad esempio, di avere due attributi di sessione, `{"x": "1", "y": "2"}`. Se il client chiama l'`PostText`operazione`PostContent` or senza specificare il`sessionAttributes` campo, Amazon Lex chiama la funzione Lambda con gli attributi di sessione memorizzati (`{"x": 1, "y": 2}`). Se la funzione Lambda non restituisce gli attributi di sessione, Amazon Lex restituisce gli attributi di sessione memorizzati all'applicazione client.

Se l'applicazione client o una funzione Lambda trasmettono gli attributi di sessione, Amazon Lex aggiorna gli attributi della sessione memorizzata. Il trasferimento di un valore esistente, come `{"x": 2}`, determina l'aggiornamento del valore archiviato. Se invii un nuovo set di attributi di sessione, ad esempio `{"z": 3}`, i valori esistenti vengono rimossi e viene mantenuto solo il nuovo valore. Quando viene trasferita una mappa vuota, `{}`, i valori archiviati vengono cancellati.

Per inviare gli attributi di sessione ad Amazon Lex, è necessario creare una string-to-string mappa degli attributi. Di seguito viene spiegato come mappare gli attributi di sessione:

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

Per l'operazione `PostText`, inserisci la mappa nel corpo della richiesta utilizzando il campo `sessionAttributes`, come segue:

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

Per l'operazione `PostContent`, applica la codifica base64 alla mappa e inviala come l'intestazione `x-amz-lex-session-attributes`.

Per inviare dati binari o strutturati in un attributo di sessione, devi dapprima trasformarli in una stringa semplice. Per ulteriori informazioni, consulta Impostazione di attributi complessi.

Impostazione degli attributi di richiesta

Gli attributi di richiesta contengono informazioni specifiche sulla richiesta e si applicano solo alla richiesta corrente. Un'applicazione client invia queste informazioni ad Amazon Lex. Utilizza gli attributi di richiesta per inviare informazioni che non devono essere conservate per l'intera sessione. Puoi creare i tuoi attributi di richiesta personali oppure utilizzare quelli predefiniti. Per inviare gli attributi di richiesta, utilizza l'intestazione `x-amz-lex-request-attributes` in un the section called "PostContent" o il campo `requestAttributes` in una richiesta the section called "PostText". Poiché, a differenza degli attributi di sessione, gli attributi di richiesta non vengono mantenuti tra richieste diverse, non vengono restituiti nelle risposte `PostContent` o `PostText`.

> **Note**
>
> Per inviare informazioni che vengano mantenute tra richieste diverse, utilizza gli attributi di sessione.

Il namespace `x-amz-lex:` è riservato agli attributi di richiesta predefiniti. Non creare attributi di richiesta con il prefisso `x-amz-lex:`.

Impostazione di attributi di richiesta predefiniti

Amazon Lex fornisce attributi di richiesta predefiniti per gestire il modo in cui elabora le informazioni inviate al bot. Gli attributi non vengono salvati per l'intera sessione, quindi dovrai inviare gli attributi predefiniti in ogni richiesta. Tutti gli attributi predefiniti sono nel namespace `x-amz-lex:`.

Oltre ai seguenti attributi predefiniti, Amazon Lex fornisce attributi predefiniti per le piattaforme di messaggistica. Per un elenco di questi attributi, consulta Distribuzione di un bot Amazon Lex su una piattaforma di messaggistica.

Impostazione del tipo di risposta

Se utilizzi due applicazioni client che hanno funzionalità diverse, potresti dover limitare il formato dei messaggi in una risposta. Ad esempio, potresti voler limitare i messaggi inviati a un client Web al testo normale, ma permettere a un client mobile di utilizzare sia testo normale sia il formato SSML (Speech Synthesis Markup Language). Per impostare il formato dei messaggi restituiti dalle

operazioni PostContent e PostText, utilizza l'attributo di richiesta `x-amz-lex:accept-content-types"`.

Puoi impostare l'attributo su una qualsiasi combinazione dei seguenti tipi di messaggi:

- `PlainText`—Il messaggio contiene testo UTF-8 semplice.
- `SSML`—Il messaggio contiene testo formattato per l'output vocale.
- `CustomPayload`—Il messaggio contiene un formato personalizzato che hai creato per il tuo cliente. Puoi definire il payload in base alle esigenze della tua applicazione.

Amazon Lex restituisce solo i messaggi con il tipo specificato`Message` nel campo della risposta. Puoi impostare più di un valore separandoli con una virgola. Se utilizzi dei gruppi di messaggi, ogni gruppo di messaggi deve contenere almeno un messaggio del tipo specificato. In caso contrario, viene visualizzato un errore `NoUsableMessageException`. Per ulteriori informazioni, consulta Gruppi di messaggi.

> **Note**
>
> L'attributo di richiesta `x-amz-lex:accept-content-types` non ha alcun effetto sul contenuto del corpo HTML. Il contenuto di una risposta dell'operazione `PostText` è sempre testo normale UTF-8. Il corpo di una risposta dell'operazione `PostContent` contiene dati nel formato impostato nell'intestazione `Accept` nella richiesta.

Impostazione del fuso orario preferito

Per impostare il fuso orario dell'utente da utilizzare per risolvere le date, utilizza l'attributo di richiesta `x-amz-lex:time-zone`. Se non specifichi un fuso orario nell'attributo `x-amz-lex:time-zone`, il valore predefinito dipende della regione che utilizzi per il tuo bot.

Regione	Fuso orario predefinito
Stati Uniti orientali (Virginia settentrionale)	`America/New_York`
Stati Uniti occidentali (Oregon)	`America/Los_Angeles`
Asia Pacifico (Singapore)	`Asia/Singapore`

Regione	Fuso orario predefinito
Asia Pacifico (Sydney)	`Australia/Sydney`
Asia Pacifico (Tokyo)	`Asia/Tokyo`
Europa (Francoforte)	`Europe/Berlin`
Europa (Irlanda)	`Europe/Dublin`
Europa (Londra)	`Europe/London`

Ad esempio, se l'utente risponde alla richiesta «`tomorrowIn` che giorno desideri che il pacco venga consegnato?» la data effettiva di consegna del pacco dipende dal fuso orario dell'utente. Ad esempio, quando sono le 02:00 del 16 settembre a New York, sono le 23:00 del 15 settembre a Los Angeles. Se il servizio è attivo nella regione degli Stati Uniti orientali (Virginia settentrionale) e una persona a Los Angeles ordina un pacco da consegnare «domani» utilizzando il fuso orario predefinito, il pacco verrà consegnato il 17, non il 16. Se invece imposti l'attributo di richiesta `x-amz-lex:time-zone` su `America/Los_Angeles`, il pacco verrà consegnato il giorno 16.

Puoi impostare l'attributo su uno dei nomi dei fusi orari della IANA (Internet Assigned Number Authority). Per l'elenco dei nomi dei fusi orari, consulta l'elenco dei fusi orari del database tz su Wikipedia.

Impostazione di attributi di richiesta definiti dall'utente

Un attributo di richiesta definito dall'utente è un'informazione che invii al tuo bot con ciascuna richiesta. Invii l'informazione nell'intestazione `amz-lex-request-attributes` di una richiesta `PostContent` o nel campo `requestAttributes` di una richiesta `PostText`.

Per inviare gli attributi della richiesta ad Amazon Lex, è necessario creare una string-to-string mappa degli attributi. Di seguito viene spiegato come mappare gli attributi di richiesta:

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

Per l'operazione `PostText`, inserisci la mappa nel corpo della richiesta utilizzando il campo `requestAttributes`, come segue:

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

Per l'operazione `PostContent`, applica la codifica base64 alla mappa e inviala come l'intestazione `x-amz-lex-request-attributes`.

Per inviare dati binari o strutturati in un attributo di richiesta, devi dapprima trasformarli in una stringa semplice. Per ulteriori informazioni, consulta Impostazione di attributi complessi.

Impostazione del timeout di sessione

Amazon Lex conserva le informazioni di contesto, ovvero i dati degli slot e gli attributi delle sessioni, fino al termine di una sessione di conversazione. Per definire la durata di una sessione per un bot, imposta il timeout di sessione. Per impostazione predefinita, la durata di una sessione è di 5 minuti, ma puoi specificarne una diversa compresa tra 0 e 1.440 minuti (24 ore).

Supponiamo, ad esempio, che tu stia creando un bot `ShoeOrdering` che supporta gli intenti `OrderShoes` e `GetOrderStatus`. Quando Amazon Lex rileva che l'intenzione dell'utente è quella di ordinare scarpe, richiede i dati relativi agli slot. come, ad esempio, numero di scarpa, colore, marchio e così via. Se l'utente fornisce alcuni dati dello slot ma non completa l'acquisto della scarpa, Amazon Lex memorizza tutti i dati dello slot e gli attributi della sessione per l'intera sessione. Se l'utente riprende la sessione prima che scada, può fornire i dati dello slot mancanti e completare l'acquisto.

Nella console Amazon Lex, imposti il timeout della sessione quando crei un bot. Con l'interfaccia a riga di comando AWS (CLI AWS) o l'API, imposti il timeout quando crei o aggiorni un bot con l'PutBotoperazione impostando ilInSeconds campo IdleSessionTTL.

Condivisione di informazioni tra intenti diversi

Amazon Lex supporta la condivisione di informazioni tra intenti. mediante l'utilizzo degli attributi di sessione.

Ad esempio, un utente del bot `ShoeOrdering` inizia ordinando delle scarpe. Il bot avvia una conversazione con l'utente, acquisendo i dati dello slot come numero di scarpa, colore e marchio.

Quando l'utente effettua un ordine, la funzione Lambda che esegue l'ordine imposta l'attributo di sessione, che contiene il numero dell'ordine. Per ricevere lo stato dell'ordine, l'utente utilizza l'intento `GetOrderStatus`. Il bot può chiedere all'utente i dati dello slot, come il numero e la data dell'ordine. Quando il bot ottiene le informazioni richieste, restituisce lo stato dell'ordine.

Se pensi che i tuoi utenti possano cambiare intento nel corso della stessa sessione, puoi progettare il tuo bot affinché restituisca lo stato relativo all'ordine più recente. Invece di chiedere le informazioni dell'ordine all'utente, utilizza l'attributo di sessione `orderNumber` per condividere le informazioni tra intenti diversi e realizzare l'intento `GetOrderStatus`. Il bot esegue questa operazione restituendo lo stato dell'ordine più recente effettuato dall'utente.

Per un esempio di condivisione di informazioni tra intenti, consulta Prenota viaggio.

Impostazione di attributi complessi

Gli attributi di sessione e richiesta sono string-to-string mappe di attributi e valori. In molti casi puoi utilizzare la mappa stringa per trasferire i valori degli attributi tra l'applicazione client e un bot, mentre in alcuni casi potresti dover trasferire i dati binari o una struttura complessa che non può essere convertita facilmente in una mappa stringa. Ad esempio, il seguente oggetto JSON rappresenta una matrice delle città più popolose degli Stati Uniti:

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

Questa matrice di dati non si traduce bene in una string-to-string mappa. In questo caso, puoi trasformare un oggetto in una stringa semplice per poterla inviare al tuo bot con le operazioni PostContent e PostText.

Ad esempio, se si utilizza JavaScript, è possibile utilizzare l'`JSON.stringify`operazione per convertire un oggetto in JSON e l'`JSON.parse`operazione per convertire il testo JSON in un JavaScript oggetto:

```
// 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);
```

Per inviare gli attributi di sessione con l'`PostContent`operazione, è necessario codificare gli attributi in base64 prima di aggiungerli all'intestazione della richiesta, come mostrato nel JavaScript codice seguente:

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

Puoi inviare i dati binari alle operazioni `PostContent` e `PostText` convertendo dapprima i dati in una stringa con codifica base64, quindi inviando la stringa come valore negli attributi di sessione:

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

Usare i punteggi di

Quando un utente fa un enunciato, Amazon Lex utilizza la comprensione del linguaggio naturale (NLU) per comprendere la richiesta dell'utente e restituire l'intento corretto. Per impostazione predefinita, Amazon Lex restituisce l'intento più probabile definito dal bot.

In alcuni casi, può essere difficile per Amazon Lex determinare l'intento più probabile. Ad esempio, l'utente potrebbe fare un enunciato ambiguo o potrebbero esserci due intenti simili. Per aiutare a determinare l'intento corretto, puoi combinare le tue conoscenze di dominio conPunteggio di attendidi un elenco di intenti alternativi. Un punteggio di fiducia è una valutazione fornita da Amazon Lex che mostra quanto sia sicuro che un intento sia l'intento corretto.

Per determinare la differenza tra due intenti alternativi, puoi confrontare i loro punteggi di confidenza. Ad esempio, se un intento ha un punteggio di confidenza di 0,95 e un altro ha un punteggio di 0,65, il primo intento è probabilmente corretto. Tuttavia, se un intento ha un punteggio di 0,75 e un altro ha un punteggio di 0,72, vi è un'ambiguità tra i due intenti che potresti essere in grado di discriminare utilizzando la conoscenza del dominio nella tua applicazione.

È inoltre possibile utilizzare i punteggi di affidabilità per creare applicazioni di test che determinano se le modifiche alle enunciate di un intento fanno differenza nel comportamento del bot. Ad esempio, è possibile ottenere i punteggi di confidenza per gli intenti di un bot utilizzando una serie di enunciati, quindi aggiornare gli intenti con nuovi enunciati. È quindi possibile controllare i punteggi di confidenza per vedere se c'è stato un miglioramento.

I punteggi di fiducia restituiti da Amazon Lex sono valori comparativi. Non dovresti fare affidamento su di loro come punteggio assoluto. I valori potrebbero cambiare in base ai miglioramenti apportati ad Amazon Lex.

Quando utilizzi i punteggi di fiducia, Amazon Lex restituisce l'intento più probabile e fino a 4 intenti alternativi con i punteggi associati in ogni risposta. Se tutti i punteggi di fiducia sono inferiori a una soglia, Amazon Lex include`AMAZON.FallbackIntent`, il`AMAZON.KendraSearchIntento` entrambi, se li hai configurati. Puoi utilizzare la soglia predefinita o impostare la soglia.

Il codice JSON seguente mostra il`alternativeIntents`campo nella risposta dalPostTextoperazione.

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

```
    ],
```

Imposta la soglia quando crei o aggiorni un bot. Puoi utilizzare l'API o la console di Amazon Lex. Per le regioni elencate di seguito è necessario optare per abilitare miglioramenti della precisione e punteggi di affidabilità. Nella console, scegli i punteggi di fiduciaOpzioni avanzatesezione. Tramite l'API, imposta il`enableModelImprovements`parametro quando si chiama ilPutBotoperazione. :

- Stati Uniti orientali (Virginia settentrionale): us-east-1
- Stati Uniti occidentali (Oregon): us-west-2
- Asia Pacifico (Sydney): ap-southeast-2
- Europa (Irlanda) (eu-west-1)

In tutte le altre regioni, i miglioramenti della precisione e il supporto del punteggio di affidabilità sono disponibili per impostazione predefinita.

Per modificare la soglia di affidabilità, impostarla nella console o utilizzando ilPutBotoperazione. La soglia deve essere un numero compreso tra 1,00 e 0,00.

Per utilizzare la console, imposta la soglia di affidabilità quando crei o aggiorni il bot.

Per impostare la soglia di affidabilità durante la creazione di un bot (console)

- SuCreare il bot, immettere un valore nelPunteggio di attendibilità.

Per aggiornare la soglia di affidabilità (console)

1. Dall'elenco dei bot, scegli il bot da aggiornare.
2. Selezionare la scheda Settings (Impostazioni).
3. Nel riquadro di navigazione a sinistra, scegliGenerale.
4. Aggiorna il valore nellaPunteggio di attendibilità.

Per impostare o aggiornare la soglia di affidabilità (SDK)

- Impostazione della proprietà`nluIntentConfidenceThreshold`parametro delPutBotoperazione. Il seguente codice JSON mostra il parametro impostato.

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

Gestione delle sessioni

Per modificare l'intento utilizzato da Amazon Lex in una conversazione con l'utente, è possibile utilizzare la risposta dalla funzione Lambda dell'hook del codice di dialogo oppure utilizzare le API di gestione delle sessioni nell'applicazione personalizzata.

Utilizzo di una funzione Lambda

Quando si utilizza una funzione Lambda, Amazon Lex la chiama con una struttura JSON che contiene l'input della funzione. La struttura JSON contiene un campo chiamato`currentIntent`che contiene l'intento che Amazon Lex ha identificato come l'intento più probabile per l'enunciato dell'utente. La struttura JSON include anche un`alternativeIntents`campo che contiene fino a quattro intenti aggiuntivi che possono soddisfare l'intento dell'utente. Ogni intento include un campo chiamato`nluIntentConfidenceScore`che contiene il punteggio di fiducia assegnato da Amazon Lex all'intento.

Per utilizzare un intento alternativo, è necessario specificarlo nella`ConfirmIntento` il`ElicitSlot`nell'azione di dialogo nella funzione Lambda.

Per ulteriori informazioni, consultare Utilizzo delle funzioni Lambda.

Utilizzo dell'API di gestione delle sessioni

Per utilizzare un intento diverso dall'intento attuale, utilizzare ilPutSessionoperazione. Ad esempio, se decidi che la prima alternativa è preferibile all'intento scelto da Amazon Lex, è possibile utilizzare il`PutSession`operazione per modificare gli intenti in modo che l'intento successivo con cui l'utente interagisce sia quello selezionato.

Per ulteriori informazioni, consultare Gestione di sessioni con l'API di Amazon Lex.

Log delle conversazioni

È possibile abilitare i log delle conversazioni per memorizzare le interazioni bot. È possibile utilizzare questi log per esaminare le prestazioni del bot e risolvere problemi relativi alle conversazioni. È possibile registrare il testo per l'operazione PostText. È possibile registrare testo e audio per l'operazione PostContent. Abilitando i log delle conversazioni si ottiene una visualizzazione dettagliata delle conversazioni tra gli utenti e il bot.

Ad esempio, una sessione con il bot ha un ID di sessione. È possibile utilizzare questo ID per ottenere la trascrizione della conversazione, incluse le enunciazioni dell'utente e le risposte

corrispondenti del bot. Puoi anche ottenere metadati quali il nome dell'intento e i valori di slot per un'enunciazione.

> **Note**
>
> Non è possibile utilizzare i log delle conversazioni con un bot soggetto al Children's Online Privacy Protection Act (COPPA).

I log delle conversazioni sono configurati per un alias. Ogni alias può avere impostazioni diverse per i log audio e di testo. È possibile abilitare log di testo, log audio o entrambi per ciascun alias. I log di testo memorizzano l'input di testo, le trascrizioni dell'input audio e CloudWatch i metadati associati. I registri audio memorizzano l'ingresso audio in Amazon S3. È possibile abilitare la crittografia dei log audio e di testo utilizzando CMK gestite dal cliente AWS KMS.

Per configurare la registrazione, utilizzare la console o l'operazione PutBotAlias. Non puoi registrare le conversazioni per l'`$LATEST`alias del tuo bot o per il bot di test disponibile nella console Amazon Lex. Dopo aver abilitato i registri delle conversazioni per un aliasPostContent oPostText l'operazione per quell'alias, registra il testo o le espressioni audio nel gruppo di log CloudWatch Logs configurato o nel bucket S3.

Argomenti

- Criteri IAM per i registri delle conversazioni
- Configurazione dei log delle conversazioni
- Crittografare i log delle conversazioni
- Visualizzazione dei log di testo in Amazon CloudWatch Logs
- Accesso ai registri audio in Amazon S3
- Monitoraggio dello stato del registro delle conversazioni con CloudWatch metriche

Criteri IAM per i registri delle conversazioni

A seconda del tipo di registrazione selezionato, Amazon Lex richiede l'autorizzazione per utilizzare i bucket Amazon CloudWatch Logs e Amazon Simple Storage Service (S3) per archiviare i log. È necessario creareAWS Identity and Access Management ruoli e autorizzazioni per consentire ad Amazon Lex di accedere a queste risorse.

Creazione di un ruolo IAM e delle policy per i log delle conversazioni

Per abilitare i log delle conversazioni, devi concedere l'autorizzazione di scrittura per CloudWatch Logs e Amazon S3. Se si abilita la crittografia degli oggetti per gli oggetti S3, è necessario concedere l'autorizzazione di accesso alle chiavi AWS KMS utilizzate per crittografare gli oggetti.

Puoi utilizzare IAMAWS Management Console, l'API IAM o ilAWS Command Line Interface per creare il ruolo e le politiche. Queste istruzioni utilizzano l'AWS CLI per creare il ruolo e le policy. Per informazioni sulla creazione di policy con la console, consulta la sezione relativa alla creazione di policy nella scheda JSON nella Guida per l'utente di AWS Identity and Access Management.

> **Note**
>
> Il codice seguente è formattato per Linux e MacOS. Per Windows, sostituire il carattere di continuazione della riga di Linux (\) con un accento circonflesso (^).

Per creare un ruolo IAM per i log delle conversazioni

1. Creare un documento nella directory corrente chiamato **`LexConversationLogsAssumeRolePolicyDocument.json`**, aggiungervi il seguente codice e salvarlo. Questo documento di policy aggiunge Amazon Lex come entità affidabile al ruolo. Ciò consente a Lex di assumere il ruolo di distribuire i log alle risorse configurate per i log delle conversazioni.

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

2. InAWS CLI, eseguire il comando seguente per creare il ruolo IAM per i log delle conversazioni.

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

Quindi, crea e collega una policy al ruolo che permetta a Amazon Lex di scrivere CloudWatch nei log.

Per creare una policy IAM per registrare il testo della conversazione CloudWatch nei log

1. Crea un documento nella directory corrente chiamata**LexConversationLogsCloudWatchLogsPolicy.json**, aggiungi la seguente politica IAM e salvalo.

```
{
  "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. NelAWS CLI, crea la policy IAM che concede l'autorizzazione di scrittura al gruppo di log CloudWatch Logs.

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

3. Collegare la policy al ruolo IAM creato per i log delle conversazioni.

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

Se stai registrando l'audio in un bucket S3, crea una policy che consenta ad Amazon Lex di scrivere nel bucket.

Per creare una policy IAM per la registrazione audio in un bucket S3

1. Creare un documento nella directory corrente chiamato **`LexConversationLogsS3Policy.json`**, aggiungervi la seguente policy e salvarlo.

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

2. NelAWS CLI, crea la policy IAM che concede l'autorizzazione di scrittura al tuo bucket S3.

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

3. Collegare la policy al ruolo creato per i log delle conversazioni.

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

Concessione dell'autorizzazione a passare un ruolo IAM

Quando usi la consoleAWS Command Line Interface, il o unAWS SDK per specificare un ruolo IAM da utilizzare per i registri delle conversazioni, l'utente che specifica il ruolo IAM dei registri delle conversazioni deve avere l'autorizzazione per passare il ruolo ad Amazon Lex. Per consentire all'utente di passare il ruolo ad Amazon Lex, è necessario concedere l'`PassRole`autorizzazione all'utente, ruolo o gruppo.

La policy seguente definisce l'autorizzazione da concedere all'utente, al ruolo o al gruppo. È possibile utilizzare le chiavi di condizione `iam:AssociatedResourceArn` e `iam:PassedToService` per limitare l'ambito dell'autorizzazione. Per ulteriori informazioni, consulta Concessione a un utente delle autorizzazioni per passare un ruolo a unAWS servizio e chiavi di contesto IAM eAWS STS Condition nella Guida per l'AWS Identity and Access Managementutente.

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

Configurazione dei log delle conversazioni

Si possono abilitare e disabilitare i log delle conversazioni utilizzando la console o il campo `conversationLogs` dell'operazione `PutBotAlias`. È possibile attivare o disattivare i log audio, i log di testo o entrambi. La registrazione inizia nelle nuove sessioni del bot. Le modifiche apportate alle impostazioni dei log non si riflettono nelle sessioni attive.

Per archiviare i log di testo, utilizza un gruppo di log Amazon CloudWatch Logs nel tuoAWS account. È possibile utilizzare qualsiasi gruppo di log valido. Il gruppo di log deve trovarsi nella stessa regione del bot Amazon Lex. Per ulteriori informazioni sulla creazione di un CloudWatch gruppo di log, consulta Utilizzo di gruppi di log e flussi di CloudWatch log.

Per archiviare i registri audio, utilizza un bucket Amazon S3 nel tuoAWS account. È possibile utilizzare qualsiasi bucket S3 valido. Il bucket deve trovarsi nella stessa regione del bot Amazon Lex.

Per ulteriori informazioni sulla creazione di un bucket S3, consulta la sezione relativa alla creazione di un bucket nella Guida alle operazioni di base di Amazon Simple Storage Service.

Devi fornire un ruolo IAM con politiche che consentano ad Amazon Lex di scrivere nel gruppo di log o nel bucket configurato. Per ulteriori informazioni, consulta Creazione di un ruolo IAM e delle policy per i log delle conversazioni.

Se si crea un ruolo collegato al servizio utilizzando ilAWS Command Line Interface, è necessario aggiungere un suffisso personalizzato al ruolo utilizzando l'`custom-suffix`opzione seguente:

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

Il ruolo IAM utilizzato per abilitare i registri delle conversazioni deve disporre dell'`iam:PassRole`autorizzazione. Al ruolo deve essere collegata la policy seguente.

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

Abilitare i log delle conversazioni

Per attivare i log utilizzando la console

1. Apri la console Amazon Lex https://console.aws.amazon.com/lex.
2. Dall'elenco, scegliere un bot.
3. Scegliere la scheda Settings (Impostazioni) quindi scegliere Conversation logs (Log delle conversazioni) dal menu a sinistra.
4. Nell'elenco degli alias, scegliere l'icona delle impostazioni per l'alias per il quale si desidera configurare i log delle conversazioni.
5. Scegliere se registrare testo, audio o entrambi.

6. Per la registrazione del testo, inserisci il nome del gruppo di log Amazon CloudWatch Logs.
7. Per la registrazione dell'audio, inserire le informazioni del bucket S3.
8. Facoltativo. Per crittografare i log audio, scegliere la chiave AWS KMS da utilizzare per la crittografia.
9. Scegli un ruolo IAM con le autorizzazioni necessarie.
10. Scegliere Save (Salva) per iniziare a registrare le conversazioni.

Per attivare i log di testo utilizzando l'API

1. Chiamare l'operazione PutBotAlias con una voce nel membro `logSettings` del campo `conversationLogs`
   - Impostare il membro `destination` su `CLOUDWATCH_LOGS`
   - Impostare il membro `logType` su `TEXT`
   - Imposta l'`resourceArn`Amazon Resource Name (ARN) del gruppo di log CloudWatch Logs che è la destinazione dei log
2. Impostai`amRoleArn` l'Amazon Resource Name (ARN) di un ruolo IAM che dispone delle autorizzazioni necessarie per abilitare i log delle conversazioni sulle risorse specificate.`conversationLogs`

Per attivare i log audio utilizzando l'API

1. Chiamare l'operazione PutBotAlias con una voce nel membro `logSettings` del campo `conversationLogs`
   - Impostare il membro `destination` su `S3`
   - Impostare il membro `logType` su `AUDIO`
   - Impostare il membro `resourceArn` sull'ARN del bucket Amazon S3 in cui sono archiviati i log audio
   - Facoltativo. Per crittografare i log audio con una chiave AWS KMS specifica, impostare il membro `kmsKeyArn` dell'ARN della chiave utilizzata per la crittografia.
2. Impostai`amRoleArn` l'Amazon Resource Name (ARN) di un ruolo IAM che dispone delle autorizzazioni necessarie per abilitare i log delle conversazioni sulle risorse specificate.`conversationLogs`

Disabilitare i log delle conversazioni

Per disattivare i log utilizzando la console

1. Apri la console Amazon Lex https://console.aws.amazon.com/lex.
2. Dall'elenco, scegliere un bot.
3. Scegliere la scheda Settings (Impostazioni) quindi scegliere Conversation logs (Log delle conversazioni) dal menu a sinistra.
4. Nell'elenco degli alias, scegliere l'icona delle impostazioni per l'alias per il quale si desidera configurare i log delle conversazioni.
5. Deselezionare la casella di controllo da testo, audio o entrambi per disattivare la registrazione.
6. Scegliere Save (Salva) per interrompere la registrazione delle conversazioni.

Per disattivare i log utilizzando l'API

- Chiamare l'operazione `PutBotAlias` senza il campo `conversationLogs`.

Per disattivare i log di testo utilizzando l'API

- • In caso di registrazione dell'audio
    - Chiamare l'operazione PutBotAlias con una voce `logSettings` solo per `AUDIO`.
    - La chiamata all'operazione `PutBotAlias` non deve avere una voce `logSettings` per `TEXT`.
  - Se non si sta registrando l'audio
    - Chiamare l'operazione PutBotAlias senza il campo `conversationLogs`.

Per disattivare i log audio utilizzando l'API

- • In caso di registrazione del testo
    - Chiamare l'operazione PutBotAlias con una voce `logSettings` solo per `TEXT`.
    - La chiamata all'operazione `PutBotAlias` non deve avere una voce `logSettings` per `AUDIO`.
  - Se non si sta registrando il testo
    - Chiamare l'operazione PutBotAlias senza il campo `conversationLogs`.

Crittografare i log delle conversazioni

È possibile utilizzare la crittografia per proteggere il contenuto dei log delle conversazioni. Per i registri di testo e audio, puoi utilizzare i CMK gestiti dalAWS KMS cliente per crittografare i dati nel gruppo di log CloudWatch Logs e nel bucket S3.

> **Note**
>
> Amazon Lex supporta solo CMK simmetriche. Non utilizzare una CMK asimmetrica per crittografare i dati.

Puoi abilitare la crittografia utilizzando unaAWS KMS chiave nel gruppo di log CloudWatch Logs che Amazon Lex utilizza per i registri di testo. Non è possibile fornire una chiave AWS KMS nelle impostazioni del log per abilitare la crittografia AWS KMS del gruppo di log. Per ulteriori informazioni, consulta Crittografia di dati di CloudWatch log nei log tramiteAWS KMS nella Guida per l'utente di Amazon CloudWatch Logs.

Per i log audio si utilizza la crittografia predefinita sul bucket S3 o si specifica una chiave AWS KMS per crittografare gli oggetti audio. Anche se il bucket S3 utilizza la crittografia predefinita, è comunque possibile specificare una chiave AWS KMS diversa per crittografare gli oggetti audio. Per ulteriori informazioni, consulta Crittografia predefinita di Amazon S3 per i bucket S3.

Amazon Lex richiedeAWS KMS autorizzazioni se scegli di crittografare i registri audio. È necessario allegare ulteriori policy al ruolo IAM utilizzato per i log delle conversazioni. Se si utilizza la crittografia predefinita nel bucket S3, la policy deve concedere l'accesso alla chiave AWS KMS configurata per tale bucket. Se si specifica una chiave AWS KMS nelle impostazioni del log audio, è necessario concedere l'accesso a tale chiave.

Se non hai creato un ruolo per i log delle conversazioni, consulta Criteri IAM per i registri delle conversazioni.

Per creare una policy IAM per l'utilizzo di unaAWS KMS chiave per la crittografia dei registri audio

1. Creare un documento nella directory corrente chiamato **`LexConversationLogsKMSPolicy.json`**, aggiungervi la seguente policy e salvarlo.

```
{
  "Version": "2012-10-17",
```

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

2. NelAWS CLI, crea la policy IAM che concede l'autorizzazione all'uso dellaAWS KMS chiave per crittografare i registri audio.

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

3. Collegare la policy al ruolo creato per i log delle conversazioni.

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

Visualizzazione dei log di testo in Amazon CloudWatch Logs

Amazon Lex archivia i log di testo per le tue conversazioni in Amazon CloudWatch Logs. Per visualizzare i log, puoi utilizzare la console o l'API CloudWatch Logs. Per ulteriori informazioni, consulta Ricerca nei dati dei log utilizzando modelli di filtro e Sintassi delle query diCloudWatch Logs Insights nella Guida per l'utente di Amazon CloudWatch Logs.

Per visualizzare i log tramite la console Amazon Lex

1. Apri la console Amazon Lex https://console.aws.amazon.com/lex.
2. Dall'elenco, scegliere un bot.
3. Scegliere la scheda Settings (Impostazioni), quindi scegliere Conversation logs (Log delle conversazioni) dal menu a sinistra.
4. Scegli il link sotto Registri di testo per visualizzare i registri dell'alias nella CloudWatch console.

Puoi anche utilizzare la CloudWatch console o l'API per visualizzare le tue voci di registro. Per trovare le voci del log, passare al gruppo di log configurato per l'alias. Puoi trovare il prefisso del flusso di log per i tuoi log nella console Amazon Lex o utilizzando l'GetBotAliasoperazione.

Le voci del log di un'enunciazione di un utente si trovano in più flussi di log. Un'enunciazione nella conversazione ha una voce in uno dei flussi di log con il prefisso specificato. Una voce nel flusso di log contiene le seguenti informazioni.

```
{
   "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"
    ...
 }
}
```

Il contenuto della voce di log dipende dal risultato di una transazione e dalla configurazione del bot e della richiesta.

- I campi `intent`, `slots` e `slotToElicit` non vengono visualizzati in una voce se il campo `missedUtterance` è `true`.
- Il campo `s3PathForAudio` non compare se i log audio sono disabilitati o se il campo `inputDialogMode` è `Text`.
- Il campo `responseCard` viene visualizzato solo quando è stata definita una scheda di risposta per il bot.

- La mappa `requestAttributes` viene visualizzata solo se nella richiesta sono stati specificati attributi di richiesta.
- Il`kendraResponse` campo è presente solo quando`AMAZON.KendraSearchIntent` effettua una richiesta di ricerca in un indice Amazon Kendra.
- Il`dialogAction` campo è vero quando è stato specificato un intento alternativo nella funzione Lambda del bot.
- La mappa `sessionAttributes` viene visualizzata solo se nella richiesta sono stati specificati attributi di sessione.
- La mappa `sentimentResponse` viene visualizzata solo se si configura il bot per restituire i valori di sentiment.

> **Note**
>
> Il formato di input potrebbe cambiare senza che corrisponda una modifica in `messageVersion`. Il codice non dovrebbe generare errori se sono presenti nuovi campi.

È necessario disporre di un ruolo e di una politica impostati per consentire ad Amazon Lex di scrivere CloudWatch nei log. Per ulteriori informazioni, consulta Criteri IAM per i registri delle conversazioni.

Accesso ai registri audio in Amazon S3

Amazon Lex archivia i log audio delle conversazioni in un bucket S3.

Per accedere ai log audio tramite la console

1. Apri la console Amazon Lex https://console.aws.amazon.com/lex.
2. Dall'elenco, scegliere un bot.
3. Scegliere la scheda Settings (Impostazioni), quindi scegliere Conversation logs (Log delle conversazioni) dal menu a sinistra.
4. Scegli il link sotto Registri audio per accedere ai registri dell'alias nella console Amazon S3.

Puoi anche utilizzare la console o l'API di Amazon S3 per accedere ai log audio. Puoi vedere il prefisso della chiave oggetto S3 dei file audio nella console Amazon Lex o nel`resourcePrefix` campo nella risposta dell'`GetBotAlias`operazione.

Monitoraggio dello stato del registro delle conversazioni con CloudWatch metriche

Usa Amazon CloudWatch per monitorare le metriche di consegna dei registri delle conversazioni. È possibile impostare allarmi sui parametri in modo da essere consapevoli dei problemi relativi alla registrazione, se dovessero verificarsi.

Amazon Lex fornisce quattro metriche nel`AWS/Lex` namespace per i registri delle conversazioni:

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

Per ulteriori informazioni, consulta CloudWatch Metriche per i registri delle conversazioni.

Le metriche di successo mostrano che Amazon Lex ha scritto correttamente i registri audio o di testo nelle rispettive destinazioni.

Le metriche relative agli errori mostrano che Amazon Lex non è in grado di fornire registri audio o di testo alla destinazione specificata. In genere si tratta di un errore di configurazione. Quando i parametri di errore sono superiori a zero, controllare quanto segue:

- Assicurati che Amazon Lex sia un'entità affidabile per il ruolo IAM.
- Per la registrazione del testo, assicurati che esista il gruppo di log CloudWatch Logs. Per la registrazione dell'audio, verificare l'esistenza del bucket S3.
- Assicurati che il ruolo IAM utilizzato da Amazon Lex per accedere al gruppo di log CloudWatch Logs o al bucket S3 disponga dell'autorizzazione di scrittura per il gruppo di log o il bucket.
- Assicurati che il bucket S3 esista nella stessa regione del bot Amazon Lex e appartenga al tuo account.
- Se utilizzi unaAWS KMS chiave per la crittografia S3, assicurati che non vi siano politiche che impediscano ad Amazon Lex di utilizzare la tua chiave e assicurati che il ruolo IAM fornito disponga delleAWS KMS autorizzazioni necessarie. Per ulteriori informazioni, consulta Criteri IAM per i registri delle conversazioni.

Gestione di sessioni con l'API di Amazon Lex

Quando un utente avvia una conversazione con il tuo bot, Amazon Lex crea unsessione. Le informazioni scambiate tra la tua applicazione e Amazon Lex costituiscono lo stato della sessione per la conversazione. Quando inoltri una richiesta, la sessione viene identificata dalla combinazione del nome del bot e di un identificatore utente specificato da te. Per ulteriori informazioni sull'identificatore utente, consulta il campo `userId` nell'operazione PostContent o PostText.

La risposta ricevuta da un'operazione di sessione include un identificatore sessione univoco che identifica una sessione specifica con un utente. È possibile utilizzare questo identificatore durante le operazioni di test o di risoluzione dei problemi del bot.

È possibile modificare lo stato della sessione scambiato tra l'applicazione e il bot. Ad esempio, è possibile creare e modificare attributi della sessione personalizzati che contengono informazioni sulla sessione ed è possibile modificare il flusso della conversazione impostando il contesto del dialogo per interpretare l'enunciazione successiva.

È possibile aggiornare lo stato della sessione in due modi. Il primo è quello di utilizzare una funzione Lambda con`PostContent`o`PostText`operazione che viene richiamata ad ogni scambio di conversazione. Per ulteriori informazioni, consultare Utilizzo delle funzioni Lambda. L'altro è utilizzando l'API di runtime di Amazon Lex nell'applicazione per modificare lo stato della sessione.

L'API di runtime di Amazon Lex fornisce operazioni che consentono di gestire le informazioni relative alla sessione per una conversazione con il bot. Le operazioni sono le seguenti: PutSession , GetSession e DeleteSession. È possibile utilizzare queste operazioni per ottenere informazioni sullo stato della sessione dell'utente con il bot e avere il controllo granulare sullo stato.

Utilizza l'operazione `GetSession` quando desideri ottenere lo stato attuale della sessione. L'operazione restituisce lo stato attuale della sessione, incluso lo stato del dialogo con l'utente, gli attributi della sessione impostati e i valori di slot per gli ultimi tre intenti con cui l'utente ha interagito.

L'operazione `PutSession` consente di manipolare direttamente lo stato attuale della sessione. È possibile impostare il tipo della prossima operazione di dialogo che il bot eseguirà, in modo da avere il controllo sul flusso della conversazione con il bot. Imposta l'azione di dialogo`type`Campo a`Delegate`per ottenere che Amazon Lex stabilisca l'operazione successiva del bot.

Puoi utilizzare l'operazione `PutSession` per creare una nuova sessione con un bot e impostare l'intento iniziale del bot. Puoi inoltre utilizzare l'operazione `PutSession` per passare da un intento a un altro. Quando crei una sessione o modifichi l'intento, puoi anche impostare lo stato della sessione,

inclusi valori di slot e attributi della sessione. Quando il nuovo intento è terminato, puoi riavviare l'intento precedente. Puoi utilizzare il pluginGetSessionper ottenere lo stato del dialogo dell'intento precedente di Amazon Lex e utilizzare le informazioni per impostare lo stato del dialogo dell'intento.

La risposta ricevuta dall'operazione PutSession contiene le stesse informazioni dell'operazione PostContent. Puoi utilizzare queste informazioni per richiedere all'utente le informazioni successive, esattamente come faresti con la risposta ricevuta dall'operazione PostContent.

Utilizza l'operazione DeleteSession per rimuovere una sessione esistente e ricominciare con una nuova sessione. Ad esempio, quando esegui il test del bot, puoi utilizzare l'operazione DeleteSession per eliminare le sessioni di test dal bot.

Le operazioni di sessione operano in collaborazione con le funzioni di adempimento di Lambda. Ad esempio, se restituisce la tua funzione LambdaFailedcome stato di adempimento è possibile utilizzare ilPutSessionoperazione per impostare il tipo di operazione di dialogo sucloseefulfillmentStateaReadyForFulfillmentper riprovare la fase di adempimento.

Di seguito sono elencate alcune azioni che è possibile effettuare con le operazioni di sessione:

- Impostare il bot in modo che avvii una conversazione invece di attendere l'utente.
- Cambiare intento durante una conversazione.
- Tornare a un intento precedente.
- Avviare o riavviare una conversazione durante un'interazione.
- Convalidare i valori di slot e impostare il bot in modo che richieda nuovamente i valori non validi.

Ognuna di queste azioni è descritta di seguito.

Cambiare intento

Puoi utilizzare l'operazione PutSession per passare da un intento a un altro e per tornare a un intento precedente. Puoi utilizzare l'operazione PutSession per impostare gli attributi di sessione o i valori di slot per il nuovo intento.

- Richiama l'operazione PutSession. Utilizza come nome dell'intento quello del nuovo intento e imposta l'operazione di dialogo su Delegate. Puoi inoltre impostare i valori di slot o gli attributi di sessione necessari per il nuovo intento.
- Amazon Lex avvierà una conversazione con l'utente utilizzando il nuovo intento.

Riprendere un intento precedente

Per riprendere un intento precedente puoi utilizzare l'operazione `GetSession` per ottenere il riepilogo dell'intento e l'operazione `PutSession` per impostare l'intento sullo stato di dialogo precedente.

- Richiama l'operazione `GetSession`. La risposta ricevuta dall'operazione include un riepilogo dello stato del dialogo degli ultimi tre intenti con cui l'utente ha interagito.
- Utilizza le informazioni del riepilogo dell'intento per richiamare l'operazione `PutSession`. In questo modo, l'utente tornerà all'intento precedente nella stessa posizione della conversazione.

In alcuni casi, può essere necessario riprendere la conversazione dell'utente con il bot. Ad esempio, immagina di aver creato un bot per il servizio clienti e che la tua applicazione stabilisca che l'utente deve parlare con un addetto per ricevere assistenza. Dopo la comunicazione con l'utente, l'addetto può aggiungere le informazioni raccolte alla conversazione con il bot.

Per riprendere una sessione, segui una procedura simile a questa:

- La tua applicazione stabilisce che l'utente deve parlare con un addetto del servizio clienti.
- Utilizza l'operazione `GetSession` per ottenere lo stato di dialogo attuale dell'intento.
- L'addetto del servizio clienti parla con l'utente e risolve il problema.
- Utilizza l'operazione `PutSession` per impostare lo stato di dialogo dell'intento. Potresti dover impostare valori di slot e attributi o modificare l'intento.
- Il bot riprende la conversazione con l'utente.

Puoi utilizzare il parametro `checkpointLabel` dell'operazione `PutSession` per etichettare un intento in modo da poterlo trovare in un secondo momento. Ad esempio, un bot che chiede informazioni a un cliente potrebbe entrare in un intento `Waiting` mentre il cliente raccoglie le informazioni. Il bot crea un'etichetta di checkpoint per l'intento corrente e quindi avvia l'intento `Waiting`. Quando il cliente restituisce il bot può trovare l'intento precedente utilizzando l'etichetta di checkpoint e tornare indietro.

L'intento deve essere presente nella struttura `recentIntentSummaryView` restituita dall'operazione `GetSession`. Se si specifica un'etichetta di checkpoint nella richiesta di operazione `GetSession`, verrà restituito un massimo di tre intenti con quell'etichetta di checkpoint.

- Utilizzare l'operazione `GetSession` per ottenere lo stato corrente della sessione.

- Utilizzare l'operazione `PutSession` per aggiungere un'etichetta di checkpoint all'ultimo intento. Se necessario puoi utilizzare questa chiamata `PutSession` per passare a un intento diverso.
- Quando è il momento di tornare all'intento etichettato, chiama l'operazione `GetSession` per ottenere un elenco di intenti recenti. Puoi utilizzare il plugin`checkpointLabelFilter`in modo che Amazon Lex restituisca solo gli intenti con l'etichetta di checkpoint specificata.

Avviare una nuova sessione

Se desideri che il bot avvii la conversazione con l'utente, puoi utilizzare l'operazione `PutSession`.

- Crea un intento di benvenuto senza slot e un messaggio conclusivo che richieda all'utente di dichiarare un intento. Ad esempio, "Cosa desideri ordinare? Puoi dire "Ordina una bevanda" oppure "Ordina una pizza"".
- Richiama l'operazione `PutSession`. Utilizza come nome dell'intento quello dell'intento di benvenuto e imposta l'operazione di dialogo su `Delegate`.
- Amazon Lex risponderà con la richiesta fornita nell'intento di benvenuto per avviare la conversazione con l'utente.

Convalidare valori di slot

Puoi convalidare le risposte inviate al bot utilizzando l'applicazione client. Se la risposta non è valida, puoi utilizzare l'operazione `PutSession` per ricevere una nuova risposta dall'utente. Ad esempio, immagina che il bot per ordinare fiori possa vendere soltanto tulipani, rose e gigli. Se l'utente ordina garofani, la tua applicazione può:

- Esaminare il valore di slot restituito dalla risposta `PostText` o `PostContent`.
- Se il valore di slot non è valido, può richiamare l'operazione `PutSession`. L'applicazione deve eliminare il valore di slot, impostare il campo `slotToElicit` e impostare il valore di `dialogAction.type` su `elicitSlot`. Facoltativamente, è possibile impostare il`message`e`messageFormat`campi se desideri modificare il messaggio che utilizza Amazon Lex per ottenere il valore di slot.

Opzioni di distribuzione di bot

Attualmente, Amazon Lex offre le seguenti opzioni di implementazione dei bot:

- SDK AWS Mobile: puoi creare applicazioni mobili che comunicano con Amazon Lex utilizzando gli SDK AWS Mobile.
- Facebook Messenger: puoi integrare la tua pagina di Facebook Messenger con il tuo bot Amazon Lex in modo che gli utenti finali di Facebook possano comunicare con il bot. Nell'implementazione corrente, questa integrazione supporta solo messaggi di input di testo.
- Slack: puoi integrare il tuo bot Amazon Lex con un'applicazione di messaggistica Slack.
- Twilio: puoi integrare il tuo bot Amazon Lex con il Twilio Simple Messaging Service (SMS).

Per alcuni esempi, consulta Distribuzione di bot Amazon Lex.

Intenti incorporati e tipi di slot

Per semplificare la creazione di bot, Amazon Lex consente di utilizzare intenti e tipi di slot incorporati standard.

Argomenti

Intenti incorporati

Per azioni comuni, puoi utilizzare la libreria di intenti standard integrata. Per creare un intento a partire da un intento incorporato, scegli un intento incorporato nella console e assegnagli un nuovo nome. Il nuovo intento ha la configurazione dell'intento di base, ad esempio gli enunciati di esempio.

Nell'implementazione corrente, non puoi effettuare le seguenti operazioni:

- Aggiunge o rimuovi gli enunciati di esempio dall'intento di base
- Configurazione di slot per intenti incorporati

Per aggiungere un intento incorporato a un bot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot a cui aggiungere l'intento integrato.
3. Nel riquadro di navigazione scegliere il segno più (+) accanto a Intenti.

4. Per Aggiungi intento, scegliere Cerca intenti esistenti.
5. Nella casella Intenti di ricerca, digita il nome dell'intento integrato da aggiungere al bot.
6. Per Copia l'intento integrato, assegna un nome all'intento, quindi scegli Aggiungi.
7. Configura l'intento come richiesto per il tuo bot.

Argomenti

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

> **Note**
>
> Per le impostazioni locali in inglese (USA) (en-US), Amazon Lex supporta gli intenti degli intenti integrati standard di Alexa. Per un elenco di intenti incorporati, consulta la sezione relativa agli intenti incorporati standard in Alexa Skills Kit.
> Amazon Lex non supporta i seguenti intenti:
>
> - `AMAZON.YesIntent`
> - `AMAZON.NoIntent`
> - Gli intenti nella libreria di intenti incorporati in Alexa Skills Kit

AMAZON.CancelIntent

Risponde a parole e frasi che indicano che l'utente desidera annullare l'interazione corrente. L'applicazione può utilizzare questo intento per rimuovere i valori dei tipi di slot e altri attributi prima di terminare l'interazione con l'utente.

Enunciati comuni:

- annullare
- non importa
- dimenticalo

AMAZON.FallbackIntent

Quando l'input di un utente a un intento non è quello che un bot si aspetta, puoi configurare Amazon Lex per richiamare un intento di fallback. Ad esempio, se l'input dell'utente «Vorrei ordinare caramelle» non corrisponde a un intento nel tuo `OrderFlowers` bot, Amazon Lex richiama l'intento di fallback per gestire la risposta.

Puoi aggiungere un intento di fallback aggiungendo il tipo di intento `AMAZON.FallbackIntent` integrato al bot. Puoi specificare l'intento utilizzando l'operazione PutBot o scegliendo l'intento dall'elenco di intenti integrati nella console.

L'invocazione di un intento di fallback utilizza due fasi. Nella prima fase l'intento di fallback viene abbinato in base all'input dell'utente. Quando l'intento di fallback viene abbinato, il modo in cui il bot si comporta dipende dal numero di nuovi tentativi configurati per un prompt. Ad esempio, se il numero massimo di tentativi di determinare un intento è 2, il bot restituisce il prompt di chiarimento del bot due volte prima di richiamare l'intento di fallback.

Amazon Lex corrisponde all'intento di fallback in queste situazioni:

- L'input dell'utente a un intento non corrisponde all'input previsto dal bot
- L'input audio è un rumore o l'input di testo non viene riconosciuto come parole.
- L'input dell'utente è ambiguo e Amazon Lex non è in grado di determinare quale intento richiamare.

L'intento di fallback viene richiamato quando:

- Il bot non riconosce l'input utente come intento dopo il numero configurato di tentativi di chiarimento quando la conversazione viene avviata.
- Un intento non riconosce l'input utente come valore di slot dopo il numero di tentativi configurato.
- Un intento non riconosce l'input utente come risposta a un prompt di conferma dopo il numero di tentativi configurato.

Puoi utilizzare quanto segue con un intento di fallback:

- Una funzione Lambda di adempimento
- Istruzione di conclusione
- Un prompt di follow-up

Non puoi aggiungere quanto segue a un intento di fallback:

- Enunciazioni
- Slot
- Una funzione Lambda di inizializzazione e convalida
- Prompt di conferma

Se hai configurato sia un'istruzione di annullamento che un intento di fallback per un bot, Amazon Lex utilizza l'intento di fallback. Se hai bisogno che il bot disponga di una dichiarazione di annullamento, puoi utilizzare la funzione fulfillment per l'intento di fallback per fornire lo stesso comportamento di una dichiarazione di annullamento. Per ulteriori informazioni, consulta il parametro `abortStatement` dell'operazione PutBot.

Utilizzo dei prompt di chiarimento

Se fornisci al bot una richiesta di chiarimento, il prompt viene utilizzato per richiedere un intento valido all'utente. Il prompt di chiarimento verrà ripetuto per il numero di volte che hai configurato. Dopodiché verrà richiamato l'intento di fallback.

Se non imposti una richiesta di chiarimento quando crei un bot e l'utente non avvia la conversazione con un intento valido, Amazon Lex richiama immediatamente il tuo intento di fallback.

Quando utilizzi un intento di fallback senza una richiesta di chiarimento, Amazon Lex non richiama il fallback nelle seguenti circostanze:

- Quando l'utente risponde a un prompt di follow-up ma non fornisce un intento. Ad esempio, in risposta a una richiesta di follow-up che dice «Vuoi qualcos'altro oggi?» , l'utente dice «Sì». Amazon Lex restituisce un'eccezione 400 Bad Request perché non dispone di un messaggio di chiarimento da inviare all'utente per ottenere un intento.

- Quando si utilizza una funzione AWS Lambda, si restituisce un tipo di finestra di dialogo `ElicitIntent`. Poiché Amazon Lex non richiede chiarimenti per ottenere un'intenzione dall'utente, restituisce un'eccezione 400 Bad Request.
- Quando utilizzi l'operazione `PutSession`, invii un tipo di finestra di dialogo `ElicitIntent`. Poiché Amazon Lex non richiede chiarimenti per ottenere un'intenzione dall'utente, restituisce un'eccezione 400 Bad Request.

Utilizzo di una funzione Lambda con un intento di fallback

Quando viene richiamato un intento di fallback, la risposta dipende dall'impostazione del parametro `fulfillmentActivity` per l'operazione PutIntent. Il bot esegue una delle seguenti operazioni:

- Restituisce le informazioni sull'intento all'applicazione client.
- Richiama la funzione Lambda di adempimento. Chiama la funzione con le variabili di sessione impostate per la sessione.

Per ulteriori informazioni sull'impostazione della risposta quando viene richiamato un intento di fallback, consulta il parametro `fulfillmentActivity` dell'operazione PutIntent.

Se utilizzi la funzione Fulfillment Lambda nel tuo intento di fallback, puoi utilizzare questa funzione per chiamare un altro intento o per eseguire qualche forma di comunicazione con l'utente, ad esempio raccogliere un numero di callback o aprire una sessione con un addetto al servizio clienti.

Puoi eseguire qualsiasi azione in una funzione Lambda con intento di fallback che puoi eseguire nella funzione di adempimento per qualsiasi altro intento. Per ulteriori informazioni sulla creazione di una funzione di adempimento tramite AWS Lambda, consulta Utilizzo delle funzioni Lambda.

Un intento di fallback può essere richiamato più volte nella stessa sessione. Ad esempio, supponiamo che la funzione Lambda utilizzi `ElicitIntent` l'azione di dialogo per richiedere all'utente un intento diverso. Se Amazon Lex non è in grado di dedurre l'intento dell'utente dopo il numero configurato di tentativi, richiama nuovamente l'intento di fallback. Richiama inoltre l'intento di fallback quando l'utente non risponde con un valore di slot valido dopo il numero di tentativi configurato.

È possibile configurare una funzione Lambda per tenere traccia del numero di volte in cui l'intento di fallback viene chiamato utilizzando una variabile di sessione. La funzione Lambda può eseguire un'azione diversa se viene chiamata più volte rispetto alla soglia impostata nella funzione Lambda. Per ulteriori informazioni sulle variabili di sessione, consulta Impostazione degli attributi di sessione.

AMAZON.HelpIntent

Risponde a parole o frasi che indicano che l'utente ha bisogno di aiuto durante l'interazione con il bot. Quando viene invocato questo intento, puoi configurare la funzione o l'applicazione Lambda per fornire informazioni sulle funzionalità del bot, porre domande di follow-up sulle aree di aiuto o affidare l'interazione a un agente umano.

Enunciati comuni:

- aiuto
- aiutami
- mi puoi aiutare

AMAZON.KendraSearchIntent

Per cercare documenti che hai indicizzato con Amazon Kendra, usa l'intento. `AMAZON.KendraSearchIntent` Quando Amazon Lex non è in grado di determinare l'azione successiva in una conversazione con l'utente, attiva l'intento di ricerca.

`AMAZON.KendraSearchIntent`È disponibile solo nella lingua inglese (Stati Uniti) (en-US) e nelle regioni Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon) ed Europa (Irlanda).

Amazon Kendra è machine-learning-based un servizio di ricerca che indicizza documenti in linguaggio naturale come documenti PDF o file Microsoft Word. Può ricercare documenti indicizzati e restituire i seguenti tipi di risposte a una domanda:

- Una risposta
- Una voce da una domanda frequente che potrebbe rispondere alla domanda
- Un documento correlato alla domanda

Per un esempio di utilizzo di `AMAZON.KendraSearchIntent`, consulta Esempio: creazione di un FAQ Bot per un indice Amazon Kendra.

Se configuri un `AMAZON.KendraSearchIntent` intento per il tuo bot, Amazon Lex chiama l'intento ogni volta che non riesce a determinare l'espressione dell'utente per uno slot o un intento. Ad esempio, se il bot sta suscitando una risposta per un tipo di slot chiamato «pizza topping» e l'utente

dice «Cos'è una pizza? », Amazon Lex li chiama `AMAZON.KendraSearchIntent` per gestire la domanda. Se non viene ricevuta alcuna risposta da Amazon Kendra, la conversazione continua come configurato nel bot.

Quando utilizzi sia la che `AMAZON.KendraSearchIntent` la `AMAZON.FallbackIntent` nello stesso bot, Amazon Lex utilizza gli intenti come segue:

1. Amazon Lex chiama il`AMAZON.KendraSearchIntent`. L'intento chiama l'operazione Amazon Kendra`Query`.
2. Se Amazon Kendra restituisce una risposta, Amazon Lex mostra il risultato all'utente.
3. Se non viene ricevuta alcuna risposta da Amazon Kendra, Amazon Lex richiede nuovamente una richiesta all'utente. L'operazione successiva dipende dalla risposta dell'utente.
   - Se la risposta dell'utente contiene un'espressione riconosciuta da Amazon Lex, ad esempio il riempimento di un valore di slot o la conferma di un intento, la conversazione con l'utente procede come configurato per il bot.
   - Se la risposta dell'utente non contiene un enunciato riconosciuto da Amazon Lex, Amazon Lex effettua un'altra chiamata all'`Query`operazione.
4. Se non viene fornita alcuna risposta dopo il numero di tentativi configurato, Amazon Lex chiama `AMAZON.FallbackIntent` e termina la conversazione con l'utente.

Esistono tre modi per inviare una richiesta `AMAZON.KendraSearchIntent` ad Amazon Kendra:

- Lascia che sia l'intento di ricerca a fare la richiesta per te. Amazon Lex chiama Amazon Kendra con l'enunciato dell'utente come stringa di ricerca. Quando crei l'intento, puoi definire una stringa di filtro di query che limiti il numero di risposte restituite da Amazon Kendra. Amazon Lex utilizza il filtro nella richiesta di query.
- Aggiungi parametri di query aggiuntivi alla richiesta per restringere i risultati della ricerca utilizzando la funzione Lambda della finestra di dialogo. Aggiungi un `kendraQueryFilterString` campo che contiene i parametri di interrogazione di Amazon Kendra all'`delegate`azione di dialogo. Quando aggiungi parametri di query alla richiesta con la funzione Lambda, hanno la precedenza sul filtro di query definito al momento della creazione dell'intento.
- Crea una nuova query utilizzando la funzione Lambda di dialogo. Puoi creare una richiesta di query Amazon Kendra completa inviata da Amazon Lex. È possibile specificare la query nel campo `kendraQueryRequestPayload` dell'operazione di dialogo `delegate`. Il campo `kendraQueryRequestPayload` ha la precedenza sul campo `kendraQueryFilterString`.

Per specificare il `queryFilterString` parametro quando crei un bot o per specificare il `kendraQueryFilterString` campo quando richiami l'`delegate`azione in una funzione Lambda di dialogo, specifichi una stringa che viene utilizzata come filtro degli attributi per la query Amazon Kendra. Se la stringa non è un filtro di attributo valido, si otterrà un'eccezione `InvalidBotConfigException` in fase di runtime. Per ulteriori informazioni sui filtri degli attributi, consulta Using document attributes to filter query nella Amazon Kendra Developer Guide.

Per avere il controllo sulla query che Amazon Lex invia ad Amazon Kendra, puoi specificare una query nel campo `kendraQueryRequestPayload` della funzione Lambda di dialogo. Se la query non è valida, Amazon Lex restituisce un'`InvalidLambdaResponseException`eccezione. Per ulteriori informazioni, consulta l'operazione di interrogazione nella Amazon Kendra Developer Guide.

Per un esempio di come utilizzare `AMAZON.KendraSearchIntent`, consulta Esempio: creazione di un FAQ Bot per un indice Amazon Kendra.

Politica IAM per Amazon Kendra Search

Per utilizzare l'`AMAZON.KendraSearchIntent`intento, devi utilizzare un ruolo che fornisca politiche AWS Identity and Access Management (IAM) che consentano ad Amazon Lex di assumere un ruolo di runtime autorizzato a richiamare l'intento di Amazon Kendra`Query`. Le impostazioni IAM che usi dipendono dal fatto che tu le crei `AMAZON.KendraSearchIntent` utilizzando la console Amazon Lex, un SDK AWS o il AWS Command Line Interface (AWS CLI). Quando usi la console, puoi scegliere se aggiungere l'autorizzazione per chiamare Amazon Kendra al ruolo collegato al servizio Amazon Lex o utilizzare un ruolo specifico per chiamare l'operazione Amazon Kendra. `Query` Quando si utilizza l'AWS CLI o un SDK per creare l'intento, è necessario utilizzare un ruolo specifico per richiamare l'operazione `Query`.

Collegamento di autorizzazioni

Puoi utilizzare la console per associare le autorizzazioni per accedere all'operazione `Query` Amazon Kendra al ruolo predefinito collegato al servizio Amazon Lex. Quando associ le autorizzazioni al ruolo collegato al servizio, non devi creare e gestire un ruolo di runtime specifico per connetterti all'indice Amazon Kendra.

L'utente, il ruolo o il gruppo che usi per accedere alla console Amazon Lex deve disporre delle autorizzazioni per gestire le politiche dei ruoli. Allega la seguente policy IAM al ruolo di accesso alla console. Quando si concedono queste autorizzazioni, il ruolo dispone delle autorizzazioni necessarie per modificare la policy del ruolo collegato ai servizi esistente.

```
{
```

```
    "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": "*"
        }
    ]
}
```

Specifica di un ruolo

Puoi utilizzare la consoleAWS CLI, l'o l'API per specificare un ruolo di runtime da utilizzare quando chiami l'operazione Amazon `Query` Kendra.

L'utente, il ruolo o il gruppo che usi per specificare il ruolo di runtime deve disporre dell'`iam:PassRole`autorizzazione. La policy seguente definisce l'autorizzazione. È possibile utilizzare le chiavi di contesto di condizione `iam:AssociatedResourceArn` e `iam:PassedToService` per limitare ulteriormente l'ambito delle autorizzazioni. Per ulteriori informazioni, consulta IAM e AWS STS Condition Context Keys nella Guida AWS Identity and Access Management per l'utente.

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

Il ruolo di runtime che Amazon Lex deve utilizzare per chiamare Amazon Kendra deve disporre delle autorizzazioni. `kendra:Query` Quando utilizzi un ruolo IAM esistente per ottenere l'autorizzazione a chiamare l'operazione Amazon `Query` Kendra, al ruolo deve essere associata la seguente policy.

Puoi utilizzare la console IAM, l'API IAM o AWS CLI creare una policy e collegarla a un ruolo. Queste istruzioni utilizzano l'AWS CLI per creare il ruolo e le policy.

> **Note**
>
> Il codice seguente è formattato per Linux e MacOS. Per Windows, sostituire il carattere di continuazione della riga di Linux (\) con un accento circonflesso (^).

Per aggiungere l'autorizzazione per l'operazione Query a un ruolo

1. Creare un documento denominato **`KendraQueryPolicy.json`** nella directory corrente, aggiungervi il seguente codice e salvarlo

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

2. InAWS CLI, esegui il comando seguente per creare la policy IAM per l'esecuzione dell'operazione Amazon `Query` Kendra.

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

3. Allega la policy al ruolo IAM che stai utilizzando per chiamare l'`Query`operazione.

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

Puoi scegliere di aggiornare il ruolo collegato al servizio Amazon Lex o di utilizzare un ruolo che hai creato quando crei il ruolo `AMAZON.KendraSearchIntent` per il tuo bot. La procedura seguente mostra come scegliere il ruolo IAM da utilizzare.

Per specificare il ruolo di runtime per AMAZON. KendraSearchIntent

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegliere il bot a cui si desidera aggiungere `AMAZON.KendraSearchIntent`.
3. Scegliete il segno più (+) accanto a Intenti.
4. In Aggiungi intento, scegliere Cerca intenti esistenti.
5. In Intenti di ricerca, immettere **`AMAZON.KendraSearchIntent`** e quindi scegliere Aggiungi.
6. In Copia intento integrato, immettere un nome per l'intento, ad esempio **`KendraSearchIntent`**, quindi scegliere Aggiungi.
7. Aprire la sezione Query Amazon Kendra.
8. In Ruolo IAM, scegliere una delle seguenti opzioni:
   - Per aggiornare il ruolo collegato al servizio Amazon Lex per consentire al bot di interrogare gli indici Amazon Kendra, scegli Aggiungi autorizzazioni Amazon Kendra.
   - Per utilizzare un ruolo autorizzato a chiamare l'operazione Amazon `Query` Kendra, scegli Usa un ruolo esistente.

Utilizzo degli attributi di richiesta e di sessione come filtri

Per filtrare la risposta di Amazon Kendra agli elementi relativi alla conversazione corrente, utilizza gli attributi di sessione e richiesta come filtri aggiungendo `queryFilterString` il parametro quando crei il bot. Specifichi un segnaposto per l'attributo quando crei l'intento, quindi Amazon Lex V2 sostituisce un valore prima di chiamare Amazon Kendra. Per ulteriori informazioni sugli attributi di richiesta, consulta Impostazione degli attributi di richiesta. Per ulteriori informazioni sugli attributi di sessione, consulta Impostazione degli attributi di sessione.

Di seguito è riportato un esempio di `queryFilterString` parametro che utilizza una stringa per filtrare la query di Amazon Kendra.

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

Di seguito è riportato un esempio di `queryFilterString` parametro che utilizza un attributo di sessione chiamato `"SourceURI"` per filtrare la query di Amazon Kendra.

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

Di seguito è riportato un esempio di `queryFilterString` parametro che utilizza un attributo request chiamato `"DepartmentName"` per filtrare la query di Amazon Kendra.

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

I `AMAZON.KendraSearchInteng` filtri utilizzano lo stesso formato dei filtri di ricerca di Amazon Kendra. Per ulteriori informazioni, consulta Utilizzo degli attributi del documento per filtrare i risultati di ricerca nella guida per sviluppatori di Amazon Kendra.

La stringa del filtro di interrogazione utilizzata con `AMAZON.KendraSearchIntent` deve utilizzare lettere minuscole per la prima lettera di ogni filtro. Ad esempio, quanto segue è un filtro di interrogazione valido per. `AMAZON.KendraSearchIntent`

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

```
        }
    ]
}
```

Utilizzo della risposta di ricerca

Amazon Kendra restituisce la risposta a una ricerca nella dichiarazione di intento. `conclusion` L'intento deve avere una `conclusion` dichiarazione a meno che una funzione Lambda di adempimento non produca un messaggio di conclusione.

Amazon Kendra offre quattro tipi di risposte.

- `x-amz-lex:kendra-search-response-question_answer-question-<N>`— La domanda di una FAQ corrispondente alla ricerca.
- `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— La risposta di una FAQ che corrisponde alla ricerca.
- `x-amz-lex:kendra-search-response-document-<N>`— Un estratto da un documento dell'indice correlato al testo dell'enunciato.
- `x-amz-lex:kendra-search-response-document-link-<N>`— L'URL di un documento nell'indice correlato al testo dell'enunciato.
- `x-amz-lex:kendra-search-response-answer-<N>`— Un estratto da un documento dell'indice che risponde alla domanda.

Le risposte vengono restituite in attributi `request`. Possono esserci fino a cinque risposte per ogni attributo, numerate da 1 a 5. Per ulteriori informazioni sulle risposte, consulta Tipi di risposta nella Amazon Kendra Developer Guide.

L'istruzione `conclusion` deve avere uno o più gruppi di messaggi. Ogni gruppo di messaggi contiene uno o più messaggi. Ogni messaggio può contenere una o più variabili segnaposto che vengono sostituite da attributi di richiesta nella risposta di Amazon Kendra. Nel gruppo di messaggi deve essere presente almeno un messaggio in cui tutte le variabili del messaggio vengono sostituite dai valori degli attributi di richiesta nella risposta runtime oppure nel gruppo deve essere presente un messaggio senza variabili segnaposto. Gli attributi della richiesta sono impostati con doppie parentesi ("((" "))"). I seguenti messaggi del gruppo di messaggi corrispondono a qualsiasi risposta di Amazon Kendra:

- «Ho trovato una domanda FAQ per te: ((x-amz-lex: kendra-search-response-question _answer-question-1)), e la risposta è ((: _answer-answer-1))» x-amz-lex kendra-search-response-question

- «Ho trovato un estratto da un documento utile: ((: -1))» x-amz-lex kendra-search-response-document
- «Penso che la risposta alle tue domande sia ((x-amz-lex: kendra-search-response-answer -1))»

Utilizzo di una funzione Lambda per gestire la richiesta e la risposta

L'`AMAZON.KendraSearchIntent`intent può utilizzare il tuo hook di dialogo e il codice di evasione ordini per gestire la richiesta ad Amazon Kendra e la risposta. Usa la funzione Dialog Code Hook Lambda quando desideri modificare la query che invii ad Amazon Kendra e la funzione Fulfillment code hook Lambda quando desideri modificare la risposta.

Creazione di una query con l'hook del codice di dialogo

Puoi utilizzare l'hook di codice di dialogo per creare una query da inviare ad Amazon Kendra. L'utilizzo dell'hook del codice di dialogo è facoltativo. Se non specifichi un hook di codice di dialogo, Amazon Lex crea una query dall'enunciato dell'utente e utilizza `queryFilterString` quello che hai fornito quando hai configurato l'intento, se ne hai fornito uno.

Puoi utilizzare due campi nella risposta del codice hook di dialogo per modificare la richiesta ad Amazon Kendra:

- `kendraQueryFilterString`— Usa questa stringa per specificare i filtri degli attributi per la richiesta Amazon Kendra. È possibile filtrare la query utilizzando uno qualsiasi dei campi di indice definiti nell'indice. Per la struttura della stringa di filtro, consulta Using document attributes to filter query nella Amazon Kendra Developer Guide. Se la stringa del filtro specificata non è valida, si otterrà un'eccezione `InvalidLambdaResponseException`. La stringa `kendraQueryFilterString` sovrascrive qualsiasi stringa di query specificata in `queryFilterString` configurato per l'intento.
- `kendraQueryRequestPayload`— Usa questa stringa per specificare una query Amazon Kendra. La tua query può utilizzare una qualsiasi delle funzionalità di Amazon Kendra. Se non si specifica una query valida, si ottiene un'eccezione `InvalidLambdaResponseException`. Per ulteriori informazioni, consulta la sezione Query nella Amazon Kendra Developer Guide.

Dopo aver creato il filtro o la stringa di query, invii la risposta ad Amazon Lex con il `dialogAction` campo della risposta impostato su`delegate`. Amazon Lex invia la query ad Amazon Kendra e quindi restituisce la risposta alla query all'hook del codice di adempimento.

Utilizzo dell'hook del codice di adempimento per la risposta

Dopo che Amazon Lex invia una query ad Amazon Kendra, la risposta alla query viene restituita alla funzione Lambda di `AMAZON.KendraSearchIntent` adempimento. L'evento di input del code hook contiene la risposta completa di Amazon Kendra. I dati della query hanno la stessa struttura di quelli restituiti dall'operazione Amazon `Query` Kendra. Per ulteriori informazioni, consulta la sintassi della risposta alle query nella Amazon Kendra Developer Guide.

L'hook del codice di adempimento è facoltativo. Se non ne esiste uno o se il code hook non restituisce un messaggio nella risposta, Amazon Lex utilizza l'`conclusione`istruzione per le risposte.

Esempio: creazione di un FAQ Bot per un indice Amazon Kendra

Questo esempio crea un bot Amazon Lex che utilizza un indice Amazon Kendra per fornire risposte alle domande degli utenti. Il bot di domande frequenti gestisce la finestra di dialogo per l'utente. Esso utilizza l'intento `AMAZON.KendraSearchIntent` per eseguire query nell'indice e presentare la risposta all'utente. Per creare il bot:

1. Creare un bot con cui i clienti interagiranno per ottenere risposte dal bot.
2. Creare un intento personalizzato. Il bot richiede almeno un intento con almeno una enunciazione. Questo intento consente la c reazione del bot, ma non viene usato altrimenti.
3. Aggiungi l'`KendraSearchIntent`intento al tuo bot e configuralo in modo che funzioni con il tuo indice Amazon Kendra.
4. Testa il bot ponendo domande a cui rispondono i documenti archiviati nel tuo indice Amazon Kendra.

Prima di poter utilizzare questo esempio, devi creare un indice Amazon Kendra. Per ulteriori informazioni, consulta Guida introduttiva a un bucket S3 (console) nella Amazon Kendra Developer Guide.

Per creare un bot di domande frequenti

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Nel riquadro di navigazione, scegliere Bot.
3. Seleziona Create (Crea).
4. Seleziona Custom bot (Bot personalizzato). Configurare il bot come segue:

- Nome del bot: assegna al bot un nome che ne indichi lo scopo, ad esempio**`KendraTestBot`**.
- Emetti voce: scegli Nessuno.
- Timeout della sessione: Invio**`5`**.
- Analisi del sentimento: scegli No.
- COPPA — Scegli No.
- Archiviazione degli enunciati utente: scegli Non archiviare.

5. Seleziona Create (Crea).

Per creare correttamente un bot, è necessario creare almeno un intento con almeno un enunciazione di esempio. Questo intento è necessario per creare il tuo bot Amazon Lex, ma non viene utilizzato per la risposta alle domande frequenti. L'enunciazione per l'intento non deve essere applicata a nessuna delle domande poste dal cliente.

Per creare l'intento richiesto

1. Nella pagina Nozioni di base sul bot scegli Crea intento.
2. Per Aggiungi intento, scegli Crea intento.
3. Nella finestra di dialogo Crea intento assegna un nome all'intento, ad esempio **`RequiredIntent`**.
4. Per Enunciazioni di esempio, digitare un'enunciazione, ad esempio **`Required utterance`**.
5. Scegliere Salva intento.

Ora, crea l'intento di cercare un indice Amazon Kendra e i messaggi di risposta che dovrebbe restituire.

Per creare un AMAZON. KendraSearchIntent messaggio di intento e risposta

1. Nel riquadro di navigazione scegliere il segno più (+) accanto a Intenti.
2. Per Aggiungi intento, scegliere Cerca intenti esistenti.
3. Nella casella Intenti di ricerca, inserisci**`AMAZON.KendraSearchIntent`**, quindi selezionalo dall'elenco.
4. Per Copia intento integrato, assegnare un nome all'intento, ad esempio **`KendraSearchIntent`**, quindi scegliere Aggiungi.
5. Nell'editor degli intenti, scegli Query Amazon Kendra per aprire le opzioni delle query.

6. Dal menu Indice Amazon Kendra scegli l'indice che desideri venga ricercato dall'intento.
7. Nella sezione Risposta aggiungere i seguenti tre messaggi:

```
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. Selezionare Salva intento, quindi Crea per creare il bot.

Infine, usa la finestra di test della console per testare le risposte dal bot. Le domande dovrebbero trovarsi nel dominio supportato dall'indice.

Per testare il bot di domande frequenti

1. Nella finestra di test della console, digitare una domanda per l'indice.
2. Verificare la risposta nella sezione delle risposte della finestra di test.
3. Per reimpostare la finestra di test per un'altra domanda, scegli Cancella cronologia chat.

AMAZON.PauseIntent

Risponde a parole e frasi che consentono all'utente di mettere in pausa un'interazione con un bot in modo da poterla riprendere in un secondo momento. La funzione o l'applicazione Lambda deve salvare i dati sugli intenti nelle variabili di sessione oppure è necessario utilizzare l'GetSessionoperazione per recuperare i dati sugli intenti quando si riprende l'intento corrente.

Enunciati comuni:

- pausa
- mettere in pausa

AMAZON.RepeatIntent

Risponde a parole e frasi che consentono all'utente di ripetere il messaggio precedente. L'applicazione deve utilizzare una funzione Lambda per salvare le informazioni sull'intento precedente nelle variabili di sessione oppure è necessario utilizzare l'GetSessionoperazione per ottenere le informazioni sull'intento precedente.

Enunciati comuni:

- ripetere
- ripetilo
- ripetilo

AMAZON.ResumeIntent

Risponde a parole e frasi che consentono all'utente di riprendere un intento precedentemente sospeso. La funzione o l'applicazione Lambda deve gestire le informazioni necessarie per riprendere l'intento precedente.

Enunciati comuni:

- riprendere
- continua
- continua

AMAZON.StartOverIntent

Risponde a parole e frasi che consentono all'utente di interrompere l'elaborazione dell'intento corrente e ricominciare da capo. È possibile utilizzare la funzione Lambda o l'`PutSession`operazione per ottenere nuovamente il valore del primo slot.

Enunciati comuni:

- ricominciare
- riavviare
- ricominciare

AMAZON.StopIntent

Risponde a parole e frasi che indicano che l'utente desidera interrompere l'elaborazione dell'intento corrente e terminare l'interazione con un bot. La funzione o l'applicazione Lambda dovrebbe cancellare tutti gli attributi e i valori del tipo di slot esistenti e quindi terminare l'interazione.

Enunciati comuni:

- arresta
- off
- zitto

Tipi di slot integrati

Amazon Lex supporta tipi di slot integrati che definiscono il modo in cui i dati nello slot vengono riconosciuti e gestiti. Puoi creare slot di questi tipi nei tuoi intenti. In questo modo, si elimina la necessità di creare valori di enumerazione per dati di slot utilizzati più frequentemente come data, ora e posizione. I tipi di slot integrati non hanno versioni.

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
Amazon.airport	Riconosce le parole che rappresentano un aeroporto.	Tutte le località	
AMAZON.AlphaNumeric	Riconosce parole costituite da lettere e numeri.	Tutte le versioni locali tranne il coreano (ko-KR)	
Amazon.city	Riconosce le parole che rappresentano una città.	Tutte le località	
Amazon.paese	Riconosce le parole che rappresentano un paese.	Tutte le località	

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
AMAZON.DATE	Riconosce le parole che rappresen tano una data e le converte in un formato standard.	Tutte le versioni locali	
AMAZON.DURATION	Riconosce le parole che rappresentano la durata e le converte in un formato standard.	Tutte le versioni locali	
AMAZZONE. EmailAddress	Riconosce le parole che rappresentano un indirizzo e-mail e le converte in un indirizzo e-mail standard.	Tutte le impostazioni locali	
AMAZON. FirstName	Riconosce le parole che rappresentano un nome.	Tutte le impostazioni locali	
AMAZZONE. LastName	Riconosce le parole che rappresentano un cognome.	Tutte le impostazioni locali	
AMAZON.NUMBER	Riconosce le parole numeriche e le converte in cifre.	Tutte le impostazioni locali	

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
AMAZON.Percentage	Riconosce le parole che rappresentano una percentuale e le converte in un numero e un segno di percentuale (%).	Tutte le impostazioni locali	
AMAZZONE. PhoneNumber	Riconosce le parole che rappresentano un numero di telefono e le converte in una stringa numerica.	Tutte le impostazioni locali	
AMAZZONE. SpeedUnit	Riconosce le parole che rappresentano un'unità di velocità e le converte in un'abbreviazione standard.	Inglese (Stati Uniti) (en-US)	
Amazon.state	Riconosce le parole che rappresentano uno stato.	Tutte le impostazioni locali	
AMAZZONE. StreetName	Riconosce le parole che rappresentano il nome di una strada.	Tutte le lingue tranne l'inglese (USA) (en-US)	
AMAZON.TIME	Riconosce le parole che indicano l'ora e le converte in un formato orario.	Tutte le impostazioni locali	

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
AMAZON. WeightUnit	Riconosce le parole che rappresentano un'unità di peso e le converte in un'abbrev iazione standard	Inglese (Stati Uniti) (en-US)	

> **Note**
>
> Per le impostazioni locali in inglese (USA) (en-US), Amazon Lex supporta i tipi di slot dell'Alexa Skill Kit. Per un elenco dei tipi di slot integrati, consultare la sezione di riferimento ai tipi di slot nella documentazione dell'Alexa Skills Kit.
>
> - Amazon Lex non supporta `AMAZON.LITERAL` i tipi di slot `AMAZON.SearchQuery` integrati.

Amazon.airport

Fornisce un elenco di aeroporti. Esempi includono:

- Aeroporto internazionale John F. Kennedy
- Aeroporto di Melbourne

AMAZON. AlphaNumeric

Riconosce stringhe costituite da lettere e numeri, ad esempio **`APQ123`**.

Questo tipo di slot non è disponibile nella versione locale coreana (ko-KR).

È possibile utilizzare il tipo di slot `AMAZON.AlphaNumeric` per stringhe che contengono:

- Caratteri alfabetici, come **`ABC`**
- Caratteri numerici, come **`123`**
- Una combinazione di caratteri alfanumerici, come **`ABC123`**

È possibile aggiungere un'espressione regolare al tipo di slot `AMAZON.AlphaNumeric` per convalidare i valori inseriti per lo slot. Ad esempio, è possibile utilizzare un'espressione regolare per convalidare:

- Codici postali del Regno Unito o del Canada
- Numeri della patente di guida
- Numeri di identificazione del veicolo

Usa un'espressione regolare standard. Amazon Lex supporta i seguenti caratteri nell'espressione regolare:

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

Amazon Lex supporta anche i caratteri Unicode nelle espressioni regolari. La forma è `\u`*`Unicode`*. Utilizzare quattro cifre per rappresentare i caratteri Unicode. Ad esempio, `[\u0041-\u005A]` è uguale a [A-Z].

I seguenti operatori di espressioni regolari non sono supportati:

- Ripetitori infiniti: *, + o {x,} senza limite superiore.
- Wild card (.)

La lunghezza massima dell'espressione regolare è di 300 caratteri. La lunghezza massima di una stringa memorizzata in AMAZON. AlphaNumeric il tipo di slot che utilizza un'espressione regolare è di 30 caratteri.

Di seguito sono riportate alcune espressioni regolari di esempio.

- Stringhe alfanumeriche, ad esempio **APQ123** o **APQ1**: `[A-Z]{3}[0-9]{1,3}` o un più vincolate `[A-DP-T]{3} [1-5]{1,3}`
- Formato US Postal Service Priority Mail International, quali **CP123456789US**: `CP[0-9]{9}US`
- Numeri di routing bancari, quali **123456789**: `[0-9]{9}`

Per impostare l'espressione regolare per un tipo di slot, utilizzare la console o l'operazione PutSlotType. L'espressione regolare viene convalidata quando si salva il tipo di slot. Se l'espressione non è valida, Amazon Lex restituisce un messaggio di errore.

Quando usi un'espressione regolare in un tipo di slot, Amazon Lex verifica l'input degli slot di quel tipo rispetto all'espressione regolare. Se l'input corrisponde all'espressione, il valore viene accettato per lo slot. Se l'input non corrisponde, Amazon Lex richiede all'utente di ripetere l'input.

Amazon.city

Fornisce un elenco di città locali e mondiali. Il tipo di slot riconosce le varianti più comuni dei nomi delle città. Amazon Lex non converte da una variante a un nome ufficiale.

Esempi:

- New York
- Reykjavik
- Tokyo
- Versailles

Amazon.paese

I nomi dei paesi di tutto il mondo. Esempi:

- Australia
- Germania
- Giappone
- Stati Uniti
- Uruguay

AMAZON.DATE

Converte le parole che rappresentano date in un formato di data.

La data viene fornita secondo le vostre intenzioni nel formato di data ISO-8601. La data di ricezione dell'intento nello slot può variare a seconda della frase specifica pronunciata dall'utente.

- Gli enunciati che corrispondono a una data specifica, ad esempio «oggi», «ora» o «venticinque novembre», vengono convertiti in una data completa:. `2020-11-25` L'impostazione predefinita è data uguale o successiva alla data corrente.
- Gli enunciati che corrispondono a una settimana specifica, ad esempio «questa settimana» o «settimana prossima», vengono convertiti nella data del primo giorno della settimana. Nel formato ISO-8601, la settimana inizia il lunedì e termina la domenica. Ad esempio, se oggi è il 25-11-2020, «settimana prossima» viene convertito in. `2020-11-30`
- Gli enunciati che corrispondono a un mese, ma non a un giorno specifico, ad esempio «mese successivo», vengono convertiti nell'ultimo giorno del mese. Ad esempio, se oggi è il 25/11/2020, «mese prossimo» verrà convertito in. `2020-12-31`
- Gli enunciati che corrispondono a un anno, ma non a un mese o giorno specifico, ad esempio «anno successivo», vengono convertiti nell'ultimo giorno dell'anno successivo. Ad esempio, se oggi è il 25/11/2020, «anno prossimo» verrà convertito in. `2021-12-31`

AMAZON.DURATION

Converte le parole che indicano la durata in una durata numerica.

La durata viene risolta in un formato basato sul formato di durata ISO-8601,. `PnYnMnWnDTnHnMnS` Pindica che si tratta di una durata, che n è un valore numerico e la lettera maiuscola che segue n è l'elemento specifico di data o ora. Ad esempio, `P3D` significa 3 giorni. A T viene utilizzato per indicare che i valori rimanenti rappresentano elementi temporali anziché elementi di data.

Esempi:

- «dieci minuti»: `PT10M`
- «cinque ore»: `PT5H`
- «tre giorni»: `P3D`
- «quarantacinque secondi»: `PT45S`
- «otto settimane»: `P8W`
- «sette anni»: `P7Y`
- «cinque ore e dieci minuti»: `PT5H10M`
- «due anni tre ore e dieci minuti»: `P2YT3H10M`

AMAZZONE. EmailAddress

Riconosce parole che rappresentano un indirizzo e-mail fornito come nomeutente@dominio. Gli indirizzi possono includere i seguenti caratteri speciali in un nome utente: sottolineatura (_), trattino (-), punto (.) e il segno più (+).

AMAZON. FirstName

Nomi di battesimo comunemente usati. Questo tipo di slot riconosce sia i nomi formali che i soprannomi informali. Il nome inviato all'intento è il valore inviato dall'utente. Amazon Lex non esegue la conversione dal nickname al nome formale.

Per i nomi che suonano allo stesso modo ma sono scritti in modo diverso, Amazon Lex invia alle tue intenzioni un unico modulo comune.

Nella lingua inglese (USA) (en-US), usa il nome dello slot Amazon.US_first_name.

Esempi:

- Emilia
- John
- Sofia

AMAZZONE. LastName

Cognomi di uso comune. Per i nomi che hanno lo stesso suono e che vengono scritti in modo diverso, Amazon Lex invia al tuo intento un'unica forma comune.

Nella lingua inglese (USA) (en-US), usa il nome dello slot Amazon.US_last_name.

Esempi:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON.NUMBER

Converte le parole o i numeri che esprimono un numero in cifre, compresi i numeri decimali. La tabella seguente mostra come il tipo di slot `AMAZON.NUMBER` acquisisce parole numeriche.

Input	Risposta
centoventitre punto quattro cinque	123.45
centoventitre punto quattro cinque	123.45
punto quattro due	0.42
punto quarantadue	0.42
232.998	232.998
50	50

AMAZON.Percentage

Converte parole e simboli che rappresentano una percentuale in un valore numerico con un segno di percentuale (%).

Se l'utente immette un numero senza un segno di percentuale o le parole "percent", il valore di slot è impostato sul numero. La tabella seguente mostra come il tipo di slot `AMAZON.Percentage` acquisisce le percentuali.

Input	Risposta
50 per cento	50%
0,4 percento	0.4%
23.5%	23.5%
venticinque percento	25%

AMAZZONE. PhoneNumber

Converte numeri o parole che rappresentano un numero di telefono in un formato stringa senza punteggiatura come segue.

Tipo	Descrizione	Input	Risultato
Numero internazionale con un segno più (+) all'inizio	Numero di 11 cifre con un segno più all'inizio.	+61 7 4445 1061 +1 (509) 555-1212	`+61744431061` `+15095551212`
Numero internazionale senza un segno più (+) all'inizio	Numero a 11 cifre senza un segno più all'inizio	1 (509) 555-1212 61 7 4445 1061	`15095551212` `61744451061`
Numero nazionale	Numero a 10 cifre senza prefisso internazionale	(03) 5115 4444 (509) 555-1212	`0351154444` `5095551212`
Numero locale	Numero di telefono a 7 cifre senza prefisso internazionale o indicativo di località	555-1212	`5551212`

AMAZZONE. SpeedUnit

Converte parole che rappresentano unità di velocità in un'abbreviazione corrispondente. Ad esempio, "miles per hour" viene convertito in `mph`.

Questo tipo di slot è disponibile solo nella lingua inglese (USA) (en-US).

I seguenti esempi mostrano come il tipo di slot `AMAZON.SpeedUnit` acquisisce unità di velocità.

Unità di velocità	Abbreviazione
miglia orarie, mph, MPH, m/h	mph
chilometri orari, km orari, kmph, KMPH, km/h	kmph

Unità di velocità	Abbreviazione
metri al secondo, mps, MPS, m/s	mps
miglia nautiche orarie, nodi, nodo	nodo

Amazon.state

I nomi delle regioni geografiche e politiche all'interno dei paesi.

Esempi:

- Baviera
- Prefettura di Fukushima
- Pacifico nord-occidentale
- Queensland
- Galles

AMAZZONE. StreetName

I nomi delle strade all'interno di un indirizzo tipico. Ciò include solo il nome della via, non il numero civico.

Questo tipo di slot non è disponibile nella versione locale inglese (USA) (en-US).

Esempi:

- Viale Canberra
- Strada anteriore
- Strada del mercato

AMAZON.TIME

Converte parole che rappresentano ore in valori orari. Include risoluzioni per tempi ambigui. Quando un utente inserisce un orario ambiguo, Amazon Lex utilizza l'`slotDetails`attributo di un evento Lambda per passare le risoluzioni per i tempi ambigui alla funzione Lambda. Ad esempio, se il tuo

bot richiede all'utente un'ora di consegna, l'utente può rispondere dicendo "10 o'clock". Questo orario è ambiguo, in quanto può intendere le 10 di mattina o le 10 di sera. In questo caso, il valore nella `slots` mappa è `null` e l'`slotDetails`entità contiene le due possibili risoluzioni dell'ora. Amazon Lex inserisce quanto segue nella funzione Lambda:

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

Quando l'utente risponde con un orario inequivocabile, Amazon Lex invia l'ora alla funzione Lambda nell'attributo `slots` dell'evento Lambda e l'attributo è vuoto. `slotDetails` Ad esempio, se l'utente risponde alla richiesta di un orario di consegna con "22:00», Amazon Lex inserisce quanto segue nella funzione Lambda:

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

Per ulteriori informazioni sui dati inviati da Amazon Lex a una funzione Lambda, consulta. Formato dell'evento di input

AMAZON. WeightUnit

Converte parole che rappresentano un'unità di peso nell'abbreviazione corrispondente. Ad esempio, "kilogram" viene convertito in `kg`.

Questo tipo di slot è disponibile solo nella lingua inglese (USA) (en-US).

Gli esempi seguenti mostrano come il tipo di slot `AMAZON.WeightUnit` acquisisce unità di peso:

Unità di peso	Abbreviazione
chilogrammi, chili, kgs, KGS	kg
grammi, gms, gm, GMS, g	g
milligrammi, mg, mgs	mg
libbre, lbs, LBS	lbs
once, oz, OZ	oz
tonnellata, t	t
chilotonnellata, kt	kt

Tipi di slot personalizzati

Per ogni intento, puoi specificare i parametri che indicano le informazioni necessarie all'intento per adempiere alla richiesta dell'utente. Questi parametri o slot sono di diversi tipi. Un tipo di slot è un elenco di valori che Amazon Lex utilizza per addestrare il modello di machine learning a riconoscere i valori di uno slot. Ad esempio, puoi definire un tipo di slot denominato "`Genres.`" Ogni valore nel tipo di slot è il nome di un genere, "commedia", "avventura", "documentario", ecc. Puoi definire un sinonimo per un valore del tipo di slot. Ad esempio, puoi definire i sinonimi "divertente" e "spiritoso" per il valore "commedia".

Puoi configurare il tipo di slot per limitare la risoluzione ai valori di slot. I valori di slot saranno utilizzati come un'enumerazione e il valore immesso dall'utente determinerà il valore di slot solo se è uguale a uno dei valori di slot o un sinonimo. Un sinonimo determina il valore di slot corrispondente. Ad esempio, se l'utente immette "divertente" determinerà il valore di slot "commedia".

In alternativa, puoi configurare il tipo di slot per espandere i valori. I valori di slot saranno utilizzati come dati di addestramento e lo slot determina il valore fornito dall'utente se è simile ai valori e sinonimi dello slot. Questo è il comportamento che segue di default.

Amazon Lex mantiene un elenco di possibili risoluzioni per uno slot. Ogni voce dell'elenco fornisce un valore di risoluzione che Amazon Lex ha riconosciuto come possibilità aggiuntive per lo slot. Un valore di risoluzione è un best effort per la corrispondenza del valore di slot. L'elenco contiene fino a cinque valori.

Quando un valore immesso dall'utente è un sinonimo, la prima voce nell'elenco dei valori di risoluzione è il valore del tipo di slot. Ad esempio, se l'utente immette "divertente" il campo `slots` contiene "divertente" e la prima voce nel campo `slotDetails` è "commedia". Puoi configurare il `valueSelectionStrategy` quando crei o aggiorni un tipo di slot con l'operazione PutSlotType in modo che il valore di slot venga compilato con il primo valore dell'elenco di risoluzione.

Se si utilizza una funzione Lambda, l'evento di input della funzione include un elenco di risoluzioni chiamato`slotDetails`. L'esempio seguente mostra la sezione dei dettagli dello slot e dello slot dell'input di una funzione Lambda:

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

Per ogni tipo di slot, si possono definire un massimo di 10.000 valori e sinonimi. Ogni bot può includere un numero totale di 50.000 sinonimi e valori di tipi di slot. Ad esempio, è possibile avere 5 tipi di slot, ognuno con 5.000 valori e 5.000 sinonimi, oppure 10 slot, ognuno con 2.500 valori e 2.500 sinonimi. Se si superano questi limiti, si riceverà un `LimitExceededException` richiamando l'operazione PutBot.

Offuscamento degli slot

Amazon Lex consente di offuscare (nascondere) il contenuto degli slot in modo che non sia visibile. Per proteggere i dati sensibili acquisiti come valori degli slot, puoi abilitare l'offuscamento degli slot per mascherare questi valori nei log delle conversazioni.

Quando si sceglie di offuscare i valori degli slot, Amazon Lex sostituisce il valore dello slot con il nome dello slot nei log delle conversazioni. Per uno slot chiamato `full_name`, il valore dello slot verrebbe offuscato nel seguente modo:

```
Before obfuscation:
```

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

Se un'enunciazione contiene delle parentesi ({}), Amazon Lex li sostituisce con due barre rovesciate (\\). Ad esempio, il testo `{John Stiles}` viene offuscato come segue:

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

I valori degli slot vengono offuscati nei log delle conversazioni. I valori dello slot sono ancora disponibili nella risposta dal`PostContente``PostText`operazioni e valori di slot sono disponibili per le funzioni di convalida e di adempimento. Se si utilizzano valori di slot nei prompt o nelle risposte, tali valori non vengono offuscati nei log delle conversazioni.

Nel primo turno di una conversazione, Amazon Lex offusca i valori degli slot se riconosce uno slot e un valore di slot nell'enunciazione. Se non viene riconosciuto alcun valore di slot, Amazon Lex non offusca l'enunciazione.

Nel secondo turno e nei successivi, Amazon Lex conosce lo slot da ottenere e se il valore dello slot deve essere offuscato. Se Amazon Lex riconosce il valore dello slot, il valore viene offuscato. Se Amazon Lex non riconosce un valore, l'intera enunciazione viene offuscata. Qualsiasi valore di slot in enunciazioni perse non verrà offuscato.

Inoltre, Amazon Lex non offusca i valori di slot archiviati negli attributi di richiesta o di sessione. Se si archiviano valori di slot che devono essere offuscati come attributo, è necessario crittografare o nascondere in altro modo il valore.

Amazon Lex non offusca il valore dello slot nell'audio. Offusca il valore dello slot nella trascrizione audio.

Non è necessario offuscare tutti gli slot in un bot. È possibile scegliere quali slot offuscare utilizzando la console o l'API di Amazon Lex. Nella console, selezionare Slot obfuscation (Offuscamento dello slot) nelle impostazioni di uno slot. Se si utilizza l'API, impostare il campo `obfuscationSetting` dello slot su `DEFAULT_OBFUSCATION` quando si invoca l'operazione PutIntent.

Analisi delle emozioni

È possibile utilizzare l'analisi del sentiment per determinare i sentimenti espressi in un enunciato utente. Con le informazioni sulle emozioni è possibile gestire il flusso di conversazione o eseguire l'analisi post-chiamata. Ad esempio, se l'emozione dell'utente è negativa, è possibile creare un flusso per passare una conversazione a un agente umano.

Amazon Lex si integra con Amazon Comprehend per rilevare le opinioni degli utenti. La risposta di Amazon Comprehend indica se il sentimento generale del testo è positivo, neutro, negativo o misto. La risposta contiene l'emozione più probabile per l'enunciato utente e i punteggi per ciascuna delle categorie di emozioni. Il punteggio rappresenta la probabilità che l'emozione sia stata rilevata correttamente.

Puoi abilitare l'analisi del sentiment per un bot utilizzando la console o utilizzando l'API Amazon Lex. Sulla console Amazon Lex, scegli la scheda Impostazioni per il tuo bot, quindi imposta l'opzione Sentiment Analysis su Sì. Se si utilizza l'API, chiamare l'operazione PutBot con il campo `detectSentiment` impostato su `true`.

Quando l'analisi del sentiment è abilitata, la risposta delle operazioni PostContent e PostText restituisce un campo chiamato `sentimentResponse` nella risposta bot con altri metadati. Il campo `sentimentResponse` ha due campi, `SentimentLabel` e `SentimentScore`, che contengono il risultato dell'analisi dell'emozione. Se si utilizza una funzione Lambda, il`sentimentResponse` campo è incluso nei dati dell'evento inviati alla funzione.

Di seguito è riportato un esempio del campo `sentimentResponse` restituito come parte della risposta `PostText` o `PostContent`. Il campo `SentimentScore` è una stringa che contiene i punteggi per la risposta.

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

Amazon Lex chiama Amazon Comprehend per tuo conto per determinare il sentimento in ogni espressione elaborata dal bot. Abilitando l'analisi del sentiment, accetti i termini e gli accordi di servizio per Amazon Comprehend. Per ulteriori informazioni sui prezzi di Amazon Comprehend, consulta Amazon Comprehend nella Guida di riferimento di Amazon Comprehend.

Per ulteriori informazioni su come funziona l'analisi del sentiment di Amazon Comprehend, consulta Determinare il sentiment nella Amazon Comprehend Developer Guide.

Assegnazione di tag alle risorse Amazon Lex

Per gestire i bot, gli alias bot e i canali bot di Amazon Lex, puoi assegnare i metadati a ciascuna risorsatag. Un tag è un'etichetta che assegni a una risorsa AWS. Ciascun tag è formato da una chiave e da un valore,

I tag ti consentono di categorizzare le tue risorse AWS in modi diversi, ad esempio, per scopo, proprietario o applicazione. I tag ti aiutano a:

- Identificazione e organizzazione delle risorse AWS. Molte risorse AWS supportano il tagging, perciò è possibile assegnare lo stesso tag a risorse di diversi servizi per indicare che queste sono correlate. Ad esempio, puoi contrassegnare un bot e le funzioni Lambda che utilizza con lo stesso tag.
- Assegnare i costi. I tag vengono attivati nel pannello di controllo AWS Billing and Cost Management. AWS usa i tag per categorizzare i costi e fornire un report di allocazione dei costi mensili. Per Amazon Lex, puoi allocare i costi per ogni alias utilizzando i tag specifici dell'alias, ad eccezione della`$LATEST`alias. Allocare i costi per il`$LATEST`Alias che utilizza i tag per il bot Amazon Lex. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing and Cost Management.
- Controllare l'accesso alle risorse di . Puoi utilizzare i tag in Amazon Lex per creare policy e controllare l'accesso alle risorse Amazon Lex. Queste policy possono essere collegate a un ruolo o a un utente IAM per abilitare il controllo dell'accesso basato su tag. Per ulteriori informazioni, consultare ABAC con Amazon Lex. Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Usa un tag per accedere a una risorsa.

Puoi lavorare con i tag utilizzando laAWS Management Console, ilAWS Command Line Interfaceo l'API Amazon Lex.

Tagging delle risorse

Se usi la console di Amazon Lex, puoi contrassegnare le risorse durante la creazione oppure aggiungere i tag in un secondo momento. È inoltre possibile utilizzare la console per aggiornare o rimuovere i tag esistenti.

Se stai usando laAWS CLIo l'API di Amazon Lex, puoi utilizzare le seguenti operazioni per gestire i tag per le risorse:

- ListTagsForResource— visualizzare i tag associati a una risorsa.
- PutBotePutBotAlias— applicare i tag quando un bot o un alias bot viene creato.
- TagResourceaggiungere e modificare i tag in una risorsa esistente.
- UntagResourcerimuovere i tag da una risorsa.

Le seguenti risorse in Amazon Lex supportano l'assegnazione di tag:

- Bots: usa un Amazon Resource Name (ARN) simile al seguente:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}`
- Alias bot: usa un ARN come il seguente:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}:${bot-alias}`
- Canali bot: usa un ARN come il seguente:
  - `arn:${partition}:lex:${region}:${account}:bot-channel:${bot-name}:${bot-alias}:${channel-name}`

Limitazioni applicate ai tag

Le seguenti restrizioni di base si applicano ai tag sulle risorse Amazon Lex:

- Numero massimo di tag - 50
- Lunghezza massima della chiave - 128 caratteri
- Lunghezza massima del valore - 256 caratteri
- Caratteri validi per chiave e valore: a—z, A-Z, 0—9, spazi e i seguenti caratteri: _.:/= + - e @
- Per chiavi e valori viene fatta distinzione tra maiuscole e minuscole.
- Non utilizzare `aws:` come prefisso per le chiavi; l'utilizzo di questo prefisso è esclusivo di AWS.

Tagging delle risorse (console)

Puoi utilizzare la console per gestire i tag per una risorsa bot, alias bot o canale bot. Puoi aggiungere i tag quando crei una risorsa oppure puoi aggiungere, modificare o rimuovere tag dalle risorse esistenti.

Per aggiungere un tag quando si crea un bot

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegliere Create (Crea) per creare un nuovo bot.
3. Nella parte inferiore della pagina Create your bot (Crea il tuo bot) scegliere Tags (Tag).
4. Scegliere Add tag (Aggiungi tag) e aggiungere uno o più tag al bot. Puoi aggiungere fino a 50 tag.

Per aggiungere un tag quando si crea un alias bot

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegliere il bot a cui si desidera aggiungere l'alias bot.
3. Seleziona Settings (Impostazioni).
4. Aggiungere il nome alias, scegliere la versione bot, quindi scegliere Add tags (Aggiungi tag).
5. Scegliere Add tag (Aggiungi tag) e aggiungere uno o più tag all'alias bot. Puoi aggiungere fino a 50 tag.

Per aggiungere un tag quando si crea un canale bot

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegliere il bot a cui si desidera aggiungere il canale bot.
3. Scegliere Channels (Canali) quindi scegliere il canale che si desidera aggiungere.
4. Aggiungere i dettagli per il canale bot, quindi scegliere Tags (Tag).
5. Scegliere Add tag (Aggiungi tag) e aggiungere uno o più tag al canale bot. Puoi aggiungere fino a 50 tag.

Per aggiungere un tag quando si importa un bot

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegliere Actions (Operazioni), quindi Import (Importa).
3. Scegliere il file zip per importare il bot.
4. Scegliere Tags (Tag), quindi scegliere Add tag (Aggiungi tag) per aggiungere uno o più tag al bot. Puoi aggiungere fino a 50 tag.

Per aggiungere, rimuovere o modificare un tag per un bot esistente

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Dal menu a sinistra, scegliere Bot e selezionare il bot che si desidera modificare.
3. Scegliere Settings (Impostazioni) quindi dal menu a sinistra scegliere General (Generale).
4. Scegliere Tags (Tag) e quindi aggiungere, modificare o rimuovere i tag per il bot.

Per aggiungere, rimuovere o modificare un tag per un alias bot

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Dal menu a sinistra, scegliere Bot e selezionare il bot che si desidera modificare.
3. Scegliere Settings (Impostazioni) quindi dal menu a sinistra scegliere Aliases (Alias).
4. Scegliere Manage tags (Gestisci tag) per l'alias che si desidera modificare, quindi aggiungere, modificare o rimuovere i tag per l'alias bot.

Per aggiungere, rimuovere o modificare un tag per un canale bot esistente

1. Eseguire l'accesso allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Dal menu a sinistra, scegliere Bot e selezionare il bot che si desidera modificare.
3. Seleziona Channels (Canali).
4. Scegliere Tags (Tag) e quindi aggiungere, modificare o rimuovere i tag per il canale bot.

Tagging delle risorse (AWS CLI)

Puoi utilizzare l'AWS CLI per gestire i tag per una risorsa bot, alias bot o canale bot. Puoi aggiungere i tag quando crei un bot o un alias bot oppure puoi aggiungere, modificare o rimuovere i tag da un bot, un alias bot o un canale bot.

Tutti gli esempi sono formattati per Linux e macOS. Per utilizzare il comando in Windows, sostituisci il carattere di continuazione Linux (\) con un accento circonflesso (^).

Per aggiungere un tag quando si crea un bot

- Il seguente comando abbreviato `put-bot` dell'AWS CLI mostra i parametri che devi utilizzare per aggiungere un tag quando crei un bot. Per creare effettivamente un bot, devi fornire altri parametri. Per ulteriori informazioni, consultare Fase 4: Nozioni di base (AWS CLI).

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

Per aggiungere un tag quando si crea un alias bot

- Il seguente comando abbreviato `put-bot-alias` dell'AWS CLI mostra i parametri che devi utilizzare per aggiungere un tag quando crei un alias bot. Per creare effettivamente un alias bot, devi fornire altri parametri. Per ulteriori informazioni, consultare Esercizio 5: Creazione di un alias (AWS CLI).

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

Per elencare i tag per una risorsa

- Utilizzare il comando `list-tags-for-resource` dell'AWS CLI per mostrare le risorse associate a un bot, un alias bot, un canale bot.

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

Per aggiungere o modificare i tag in una risorsa

- Utilizzare il comando `tag-resource` dell'AWS CLI per aggiungere o modificare un bot, un alias bot o un canale bot.

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

Per rimuovere i tag da una risorsa

- Utilizzare il comando `untag-resource` dell'AWS CLI per rimuovere i tag da un bot, un alias bot o un canale bot.

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

Guida introduttiva ad Amazon Lex

Amazon Lex fornisce operazioni API che puoi integrare con le tue applicazioni esistenti. Per un elenco delle operazioni supportate, consultare la Documentazione di riferimento delle API. È possibile utilizzare una qualsiasi delle seguenti opzioni:

- SDK AWS: quando utilizzi gli SDK, le tue richieste ad Amazon Lex vengono firmate e autenticate automaticamente utilizzando le credenziali fornite. Questa è la soluzione consigliata per la creazione di applicazioni.
- AWS CLI — Puoi utilizzarlo AWS CLI per accedere a qualsiasi funzionalità di Amazon Lex senza dover scrivere alcun codice.
- Console AWS: la console è il modo più semplice per iniziare a testare e utilizzare Amazon Lex

Se non conosci Amazon Lex, ti consigliamo di leggere primaAmazon Lex: come funziona: come funziona.

Argomenti

- Fase 1: configurare un AWS account e creare un utente amministratore
- Fase 2: Configurare il AWS Command Line Interface
- Fase 3: nozioni di base (console)
- Fase 4: Nozioni di base (AWS CLI)

Fase 1: configurare un AWS account e creare un utente amministratore

Prima di utilizzare Amazon Lex per la prima volta, completa le seguenti attività:

1. Registrati per AWS
2. Creazione di un utente

Registrati per AWS

Se hai già un AWS account, salta questa attività.

Quando ti registri ad Amazon Web Services (AWS), il tuo AWS account viene automaticamente registrato per tutti i servizi in AWS, incluso Amazon Lex. Ti vengono addebitati solo i servizi che utilizzi.

Con Amazon Lex, paghi solo per le risorse che utilizzi. Se sei un nuovo AWS cliente, puoi iniziare a usare Amazon Lex gratuitamente. Per ulteriori informazioni, consulta Piano di utilizzo gratuito di AWS.

Se hai già un AWS account, passa all'attività successiva. Se non disponi di un account AWS , utilizza la seguente procedura per crearne uno.

Per creare un account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come procedura consigliata in materia di sicurezza, assegnate l'accesso amministrativo a un utente e utilizzate solo l'utente root per eseguire attività che richiedono l'accesso da parte dell'utente root.

Annota l'ID AWS del tuo account perché ti servirà per l'attività successiva.

Creazione di un utente

I servizi in AWS, come Amazon Lex, richiedono l'immissione di credenziali al momento dell'accesso, in modo che il servizio possa determinare se disponi delle autorizzazioni per accedere alle risorse di proprietà di quel servizio. Per la console è necessaria la password. Tuttavia, non ti consigliamo di accedere AWS utilizzando le credenziali del tuo account. AWS Consigliamo invece di effettuare queste operazioni:

- Usa AWS Identity and Access Management (IAM) per creare un utente
- Aggiungi l'utente a un gruppo IAM con autorizzazioni amministrative
- Concedere autorizzazioni amministrative per l'utente creato.

Puoi quindi accedere AWS utilizzando un URL speciale e le credenziali dell'utente.

Per eseguire gli esercizi Nozioni di base di questa guida si presuppone che tu abbia creato un utente (`adminuser`) con privilegi di amministratore. Per creare un `adminuser` nel tuo account, utilizza la procedura indicata di seguito.

Per creare un utente amministratore e accedere alla console

1. Crea nel tuo account AWS un utente amministratore denominato `adminuser`. Per istruzioni, consulta Creazione del primo gruppo di utenti e amministratori nella Guida per l'utente IAM.
2. Come utente, puoi accedere a AWS Management Console utilizzando un URL speciale. Per ulteriori informazioni, consulta la sezione How Users Sign In to Your Account (Come gli utenti accedono al tuo account) nella IAM User Guide (Guida per l'utente di IAM).

Per ulteriori informazioni su IAM, consulta:

- AWS Identity and Access Management (IAM)
- Nozioni di base
- Guida per l'utente di IAM

Fase succcessiva

Fase 2: Configurare il AWS Command Line Interface

Fase 2: Configurare il AWS Command Line Interface

Se preferisci usare Amazon Lex con AWS Command Line Interface (AWS CLI), scaricalo e configuralo.

> **Important**
>
> Non è necessario che AWS CLI esegua i passaggi degli esercizi introduttivi. Tuttavia, alcuni esercizi successivi di questa guida si basano sull'utilizzo di AWS CLI. Se preferisci iniziare utilizzando la console, ignora questa fase e passa a Fase 3: nozioni di base (console). Successivamente, quando ne avrai bisogno AWS CLI, torna qui per configurarlo.

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente di AWS Command Line Interface :

   - Configurarsi con il AWS Command Line Interface
   - Configurazione della AWS Command Line Interface
2. Aggiungi un profilo denominato per l'utente amministratore alla fine del file di AWS CLI configurazione. Questo profilo viene utilizzato durante l'esecuzione dei AWS CLI comandi. Per ulteriori informazioni sui profili denominati, consulta Profili denominati in Guida per l'utente dell'AWS Command Line Interface .

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

   Per un elenco delle AWS regioni disponibili, consulta Regioni ed endpoint in. Riferimenti generali di Amazon Web Services
3. Verifica la configurazione digitando il comando help al prompt dei comandi:

   ```
   aws help
   ```

Fase 3: nozioni di base (console)

Fase 3: nozioni di base (console)

Il modo più semplice per imparare a usare Amazon Lex è usare la console. Per iniziare, abbiamo creato i seguenti esercizi, ognuno dei quali si basa sull'uso della console:

- Esercizio 1: crea un bot Amazon Lex utilizzando un blueprint, un bot predefinito che fornisce tutte le configurazioni di bot necessarie. Fai solo un minimo di lavoro per testare la end-to-end configurazione.

  Inoltre, si utilizza il blueprint della funzione Lambda, fornito da AWS Lambda, per creare una funzione Lambda. Questa funzione rappresenta un hook di codice che utilizza codice predefinito compatibile con il bot.

- Esercizio 2: creare un bot personalizzato creando e configurando manualmente un bot. Puoi anche creare una funzione Lambda come code hook. Viene fornito il codice di esempio.
- Esercizio 3: pubblicate un bot, quindi createne una nuova versione. Parte di questo esercizio consiste nella creazione di un alias che punta alla versione del bot.

Argomenti

Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console)

In questo esercizio, devi effettuare le seguenti operazioni:

- Crea il tuo primo bot Amazon Lex e testalo nella console Amazon Lex.

  Per questo esercizio, si utilizza il OrderFlowersblueprint. Per ulteriori informazioni sui piani, consulta l'argomento Amazon Lex eAWS LambdaPiani.

- Create unaAWS Lambda funzione e testarla nella console Lambda. Durante l'elaborazione di una richiesta, il bot chiama questa funzione Lambda. Per questo esercizio, si utilizza un blueprint Lambda (lex-order-flowers-python) fornito nellaAWS Lambda console per creare la funzione Lambda. Il codice del blueprint illustra come utilizzare la stessa funzione Lambda per eseguire l'inizializzazione e la convalida e per soddisfare l'`OrderFlowers`intento.

- Aggiorna il bot per aggiungere la funzione Lambda come hook di codice per soddisfare l'intento. Prova l' end-to-end esperienza.

Le seguenti sezioni illustrano cosa fanno i piani.

Amazon Lex Bot: panoramica del progetto

Utilizzi il OrderFlowersblueprint per creare un bot Amazon Lex. Per ulteriori informazioni sulla struttura di un bot, consultaAmazon Lex: come funziona: come funziona. Il bot è preconfigurato come segue:

- Intento — OrderFlowers
- Tipi di slot: un tipo di slot personalizzato denominato `FlowerTypes` con i valori di enumerazione: `roses`, `lilies` e `tulips`.
- Slot: prima che il bot possa realizzare l'intento, quest'ultimo richiede le informazioni riportate di seguito (slot).
  - `PickupTime` (tipo integrato AMAZON.TIME)
  - `FlowerType`(tipoFlowerTypes personalizzato)
  - `PickupDate` (tipo integrato AMAZON.DATE)
- Enunciazione: le seguenti enunciazioni di esempio indicano l'intento dell'utente:
  - "I would like to pick up flowers."
  - "I would like to order some flowers."
- Prompt: dopo che ha identificato l'intento, il bot utilizza i seguenti prompt per riempire gli slot:
  - Prompt per lo slot `FlowerType`: "What type of flowers would you like to order?"
  - Richiedi lo`PickupDate` slot: «In che giorno vuoi che il {FlowerType} venga ritirato?»
  - Richiedi lo`PickupTime` slot: «A che ora vuoi che il {FlowerType} venga ritirato?»
  - Dichiarazione di conferma: «Ok, il tuo {FlowerType} sarà pronto per il ritiro entro {PickupTime} il {PickupDate}. Does this sound okay?"

Funzione AWS Lambda: riepilogo del piano

La funzione Lambda in questo esercizio esegue sia le attività di inizializzazione che di convalida e di adempimento. Pertanto, dopo aver creato la funzione Lambda, si aggiorna la configurazione degli intenti specificando la stessa funzione Lambda come hook di codice per gestire sia le attività di inizializzazione che di convalida ed esecuzione.

- Come hook di codice di inizializzazione e convalida, la funzione Lambda esegue la convalida di base. Ad esempio, se l'utente fornisce un orario per il ritiro al di fuori del normale orario lavorativo, la funzione Lambda ordina ad Amazon Lex di richiedere nuovamente all'utente l'ora.

- Come parte del codice logistico, la funzione Lambda restituisce un messaggio di riepilogo che indica che l'ordine dei fiori è stato effettuato (ovvero che l'intento è stato soddisfatto).

Fase successiva

Fase 1: creazione Amazon Lex bot (console)

Fase 1: creazione Amazon Lex bot (console)

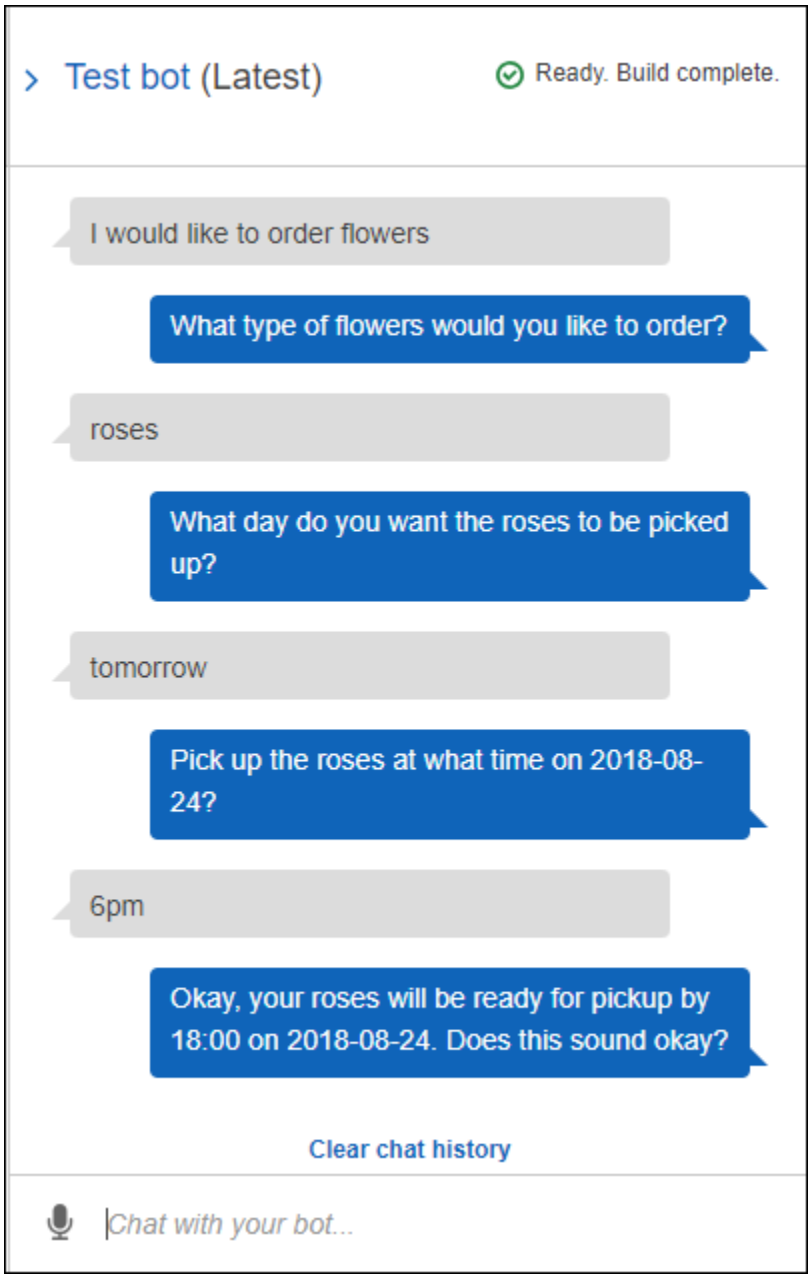
Per questo esercizio, crea un bot per ordinare fiori, chiamato OrderFlowersBot.

Per creare un bot Amazon Lex (console)

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Se questo è il tuo primo bot, seleziona Get Started (Inizia); altrimenti, nella pagina Bots (Bot), scegli Crea.
3. Nella pagina Create your Lex bot (Crea bot di Lex), fornisci le seguenti informazioni, quindi scegli Create (Crea).
   - Scegli il OrderFlowersblueprint.
   - Lascia il nome del bot predefinito (OrderFlowers).
   - Per COPPA, scegliere **No**.
   - Per la memorizzazione delle espressioni dell'utente, scegli la risposta appropriata.
4. Seleziona Create (Crea). La console invia le richieste necessarie ad Amazon Lex per salvare la configurazione. La console visualizza la finestra dell'editor il bot.
5. Attendere la conferma che il bot è stato creato.
6. Esegui il test del bot.

> **Note**
>
> Puoi eseguire il test del bot digitando una parte di testo nella finestra di prova oppure, nei browser compatibili, scegliendo il pulsante del microfono nella finestra di prova e parlando.

Utilizza il seguente testo di esempio per avviare una conversazione con il bot per l'ordine dei fiori:



Da questo input, il bot deduce l'intento `OrderFlowers` e richiede i dati dello slot. Una volta forniti tutti i dati di slot necessari, il bot realizza l'intento (`OrderFlowers`) restituendo tutte le informazioni all'applicazione client (in questo caso, la console). La console mostra le informazioni nella finestra di prova.

Nello specifico:

- Nell'istruzione "What day do you want the roses to be picked up?", il termine "roses" appare perché il prompt dello slot `pickupDate` viene configurato utilizzando delle sostituzioni, `{FlowerType}`. Verificare nella console.
- L'istruzione "Okay, your roses will be ready..." è il prompt di conferma che hai configurato.
- L'ultima istruzione ("`FlowerType:roses...`") indica semplicemente i dati dello slot restituiti al client, in questo caso, nella finestra di prova. Nell'esercizio successivo, si utilizza una funzione Lambda per soddisfare l'intento, nel qual caso viene visualizzato un messaggio che indica che l'ordine è stato evaso.

Fase successiva

Fase 2 (facoltativo): Revisione dei dettagli del flusso di informazioni (console)

Fase 2 (facoltativo): Revisione dei dettagli del flusso di informazioni (console)

Questa sezione spiega il flusso di informazioni tra un cliente e Amazon Lex per ogni input dell'utente nella nostra conversazione di esempio.

L'esempio utilizza la finestra di test della console per la conversazione con il bot.

Per aprire la finestra di test di Amazon Lex

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot da testare.
3. Sul lato destro della console, scegliere Test del chatbot.

Per visualizzare il flusso di informazioni relativo ai contenuti parlati o digitati, scegli l'argomento appropriato.

Argomenti

- Fase 2a (facoltativo): Revisione dei dettagli del flusso di informazioni parlate (console)
- Fase 2b (facoltativo): Revisione dei dettagli del flusso di informazioni digitate (console)

Fase 2a (facoltativo): Revisione dei dettagli del flusso di informazioni parlate (console)

Questa sezione spiega il flusso di informazioni tra il client e Amazon Lex quando il client utilizza la voce per inviare richieste. Per ulteriori informazioni, consulta PostContent.

1. L'utente afferma: Vorrei ordinare dei fiori.

   a. Il client (console) invia la seguente richiesta PostContent ad 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
```

   Sia l'URI della richiesta che il corpo forniscono informazioni ad Amazon Lex:

   - URI di richiesta: fornisce il nome del bot (`OrderFlowers`), l'alias del bot ($LATEST) e il nome utente (una stringa casuale che identifica l'utente). `content`indica che si tratta di una richiesta`PostContent` API (non di una`PostText` richiesta).
   - Intestazioni della richiesta
     - `x-amz-lex-session-attributes`— Il valore con codifica base64 rappresenta «{}». Quando il client effettua la prima richiesta non vi sono attributi della sessione.
     - `Content-Type`: riflette il formato audio.
   - Corpo della richiesta: il flusso audio dell'input utente ("I would like to order some flowers.").

   > **Note**
   >
   > Se invece di usare la voce, l'utente sceglie di inviare una parte di testo ("I would like to order some flowers") all'API `PostContent`, il corpo della richiesta sarà l'input utente. L'intestazione `Content-Type` viene impostata di conseguenza:
   >
   > ```
   > 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. Dal flusso di input, Amazon Lex rileva l'intento (`OrderFlowers`). Quindi, sceglie uno degli slot di intenti (in questo caso, `FlowerType`) e uno dei relativi prompt per l'ottenimento del valore, infine invia una risposta con le seguenti intestazioni:

```
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==
```

I valori dell'intestazione forniscono le informazioni seguenti:

- `x-amz-lex-input-transcript`: fornisce la trascrizione dell'audio (input utente) dalla richiesta
- `x-amz-lex-message`— Fornisce la trascrizione dell'audio che Amazon Lex ha restituito nella risposta
- `x-amz-lex-slots`: la versione codificata base64 di slot e valori:

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

- `x-amz-lex-session-attributes`: la versione codificata base64 degli attributi della sessione ({})

Il client riproduce l'audio nel corpo della risposta.

2. L'utente pronuncia: rose

   a. Il client (console) invia la seguente richiesta PostContent ad 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")
```

Il corpo della richiesta è il flusso audio dell'input utente (rose). `sessionAttributes` rimane vuoto.

b. Amazon Lex interpreta il flusso di input nel contesto dell'intento corrente (ricorda di aver chiesto a questo utente informazioni relative allo`FlowerType` slot). Amazon Lex aggiorna innanzitutto il valore dello slot per l'intento corrente. Quindi sceglie un altro slot (`PickupDate`), insieme a uno dei relativi messaggi di richiesta (Quando vuoi ritirare le rose?), e restituisce una risposta con le seguenti intestazioni:

```
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=
```

I valori dell'intestazione forniscono le informazioni seguenti:

- `x-amz-lex-slots`: la versione codificata base64 di slot e valori:

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

- `x-amz-lex-session-attributes`: la versione codificata base64 degli attributi della sessione ({})

Il client riproduce l'audio nel corpo della risposta.

3. L'utente pronuncia: domani

   a. Il client (console) invia la seguente richiesta PostContent ad 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")
```

Il corpo della richiesta è il flusso audio dell'input utente ("tomorrow"). `sessionAttributes` rimane vuoto.

b. Amazon Lex interpreta il flusso di input nel contesto dell'intento corrente (ricorda di aver chiesto a questo utente informazioni relative allo`PickupDate` slot). Amazon Lex aggiorna il valore dello slot (`PickupDate`) per l'intento corrente. Quindi sceglie un altro slot per recuperare il valore di (`PickupTime`) e uno dei prompt di ottenimento del valore (Quando vuoi ritirare le rose il 18/03/2018?), infine restituisce una risposta con le seguenti intestazioni:

```
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
```

I valori dell'intestazione forniscono le informazioni seguenti:

- `x-amz-lex-slots`: la versione codificata base64 di slot e valori:

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

- `x-amz-lex-session-attributes`: la versione codificata base64 degli attributi della sessione ({})

Il client riproduce l'audio nel corpo della risposta.

4. L'utente pronuncia: 18

   a. Il client (console) invia la seguente richiesta PostContent ad 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")
```

   Il corpo della richiesta è il flusso audio dell'input utente ("6 pm"). `sessionAttributes` rimane vuoto.

   b. Amazon Lex interpreta il flusso di input nel contesto dell'intento corrente (ricorda di aver chiesto a questo utente informazioni relative allo`PickupTime` slot). Innanzitutto aggiorna il valore dello slot dell'intento corrente.

   Ora Amazon Lex rileva di avere informazioni per tutti gli slot. Tuttavia, poiché l'intento `OrderFlowers` è configurato con un messaggio di conferma, Pertanto, Amazon Lex necessita di una conferma esplicita da parte dell'utente prima di poter procedere a soddisfare l'intento. e pertanto invia una risposta con le intestazioni seguenti per richiedere una conferma prima di ordinare i fiori:

```
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
```

   I valori dell'intestazione forniscono le informazioni seguenti:

- `x-amz-lex-slots`: la versione codificata base64 di slot e valori:

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

- `x-amz-lex-session-attributes`: la versione codificata base64 degli attributi della sessione ({})

Il client riproduce l'audio nel corpo della risposta.

5. L'utente pronuncia: Sì

   a. Il client (console) invia la seguente richiesta PostContent ad 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")
```

   Il corpo della richiesta è il flusso audio dell'input utente ("Yes"). `sessionAttributes` rimane vuoto.

   b. Amazon Lex interpreta il flusso di input e capisce che l'utente desidera procedere con l'ordine. L'intento `OrderFlowers` è configurato con `ReturnIntent` come attività di l'attività di adempimento. Questo indica ad Amazon Lex di restituire tutti i dati relativi alle intenzioni al client. Amazon Lex restituisce una risposta con quanto segue:

```
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
```

L'intestazione della risposta `x-amz-lex-dialog-state` è impostata su `ReadyForFulfillment`. Quindi il client può realizzare l'intento.

6. A questo punto, esegui nuovamente il test del bot. Per stabilire un nuovo contesto (utente), scegli il collegamento Clear (Cancella) nella console. Fornisci i dati per l'intento `OrderFlowers` e includi alcuni dati non validi. Ad esempio:

   - Gelsomino come tipo di fiore (non è uno dei tipi di fiore supportati)
   - Ieri come giorno in cui si desidera ritirare i fiori

   Nota che il bot accetta questi valori perché non disponi di alcun codice per inizializzare e convalidare i dati utente. Nella sezione successiva, aggiungi una funzione Lambda per eseguire questa operazione. Si noti quanto segue in relazione alla funzione Lambda:

   - Convalida i dati dello slot dopo ogni input utente. Soddisfa l'intento alla fine. Ciò significa che il bot elabora l'ordine di fiori e restituisce un messaggio all'utente invece che restituire semplicemente i dati dello slot al client. Per ulteriori informazioni, consulta Utilizzo delle funzioni Lambda.
   - Inoltre, imposta gli attributi di sessione. Per ulteriori informazioni sugli attributi di sessione, consulta PostText.

     Al termine della sezione Nozioni di base, potrai eseguire gli esercizi aggiuntivi (Esempi aggiuntivi: creazione di bot Amazon Lex). Prenota viaggio utilizza gli attributi di sessione per condividere le informazioni tra intenti per impegnarsi in una conversazione dinamica con l'utente.

Fase successiva

Fase 3: creazione di una funzione Lambda

Fase 2b (facoltativo): Revisione dei dettagli del flusso di informazioni digitate (console)

Questa sezione illustra il flusso di informazioni tra il client e Amazon Lex in cui il client utilizza l'API `PostText` per inviare le richieste. Per ulteriori informazioni, consulta PostText.

1. L'utente digita: Vorrei ordinare dei fiori

   a. Il client (console) invia la seguente richiesta PostText ad 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": {}
}
```

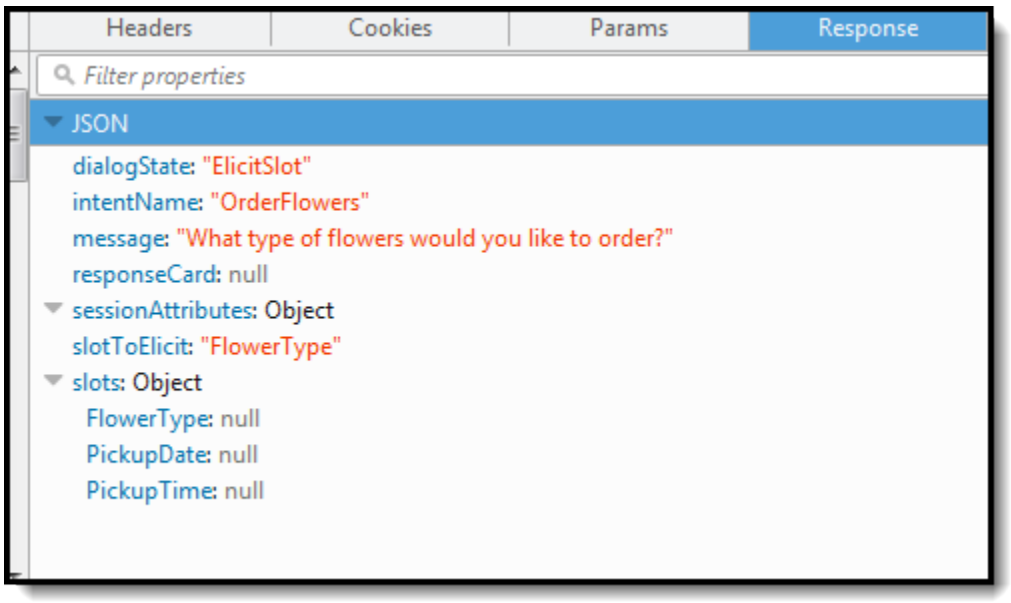
Sia l'URI della richiesta che il corpo forniscono informazioni ad Amazon Lex:

- URI di richiesta: fornisce il nome del bot (`OrderFlowers`), l'alias del bot ($LATEST) e il nome utente (una stringa casuale che identifica l'utente). Il codice `text` finale indica che si tratta di una richiesta API `PostText` (non `PostContent`).

- Corpo della richiesta: include l'input utente (`inputText`) e il campo `sessionAttributes` vuoto. Quando il client effettua la prima richiesta non vi sono attributi della sessione. La funzione Lambda li avvia in un secondo momento.

b. Da lì`inputText`, Amazon Lex rileva l'intento (`OrderFlowers`). Questo intento non prevede alcun hook di codice (ovvero le funzioni Lambda) per l'inizializzazione e la convalida dell'input o dell'adempimento dell'utente.

   Amazon Lex sceglie uno degli slot dell'intento (`FlowerType`) per ottenere il valore. Inoltre, seleziona uno dei prompt per l'ottenimento del valore dello slot (tutti parte della configurazione dell'intento) e invia la seguente risposta al client. La console mostra il messaggio nella risposta all'utente.



Il client visualizza il messaggio nella risposta.

2. L'utente digita: rose

   a. Il client (console) invia la seguente richiesta PostText ad Amazon Lex:

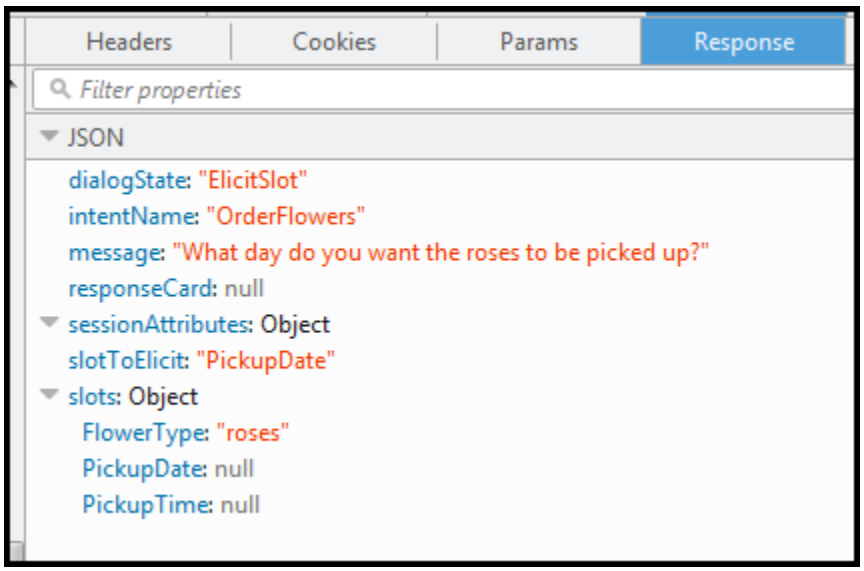
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   `inputText` nel corpo della richiesta fornisce l'input utente. `sessionAttributes` rimane vuoto.

   b. Amazon Lex lo interpreta innanzitutto`inputText` nel contesto dell'intenzione corrente: il servizio ricorda di aver chiesto all'utente specifico informazioni sullo`FlowerType` slot. Amazon Lex aggiorna innanzitutto il valore dello slot per l'intento corrente e ne sceglie un altro (`PickupDate`) insieme a uno dei suoi messaggi rapidi: in che giorno vuoi che vengano ritirate le rose? — per lo slot.

   Quindi, Amazon Lex restituisce la risposta:



Il client visualizza il messaggio nella risposta.

3. L'utente digita: domani

   a. Il client (console) invia la seguente richiesta PostText ad Amazon Lex:

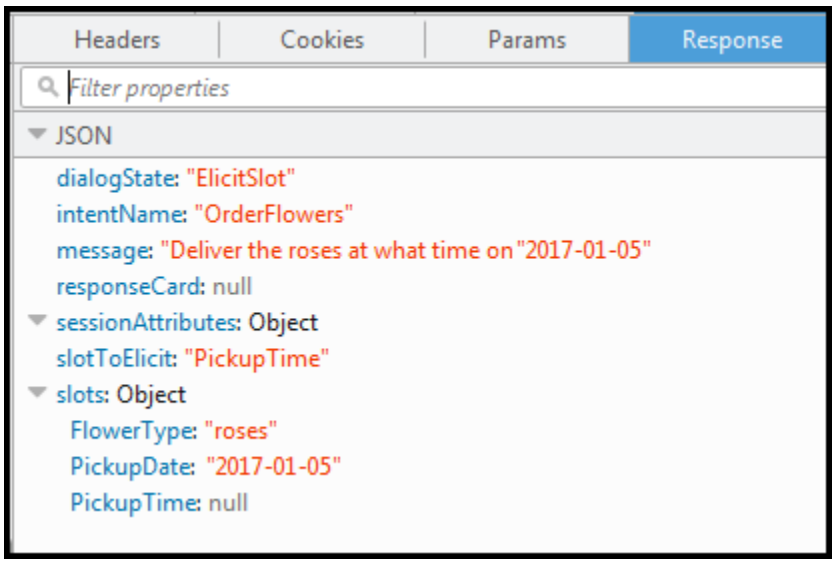
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   `inputText` nel corpo della richiesta fornisce l'input utente. `sessionAttributes` rimane vuoto.

   b. Amazon Lex lo interpreta innanzituttoinputText nel contesto dell'intenzione corrente: il servizio ricorda di aver chiesto all'utente specifico informazioni sulloPickupDate slot. Amazon Lex aggiorna il valore dello slot (`PickupDate`) per l'intento corrente. e sceglie un altro slot per ottenere il valore di (`PickupTime`). Restituisce una delle richieste di promozione del valore: consegnare le rose a che ora il 05/01/2017? — al cliente.

   Amazon Lex restituisce quindi la risposta:



Il client visualizza il messaggio nella risposta.

4. L'utente digita: 18:00

   a. Il client (console) invia la seguente richiesta PostText ad Amazon Lex:

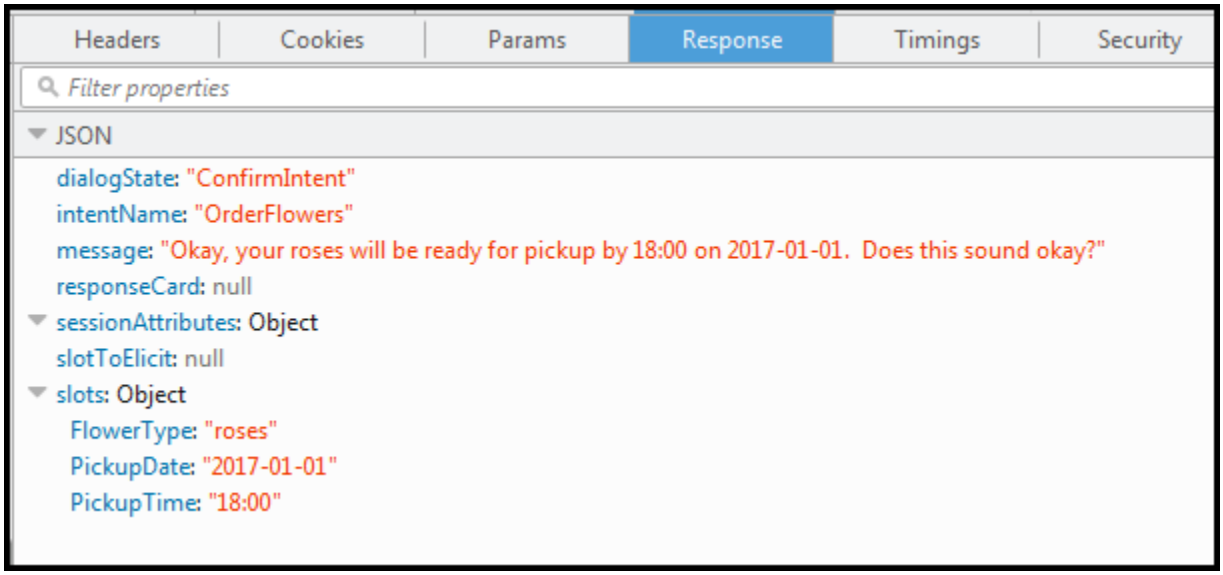
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   `inputText` nel corpo della richiesta fornisce l'input utente. `sessionAttributes` rimane vuoto.

   b. Amazon Lex lo interpreta innanzitutto`inputText` nel contesto dell'intenzione corrente: il servizio ricorda di aver chiesto all'utente specifico informazioni sullo`PickupTime` slot. Amazon Lex aggiorna innanzitutto il valore dello slot per l'intento corrente. Ora Amazon Lex rileva di avere informazioni per tutti gli slot.

   Tuttavia, poiché l'intento `OrderFlowers` è configurato con un messaggio di conferma, Pertanto, Amazon Lex necessita di una conferma esplicita da parte dell'utente prima di poter procedere a soddisfare l'intento. Amazon Lex invia il seguente messaggio al cliente richiedendo conferma prima di ordinare i fiori:



Il client visualizza il messaggio nella risposta.

5. L'utente digita: Sì

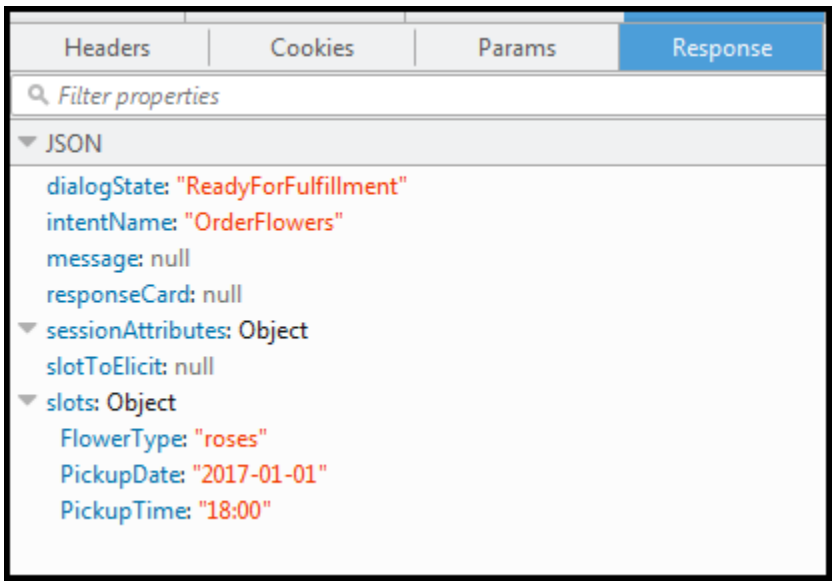
   a. Il client (console) invia la seguente richiesta PostText ad Amazon Lex:

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

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

   `inputText` nel corpo della richiesta fornisce l'input utente. `sessionAttributes` rimane vuoto.

   b. Amazon Lex lo interpreta`inputText` nel contesto della conferma dell'intento attuale. e comprende che l'utente desidera procedere con l'ordine. L'`OrderFlowers`intento è configurato`ReturnIntent` come attività di adempimento (non esiste una funzione Lambda per soddisfare l'intento). Pertanto, Amazon Lex restituisce al client i seguenti dati di slot.



Amazon Lex ha impostato`dialogState` su`ReadyForFulfillment`. Quindi il client può realizzare l'intento.

6. A questo punto, esegui nuovamente il test del bot. Per farlo, devi scegliere il collegamento Clear (Cancella) nella console per stabilire un nuovo contesto (utente). Ora, quando fornisci i dati per l'intento relativo all'ordine di fiori, prova a fornire dei dati non validi. Ad esempio:

   - Gelsomino come tipo di fiore (non è uno dei tipi di fiore supportati).
   - Ieri come giorno in cui si desidera ritirare i fiori.

   Nota che il bot accetta questi valori perché non disponi di alcun codice per inizializzare/convalidare i dati utente. Nella sezione successiva, aggiungi una funzione Lambda per eseguire questa operazione. Si noti quanto segue in relazione alla funzione Lambda:

   - La funzione Lambda convalida i dati degli slot dopo ogni input dell'utente. Soddisfa l'intento alla fine. Ciò significa che il bot elabora l'ordine di fiori e restituisce un messaggio all'utente invece che restituire semplicemente i dati dello slot al client. Per ulteriori informazioni, consulta Utilizzo delle funzioni Lambda.
   - La funzione Lambda imposta anche gli attributi della sessione. Per ulteriori informazioni sugli attributi di sessione, consulta PostText.

     Al termine della sezione Nozioni di base, potrai eseguire gli esercizi aggiuntivi (Esempi aggiuntivi: creazione di bot Amazon Lex). Prenota viaggio utilizza gli attributi di sessione per

condividere le informazioni tra intenti per impegnarsi in una conversazione dinamica con l'utente.

Fase successiva

[Fase 3: creazione di una funzione Lambda](#)

Fase 3: creazione di una funzione Lambda

Crea una funzione Lambda (utilizzando il lex-order-flowers-pythonblueprint) ed esegui una chiamata di test utilizzando i dati degli eventi di esempio nellaAWS Lambda console.

Ritorni alla console Amazon Lex e aggiungi la funzione Lambda come codice hook per soddisfare l'`OrderFlowers`intento`OrderFlowersBot` che hai creato nella sezione precedente.

Per creare la funzione Lambda (console)

1. Accedere alla AWS Management Console e aprire la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegliere Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione) scegliere Use a blueprint (Usa un piano). Digita **`lex-`** nella casella di testo di filtro, quindi premi `Enter` per trovare il piano e scegli il piano `lex-order-flowers-python`.

   I blueprint delle funzioni Lambda sono forniti sia in Node.js che in Python. Per questo esercizio, utilizza il piano basato su Python.
4. Nella pagina Basic information (Informazioni di base), procedere come segue:
   - Digitare il nome di una funzione Lambda (`OrderFlowersCodeHook`).
   - Per il ruolo di esecuzione, scegliere Create a new role with basic Lambda permissions (Crea un nuovo ruolo con le autorizzazioni Lambda di base).
   - Non modificare gli altri valori di default.
5. Scegli Create function (Crea funzione).
6. Se utilizzi una lingua diversa dall'inglese (USA) (en-US), aggiorna i nomi degli intenti come descritto inAggiornamento di un blueprint per una specifica impostazione locale.
7. Test della funzione Lambda.

a. Seleziona Select a test events (Seleziona eventi test), Configure test event (Configura evento test).

b. Seleziona Amazon Lex Order Flowers nell'elenco Event template (Modello eventi). Questo evento di esempio corrisponde al modello di richiesta/risposta di Amazon Lex (vediUtilizzo delle funzioni Lambda). Assegna un nome all'evento di test (`LexOrderFlowersTest`).

c. Seleziona Create (Crea).

d. Scegli Test per testare l'hook del codice.

e. Eseguire le operazioni di verifica del corretto funzionamento della funzione Lambda. La risposta in questo caso corrisponde al modello di risposta di Amazon Lex.

Fase successiva

Passaggio 4: aggiungere la funzione Lambda come Code Hook (console)

Passaggio 4: aggiungere la funzione Lambda come Code Hook (console)

In questa sezione, si aggiorna la configurazione dell' OrderFlowersintento di utilizzare la funzione Lambda come segue:

- Utilizzate innanzitutto la funzione Lambda come codice hook per realizzare l'`OrderFlowers`intento. Testi il bot e verifichi di aver ricevuto un messaggio di evasione dalla funzione Lambda. Amazon Lex richiama la funzione Lambda solo dopo aver fornito i dati per tutti gli slot necessari per ordinare i fiori.
- Configura la stessa funzione Lambda come hook di codice per eseguire l'inizializzazione e la convalida. Si verifica e si verifica che la funzione Lambda esegua la convalida (quando si forniscono i dati dello slot).

Per aggiungere una funzione Lambda come codice hook (console)

1. Nella console Amazon Lex, seleziona il OrderFlowersbot. La console mostra l'OrderFlowersintento. Accertati che la versione dell'intento sia impostata su `$LATEST` poiché questa è l'unica versione che è possibile modificare.
2. Aggiungi la funzione Lambda come hook del codice di adempimento e testala.

a. Nell'Editor, scegli AWS Lambdala funzione Fulfillment e seleziona la funzione Lambda creata nel passaggio precedente (`OrderFlowersCodeHook`). Scegli OK per autorizzare Amazon Lambda.

   Stai configurando questa funzione Lambda come un hook di codice per soddisfare l'intento. Amazon Lex richiama questa funzione solo dopo aver ricevuto dall'utente tutti i dati di slot necessari per soddisfare l'intento.

b. Specifica un messaggio di saluto (Goodbye message).

c. Seleziona Build (Crea).

d. Esegui il test del bot utilizzando la conversazione precedente.

L'ultima dichiarazione «Grazie, il tuo ordine per le rose...» è una risposta della funzione Lambda che hai configurato come hook di codice. Nella sezione precedente, non esisteva una funzione Lambda. Ora stai usando una funzione Lambda per soddisfare effettivamente l'`OrderFlowers`intento.

3. Aggiungi la funzione Lambda come hook di codice di inizializzazione e convalida ed esegui il test.

   Il codice di funzione Lambda di esempio che stai utilizzando può sia eseguire la convalida che l'evasione degli input dell'utente. L'evento di input ricevuto dalla funzione Lambda ha un campo (`invocationSource`) che il codice utilizza per determinare quale parte del codice eseguire. Per ulteriori informazioni, consulta Formato di evento di input e di risposta della funzione Lambda.

   a. Seleziona la versione $LATEST dell'intento `OrderFlowers`. Questa è l'unica versione che puoi aggiornare.

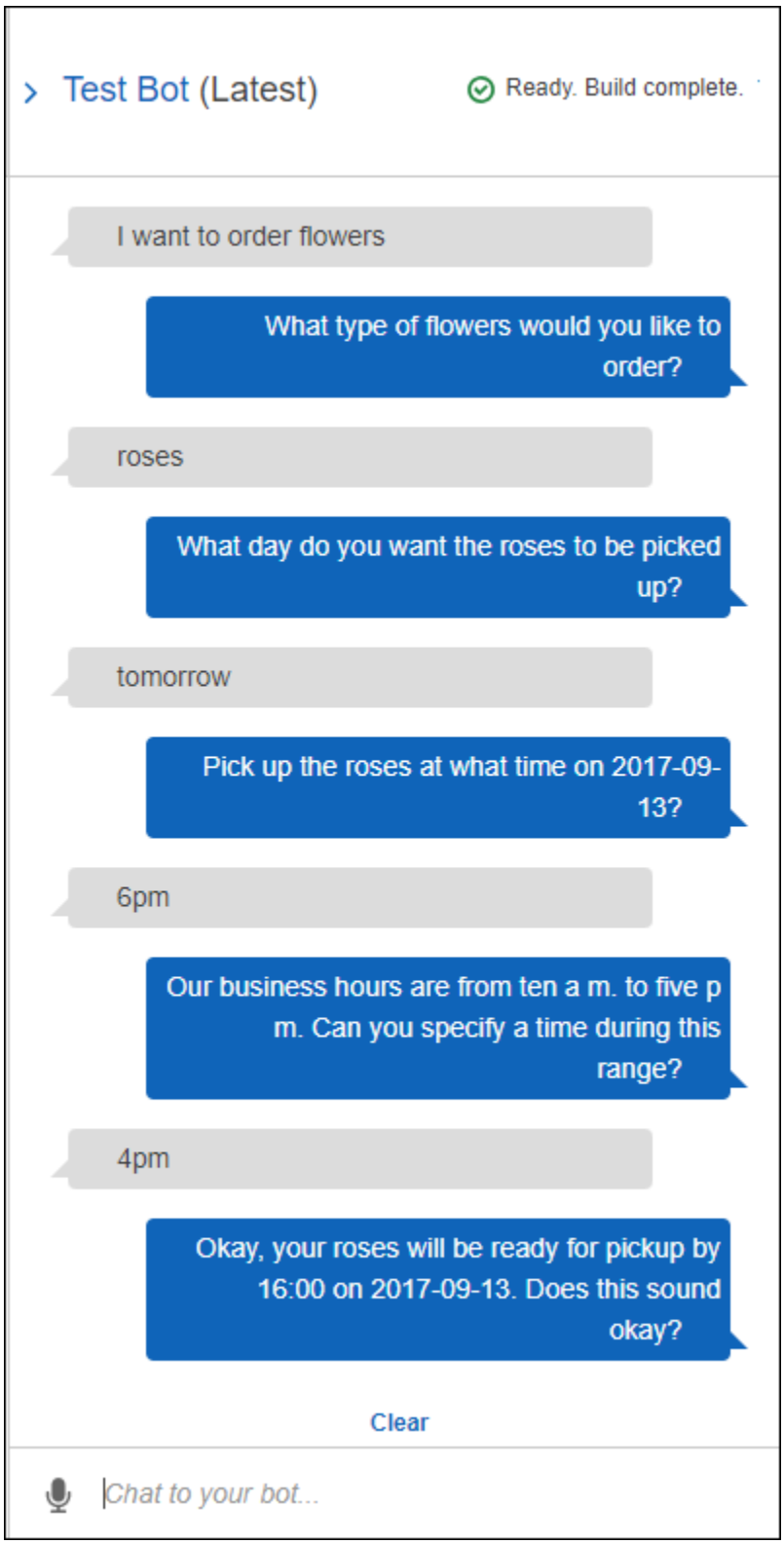
   b. Nell'editor, scegli Initialization and validation (Inizializzazione e convalida) nella sezione Options (Opzioni).

   c. Ancora una volta, seleziona la stessa funzione Lambda.

   d. Seleziona Build (Crea).

   e. Esegui il test del bot.

      Ora sei pronto per conversare con Amazon Lex come nell'immagine seguente. Per testare la parte di convalida, scegli l'ora alle 18:00 e la funzione Lambda restituisce una risposta («Il nostro orario lavorativo è dalle 10:00 alle 17:00») e ti chiede di nuovo. Dopo aver fornito tutti i dati validi dello slot, la funzione Lambda soddisfa l'ordine.



Fase successiva

Fase 5 (facoltativo): Revisione dei dettagli del flusso di informazioni (console)

Fase 5 (facoltativo): Revisione dei dettagli del flusso di informazioni (console)

Questa sezione spiega il flusso di informazioni tra il client e Amazon Lex per ogni input dell'utente, inclusa l'integrazione della funzione Lambda.

> **Note**
>
> La sezione presuppone che il client invii richieste ad Amazon Lex utilizzando l'API`PostText` di runtime e mostri i dettagli della richiesta e della risposta di conseguenza. Per un esempio del flusso di informazioni tra il client e Amazon Lex in cui il client utilizza l'`PostContent`API, consultaFase 2a (facoltativo): Revisione dei dettagli del flusso di informazioni parlate (console) .

Per ulteriori informazioni sull'API di runtime `PostText` e ulteriori dettagli sulle richieste e le risposte mostrate nelle fasi seguenti, consulta l'argomento PostText.

1. Utente: Vorrei ordinare dei fiori.

   a. Il client (console) invia la seguente richiesta PostText ad 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": {}
   }
   ```

   Sia l'URI della richiesta che il corpo forniscono informazioni ad Amazon Lex:

   - URI di richiesta: fornisce il nome del bot (`OrderFlowers`), l'alias del bot ($LATEST) e il nome utente (una stringa casuale che identifica l'utente). Il codice `text` finale indica che si tratta di una richiesta API `PostText` (non `PostContent`).
   - Corpo della richiesta: include l'input utente (`inputText`) e il campo `sessionAttributes` vuoto. Quando il client effettua la prima richiesta non vi sono attributi della sessione. La funzione Lambda li avvia in un secondo momento.

b. Da lì`inputText`, Amazon Lex rileva l'intento (`OrderFlowers`). Questo intento è configurato con una funzione Lambda come hook di codice per l'inizializzazione e la convalida dei dati utente. Pertanto, Amazon Lex richiama quella funzione Lambda passando le seguenti informazioni come dati dell'evento:

```
{
    "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"
    }
}
```

Per ulteriori informazioni, consulta Formato dell'evento di input.

Oltre alle informazioni inviate dal cliente, Amazon Lex include anche i seguenti dati aggiuntivi:

- `messageVersion`— Attualmente Amazon Lex supporta solo la versione 1.0.
- `invocationSource`— Indica lo scopo dell'invocazione della funzione Lambda. In questo caso, lo scopo è eseguire l'inizializzazione e la convalida dei dati utente. Al momento, Amazon Lex sa che l'utente non ha fornito tutti i dati dello slot necessari per soddisfare l'intento.
- Informazioni su `currentIntent` con tutti i valori dello slot impostati su null.

c. Al momento, tutti i valori dello slot sono nulli. Non c'è nulla da convalidare per la funzione Lambda. La funzione Lambda restituisce la seguente risposta ad Amazon Lex:

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

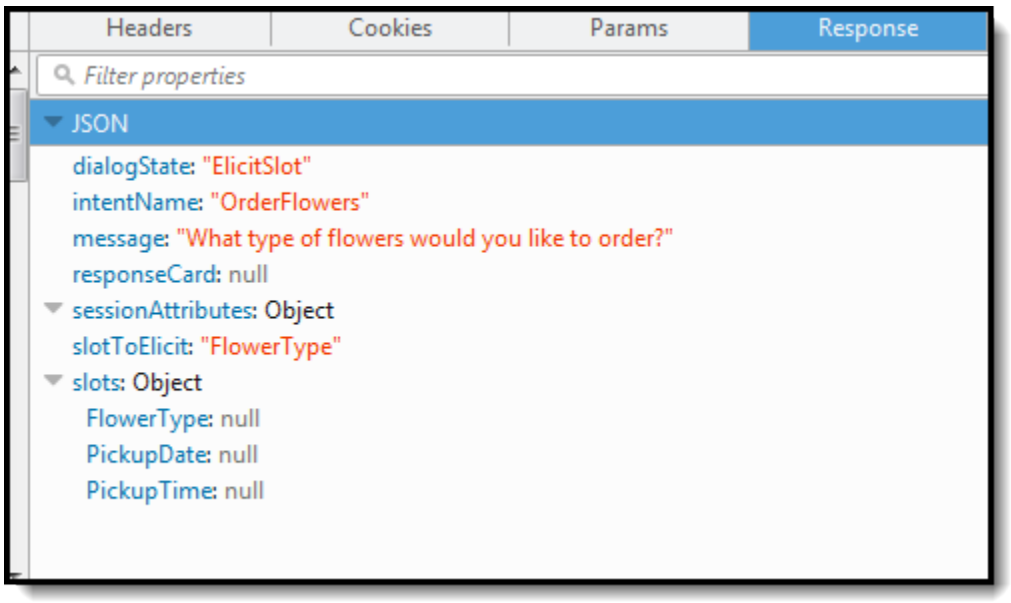
Per informazioni sul formato della risposta, consulta l'argomento Formato della risposta.

Tieni presente quanto segue:

- `dialogAction.type`— Impostando questo valore su`Delegate`, la funzione Lambda delega ad Amazon Lex la responsabilità di decidere la prossima linea d'azione.

> **Note**
>
> Se la funzione Lambda rileva qualcosa nella convalida dei dati utente, indica ad Amazon Lex cosa fare dopo, come mostrato nei passaggi successivi.

d. Secondo il`dialogAction.type`, Amazon Lex decide la prossima linea d'azione. Poiché nessuno slot è popolato, decide di ottenere il valore dello slot `FlowerType` Inoltre, seleziona uno dei messaggi di richiesta per l'ottenimento del valore ("What type of flowers would you like to order?") dello slot e invia la seguente risposta al client:



Il client visualizza il messaggio nella risposta.

2. Utente: rosa

   a. Il client invia la seguentePostText richiesta ad Amazon Lex:

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

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

   Nel corpo della richiesta, `inputText` fornisce l'input utente. `sessionAttributes` rimane vuoto.

   b. Amazon Lex lo interpreta innanzituttoinputText nel contesto dell'intento attuale. Il servizio ricorda che aveva richiesto all'utente specifico le informazioni sullo slot `FlowerType` Aggiorna il valore dello slot nell'intento corrente e richiama la funzione Lambda con i seguenti dati sugli eventi:

```
{
    "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"
    }
}
```

Tieni presente quanto segue:

- `invocationSource`: continua a essere `DialogCodeHook` (stiamo semplicemente convalidando i dati utente).
- `currentIntent.slots`— Amazon Lex ha aggiornato lo`FlowerType` slot alle rose.

c. In base al`invocationSource` valore di`DialogCodeHook`, la funzione Lambda esegue la convalida dei dati utente. Riconosce`roses` come valore di slot valido (e viene impostato`Price` come attributo di sessione) e restituisce la seguente risposta ad Amazon Lex.

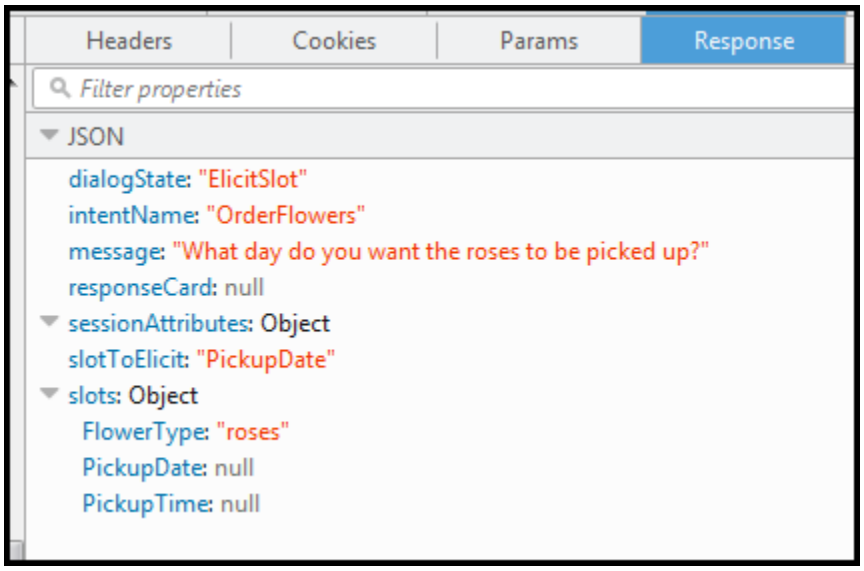
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": null
        }
    }
```

```
}
```

Tieni presente quanto segue:

- `sessionAttributes`— La funzione Lambda è stata aggiunta`Price` (delle rose) come attributo di sessione.
- `dialogAction.type`: è impostato su `Delegate`. I dati utente erano validi, quindi la funzione Lambda indirizza Amazon Lex a scegliere la prossima linea d'azione.

d. Secondo il`dialogAction.type`, Amazon Lex sceglie la prossima linea d'azione. Amazon Lex sa di aver bisogno di più dati sugli slot, quindi sceglie il successivo slot non riempito (`PickupDate`) con la massima priorità in base alla configurazione dell'intento. Amazon Lex seleziona uno dei messaggi di richiesta di promozione del valore: «In che giorno vuoi che vengano ritirate le rose?» —per questo slot in base alla configurazione dell'intento, quindi invia la seguente risposta al client:



Il client visualizza semplicemente il messaggio nella risposta: "What day do you want the roses to be picked up?"

3. Utente: domani

a. Il client invia la seguentePostText richiesta ad Amazon Lex:

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

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

{
    "inputText": "tomorrow",
    "sessionAttributes": {
        "Price": "25"
    }
}
```

Nel corpo della richiesta, `inputText` fornisce l'input utente e il client trasferisce gli attributi di sessione al servizio.

b. Amazon Lex ricorda il contesto, ossia che stava raccogliendo dati per lo`PickupDate` slot. In questo contesto, sa che il valore `inputText` è per lo slot `PickupDate`. Amazon Lex richiama quindi la funzione Lambda inviando il seguente evento:

```
{
    "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 ha aggiornato il`currentIntent.slots` impostando il`PickupDate` valore. Si noti inoltre che il servizio passa il`sessionAttributes` file così com'è alla funzione Lambda.

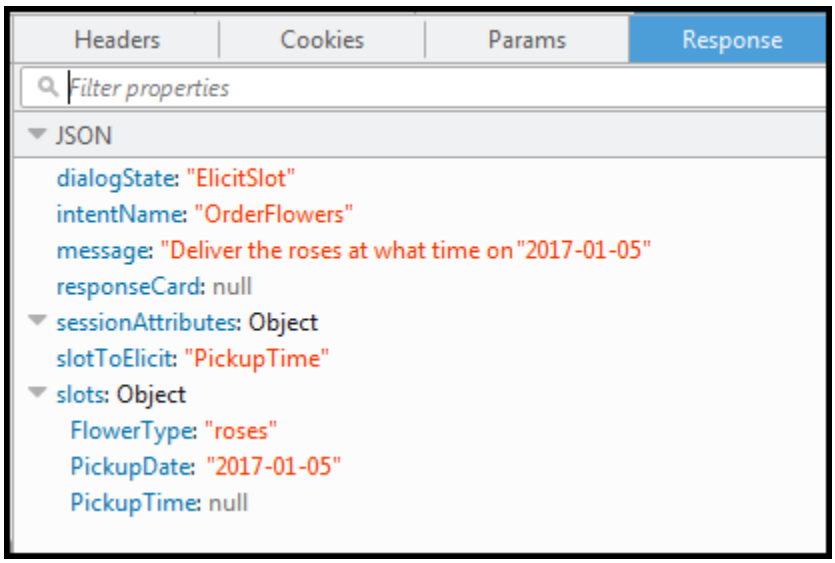
c. In base al`invocationSource` valore di`DialogCodeHook`, la funzione Lambda esegue la convalida dei dati utente. Riconosce che il valore`PickupDate` dello slot è valido e restituisce la seguente risposta ad Amazon Lex:

```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        }
    }
}
```

Tieni presente quanto segue:

- `sessionAttributes`: nessuna modifica.
- `dialogAction.type`: è impostato su `Delegate`. I dati utente erano validi e la funzione Lambda indica ad Amazon Lex di scegliere la prossima linea d'azione.

d. Secondo il`dialogAction.type`, Amazon Lex sceglie la prossima linea d'azione. Amazon Lex sa di aver bisogno di più dati sugli slot, quindi sceglie il successivo slot non riempito (`PickupTime`) con la massima priorità in base alla configurazione dell'intento. Amazon Lex seleziona uno dei messaggi rapidi («Consegnare le rose a che ora il 05/01/2017?») per questo slot in base alla configurazione dell'intento e invia la seguente risposta al client:



Il cliente visualizza il messaggio nella risposta: «Consegnare le rose a che ora il 05/01/2017?»

4. Utente: 16:00

   a. Il client invia la seguentePostText richiesta ad 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"
    }
}
```

   Nel corpo della richiesta, `inputText` fornisce l'input utente. Il client trasferisce `sessionAttributes` nella richiesta.

   b. Amazon Lex comprende il contesto. Comprende inoltre che stava richiedendo i dati dello slot `PickupTime`. In questo contesto, sa che il`inputText` valore è per lo`PickupTime` slot. Amazon Lex richiama quindi la funzione Lambda inviando il seguente evento:

```
{
    "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 ha aggiornato il`currentIntent.slots` impostando il`PickupTime` valore.

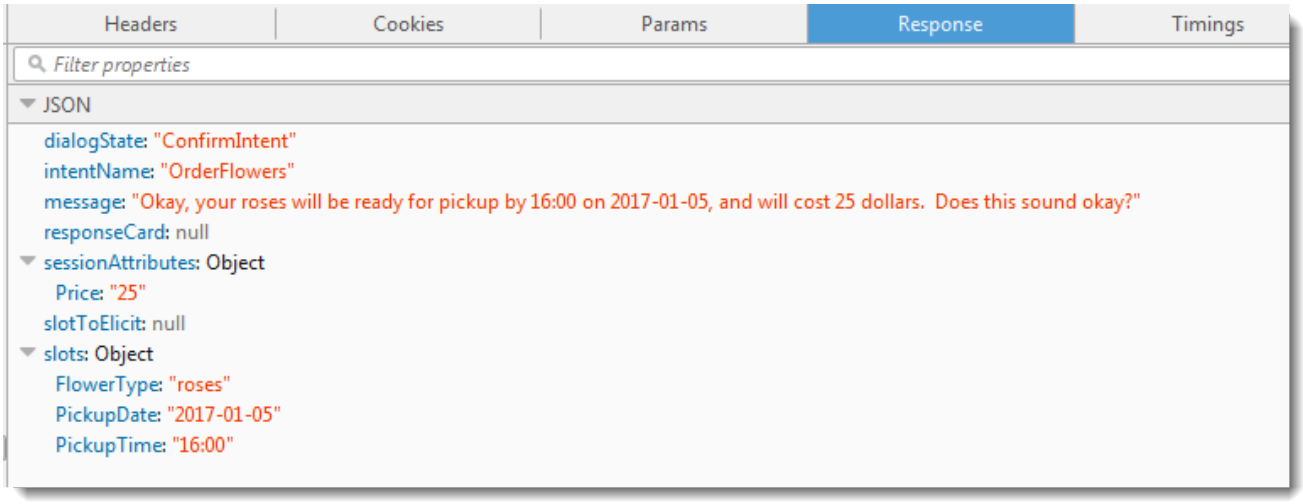
c. In base al`invocationSource` valore di`DialogCodeHook`, la funzione Lambda esegue la convalida dei dati utente. Riconosce che il valore`PickupDate` dello slot è valido e restituisce la seguente risposta ad Amazon Lex.

```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        }
    }
}
```

Tieni presente quanto segue:

- `sessionAttributes`: nessuna modifica dell'attributo di sessione.
- `dialogAction.type`: è impostato su `Delegate`. I dati utente erano validi, quindi la funzione Lambda indirizza Amazon Lex a scegliere la prossima linea d'azione.

d. Al momento Amazon Lex sa di avere tutti i dati relativi agli slot. Tuttavia, poiché questo intento è configurato con un prompt di conferma, Pertanto, Amazon Lex invia la seguente risposta all'utente chiedendo conferma prima di soddisfare l'intento:



Il client visualizza semplicemente il messaggio nella risposta e attende la risposta dell'utente.

5. Utente: Sì

a. Il client invia la seguentePostText richiesta ad 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 lo interpretainputText nel contesto della conferma dell'intento attuale. Amazon Lex comprende che l'utente desidera procedere con l'ordine. Questa volta Amazon Lex richiama la funzione Lambda per soddisfare l'intento inviando il seguente evento,

che imposta il`invocationSource` valore to`FulfillmentCodeHook` nel caso in cui venga inviato alla funzione Lambda. Amazon Lex imposta anche`confirmationStatus` il`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"
    }
}
```

Tieni presente quanto segue:

- `invocationSource`— Questa volta Amazon Lex ha impostato questo valore su`FulfillmentCodeHook`, indirizzando la funzione Lambda a soddisfare l'intento.
- `confirmationStatus`: è impostato su `Confirmed`.

c. Questa volta, la funzione Lambda soddisfa l'`OrderFlowers`intento e restituisce la seguente risposta:

```
{
    "sessionAttributes": {
        "Price": "25"
    },
    "dialogAction": {
```

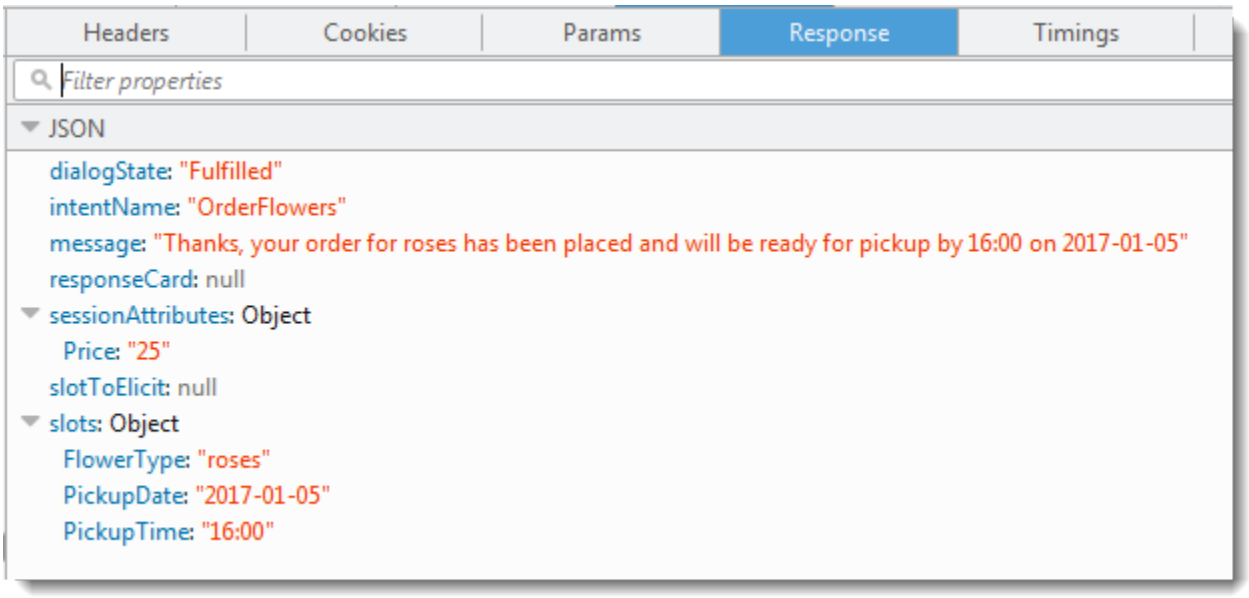
```
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}
```

Tieni presente quanto segue:

- Imposta il valore`dialogAction.type` — La funzione Lambda imposta questo valore su`Close`, indicando ad Amazon Lex di non aspettarsi una risposta da parte dell'utente.
- `dialogAction.fulfillmentState`: è impostato su Fulfilled (Soddisfatto) e include un elemento `message` appropriato da trasmettere all'utente.

d. Amazon Lex lo esamina`fulfillmentState` e invia al client la seguente.

Amazon Lex restituisce quindi quanto segue al cliente:



Nota:

- `dialogState`— Amazon Lex imposta questo valore su`fulfilled`.
- `message`— è lo stesso messaggio fornito dalla funzione Lambda.

Il client visualizza il messaggio.

6. A questo punto, esegui nuovamente il test del bot. Per stabilire un nuovo contesto (utente), scegli il collegamento Clear (Cancella) nella finestra di prova. A questo punto, fornisci dei dati non validi dello slot per l'intento `OrderFlowers`. Questa volta la funzione Lambda esegue la convalida dei dati, reimposta il valore dei dati dello slot non valido su nullo e chiede ad Amazon Lex di richiedere all'utente di immettere dati validi. Ad esempio, prova quanto seguente:

   - Gelsomino come tipo di fiore (non è uno dei tipi di fiore supportati).
   - Ieri come giorno in cui si desidera ritirare i fiori.
   - Dopo aver effettuato l'ordine, inserisci un altro tipo di fiore invece che rispondere "yes" per confermare l'ordine. In risposta, la funzione Lambda aggiorna l'attributo`Price` in the session, mantenendo un totale continuo degli ordini di fiori.

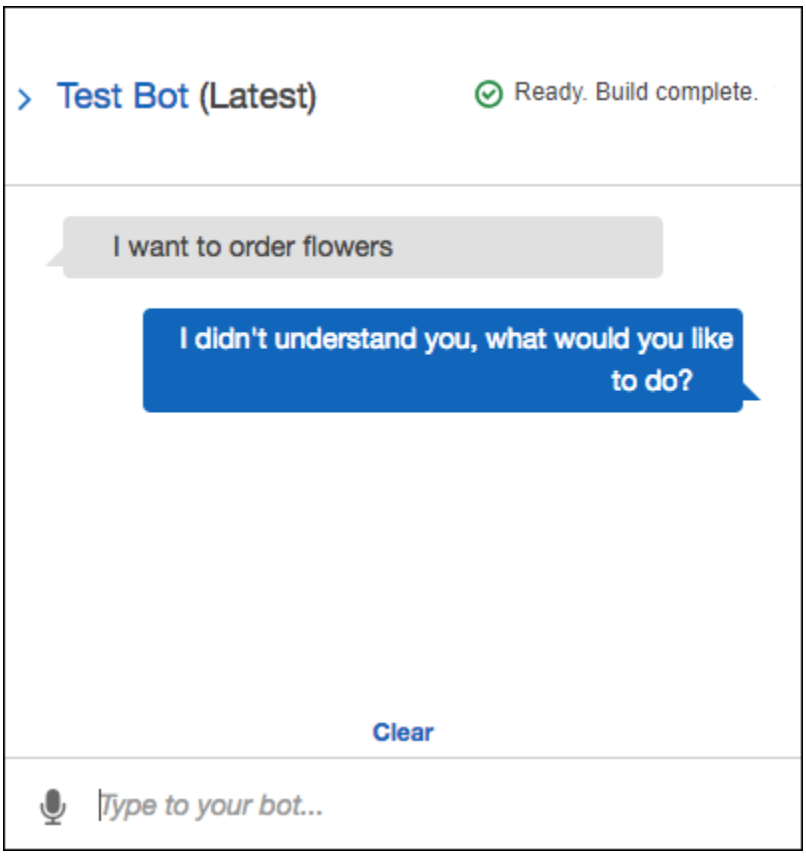
   La funzione Lambda svolge anche l'attività di adempimento.

Fase successiva

[Fase 6: Aggiornamento della configurazione dell'intento per aggiungere un'enunciazione (console)]

Fase 6: Aggiornamento della configurazione dell'intento per aggiungere un'enunciazione (console)

Il bot `OrderFlowers` è configurato con due sole enunciazioni. Ciò fornisce ad Amazon Lex informazioni limitate per creare un modello di apprendimento automatico che riconosca e risponda all'intento dell'utente. Prova a digitare «Voglio ordinare fiori», come nella seguente finestra di prova. Amazon Lex non riconosce il testo e risponde con «Non ti ho capito, cosa vorresti fare?» Puoi migliorare il modello di machine learning aggiungendo altre enunciazioni.



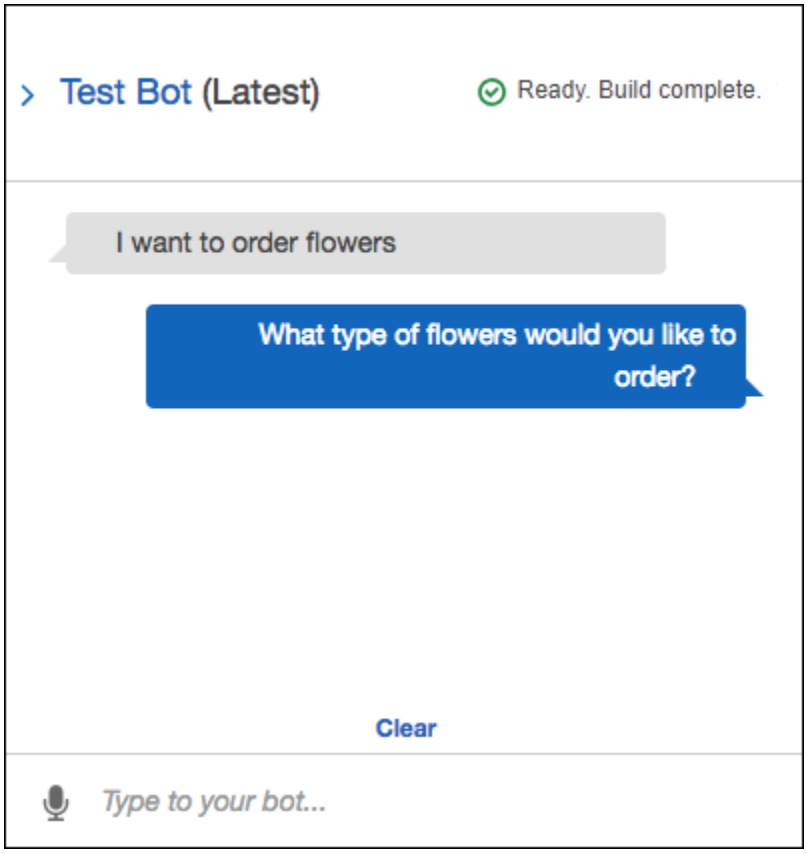
Ogni espressione che aggiungi fornisce ad Amazon Lex ulteriori informazioni su come rispondere ai tuoi utenti. Non è necessario aggiungere un'enunciazione esatta, Amazon Lex generalizza in base agli esempi che fornisci per riconoscere sia le corrispondenze esatte che gli input simili.

Per aggiungere un'enunciazione (console)

1. Aggiungi l'espressione «Voglio fiori» all'intento digitandola nella sezione Espressioni di esempio dell'editor degli intenti, come nell'immagine seguente, e quindi facendo clic sull'icona con il segno più accanto alla nuova espressione.

2. Crea il tuo bot per ottenere la modifica. Seleziona Build (Crea), quindi di nuovo Build (Crea).

3. Esegui il test del bot per verificare che riconosca la nuova enunciazione. Nella finestra di test, come nell'immagine seguente, digita «Voglio ordinare fiori». Amazon Lex riconosce la frase e risponde con «Che tipo di fiori desideri ordinare?».



Fase successiva

Fase 7 (facoltativo): Pulizia (Console)

Fase 7 (facoltativo): Pulizia (Console)

Adesso, elimina le risorse che hai creato e pulisci il tuo account.

Puoi eliminare solo le risorse che non sono in uso. In generale, devi eliminare le risorse nell'ordine seguente:

- Elimina i bot per liberare risorse di intenti.
- Elimina gli intenti per liberare risorse di tipo di slot.
- Elimina i tipi di slot come ultima cosa.

Per pulire il tuo account (console)

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegliere la casella di controllo accanto a OrderFlowers.
3. Per eliminare il bot, scegli Delete (Elimina), quindi scegli Continue (Continua) nella finestra di dialogo di conferma.
4. Nel riquadro a sinistra seleziona Intents (Intenti).
5. Nell'elenco degli intenti, scegli OrderFlowersIntent.
6. Per eliminare l'intento, scegli Delete (Elimina), quindi scegli Continue (Continua) nella finestra di dialogo di conferma.
7. Nel riquadro a sinistra seleziona Slot types (Tipi di slot).
8. Nell'elenco di tipi di slot, scegli Flowers (Fiori).
9. Per eliminare il tipo di slot, scegli Delete (Elimina), quindi scegli Continue (Continua) nella finestra di dialogo di conferma.

Hai rimosso tutte le risorse Amazon Lex che hai creato e hai ripulito il tuo account. Se lo si desidera, è possibile utilizzare la console Lambda per eliminare la funzione Lambda utilizzata in questo esercizio.

Esercizio 2: Creare un Amazon Lex Bot personalizzato

In questo esercizio, utilizzi la console Amazon Lex per creare un bot personalizzato che ordini pizza (`OrderPizzaBot`). Il bot verrà configurato con l'aggiunta di un intento personalizzato (`OrderPizza`), la definizione di tipi di slot personalizzati e degli slot necessari per la realizzazione dell'ordine di una pizza (bordo della pizza, dimensioni e così via). Per ulteriori informazioni sui tipi di slot e sugli slot, consultare Amazon Lex: come funziona: come funziona.

Argomenti

- Fase 1: Creazione di una funzione Lambda
- Fase 2: creazione di un bot
- Fase 3: creazione e test del bot
- Fase 4 (facoltativo): pulizia

Fase 1: Creazione di una funzione Lambda

Innanzitutto, crea una funzione Lambda che soddisfi un ordine di pizza. Si specificherà questa funzione nel bot Amazon Lex, che si creerà nella sezione successiva.

Per creare una funzione Lambda

1. Accedere alla AWS Management Console e aprire la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegli Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), scegliere Author from scratch (Crea da zero).

   Per creare una funzione Lambda si utilizzerà il codice personalizzato fornito in questo esercizio, pertanto è necessario selezionare la funzione di creazione da zero.

   Esegui questa operazione:

   a. Digita il nome (`PizzaOrderProcessor`).
   b. Per il Runtime (Runtime), scegliere la versione più recente di Node.js.
   c. Seleziona Create new role from template(s) (Crea nuovo ruolo da modello/i) come Role (Ruolo).
   d. Immetti un nuovo nome ruolo (`PizzaOrderProcessorRole`).
   e. Scegli Create function (Crea funzione).
4. Nella pagina function (Configura funzione) segui la procedura riportata di seguito.

   Nella sezione Function code (Codice funzione) seleziona Edit code inline (Modifica codice in linea) e quindi copia e incolla il seguente codice funzione Node.js nella finestra.

```
'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. Seleziona Salva.

Test della funzione Lambda mediante dati di esempio sugli eventi

Nella console, testate la funzione Lambda utilizzando i dati degli eventi di esempio per richiamarla manualmente.

Verifica della funzione Lambda:

1. Accedere alla AWS Management Console e aprire la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Nella pagina della funzione Lambda, scegli la funzione Lambda (`PizzaOrderProcessor`).
3. Nella pagina delle funzioni, nell'elenco degli eventi di test, seleziona Configure test events (Configura eventi di test).
4. Nella pagina Configure test event (Configura evento di test) segui la procedura riportata di seguito.

   a. Scegliere Create new test event (Creare nuovo evento di test).

   b. Nel campo Event name (Nome evento) immetti il nome per l'evento (`PizzaOrderProcessorTest`).

   c. Copia il seguente evento Amazon Lex nella finestra.

```
{
  "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. Seleziona Create (Crea).

Dopo la creazione del test in AWS Lambda, torna alla pagina delle funzioni. Scegli Test e Lambda eseguirà la tua funzione Lambda.

Nella casella dei risultati, seleziona Details (Dettagli). Il seguente output verrà visualizzato nel riquadro Execution result (Risultato esecuzione) della console.

```
{
  "sessionAttributes": {},
  "dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled",
    "message": {
      "contentType": "PlainText",
      "content": "Okay, I have ordered your large meat pizza on thin crust."
    }
}
```

Fase successiva

Fase 2: creazione di un bot

Fase 2: creazione di un bot

In questa fase verrà creato un bot per la gestione degli ordini di pizza.

Argomenti

- Creazione del bot
- Creazione di un intento
- Creazione dei tipi di slot
- Configurazione dell'intento
- Configurazione del bot

Creazione del bot

Crea il bot `PizzaOrderingBot` con le informazioni minime necessarie. In seguito, verrà aggiunto un intento, ovvero l'operazione che l'utente desidera eseguire.

Per creare il bot

1. Accedi allaAWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Crea un bot.

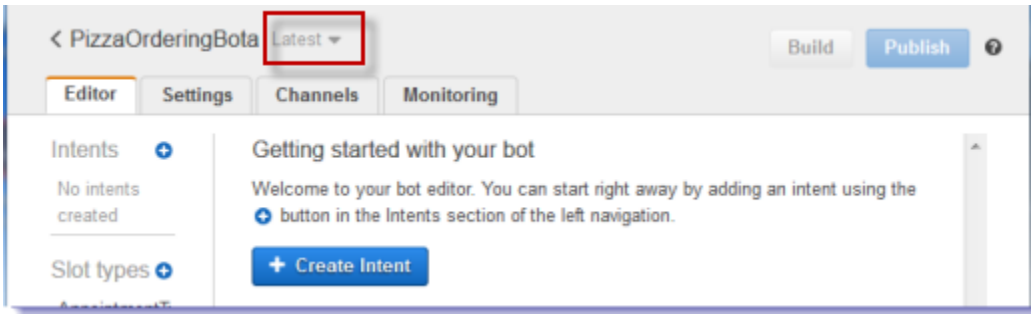
   a. Se si tratta della tua prima creazione di un bot, seleziona Get Started (Inizia). In caso contrario, seleziona Bots, quindi Create (Crea).

   b. Nella pagina Create your Lex bot (Crea bot di Lex), seleziona Custom bot (Bot personalizzato) e fornisci le seguenti informazioni:

      - Nome del bot: PizzaOrderingBot
      - Lingua: scegli la lingua e le impostazioni locali per il tuo bot.
      - Output vocale: Salli
      - Timeout di sessione: 5 minuti
      - COPPA: scegli la risposta appropriata.
      - Memorizzazione delle espressioni dell'utente: scegli la risposta appropriata.

   c. Seleziona Create (Crea).

      La console invia ad Amazon Lex una richiesta per creare un nuovo bot. Amazon Lex imposta la versione bot su`$LATEST`. Dopo aver creato il bot, Amazon Lex mostra la scheda Bot Editor, come nell'immagine seguente:



      - La versione Latest (Più recente) del bot appare accanto al nome del bot nella console. Le nuove risorse Amazon Lex hanno`$LATEST` come versione. Per ulteriori informazioni, consulta Funzione Versioni multiple e alias.
      - Poiché non è stato creato alcun tipo di intento o di slot, nell'elenco non è visualizzato alcun tipo.

- Build (Crea) e Publish (Pubblica) sono attività a livello di bot. Dopo aver configurato l'intero bot, potrai scoprire ulteriori informazioni su queste attività.

Fase successiva

Creazione di un intento

Creazione di un intento

A questo punto, con le informazioni minime necessarie, verrà creato l'intento `OrderPizza`, ovvero l'operazione che l'utente desidera eseguire. Verranno aggiunti i tipi di slot per l'intento e quindi configurato l'intento in un momento successivo.

Per creare un intento

1. Nella console Amazon Lex, scegli il segno più (+) accanto a Intents, quindi scegli Crea nuovo intento.
2. Nella finestra di dialogo Create intent (Crea intento) digita il nome dell'intento (`OrderPizza`) e quindi seleziona Add (Aggiungi).

La console invia una richiesta ad Amazon Lex per creare l'`OrderPizza`intento. In questo esempio si creano slot per l'intento dopo aver creato i tipi di slot.

Fase successiva

Creazione dei tipi di slot

Creazione dei tipi di slot

A questo punto è necessario creare i tipi di slot o i valori dei parametri utilizzati dall'intento `OrderPizza`.

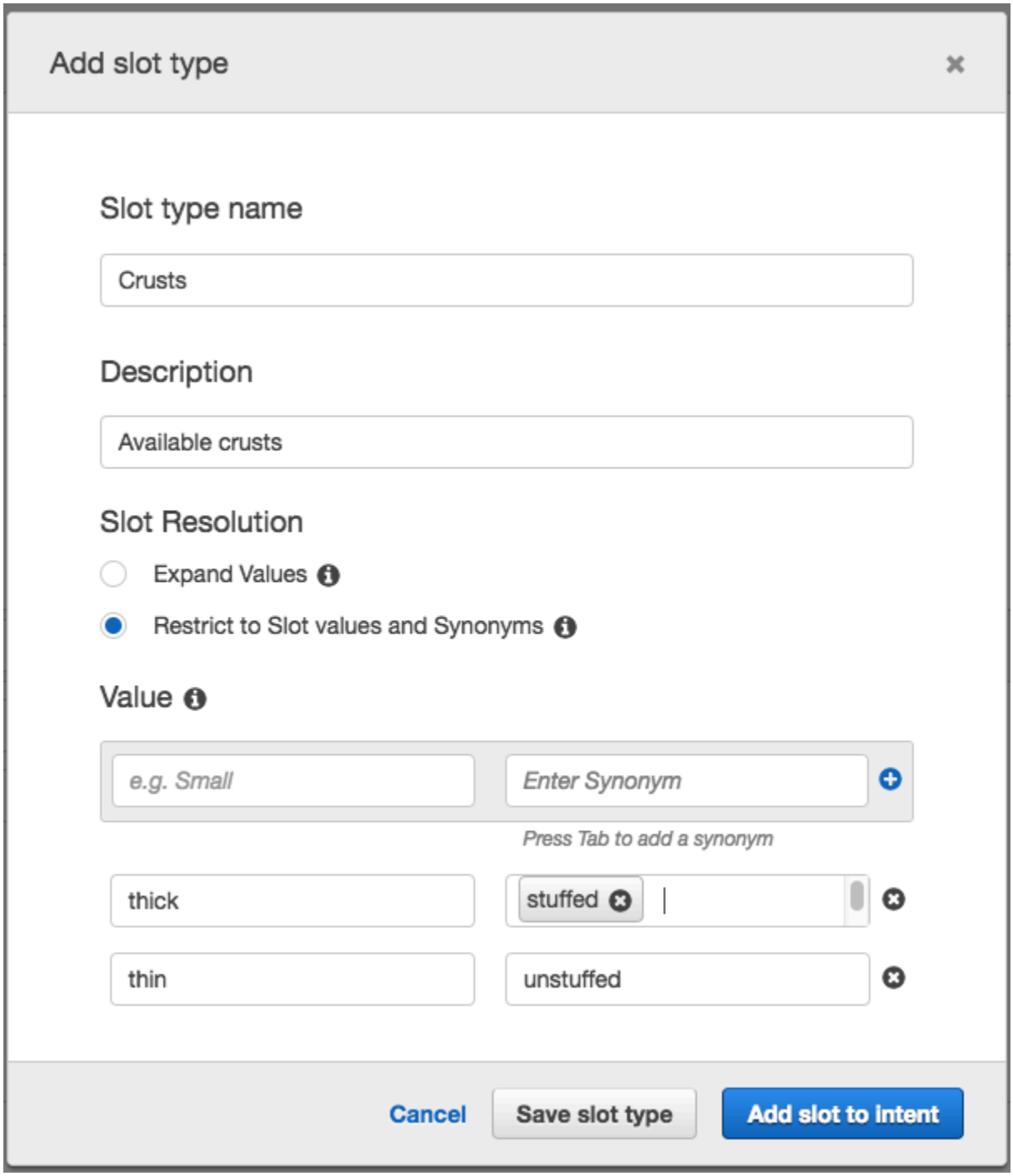
Per creare i tipi di slot

1. Nel menu a sinistra, seleziona il segno più (+) accanto a Slot types (Tipi di slot).
2. Nella finestra di dialogo Add slot type (Aggiungi tipo di slot) aggiungi i seguenti elementi:

   - Slot type name (Nome tipo di slot): Bordi
   - Description (Descrizione): Bordi disponibili

- Seleziona l'opzione Restrict to Slot values and Synonyms (Limita a valori slot e sinonimi).
- Valore: tipo**`thick`**. Premi il tasto tab e nel campo Sinonimo digita**`stuffed`**. Seleziona il segno più (+). Digita **`thin`** e quindi scegli di nuovo il segno più (+).

La finestra di dialogo dovrebbe essere simile all'immagine sotto elencata:



3. Seleziona Add slot to intent (Aggiungi slot a intento).

4. Nella pagina Intent (Intento) seleziona Required (Obbligatorio). Modifica il nome dello slot da **`slotOne`** a **`crust`**. Cambia il messaggio di richiesta in **`What kind of crust would you like?`**
5. Ripeti Step 1 a Step 4 utilizzando i valori della seguente tabella:

Nome	Descrizione	Valori	Nome slot	Prompt
Dimensioni	Dimensioni disponibili	small, medium, large	formato	Quali dimension i per la pizza?
PizzaKind	Pizze disponibili	alle verdure, ai formaggi	pizzaKind	Desideri una pizza alle verdure o ai formaggi?

Fase successiva

Configurazione dell'intento

Configurazione dell'intento

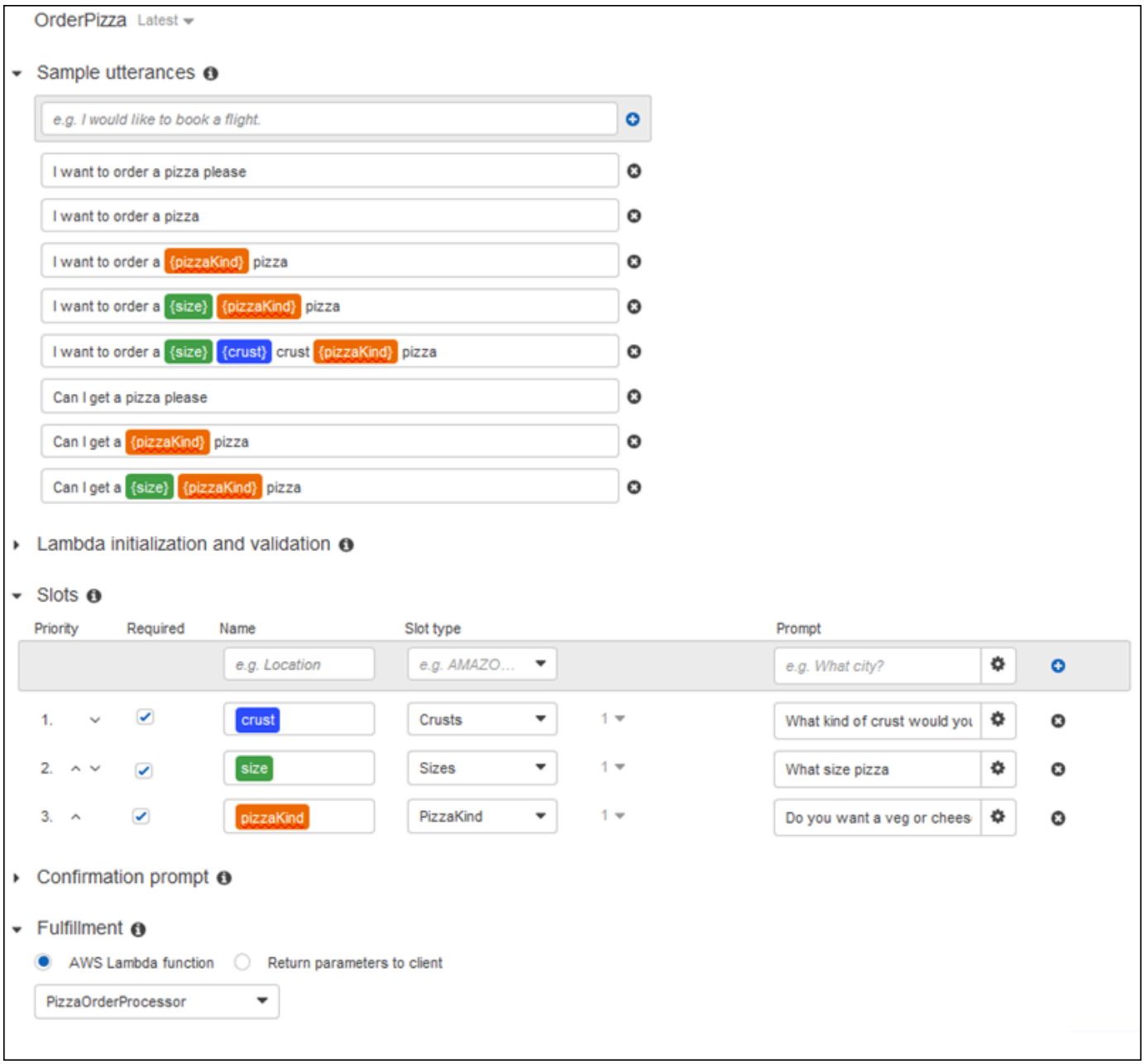
Configura l'intento `OrderPizza` per soddisfare la richiesta di un utente che desidera ordinare una pizza.

Per configurare un intento

- Nella pagina OrderPizzadi configurazione, configura l'intento come segue:
  - Espressioni di esempio: digita le seguenti stringhe. I nomi degli slot sono racchiusi tra parentesi graffe {}.
    - Vorrei ordinare una pizza
    - Voglio ordinare una pizza
    - Voglio ordinare una pizza {pizzaKind}
    - Voglio ordinare una pizza {size} {pizzaKind}
    - Voglio ordinare una pizza {size} con bordo
    - È possibile ordinare una pizza?
    - È possibile ordinare una pizza {pizzaKind}?

    - È possibile ordinare una pizza {size} {pizzaKind}?
- Lambda initialization and validation (Inizializzazione e convalida Lambada): mantieni le impostazioni predefinite.
- Confirmation prompt (Prompt di conferma): mantieni le impostazioni predefinite.
- Adempimento: eseguire le operazioni sotto elencate:
    - Scegli AWS Lambda function (Funzione).
    - Scegli **`PizzaOrderProcessor`**.
    - Se viene visualizzata la finestra di dialogo Aggiungi autorizzazione alla funzione Lambda, scegli OK per concedere all'`OrderPizza`intento il permesso di chiamare la funzione`PizzaOrderProcessor` Lambda.
    - Lascia selezionato il valore None (Nessuno).

L'intento sarà simile al seguente:



Fase successiva

[Configurazione del bot]

Configurazione del bot

Configura la gestione degli errori per il bot `PizzaOrderingBot`.

1. Passa al bot `PizzaOrderingBot`. Scegliete Editor, quindi scegliete Gestione errori, come nell'immagine seguente:

2. Utilizza la scheda Editor per configurare la gestione degli errori del bot.

   - Le informazioni fornite in Clarification Prompts(Prompt di chiarimento) eseguono la mappatura alla configurazione del clarificationPrompt del bot.

     Quando Amazon Lex non è in grado di determinare l'intento dell'utente, il servizio restituisce una risposta con questo messaggio.
   - Le informazioni fornite nella frase di Hang-up (Sospensione) eseguono la mappatura alla configurazione di abortStatement del bot.

     Se il servizio non è in grado di determinare l'intento dell'utente dopo un determinato numero di richieste consecutive, Amazon Lex restituisce una risposta con questo messaggio.

   Mantieni le impostazioni predefinite.

Fase successiva

Fase 3: creazione e test del bot

Fase 3: creazione e test del bot

Assicurati che il bot funzioni creandolo ed eseguendo il test.

Per creare ed eseguire il test del bot

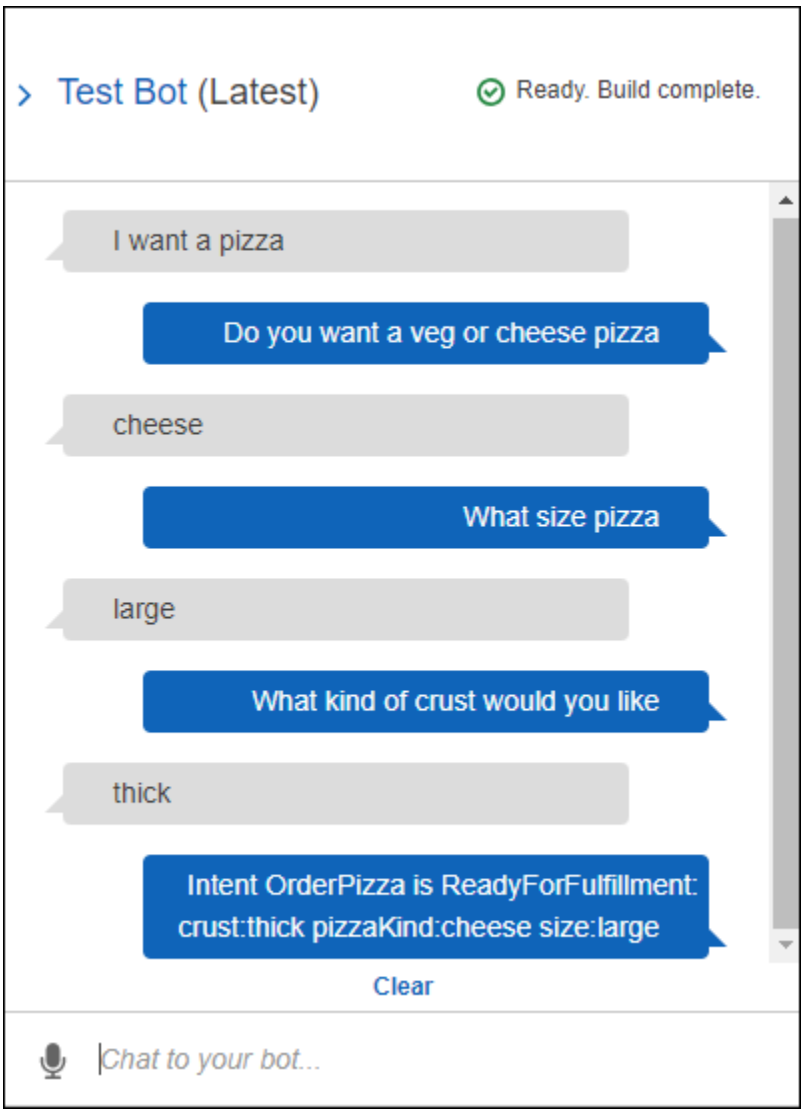
1. Per creare il bot `PizzaOrderingBot` seleziona Build (Crea).

   Amazon Lex crea un modello di apprendimento automatico per il bot. Quando si testa il bot, la console utilizza l'API di runtime per inviare l'input dell'utente ad Amazon Lex. Amazon Lex utilizza quindi il modello di apprendimento automatico per interpretare l'input dell'utente.

   La creazione potrebbe richiedere un po' di tempo.

2. Per testare il bot, nella finestra Test Bot, inizia a comunicare con il tuo bot Amazon Lex.

   - Ad esempio, potresti dire o digitare quanto segue:



- Per eseguire il test del bot, utilizza le enunciazioni di esempio configurate nell'intento `OrderPizza`. Ad esempio, quanto segue è una delle enunciazioni di esempio configurata per l'intento `PizzaOrder`:

  ```
  I want a {size} {crust} crust {pizzaKind} pizza
  ```

  Per eseguire il test, digita quanto segue:

  ```
  I want a large thin crust cheese pizza
  ```

  Quando digiti «Voglio ordinare una pizza», Amazon Lex rileva l'intento (`OrderPizza`). Quindi, Amazon Lex richiede informazioni sugli slot.

  Dopo aver fornito tutte le informazioni sullo slot, Amazon Lex richiama la funzione Lambda che hai configurato per l'intento.

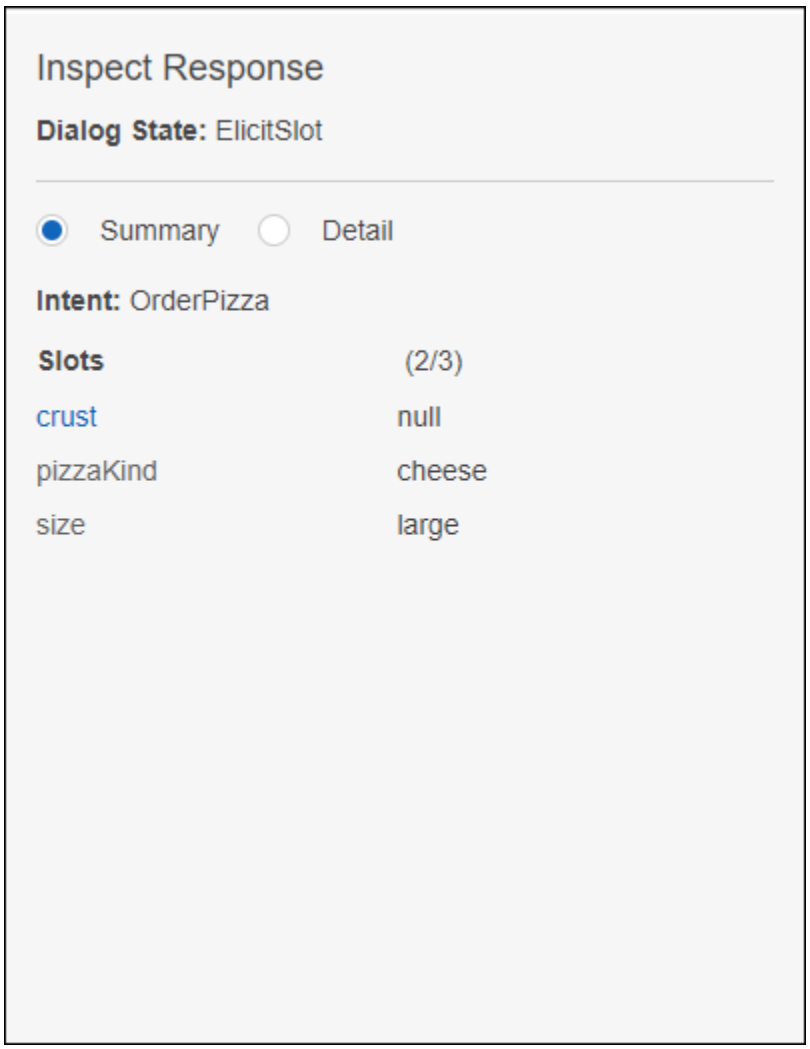
  La funzione Lambda restituisce un messaggio («Ok, ho ordinato il tuo...») ad Amazon Lex, che Amazon Lex ti restituisce.

Esame della risposta

Sotto la finestra di chat è presente un riquadro che consente di esaminare la risposta di Amazon Lex. Il riquadro fornisce informazioni complete sullo stato del bot con modifiche dinamiche che riflettono l'interazione in corso tra l'utente il bot. Il contenuto dei riquadri mostra lo stato corrente dell'operazione.

- Stato della finestra di dialogo: lo stato corrente della conversazione con l'utente. Può essere `ElicitIntent`, `ElicitSlot`, `ConfirmIntent` o `Fulfilled`.

- Riepilogo: mostra una visualizzazione semplificata della finestra di dialogo che mostra i valori degli slot relativi all'intento da soddisfare in modo da poter tenere traccia del flusso di informazioni. Vengono mostrati il nome dell'intento, il numero di slot, il numero di slot popolati e un elenco di tutti gli slot e dei relativi valori associati. Vedi l'immagine seguente:



- Dettaglio: mostra la risposta JSON non elaborata dal chatbot per darti una visione più approfondita dell'interazione con il bot e dello stato corrente della finestra di dialogo durante il test e il debug del chatbot. Se si digita nella finestra della chat, il riquadro di ispezione mostra la risposta JSON dell'operazione PostText. Se si utilizza la comunicazione vocale nella finestra della chat, il riquadro di ispezione mostra le intestazioni della risposta dall'operazione PostContent. Vedi l'immagine seguente:

Fase successiva

Fase 4 (facoltativo): pulizia

Fase 4 (facoltativo): pulizia

È possibile eliminare le risorse create ed effettuare la pulizia dell'account per evitare di incorrere in ulteriori costi per le risorse create.

Puoi eliminare solo le risorse che non sono in uso. Ad esempio, non è possibile eliminare un tipo di slot a cui fa riferimento un intento. Non è possibile eliminare un intento a cui fa riferimento un bot.

Per eliminare le risorse, procedere in questo ordine:

- Elimina i bot per liberare risorse di intenti.

- Elimina gli intenti per liberare risorse di tipo di slot.
- Elimina i tipi di slot come ultima cosa.

Per effettuare la pulizia dell'account:

1. Accedi allaAWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli PizzaOrderingBot.
3. Per eliminare il bot, seleziona Delete (Elimina) e quindi Continue (Continua).
4. Nel riquadro a sinistra seleziona Intents (Intenti).
5. Nell'elenco degli intenti, scegli OrderPizza.
6. Per eliminare l'intento, seleziona Delete (Elimina) e quindi Continue (Continua).
7. Nel menu a sinistra seleziona Slot types (Tipi di slot).
8. Nell'elenco di tipi di slot, seleziona Crusts (Bordi).
9. Per eliminare il tipo di bot seleziona Delete (Elimina) e quindi Continue (Continua).
10. Ripeti Step 8 e Step 9 per i tipi di slot `Sizes` e `PizzaKind`.

Hai rimosso tutte le risorse che hai creato e pulito il tuo account.

Fasi successive

- Pubblicazione di una versione e creazione di un alias
- Crea un bot Amazon Lex conAWS Command Line Interface

Esercizio 3. Pubblicazione di una versione e creazione di un alias

Negli esercizi introduttivi 1 e 2 hai creato un bot e ne hai eseguito il test. In questo esercizio, devi effettuare le seguenti operazioni:

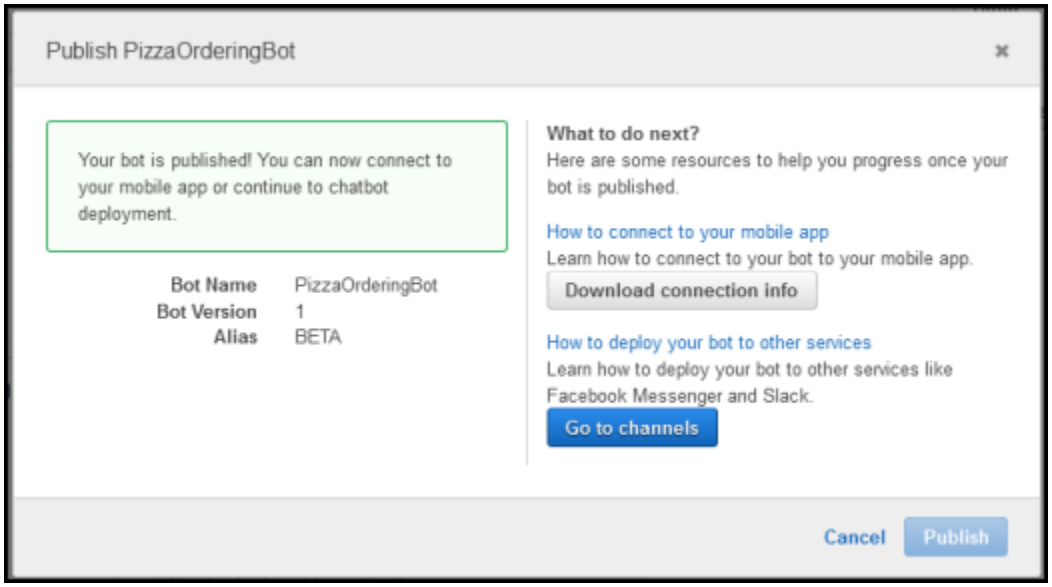
- Pubblica una nuova versione del bot. Amazon Lex scatta una copia istantanea della`$LATEST` versione per pubblicarne una nuova.
- Creazione di un alias che punta alla nuova versione.

Per ulteriori informazioni sulla funzione Versioni multiple e sugli alias, consulta Funzione Versioni multiple e alias.

Per pubblicare una versione di un bot creato per questo esercizio, procedi come descritto di seguito:

1. Nella console Amazon Lex, scegli uno dei bot che hai creato.

   Verifica che la console visualizzi la versione `$LATEST` accanto al nome del bot.
2. Seleziona Publish (Pubblica).
3. Nella procedura guidata Publish (Pubblica) ***`nome bot`***, specifica l'alias **`BETA`**, quindi scegli Publish (Pubblica).
4. Verifica che la console Amazon Lex mostri la nuova versione accanto al nome del bot, come nell'immagine seguente.



Ora che disponi di un bot con un alias e una versione pubblicata, puoi distribuirlo (nella tua applicazione per dispositivi mobili o integrarlo in Facebook Messenger). Per un esempio, consulta Integrazione di un Amazon Lex Bot con Facebook Messenger.

Fase 4: Nozioni di base (AWS CLI)

In questa fase, utilizzerai ilAWS CLIper creare, testare e modificare un bot Amazon Lex. Per completare questi esercizi, dovrai acquisire familiarità con l'uso dell'interfaccia a riga di comando (CLI) e disporre di un editor di testo. Per ulteriori informazioni, consultaFase 2: Configurare il AWS Command Line Interface

- Esercizio 1: crea e prova un bot Amazon Lex. L'esercizio fornisce tutti gli oggetti JSON necessari per creare un tipo di slot personalizzato, un intento e un bot. Per ulteriori informazioni, consultaAmazon Lex: come funziona: come funziona
- Esercizio 2. Aggiornamento del bot creato nell'esercizio 1 per aggiungere un'ulteriore enunciazione di esempio. Amazon Lex utilizza le enunciazioni di esempio per creare il modello di machine learning per il tuo bot.
- Esercizio 3. Aggiornamento del bot creato nell'esercizio 1 per aggiungere una funzione Lambda per convalidare l'input dell'utente e realizzare l'intento.
- Esercizio 4. Pubblicazione di una versione delle risorse del tipo di slot, dell'intento e del bot create nell'esercizio 1. Una versione è uno snapshot di una risorsa che non può essere modificata.
- Esercizio 5. Creazione di un alias per il bot creato nell'esercizio 1.
- Esercizio 6. Pulizia dell'account tramite l'eliminazione del tipo di slot, dell'intento e del bot creati nell'esercizio 1 e dell'alias creato nell'esercizio 5.

Argomenti

Esercizio 1: Creazione di un bot Amazon Lex (AWS CLI)

In generale, quando crei bot:

1. Crei tipi di slot per definire le informazioni che verranno utilizzate dal tuo bot.
2. Crei intenti che definiscano le operazioni dell'utente supportate dal bot. Usa i tipi di slot personalizzati che hai creato in precedenza per definire gli slot, o parametri, richiesti dal tuo intento.
3. Crei un bot che utilizzi gli intenti da te definiti.

In questo esercizio devi creare e testare un nuovo bot Amazon Lex utilizzando l'interfaccia a riga di comando (CLI). Usa le strutture JSON fornite per creare il bot. Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Argomenti

- Fase 1: Creazione di un ruolo collegato ai servizi (AWS CLI)
- Fase 2: Creazione di un tipo di slot personalizzato (AWS CLI)
- Fase 3: Creazione di un intento (AWS CLI)
- Fase 4: Creazione di un bot (AWS CLI)
- Fase 5: Test di un bot (AWS CLI)

Fase 1: Creazione di un ruolo collegato ai servizi (AWS CLI)

Amazon Lex presupponeAWS Identity and Access Managementruoli collegati al servizio da chiamareAWSservizi per conto dei tuoi bot. I ruoli presenti nel tuo account sono collegati ai casi d'uso di Amazon Lex e dispongono di autorizzazioni predefinite. Per ulteriori informazioni, consultare Utilizzo di ruoli collegati ai servizi per Amazon Lex.

Se hai già creato un bot Amazon Lex utilizzando la console, il ruolo collegato ai servizi è stato creato automaticamente. Passa a Fase 2: Creazione di un tipo di slot personalizzato (AWS CLI).

Come creare un ruolo collegato ai servizi (AWS CLI)

1. In AWS CLI, digita il comando seguente:

   ```
   aws iam create-service-linked-role --aws-service-name lex.amazonaws.com
   ```

2. Controlla la policy utilizzando il comando seguente:

   ```
   aws iam get-role --role-name AWSServiceRoleForLexBots
   ```

   La risposta è:

   ```
   {
       "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"
}
```

Fase successiva

Fase 2: Creazione di un tipo di slot personalizzato (AWS CLI)

Fase 2: Creazione di un tipo di slot personalizzato (AWS CLI)

Crea un tipo di slot personalizzato con i valori di enumerazione dei fiori che è possibile ordinare. Userai questo tipo nella fase successiva per creare l'intento `OrderFlowers`. Un tipo di slot definisce i valori possibili per uno slot, o parametro, dell'intento.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Per creare un tipo di slot personalizzato (AWS CLI)

1. Crea un file di testo denominato **`FlowerTypes.json`**. Copia il codice JSON da FlowerTypes.json nel file di testo.
2. Chiama l'operazione PutSlotType utilizzando AWS CLI per creare il tipo di slot. L'esempio è formattato per Unix, Linux e macOS. Per Windows, sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

```
aws lex-models put-slot-type \
    --region region \
    --name FlowerTypes \
    --cli-input-json file://FlowerTypes.json
```

La risposta del server è:

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "checksum": "checksum",
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Fase successiva

Fase 3: Creazione di un intento (AWS CLI)

FlowerTypes.json

Il codice seguente rappresenta i dati JSON necessari per creare il tipo di slot personalizzato `FlowerTypes`:

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
```

```
        }
    ],
    "name": "FlowerTypes",
    "description": "Types of flowers to pick up"
}
```

Fase 3: Creazione di un intento (AWS CLI)

Crea un intento per il bot `OrderFlowersBot` e fornisci tre slot, o parametri. Gli slot consentono al bot di realizzare l'intento:

- `FlowerType` è un tipo di slot personalizzato che specifica quali tipi di fiori è possibile ordinare.
- `AMAZON.DATE` e `AMAZON.TIME` sono tipi di slot integrati utilizzati per stabilire la data e l'ora della consegna dei fiori da parte dell'utente.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Per creare l'intento **`OrderFlowers`** (AWS CLI)

1. Crea un file di testo denominato **`OrderFlowers.json`**. Copia il codice JSON da OrderFlowers.json nel file di testo.
2. In AWS CLI, chiama l'operazione PutIntent per creare l'intento. L'esempio è formattato per Unix, Linux e macOS. Per Windows, sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

   ```
   aws lex-models put-intent \
      --region region \
      --name OrderFlowers \
      --cli-input-json file://OrderFlowers.json
   ```

   Il server risponde come segue:

   ```
   {
       "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"
}
```

Fase successiva

Fase 4: Creazione di un bot (AWS CLI)

OrderFlowers.json

Il codice seguente rappresenta i dati JSON necessari per creare l'intento `OrderFlowers`:

```
{
    "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"
}
```

Fase 4: Creazione di un bot (AWS CLI)

Il bot `OrderFlowersBot` include l'intento `OrderFlowers` creato nella fase precedente. Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica `"\$LATEST"` in `$LATEST`.

Per creare il bot **`OrderFlowersBot`** (AWS CLI)

1. Crea un file di testo denominato **`OrderFlowersBot.json`**. Copia il codice JSON da OrderFlowersBot.json nel file di testo.
2. In AWS CLI, chiama l'operazione PutBot per creare il bot. L'esempio è formattato per Unix, Linux e macOS. Per Windows, sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

   ```
   aws lex-models put-bot \
       --region region \
       --name OrderFlowersBot \
       --cli-input-json file://OrderFlowersBot.json
   ```

   La risposta del server è riportata di seguito. Quando crei o aggiorni il bot, il campo `status` è impostato su `BUILDING`, a indicare che il bot non è pronto all'uso. Usa l'operazione GetBot della fase successiva per determinare quando il bot è pronto all'uso.

   ```
   {
       "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. Per determinare se il tuo nuovo bot è pronto all'uso, esegui il comando riportato di seguito. Ripeti questo comando finché il campo `status` non restituisce `READY`. L'esempio è formattato per Unix, Linux e macOS. Per Windows, sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST"
```

Individua il campo `status` nella risposta:

```
{
    "status": "READY",

    ...

}
```

Fase successiva

[Fase 5: Test di un bot (AWS CLI)]

OrderFlowersBot.json

Il codice seguente fornisce i dati JSON necessari per creare l'`OrderFlowers`Supporto Amazon Lex:

```
{
    "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"
}
```

Fase 5: Test di un bot (AWS CLI)

Per testare il bot, puoi utilizzare un test basato sull'input vocale o di testo.

Argomenti

- Test di un bot utilizzando l'input di testo (AWS CLI)
- Test di un bot utilizzando l'input vocale (AWS CLI)

Test di un bot utilizzando l'input di testo (AWS CLI)

Per verificare che il bot funzioni correttamente con l'input di testo, usa l'operazione PostText. Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote di servizio runtime.

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica `"\$LATEST"` in `$LATEST` e sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

Per testare il bot utilizzando l'input di testo (AWS CLI)

1. In AWS CLI, avvia una conversazione con il bot `OrderFlowersBot`. L'esempio è formattato per Unix, Linux e macOS. Per Windows, sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

   ```
   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 riconosce l'intento dell'utente e avvia una conversazione restituendo la risposta seguente:

```
{
    "slotToElicit": "FlowerType",
    "slots": {
        "PickupDate": null,
        "PickupTime": null,
        "FlowerType": null
    },
    "dialogState": "ElicitSlot",
    "message": "What type of flowers would you like to order?",
    "intentName": "OrderFlowers"
}
```

2. Esegui i comandi riportati di seguito per completare la conversazione con il bot.

```
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"
```

Una volta confermato l'ordine, Amazon Lex invia una risposta di soddisfazione per completare la conversazione:

```
{
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    },
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers"
}
```

Fase successiva

Test di un bot utilizzando l'input vocale (AWS CLI)

Test di un bot utilizzando l'input vocale (AWS CLI)

Per testare il bot utilizzando file audio, usa l'operazione PostContent. Genera i file audio utilizzando le operazioni di sintesi vocale di Amazon Polly.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi di Amazon Lex e Amazon Polly. Per l'elenco delle regioni relative ad Amazon Lex, consultaQuote di servizio runtime. Per l'elenco delle regioni relative ad Amazon Polly, consultaAWSRegioni ed endpointnellaRiferimento generale di Amazon Web Services.

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica `"\$LATEST"` in `$LATEST` e sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

Per testare il bot utilizzando l'input vocale (AWS CLI)

1. NellaAWS CLI, crea un file audio utilizzando Amazon Polly. L'esempio è formattato per Unix, Linux e macOS. Per Windows, sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

   ```
   aws polly synthesize-speech \
       --region region \
       --output-format pcm \
       --text "i would like to order flowers" \
       --voice-id "Salli" \
       IntentSpeech.mpg
   ```

2. Per inviare il file audio ad Amazon Lex, esegui il comando riportato di seguito. Amazon Lex salva l'audio della risposta nel file di output specificato.

   ```
   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 risponde con una richiesta per il primo slot, quindi salva l'audio della risposta nel file di output specificato.

   ```
   {
       "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. Per ordinare rose, crea i file audio seguenti e inviali ad Amazon Lex:

```
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. Per impostare la data di consegna, crea i file audio seguenti e inviali ad Amazon Lex:

```
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. Per impostare l'orario di consegna, crea i file audio seguenti e inviali ad Amazon Lex:

```
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. Per confermare la consegna, crea i file audio seguenti e inviali ad Amazon Lex:

```
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
```

Una volta confermata la consegna, Amazon Lex invia una risposta che conferma la realizzazione dell'intento:

```
{
    "contentType": "text/plain;charset=utf-8",
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers",
    "inputTranscript": "yes",
    "slots": {
        "PickupDate": "2017-05-16",
```

```
        "PickupTime": "10:00",
        "FlowerType": "roses"
    }
}
```

Fase successiva

[Esercizio 2: Aggiungere un nuovo enunciazione (AWS CLI)]

Esercizio 2: Aggiungere un nuovo enunciazione (AWS CLI)

Per migliorare il modello di machine learning che Amazon Lex utilizza per riconoscere le richieste da parte degli utenti, aggiungi un'altra enunciazione di esempio al bot.

L'aggiunta di una nuova enunciazione è un processo diviso in quattro fasi.

1. Utilizzo dell'GetIntentoperazione per ottenere un intento da Amazon Lex.
2. Aggiorna l'intento.
3. Utilizzo dell'PutIntentoperazione per inviare l'intento aggiornato nuovamente ad Amazon Lex.
4. Usa le operazioni GetBot e PutBot per ricreare qualsiasi bot che utilizza l'intento.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

La risposta dell'operazione `GetIntent` contiene un campo denominato `checksum` che identifica una specifica revisione dell'intento. È necessario fornire il valore checksum quando si utilizza l'operazione PutIntent per aggiornare un intento. In caso contrario, verrà visualizzato il messaggio di errore seguente:

```
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**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica "\$LATEST" in $LATEST e sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

Per aggiornare l'intento **`OrderFlowers`** (AWS CLI)

1. NellaAWS CLI, ottieni l'intento da Amazon Lex. Amazon Lex invia l'output a un file denominato**`OrderFlowers-V2.json`.**

   ```
   aws lex-models get-intent \
       --region region \
       --name OrderFlowers \
       --intent-version "\$LATEST" > OrderFlowers-V2.json
   ```

2. Aprire **`OrderFlowers-V2.json`** in un editor di testo.

   1. Individua e cancella i campi `createdDate`, `lastUpdatedDate` e `version`.
   2. Aggiungi il testo seguente nel campo `sampleUtterances`:

      ```
      I want to order flowers
      ```

   3. Salvare il file.

3. Invia l'intento aggiornato ad Amazon Lex con il comando seguente:

   ```
   aws lex-models put-intent  \
       --region region \
       --name OrderFlowers \
       --cli-input-json file://OrderFlowers-V2.json
   ```

   Amazon Lex invia la risposta seguente:

   ```
   {
       "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"
}
```

Ora che l'intento è stato aggiornato, ricrea qualsiasi bot che lo utilizza.

Per ricreare il bot **`OrderFlowersBot`** (AWS CLI)

1. In AWS CLI, ottieni la definizione del bot `OrderFlowersBot` e salvala in un file con il comando seguente:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V2.json
```

2. In un editor di testo, apri **`OrderFlowersBot-V2.json`**. Rimuovi i campi `createdDate`, `lastUpdatedDate`, `status` e `version`.
3. In un editor di testo, aggiungi la seguente riga alla definizione del bot:

```
"processBehavior": "BUILD",
```

4. In AWS CLI, crea una nuova revisione del bot eseguendo il comando riportato di seguito:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V2.json
```

   La risposta del server è:

```
{
    "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"
}
```

Fase successiva

Esercizio 3: Aggiungere una funzione Lambda (AWS CLI)

Esercizio 3: Aggiungere una funzione Lambda (AWS CLI)

Aggiungi una funzione Lambda che convalidi l'input dell'utente e realizzi l'intento dell'utente per il bot.

L'aggiunta di un'espressione Lambda è un processo in cinque fasi.

1. Usa LambdaAddPermissionfunzione per abilitare il`OrderFlowers`intento di chiamare la LambdaRichiamooperazione.
2. Utilizzo dell'GetIntentoperazione per ottenere l'intento da Amazon Lex.
3. Aggiorna l'intento per aggiungere la funzione Lambda.
4. Utilizzo dell'PutIntentoperazione per inviare l'intento aggiornato nuovamente ad Amazon Lex.
5. Usa le operazioni GetBot e PutBot per ricreare qualsiasi bot che utilizza l'intento.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Se aggiungi una funzione Lambda ad un intento prima di aggiungere la`InvokeFunction`autorizzazione, viene visualizzato il seguente messaggio di errore:

```
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.
```

La risposta dell'operazione `GetIntent` contiene un campo denominato `checksum` che identifica una specifica revisione dell'intento. È necessario fornire il valore checksum quando si utilizza l'operazione PutIntent per aggiornare un intento. In caso contrario, verrà visualizzato il messaggio di errore seguente:

```
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.
```

Questo esercizio utilizza la funzione LambdaEsercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console). Per istruzioni su come creare la funzione Lambda, consultaFase 3: creazione di una funzione Lambda.

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica `"\$LATEST"` in `$LATEST`.

Per aggiungere una funzione Lambda a un intento

1. In AWS CLI, aggiungi l'autorizzazione `InvokeFunction` per l'intento `OrderFlowers`:

```
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 invia la risposta seguente:

```
{
    "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. Ottieni l'intento da Amazon Lex. Amazon Lex invia l'output a un file denominato**`OrderFlowers-V3.json`**.

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V3.json
```

3. In un editor di testo aprire il file **`OrderFlowers-V3.json`**.

   1. Individua e cancella i campi `createdDate`, `lastUpdatedDate` e `version`.
   2. Aggiorna il campo `fulfillmentActivity`:

```
    "fulfillmentActivity": {
        "type": "CodeHook",
        "codeHook": {
```

```
            "uri": "arn:aws:lambda:region:account
ID:function:OrderFlowersCodeHook",
            "messageVersion": "1.0"
        }
    }
```

3. Salvare il file.

4. NellaAWS CLI, invia l'intento aggiornato ad Amazon Lex:

```
aws lex-models put-intent \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V3.json
```

Ora che l'intento è stato aggiornato, ricrea il bot.

Per ricreare il bot **`OrderFlowersBot`**

1. In AWS CLI, ottieni la definizione del bot `OrderFlowersBot` e salvala in un file:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V3.json
```

2. In un editor di testo, apri **`OrderFlowersBot-V3.json`**. Rimuovi i campi `createdDate`, `lastUpdatedDate`, `status` e `version`.

3. Nell'editor di testo, aggiungi la seguente riga alla definizione del bot:

```
"processBehavior": "BUILD",
```

4. In AWS CLI, crea una nuova revisione del bot:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V3.json
```

La risposta del server è:

```
{
    "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"
}
```

Fase successiva

[Esercizio 4: Pubblicazione di una versione (AWS CLI)]

Esercizio 4: Pubblicazione di una versione (AWS CLI)

A questo punto, crea una versione del bot creato nell'esercizio 1. Una versione è uno snapshot del bot. Una volta creata la versione, non potrai modificarla. L'unica versione di un bot che è possibile aggiornare è la versione `$LATEST`. Per ulteriori informazioni sulle versioni, consulta Funzione Versioni multiple e alias.

Prima di pubblicare una versione di un bot, è necessario pubblicare i relativi intenti. Analogamente, è necessario pubblicare i tipi di slot cui fanno riferimento tali intenti. In generale, per pubblicare una versione di un bot:

1. Pubblichi una versione di un tipo di slot con l'operazione CreateSlotTypeVersion.
2. Pubblichi una versione di un intento con l'operazione CreateIntentVersion.
3. Pubblichi una versione di un bot con l'operazione CreateBotVersion.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Argomenti

- Fase 1: Pubblicazione del tipo di slot (AWS CLI)
- Fase 2: Pubblicazione dell'intento (AWS CLI)
- Fase 3: Pubblicazione del bot (AWS CLI)

Fase 1: Pubblicazione del tipo di slot (AWS CLI)

Prima di pubblicare una versione di qualsiasi intento che utilizza un tipo di slot, è necessario pubblicare una versione di tale tipo di slot. In questo caso, dovrai pubblicare il tipo di slot `FlowerTypes`.

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica `"\$LATEST"` in `$LATEST` e sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

Per pubblicare un tipo di slot (AWS CLI)

1. In AWS CLI, ottieni la versione più recente del tipo di slot:

```
aws lex-models get-slot-type \
    --region region \
    --name FlowerTypes \
    --slot-type-version "\$LATEST"
```

La risposta di Amazon Lex è riportata di seguito. Registra il checksum per l'attuale revisione della versione $LATEST.

```
{
    "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. Pubblica una versione del tipo di slot. Usa il checksum registrato nella fase precedente.

```
aws lex-models create-slot-type-version \
    --region region \
    --name FlowerTypes \
    --checksum "checksum"
```

La risposta di Amazon Lex è riportata di seguito. Registra il numero di versione per la fase successiva.

```
{
    "version": "1",
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Fase successiva

Fase 2: Pubblicazione dell'intento (AWS CLI)

Fase 2: Pubblicazione dell'intento (AWS CLI)

Prima di pubblicare un intento, è necessario pubblicare tutti i tipi di slot cui fa riferimento l'intento. I tipi di slot devono essere versioni numerate, anziché la versione `$LATEST`.

Innanzitutto, aggiorna l'intento `OrderFlowers` per usare la versione del tipo di slot `FlowerTypes` pubblicata nella fase precedente. Quindi pubblica una nuova versione dell'intento `OrderFlowers`.

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica `"\$LATEST"` in `$LATEST` e sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

Per pubblicare una versione di un intento (AWS CLI)

1. In AWS CLI, ottieni la versione $LATEST dell'intento `OrderFlowers` e salvala in un file:

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4.json
```

2. In un editor di testo, aprire il file **`OrderFlowers_V4.json`**. Cancella i campi `createdDate`, `lastUpdatedDate` e `version`. Individua il tipo di slot `FlowerTypes` e modifica la versione nel numero di versione registrato nella fase precedente. Il seguente frammento del file **`OrderFlowers_V4.json`** mostra la posizione della modifica:

```
       {
           "slotType": "FlowerTypes",
           "name": "FlowerType",
           "slotConstraint": "Required",
           "valueElicitationPrompt": {
               "maxAttempts": 2,
               "messages": [
                   {
                       "content": "What type of flowers?",
                       "contentType": "PlainText"
                   }
               ]
           },
           "priority": 1,
           "slotTypeVersion": "version",
           "sampleUtterances": []
       },
```

3. In AWS CLI, salva la revisione dell'intento:

```
aws lex-models put-intent \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers_V4.json
```

4. Ottieni il checksum della versione più recente dell'intento:

```
aws lex-models get-intent \
    --region region \
```

```
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4a.json
```

Il seguente frammento della risposta mostra il checksum dell'intento. Registra questo valore per la fase successiva.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
```

5. Pubblica una nuova versione dell'intento:

```
aws lex-models create-intent-version \
    --region region \
    --name OrderFlowers \
    --checksum "checksum"
```

Il seguente frammento della risposta mostra la nuova versione dell'intento. Registra il numero di versione per la fase successiva.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "version",
```

Fase successiva

Fase 3: Pubblicazione del bot (AWS CLI)

Fase 3: Pubblicazione del bot (AWS CLI)

Dopo aver pubblicato tutti i tipi di slot e gli intenti utilizzati dal tuo bot, potrai pubblicare il bot.

Aggiorna il bot `OrderFlowersBot` per utilizzare l'intento `OrderFlowers` aggiornato nella fase precedente. Quindi pubblica una nuova versione del bot `OrderFlowersBot`.

> **Note**
>
> L'esempio seguente di AWS CLI è formattato per Unix, Linux e macOS. Per Windows, modifica "\$LATEST" in $LATEST e sostituisci il carattere di continuazione UNIX barra rovesciata (\) al termine di ogni riga con un accento circonflesso (^).

Per pubblicare una versione di un bot (AWS CLI)

1. In AWS CLI, ottieni la versione $LATEST del bot `OrderFlowersBot` e salvala in un file:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot_V4.json
```

2. In un editor di testo, aprire il file **`OrderFlowersBot_V4.json`**. Cancella i campi `createdDate`, `lastUpdatedDate`, `status` e `version`. Individua l'intento `OrderFlowers` e modifica la versione nel numero di versione registrato nella fase precedente. Il seguente frammento di **`OrderFlowersBot_V4.json`** mostra la posizione della modifica.

```
    "intents": [
        {
            "intentVersion": "version",
            "intentName": "OrderFlowers"
        }
```

3. In AWS CLI, salva la nuova revisione del bot: Prendere nota del numero di versione restituito dalla chiamata a `put-bot`.

```
aws lex-models put-bot \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot_V4.json
```

4. Ottieni il checksum della versione più recente del bot: Utilizzare il numero di versione restituito al passaggio 3.

```
aws lex-models get-bot \
    --region region \
    --version-or-alias version \
```

```
    --name OrderFlowersBot > OrderFlowersBot_V4a.json
```

Il seguente frammento della risposta mostra il checksum del bot. Registra questo valore per la fase successiva.

```
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
```

5. Pubblica una nuova versione del bot:

```
aws lex-models create-bot-version \
    --region region \
    --name OrderFlowersBot \
    --checksum "checksum"
```

Il seguente frammento della risposta mostra la nuova versione del bot.

```
    "checksum": "checksum",
    "abortStatement": {
        ...
    },
    "version": "1",
    "lastUpdatedDate": timestamp,
```

Fase successiva

Esercizio 5: Creazione di un alias (AWS CLI)

Esercizio 5: Creazione di un alias (AWS CLI)

Un alias è un puntatore a una specifica versione di un bot Con un alias è possibile aggiornare facilmente la versione utilizzata dalle applicazioni client. Per ulteriori informazioni, consulta Funzione Versioni multiple e alias. Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Per creare un alias (AWS CLI)

1. In AWS CLI, ottieni la versione del bot `OrderFlowersBot` creato in Esercizio 4: Pubblicazione di una versione (AWS CLI).

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias version > OrderFlowersBot_V5.json
```

2. In un editor di testo, apri **`OrderFlowersBot_v5.json`**. Individua e registra il numero di versione.
3. In AWS CLI, crea l'alias del bot:

```
aws lex-models put-bot-alias  \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot \
    --bot-version version
```

Di seguito è riportata la risposta del server:

```
{
    "name": "PROD",
    "createdDate": timestamp,
    "checksum": "checksum",
    "lastUpdatedDate": timestamp,
    "botName": "OrderFlowersBot",
    "botVersion": "1"
}}
```

Fase successiva

Esercizio 6: Pulizia (AWS CLI)

Esercizio 6: Pulizia (AWS CLI)

Elimina le risorse che hai creato e pulisci il tuo account.

Puoi eliminare solo le risorse che non sono in uso. In generale, devi eliminare le risorse nell'ordine seguente.

1. Elimina gli alias per liberare risorse di bot.

2. Elimina i bot per liberare risorse di intenti.
3. Elimina gli intenti per liberare risorse di tipo di slot.
4. Elimina i tipi di slot.

Per eseguire i comandi di questo esercizio, devi conoscere la regione in cui verranno eseguiti i comandi. Per l'elenco delle regioni, consulta Quote per la creazione di modelli .

Per effettuare la pulizia dell'account (AWS CLI)

1. Dalla riga di comando di AWS CLI, elimina l'alias:

```
aws lex-models delete-bot-alias \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot
```

2. Dalla riga di comando di AWS CLI, elimina il bot:

```
aws lex-models delete-bot \
    --region region \
    --name OrderFlowersBot
```

3. Dalla riga di comando di AWS CLI, elimina l'intento:

```
aws lex-models delete-intent \
    --region region \
    --name OrderFlowers
```

4. Dalla riga di comando di AWS CLI, elimina il tipo di slot:

```
aws lex-models delete-slot-type \
    --region region \
    --name FlowerTypes
```

Hai rimosso tutte le risorse che hai creato e pulito il tuo account.

Funzione Versioni multiple e alias

Amazon Lex supporta la pubblicazione di versioni di bot, intenti e tipi di slot per controllare l'implementazione utilizzata dalle applicazioni client. Una versione è una snapshot numerata del lavoro dell'utente che è possibile pubblicare per l'utilizzo in diverse parti del flusso di lavoro, quali lo sviluppo, la distribuzione beta e la produzione.

I bot di Amazon Lex supportano anche gli alias. Un alias è un puntatore a una specifica versione di un bot Con un alias è possibile aggiornare facilmente la versione utilizzata dalle applicazioni client. Ad esempio, si potrebbe associare un alias alla versione 1 di un bot. Quando si è pronti per aggiornare il bot, è possibile pubblicare la versione 2 del bot e modificare l'alias in modo che punti alla nuova versione. Poiché le applicazioni utilizzano l'alias anziché una versione specifica, tutti i client otterranno la nuova funzionalità senza necessità di un aggiornamento.

Argomenti

- Funzione Controllo delle versioni
- Alias

Funzione Controllo delle versioni

Quando si esegue la versione di una risorsa Amazon Lex, viene creata una snapshot della risorsa in modo che sia possibile utilizzare la risorsa esistente al momento della creazione della versione. Una volta creata una versione, mentre si continua a lavorare sull'applicazione questa versione rimarrà invariata.

La versione $LATEST

Quando si crea un bot, un intento o un tipo di slot di Amazon Lex è disponibile una sola versione:`$LATEST`Versione .

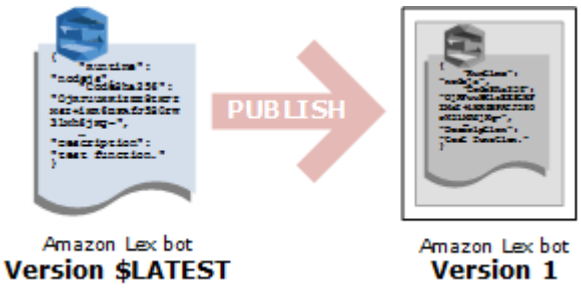
La versione $LATEST è la copia di lavoro della risorsa. Fino a quando non si pubblica la prima versione della risorsa, è possibile aggiornare solo la versione $LATEST. La versione $LATEST è l'unica versione della risorsa disponibile.

Solo la versione $LATEST di una risorsa può utilizzare la versione $LATEST di un'altra risorsa. Ad esempio, la versione $LATEST di un bot può utilizzare la versione $LATEST di un intento e la versione $LATEST di un intento può utilizzare la versione $LATEST di un tipo di slot.

La$LATESTLa versione del bot deve essere utilizzata solo per i test manuali. Amazon Lex limita il numero di richieste di runtime che è possibile effettuare per$LATESTversione del bot.

Pubblicazione di una versione delle risorse Amazon Lex

Quando si pubblica una risorsa, Amazon Lex crea una copia del$LATESTla salva come versione numerata. La versione pubblicata non può essere modificata.



È possibile creare e pubblicare versioni tramite la console Amazon Lex o ilCreateBotVersionoperazione. Per un esempio, consultare Esercizio 3. Pubblicazione di una versione e creazione di un alias.

Quando si modifica la versione $LATEST di una risorsa, è possibile pubblicare la nuova versione per rendere disponibili le modifiche per le applicazioni client. Ogni volta che pubblichi una versione, Amazon Lex copia il$LATESTversione per creare la nuova versione e incrementa il numero della versione di 1 numero. I numeri di versione non vengono mai riutilizzati. Ad esempio, se si elimina una versione con numero di risorsa 10 e quindi la si crea nuovamente, il numero della nuova versione assegnato da Amazon Lex sarà 11.

Prima di poter pubblicare un bot, è necessario farlo puntare a una versione numerata di qualsiasi evento utilizzato. Se si tenta di pubblicare una nuova versione di un bot che utilizza la versione $LATEST di un intento, Amazon Lex restituisce un'eccezione di richiesta non valida HTTP 400. Prima di poter pubblicare una versione numerata dell'intento, è necessario far puntare l'intento a una

versione numerata di qualsiasi tipo di slot utilizzato. In caso contrario, verrà restituita un'eccezione di richiesta non valida 400 HTTP.

> **Note**
>
> Amazon Lex pubblica una nuova versione solo se l'ultima versione pubblicata è diversa dalla`$LATEST`Versione . Se provi a pubblicare il`$LATEST`senza modificarla, Amazon Lex non crea né pubblica una nuova versione.

Aggiornamento di una risorsa Amazon Lex

È possibile aggiornare solo il`$LATEST`versione di un bot, intento o tipo di slot di Amazon Lex. Non è possibile modificare le versioni pubblicate. È possibile pubblicare una nuova versione in qualsiasi momento dopo aver aggiornato una risorsa tramite la console o le operazioni CreateBotVersion, CreateIntentVersion o CreateSlotTypeVersion.

Eliminazione di una risorsa o versione di Amazon Lex

Amazon Lex supporta l'eliminazione di una risorsa o versione tramite la console o una delle operazioni API:

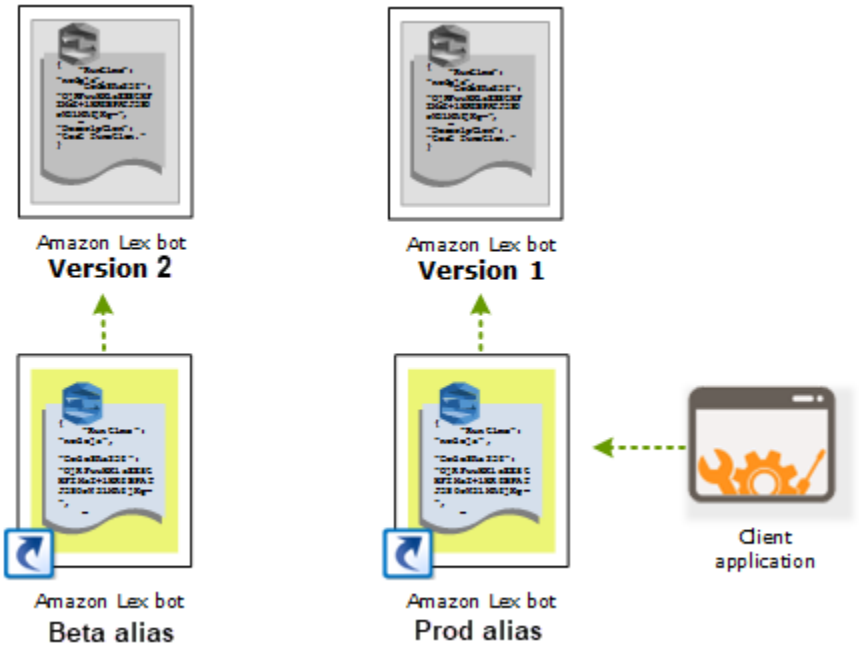
- DeleteBot
- DeleteBotVersion
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType

- DeleteSlotTypeVersion

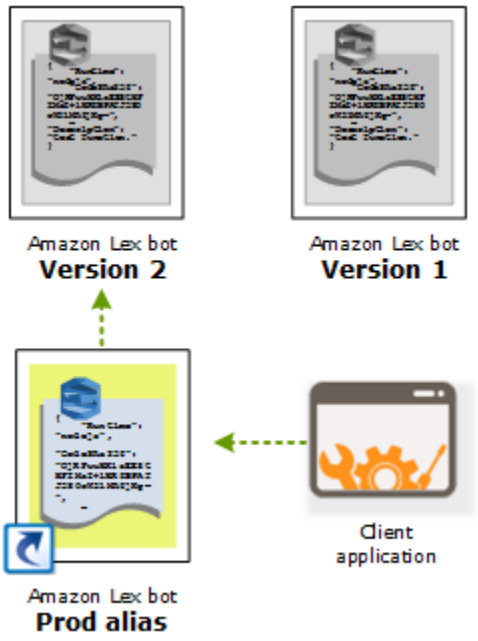
Alias

Un alias è un puntatore a una specifica versione di un bot Amazon Lex. È possibile utilizzare un alias per consentire alle applicazioni client di utilizzare una versione specifica del bot senza richiedere all'applicazione di tenere traccia di quale versione sia.

L'esempio seguente mostra due versioni di un bot Amazon Lex: la versione 1 e la versione 2. A ciascuna di queste versioni di bot è associato un alias: rispettivamente BETA e PROD. Le applicazioni client utilizzano l'alias PROD per accedere al bot.



Quando si crea una seconda versione del bot, è possibile aggiornare l'alias utilizzando la console o l'operazione PutBot per fare in modo che punti alla nuova versione del bot. Se si modifica l'alias, tutte le applicazioni client utilizzeranno la nuova versione. Se si verifica un problema relativo alla nuova versione, è possibile tornare alla versione precedente semplicemente modificando l'alias in modo che punti a tale versione.



> **Note**
>
> Sebbene sia possibile effettuare un test della versione $LATEST di un bot nella console, quando si integra un bot con l'applicazione client si consiglia di pubblicare prima una versione e creare un alias che punta a tale versione. Utilizzare l'alias nell'applicazione client per i motivi illustrati in questa sezione. Quando si aggiorna un alias, prima di iniziare a utilizzare la nuova versione Amazon Lex attenderà fino allo scadere del timeout di sessione per tutte le sessioni correnti. Per ulteriori informazioni sul timeout di sessione, consultare the section called “Impostazione del timeout di sessione”.

Utilizzo delle funzioni Lambda

Puoi creareAWS Lambdafunzioni da utilizzare come hook di codice per il bot di Amazon Lex. Nella configurazione dell'intento, puoi identificare le funzioni Lambda per eseguire operazioni di inizializzazione e convalida, di adempimento o entrambi.

Ti consigliamo di utilizzare una funzione Lambda come hook di codice per il bot. Senza una funzione Lambda, il bot restituisce l'informazione sull'intento all'applicazione client per l'adempimento.

Argomenti

Formato di evento di input e di risposta della funzione Lambda

Questa sezione descrive la struttura dei dati dell'evento che Amazon Lex fornisce a una funzione Lambda. Utilizza queste informazioni per analizzare l'input nel codice Lambda. Illustra inoltre il formato della risposta che Amazon Lex prevede di ricevere in restituzione dalla funzione Lambda.

Argomenti

Formato dell'evento di input

Di seguito è riportato il formato generale di un evento Amazon Lex passato a una funzione Lambda. Fai riferimento a queste informazioni quando scrivi la tua funzione Lambda.

> **Note**
>
> Il formato di input potrebbe cambiare senza che corrisponda una modifica in `messageVersion`. Il codice non dovrebbe generare errori se sono presenti nuovi campi.

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

Tieni conto delle ulteriori informazioni seguenti sui campi di evento:

- currentIntent: fornisce i campi `name`, `slots`, `slotDetails` e `confirmationStatus` dell'intento.

  `nluIntentConfidenceScore`è la certezza che Amazon Lex ha che l'intento attuale è quello che meglio corrisponde all'intento attuale dell'utente.

  `slots`è una mappa di nomi di slot, configurati per l'intento, per i valori degli slot che Amazon Lex ha riconosciuto nella conversazione con l'utente. Un valore di slot rimane null fino a quando l'utente non fornisca un valore.

  Il valore dello slot nell'evento di input potrebbe non corrispondere a uno dei valori configurati per lo slot. Ad esempio, se l'utente risponde al messaggio di richiesta "Quale colore preferisci per l'auto?" con «pizza», Amazon Lex restituirà «pizza» come valore dello slot. La funzione dovrebbe convalidare i valori per assicurare che questi abbiano senso nel contesto.

`slotDetails` fornisce ulteriori informazioni su un valore dello slot. La matrice `resolutions` contiene un elenco di valori aggiuntivi riconosciuti per lo slot. Ciascuno slot può avere un massimo di cinque valori.

Il campo `originalValue` contiene il valore che l'utente ha inserito per lo slot. Quando il tipo di slot è configurato per restituire il valore di risoluzione superiore come valore dello slot, `originalValue` potrebbe essere diverso dal valore del campo `slots`.

`confirmationStatus` fornisce la risposta dell'utente a un messaggio di richiesta di conferma, se disponibile. Ad esempio, se Amazon Lex chiede «Vuoi ordinare una pizza grande al formaggio? », in base alla risposta dell'utente, il valore di questo campo può essere`Confirmed`o`Denied`. In caso contrario, il valore di questo campo è `None`.

Se l'utente conferma l'intento, Amazon Lex imposta il campo su`Confirmed`. Se l'utente rifiuta l'intento, Amazon Lex imposta il valore su`Denied`.

Nella risposta di conferma, un'enunciazione utente potrebbe fornire aggiornamenti dello slot. Ad esempio, l'utente potrebbe dire "Sì, modifica dimensione a media". In questo caso, l'evento Lambda successivo presenterà il valore dello slot aggiornato,`PizzaSize`impostato su`medium`. Amazon Lex imposta il`confirmationStatus`a`None`, in quanto l'utente ha modificato alcuni dati dello slot, richiedendo alla funzione Lambda di eseguire la convalida dei dati utente.

- Intenti alternativi— Se abiliti i punteggi di fiducia, Amazon Lex restituisce fino a quattro intenti alternativi. Ogni intento include un punteggio che indica il livello di sicurezza di Amazon Lex che l'intento è l'intento corretto in base all'enunciato dell'utente.

Il contenuto degli intenti alternativi è lo stesso del contenuto della`currentIntent`. Per ulteriori informazioni, consultare Usare i punteggi di.

- bot: informazioni sul bot che ha elaborato la richiesta.
  - `name`: nome del bot che ha elaborato la richiesta.
  - `alias`: alias della versione del bot che ha elaborato la richiesta.
  - `version`: versione del bot che ha elaborato la richiesta.

- userId— Questo valore è fornito dall'applicazione client. Amazon Lex lo passa alla funzione Lambda.

- inputTranscript: testo utilizzato per elaborare la richiesta.

  Se si tratta di un input di testo, il campo `inputTranscript` contiene il testo inserito dall'utente.

  In caso di input di flusso audio, il campo `inputTranscript` contiene il testo estratto dal flusso audio. Questo è il testo che viene effettivamente elaborato per riconoscere gli intenti e i valori dello slot.

- invocationSource: per indicare perché Amazon Lex invoca la funzione Lambda imposta uno dei seguenti valori:
  - `DialogCodeHook`: Amazon Lex imposta questo valore per indirizzare la funzione Lambda all'inizializzazione della funzione e alla convalida dell'input di dati dell'utente.

    Quando l'intento è configurato per invocare una funzione Lambda come hook di codice di inizializzazione e di convalida, Amazon Lex invoca la funzione Lambda specificata su ciascun input dell'utente (enunciazione) dopo che Amazon Lex abbia compreso l'intento.

> **Note**
>
> Se l'intento non è chiaro, Amazon Lex non sarà in grado di invocare la funzione Lambda.

- `FulfillmentCodeHook`— Amazon Lex imposta questo valore per indirizzare la funzione Lambda all'adempimento di un intento.

  Se l'intento è configurato per invocare una funzione Lambda come hook di codice di adempimento, Amazon Lex imposta il`invocationSource`a questo valore solo quando dispone di tutti i dati dello slot necessari per adempiere l'intento.

  Nella configurazione dell'intento, puoi avere due funzioni Lambda separate per inizializzare e convalidare i dati utente e adempiere l'intento. Puoi inoltre utilizzare una funzione Lambda per entrambe le operazioni. In questo caso, la funzione Lambda può utilizzare`invocationSource`valore per seguire il percorso corretto del codice.

- outputDialogMode: per ogni input dell'utente, il client invia la richiesta ad Amazon Lex utilizzando una delle operazioni API di runtime,PostContentoPostText. Amazon Lex utilizza i parametri della richiesta per determinare se la risposta al client è di testo o vocale, quindi imposta il campo di conseguenza.

  La funzione Lambda può utilizzare queste informazioni per generare un messaggio appropriato. Ad esempio, se il client prevede una risposta vocale, la funzione Lambda può restituire Speech Synthesis Markup Language (SSML) al posto del testo.

- messageVersion: la versione del messaggio che identifica il formato dei dati di evento che giungono alla funzione Lambda e il formato previsto della risposta da parte di una funzione Lambda.

> **Note**
>
> Puoi configurare questo valore quando definisci un intento. Nell'implementazione corrente, è supportata solo la versione 1.0 dei messaggi. Pertanto, la console presuppone il valore predefinito di 1.0 e non mostra la versione del messaggio.

- sessionAttributes— Attributi di sessione specifici dell'applicazione che il client invia nella richiesta. Se vuoi che Amazon Lex li includa nella risposta al client, la funzione Lambda deve inviarli ad Amazon Lex nella risposta. Per ulteriori informazioni, consultaImpostazione degli attributi di sessione

- requestAttributes— Attributi specifici della richiesta che il client invia nella richiesta. Utilizza gli attributi di richiesta per inviare informazioni che non devono essere conservate per l'intera sessione. Se non ci sono attributi della richiesta, il valore sarà null. Per ulteriori informazioni, consultaImpostazione degli attributi di richiesta

- recentIntentSummaryView— Informazioni sullo stato di un intento. È possibile visualizzare informazioni sugli ultimi tre intenti utilizzati. Queste informazioni possono essere usate per impostare i valori nell'intento o per tornare a un intento precedente. Per ulteriori informazioni, consultare Gestione di sessioni con l'API di Amazon Lex.

- sentimentResponse— Il risultato dell'analisi di un sentiment di Amazon Comprehend dell'ultimo enunciato. È possibile utilizzare queste informazioni per gestire il flusso di conversazione del bot in base all'emozione espressa dall'utente. Per ulteriori informazioni, consultare Analisi delle emozioni.

- kendraResponse— Il risultato di una query su un indice Amazon Kendra. Presente solo nell'input a un hook di codice di adempimento e solo quando l'intento estende l'intento integrato `AMAZON.KendraSearchIntent`. Il campo contiene l'intera risposta della ricerca di Amazon Kendra. Per ulteriori informazioni, consultare AMAZON.KendraSearchIntent.

- Contesti attivi— Uno o più contesti attivi durante questo turno di una conversazione con l'utente.

- TimeToLive— Il periodo di tempo o il numero di turni nella conversazione con l'utente in cui il contesto rimane attivo.
- nome— il nome del contesto.
- parametriun elenco di coppie chiave/valore che contiene il nome e il valore degli slot dall'intento che ha attivato il contesto.

Per ulteriori informazioni, consultare Impostazione del contesto dell'intento.

Formato della risposta

Amazon Lex prevede una risposta da una funzione Lambda nel formato seguente:

```
{
  "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.
  }
}
```

La risposta è costituita da quattro campi. La`sessionAttributes`,`recentIntentSummaryView`, `activeContexts`si campi sono facoltativi,`dialogAction`è obbligatorio. Il contenuto del campo `dialogAction` dipende dal valore del campo `type`. Per dettagli, consultare dialogAction.

sessionAttributes

Facoltativo. Se includi il campo `sessionAttributes`, questo può essere vuoto. Se la funzione Lambda non restituisce gli attributi di sessione, l'ultimo notosessionAttributespassata tramite l'API o la funzione Lambda rimangono. Per maggiori informazioni, consulta le operazioni PostContent e PostText.

```
  "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"
  }
```

recentIntentSummaryView

Facoltativo. Se incluso, imposta i valori per uno o più intenti recenti. È possibile includere informazioni per un massimo di tre intenti. Ad esempio, puoi impostare i valori per gli intenti precedenti in base alle informazioni raccolte dall'intento corrente. Le informazioni contenute nel riepilogo devono essere valide per l'intento. Ad esempio, il nome dell'intento deve essere un intento nel bot. Se includi un valore di slot nella visualizzazione riepilogo, lo slot deve essere presente nell'intento. Se non includi la `recentIntentSummaryView` nella risposta, tutti i valori degli intenti recenti rimangono invariati. Per ulteriori informazioni, vedere l'operazione PutSession o il tipo di dati IntentSummary.

```
"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"
   }
 ]
```

Contesti attivi

Facoltativo. Se incluso, imposta il valore di uno o più contesti. Ad esempio, puoi includere un contesto per rendere idonei al riconoscimento uno o più intenti che hanno quel contesto come input idoneo per il riconoscimento nel prossimo turno della conversazione.

Tutti i contesti attivi che non sono inclusi nella risposta hanno i valori di time-to-live diminuiti e possono essere ancora attivi sulla richiesta successiva.

Se si specifica un time-to-live pari a 0 per un contesto incluso nell'evento di input, sarà inattivo nella richiesta successiva.

Per ulteriori informazioni, consultare Impostazione del contesto dell'intento.

dialogAction

Campo obbligatorio. La`dialogAction`indirizza Amazon Lex verso l'operazione successiva e descrive cosa prevedere da parte dell'utente dopo che Amazon Lex abbia restituito una risposta al client.

Il campo `type` indica l'operazione successiva. Determina inoltre gli altri campi che la funzione Lambda deve fornire come parte dell'`dialogAction`value.

- `Close`: informa Amazon Lex di non prevedere una risposta da parte dell'utente. Ad esempio, "L'ordine della tua pizza è stato registrato" non richiede una risposta.

Il campo `fulfillmentState` è obbligatorio. Amazon Lex utilizza questo valore per impostare`dialogState`nel campoPostContentoPostTextrisposta all'applicazione client. I campi `message` e `responseCard` sono facoltativi. Se non specifichi un messaggio, Amazon Lex utilizza il messaggio di saluto o il messaggio di prosecuzione configurato per l'intento.

```
"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`: informa Amazon Lex che l'utente deve fornire una risposta affermativa o negativa per confermare o negare l'intento corrente.

  È necessario includere i campi `intentName` e `slots`. Il campo `slots` deve contenere una voce per ciascuno slot riempito per l'intento specificato. Non è necessario includere una voce nel campo

`slots` per gli slot che non vengono riempiti. È necessario includere il campo `message` se il campo intento `confirmationPrompt` è nullo. Contenuti di`message`Il campo restituito dalla funzione Lambda ha la precedenza rispetto al`confirmationPrompt`specificato nell'intento. Il campo `responseCard` è facoltativo.

```
"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`: indirizza Amazon Lex a scegliere la prossima operazione in base alla configurazione del bot. Se la risposta non include alcun attributo di sessione, Amazon Lex mantiene gli attributi

esistenti. Per ottenere un valore null di uno slot, non occorre includere il campo slot nella richiesta. Si riceverà un'eccezione `DependencyFailedException` se la funzione di adempimento restituisce l'operazione di dialogo `Delegate` senza rimuovere alcuno slot.

I campi `kendraQueryRequestPayload` e `kendraQueryFilterString` sono facoltativi e utilizzati solo quando l'intento è derivato dall'intento integrato `AMAZON.KendraSearchIntent`. Per ulteriori informazioni, consulta 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`: informa Amazon Lex che l'utente deve rispondere con un'enunciazione contenente un intento. Ad esempio, "Voglio una pizza grande", che indica l'intento `OrderPizzaIntent`. L'enunciazione «grande», invece, non è sufficiente affinché Amazon Lex deduca l'intento dell'utente.

  I campi `message` e `responseCard` sono facoltativi. Se non fornisci un messaggio, Amazon Lex utilizza uno dei messaggi di richiesta di chiarimento del bot. Se non viene definito alcun prompt di chiarimento, Amazon Lex restituisce un'eccezione 400 Bad Request (Richiesta non valida).

```
{
  "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`: informa Amazon Lex che l'utente dovrebbe fornire un valore dello slot nella risposta.

  I campi `intentName`, `slotToElicit` e `slots` sono obbligatori. I campi `message` e `responseCard` sono facoltativi. Se non specifichi un messaggio, Amazon Lex utilizza uno dei messaggi di richiesta di sollecitazione dello slot configurati per lo slot.

```
 "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 eAWS LambdaPiani

La console di Amazon Lex offre bot di esempio (chiamati piani di bot) preconfigurati in modo da poter creare e testare rapidamente un bot nella console. Per ciascuno di questi piani di bot, sono offerti anche piani della funzione Lambda. Questi piani forniscono il codice di esempio funzionante con i bot corrispondenti. Puoi utilizzare questi piani per creare rapidamente un bot configurato con una funzione Lambda come hook di codice, oltre a testare la configurazione end-to-end, senza scrivere codice.

Puoi utilizzare i piani di bot di Amazon Lex seguenti e i corrispondentiAWS Lambdapiani della funzione come hook di codice per bot:

- Piani di Amazon Lex:`OrderFlowers`
  - AWS Lambdapiano —`lex-order-flowers-python`
- Piani di Amazon Lex:`ScheduleAppointment`
  - AWS Lambdapiano —`lex-make-appointment-python`
- Piani di Amazon Lex:`BookTrip`
  - AWS Lambdapiano —`lex-book-trip-python`

Per creare un bot utilizzando un piano e configurarlo affinché utilizzi una funzione Lambda come hook di codice, consultaEsercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console). Per esempi di utilizzo di altri piani, consulta Esempi aggiuntivi: creazione di bot Amazon Lex.

Aggiornamento di un blueprint per una specifica impostazione locale

Se si utilizza un blueprint in una lingua diversa dall'inglese (USA) (en-US), è necessario aggiornare il nome di qualsiasi intenzione per includere le impostazioni locali. Ad esempio, se si sta utilizzando il`OrderFlowers`piano, devi eseguire le seguenti operazioni.

- Trova l'`dispatch`funzione vicino alla fine del codice funzione Lambda.
- Nella`dispatch`funzione, aggiorna il nome dell'intento per includere le impostazioni locali utilizzate. Ad esempio, se si utilizza la lingua inglese (australiana) (en-AU), modificare la riga:

  ```
  if intent_name == 'OrderFlowers':
  ```

  to

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

Altri blueprint utilizzano altri nomi di intento, dovrebbero essere aggiornati come sopra prima di utilizzarli.

Distribuzione di bot Amazon Lex

Questa sezione fornisce esempi di distribuzione di bot di Amazon Lex su differenti piattaforme di messaggistica e in applicazioni per dispositivi mobili.

Argomenti

Distribuzione di un bot Amazon Lex su una piattaforma di messaggistica

Questa sezione descrive come distribuire bot di Amazon Lex sulle piattaforme di messaggistica Facebook, Slack e Twilio.

> **Note**
>
> Per l'archiviazione delle configurazioni di Facebook, Slack o Twilio, Amazon Lex utilizzaAWS Key Management Servicechiavi gestite dal cliente per crittografare le informazioni. La prima volta che crei un canale a una di queste piattaforme di messaggistica, Amazon Lex crea una chiave gestita dal cliente predefinita (`aws/lex`). In alternativa, puoi creare la tua chiave gestita dal cliente conAWS KMS. Questa opzione ti offre una maggiore flessibilità, tra cui la possibilità di creare, ruotare e disabilitare le chiavi. Puoi anche definire controlli di accesso e controllare le chiavi di crittografia utilizzate per proteggere i dati. Per ulteriori informazioni, consulta la Guida per gli sviluppatori di AWS Key Management Service.

Quando una piattaforma di messaggistica invia una richiesta ad Amazon Lex, include informazioni specifiche della piattaforma come attributo di richiesta per la funzione Lambda. Utilizza questi attributi per personalizzare il comportamento del bot. Per ulteriori informazioni, consultare Impostazione degli attributi di richiesta.

Tutti gli attributi acquisiscono il nomespace `x-amz-lex:` come prefisso. Ad esempio, l'attributo `user-id` viene denominato `x-amz-lex:user-id`. Oltre agli attributi specifici di una determinata piattaforma, vi sono alcuni attributi comuni che vengono inviati da tutte le piattaforme di

messaggistica. La tabella seguente elenca gli attributi di richiesta che le piattaforme di messaggistica inviano alla funzione Lambda del tuo bot.

Attributi di richiesta comuni

Attributo	Descrizione
`channel-id`	L'identificatore dell'endpoint del canale di Amazon Lex.
`channel-name`	Il nome di canale di Amazon Lex.
`channel-type`	Uno dei seguenti valori: • `Facebook` • `Kik` • `Slack` • `Twilio-SMS`
`webhook-endpoint-url`	L'endpoint Amazon Lex per il canale.

Attributi di richiesta di Facebook

Attributo	Descrizione
`user-id`	L'identificatore Facebook del mittente. Consulta https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.
`facebook-page-id`	L'identificatore di pagina Facebook del destinatario. Consulta https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.

Attributi di richiesta di Kik

Attributo	Descrizione
`kik-chat-id`	Identificatore della conversazione in cui è coinvolto il tuo bot. Per ulteriori informazioni, consulta https://dev.kik.com/#/docs/messaging#message-formats.
`kik-chat-type`	Il tipo di conversazione da cui è stato creato il messaggio. Per ulteriori informazioni, consulta https://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-id`	Un UUID che identifica il messaggio. Per ulteriori informazioni, consulta https://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-type`	Il tipo di messaggio. Per ulteriori informazioni, consulta https://dev.kik.com/#/docs/messaging#message-types.

Attributi di richiesta di Twilio

Attributo	Descrizione
`user-id`	Il numero di telefono del mittente ("From"). Consulta https://www.twilio.com/docs/api/rest/message.
`twilio-target-phone-number`	Il numero di telefono del destinatario ("To"). Consulta https://www.twilio.com/docs/api/rest/message.

Attributi di richiesta di Slack

Attributo	Descrizione
`user-id`	L'identificatore utente di Slack. Consulta https://api.slack.com/types/user.
`slack-team-id`	L'identificatore del team che ha inviato il messaggio. Consulta https://api.slack.com/methods/team.info.

Attributo	Descrizione
`slack-bot-token`	Il token dello sviluppatore che fornisce al bot l'accesso alle API di Slack. Consulta https://api.slack.com/docs/token-types.

Integrazione di un Amazon Lex Bot con Facebook Messenger

Questo esercizio mostra come integrare Facebook Messenger con il tuo bot Amazon Lex. Completa la seguente procedura:

1. Crea un bot Amazon Lex
2. Creazione di un'applicazione Facebook
3. Integra Facebook Messenger con il tuo bot Amazon Lex
4. Convalida dell'integrazione

Argomenti

- Fase 1: Creazione di un Amazon Lex Bot
- Fase 2: Creazione di un'applicazione Facebook
- Passaggio 3: integra Facebook Messenger con Amazon Lex Bot
- Fase 4: testa l'integrazione

Fase 1: Creazione di un Amazon Lex Bot

Se non si dispone di un bot Amazon Lex, crearne uno. In questo argomento si presuppone che tu stia utilizzando il bot creato nell'esercizio 1 "Nozioni di base". Tuttavia, puoi utilizzare uno qualsiasi dei bot di esempio forniti in questa guida. Per l'esercizio 1 "Nozioni di base", consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).

1. Crea un bot Amazon Lex. Per istruzioni, consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).
2. Distribuisci il bot e crea un alias. Per istruzioni, consulta Esercizio 3. Pubblicazione di una versione e creazione di un alias.

Fase 2: Creazione di un'applicazione Facebook

Sul portale per gli sviluppatori di Facebook, crea un'applicazione e una pagina Facebook. Per istruzioni, consulta il documento Quick Start nella documentazione della piattaforma di Facebook Messenger. Prendi nota dei dati seguenti:

- L'App Secret (Chiave segreta app) per l'applicazione di Facebook
- Il Page Access Token (Token accesso pagina) per la pagina di Facebook

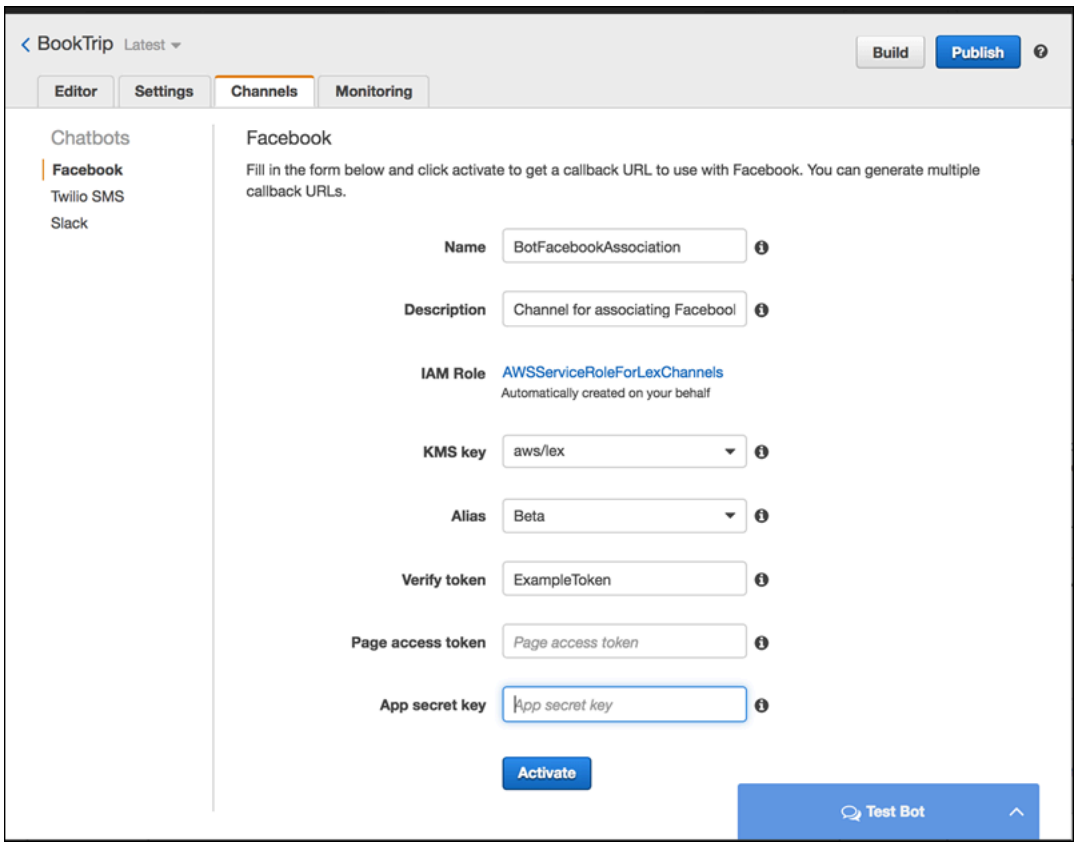
Passaggio 3: integra Facebook Messenger con Amazon Lex Bot

In questa sezione, integri Facebook Messenger con il tuo bot Amazon Lex.

Una volta completata questa operazione, la console offre un URL di callback. Prendi nota di questo URL.

Per integrare Facebook Messenger con il tuo bot

1. a. Accedere aAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
   b. Scegli il tuo bot Amazon Lex.
   c. Seleziona Channels (Canali).
   d. Seleziona Facebook nella sezione Chatbots (Chatbot). La console visualizza la pagina di integrazione di Facebook.
   e. In questa pagina, effettua quanto segue:
      - Digita il seguente nome: `BotFacebookAssociation`.
      - Per KMS key (Chiave KMS), scegli aws/lex.
      - Per l'opzione Alias, seleziona l'alias del bot.
      - Per Verify token (Verifica token), digita un token. Questo può essere qualsiasi stringa vuoi (ad esempio, `ExampleToken`). Utilizzerai questo token in un secondo momento nel portale per gli sviluppatori di Facebook, al momento della configurazione del webhook.
      - Per l'opzione Page access token (Token accesso pagina), digita il token che hai ottenuto alla Fase 2 da Facebook.
      - Per App secret key (Chiave segreta app), digita la chiave che hai ottenuto alla Fase 2 da Facebook.



f. Seleziona Activate (Attiva).

La console crea l'associazione del canale del bot e restituisce un URL di callback. Prendi nota di questo URL.

2. Sul portale per gli sviluppatori di Facebook, scegli la tua applicazione.
3. Seleziona il prodotto Messenger, quindi Setup webhooks (Webhook di configurazione) nella sezione Webhooks (Webhook) della pagina.

   Per istruzioni, consulta il documento Quick Start nella documentazione della piattaforma di Facebook Messenger.
4. Nella pagina webhook della procedura guidata per la registrazione, effettua quanto segue:

   - Per Callback URL, digita l'URL di callback fornito nella console Amazon Lex in precedenza nella procedura.
   - Per Verify Token, digita lo stesso token che hai usato in Amazon Lex.
   - Seleziona Subscription Fields (Campi registrazione) (messages, messaging_postbacks e messaging_optins).

- Seleziona Verify and Save (Verifica e salva). Questo dà inizio a una stretta di mano tra Facebook e Amazon Lex.

5. Abilita l'integrazione degli webhook. Seleziona la pagina creata, quindi scegli subscribe (effettua registrazione).

> **Note**
>
> Se aggiorni o crei nuovamente un webhook, annulla la registrazione ed eseguila nuovamente per la pagina.

Fase 4: testa l'integrazione

Ora puoi iniziare una conversazione da Facebook Messenger con il tuo bot Amazon Lex.

1. Apri la tua pagina Facebook e scegli Messaggio.
2. Nella finestra di Messenger, utilizza le stesse enunciazioni di prova fornite in Fase 1: creazione Amazon Lex bot (console).

Integrazione di un bot Amazon Lex con Kik

Questo esercizio fornisce istruzioni per l'integrazione di un bot Amazon Lex con l'applicazione di messaggistica Kik. Completa la seguente procedura:

1. Crea un bot Amazon Lex.
2. Crea un bot di Kik utilizzando l'app e il sito Web di Kik.
3. Integra il tuo bot Amazon Lex con il bot Kik utilizzando la console Amazon Lex.
4. Partecipa a una conversazione con il tuo bot Amazon Lex usando Kik per testare l'associazione tra il tuo bot Amazon Lex e Kik.

Argomenti

Fase 1: Creazione di un Amazon Lex Lex Bot

Se non disponi già di un bot Amazon Lex, devi crearne uno. In questo argomento si presuppone che tu stia utilizzando il bot creato nell'esercizio 1 "Nozioni di base". Tuttavia, puoi utilizzare uno qualsiasi dei bot di esempio forniti in questa guida. Per l'esercizio 1 "Nozioni di base", consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console)

1. Crea un bot Amazon Lex. Per istruzioni, consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).
2. Distribuisci il bot e crea un alias. Per istruzioni, consulta Esercizio 3. Pubblicazione di una versione e creazione di un alias.

Fase successiva

Fase 2: crea un bot di Kik

Fase 2: crea un bot di Kik

In questa fase puoi utilizzare l'interfaccia utente di Kik per creare un bot di Kik. Utilizzi le informazioni generate durante la creazione del bot per collegarlo al tuo bot Amazon Lex.

1. Se non lo hai già fatto, scarica e installa l'app di Kik e crea un account di Kik. Se disponi di un account, effettua l'accesso.
2. Apri il sito Web di Kik all'indirizzo https://dev.kik.com/. Lascia la finestra del browser aperta.
3. Nell'app di Kik, seleziona l'icona a forma di ingranaggio per aprire le impostazioni, quindi seleziona Your Kik Code (Il tuo codice di Kik).
4. Scansiona il codice di Kik sul sito Web di Kit per aprire la chatbot di Botsworth. Seleziona Yes (Sì) per aprire il pannello di controllo del bot.
5. Nell'app di Kik, seleziona Create a Bot (Crea un bot). Segui le istruzioni per creare il bot di Kik.
6. Una volta creato il bot, seleziona Configuration (Configurazione) nel browser. Assicurati che sia selezionato il nuovo bot.
7. Annota il nome del bot e la chiave API per la sezione successiva.

Fase successiva

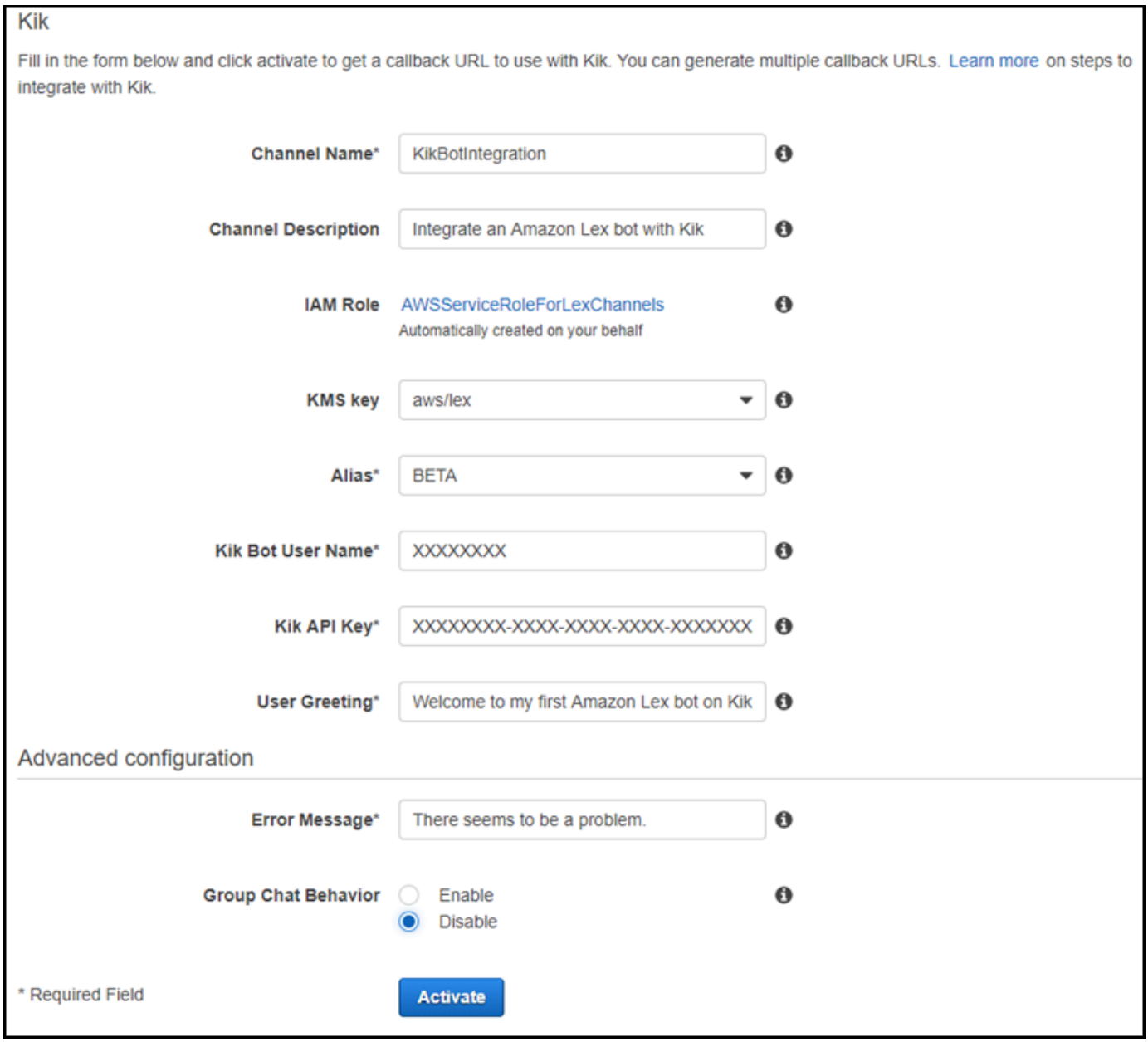
Passaggio 3: integra Kik Bot con Amazon Lex Bot

Passaggio 3: integra Kik Bot con Amazon Lex Bot

Ora che hai creato un bot Amazon Lex e un bot Kik, sei pronto per creare un'associazione di canali tra di loro in Amazon Lex. Quando l'associazione è attivata, Amazon Lex imposta automaticamente un URL di callback con Kik.

1. Accedere alla Console di gestione AWS e aprire la console Amazon Lex e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot Amazon Lex creato nella Fase 1.
3. Seleziona la scheda Channels (Canali).
4. Nella sezione Channels (Canali), seleziona Kik.
5. Nella pagina di Kik, inserisci quanto segue:

   - Digita un nome. Ad esempio, `BotKikIntegration`.
   - Digita una descrizione.
   - Seleziona "aws/lex" dall'elenco a discesa KMS key (Chiave KMS).
   - In Alias, seleziona un alias nell'elenco a discesa.
   - In Kik bot user name (Nome utente bot di Kik), digita il nome assegnato al bot su Kik.
   - In Kik API key (Chiave API di Kik) digita la chiave API assegnata al bot su Kik.
   - In User greeting (Saluto utente), digita il saluto che vuoi che il bot invii la prima volta che un utente conversa con lui.
   - In Error message (Messaggio di errore), inserisci un messaggio di errore visualizzato dall'utente quando parte della conversazione non viene compresa.
   - In Group chat behavior (Comportamento chat di gruppo), seleziona una delle opzioni:
     - Enable (Abilita): abilita l'intero gruppo di chat ad interagire con il bot in un'unica conversazione.
     - Disable (Disabilita): limita la conversazione a un utente nel gruppo di chat.
   - Seleziona Activate (Attiva) per creare l'associazione e collegarla al bot di Kik.



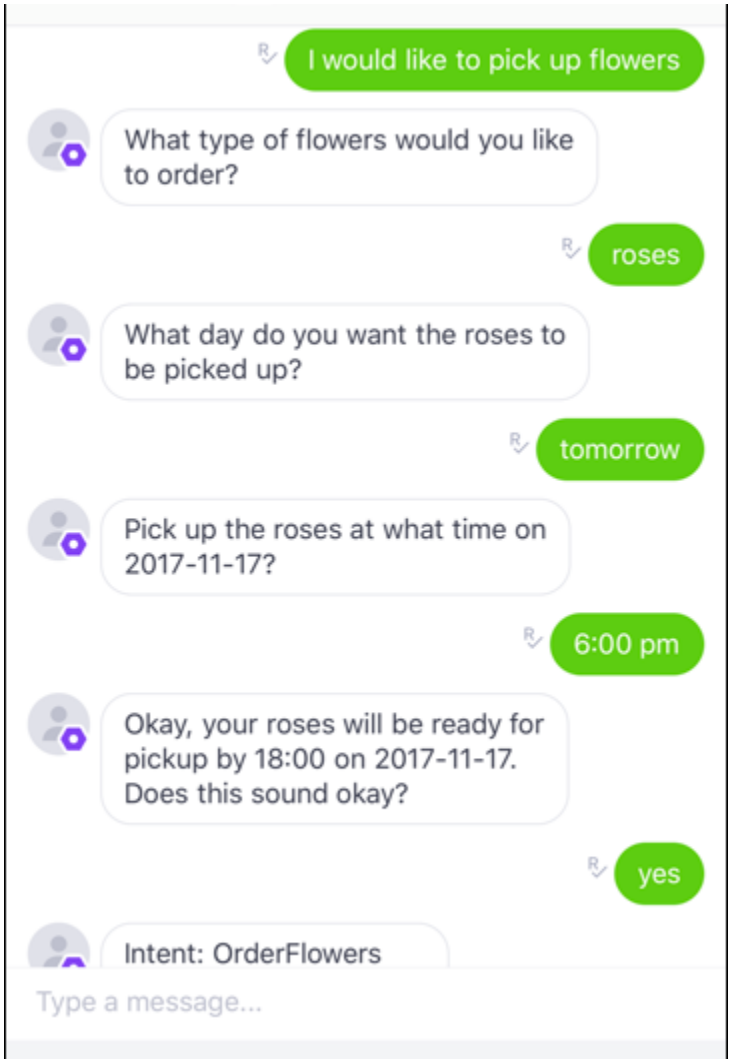
Fase successiva

[Fase 4: testa l'integrazione]

Fase 4: testa l'integrazione

Ora che hai creato un'associazione tra il tuo bot Amazon Lex e Kik, puoi utilizzare l'app Kik per testare l'associazione.

1. Avvia l'app di Kik ed effettua l'accesso. Seleziona il bot creato.
2. Puoi testare il bot come riportato di seguito:



   Quando inserisci ogni frase, il tuo bot Amazon Lex risponderà tramite Kik con il prompt che hai creato per ogni slot.

Integrazione di un bot Amazon Lex con Slack

Questo esercizio fornisce istruzioni per l'integrazione di un bot Amazon Lex con l'applicazione di messaggistica Slack. Completa la seguente procedura:

1. Crea un bot Amazon Lex.
2. Crea un'applicazione di messaggistica Slack.
3. Integra l'applicazione Slack con il tuo bot Amazon Lex.

4. Testa l'integrazione interagendo con il tuo bot Amazon Lex. Invia messaggi con l'applicazione Slack e completa una verifica in una finestra del browser.

Argomenti

Fase 1: Creazione di un bot Amazon Lex

Se non disponi ancora di un bot Amazon Lex, crearne e distribuirlo. In questo argomento si presuppone che tu stia utilizzando il bot creato nell'esercizio 1 "Nozioni di base". Tuttavia, puoi utilizzare uno qualsiasi dei bot di esempio forniti in questa guida. Per l'esercizio 1 "Nozioni di base", consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console)

1. Crea un bot Amazon Lex. Per istruzioni, consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).
2. Distribuisci il bot e crea un alias. Per istruzioni, consulta Esercizio 3. Pubblicazione di una versione e creazione di un alias.

Fase successiva

Fase 2: registrati a Slack e crea un team Slack

Fase 2: registrati a Slack e crea un team Slack

Registra un account Slack e crea un team Slack. Per istruzioni, consulta Utilizzo di Slack. Nella prossima sezione dovrai creare un'applicazione Slack che tutto il team Slack possa installare.

Fase successiva

Fase 3: crea un'applicazione Slack

Fase 3: crea un'applicazione Slack

In questa sezione effettuerai le operazioni seguenti:

1. Crea un'applicazione di Slack nella console API Slack
2. Configura l'applicazione per aggiungere messaggi interattivi al tuo bot:

Al termine della sessione riceverai le credenziali dell'applicazione (ID client, Client Secret e token di verifica). Nella sezione successiva, utilizzi queste informazioni per configurare l'associazione dei canali bot nella console Amazon Lex.

1. Accedi alla console API Slack su http://api.slack.com .
2. Crea un'applicazione di .

   Dopo aver creato correttamente l'applicazione, Slack visualizza la pagina delle informazioni di base per l'applicazione.
3. Configura le funzioni dell'applicazione nel modo seguente:

   - Nel menu a sinistra, seleziona Interanze e scorciatoie.
     - Scegli il toggle per attivare i componenti interattivi.
     - Nella casella Request URL (URL richiesta), specifica un URL valido, Per esempio, è possibile utilizzare **`https://slack.com`**.

       > **Note**
       >
       > Per il momento, immetti un URL valido per ottenere il token di verifica che ti servirà nella fase successiva. Aggiornerai questo URL dopo aver aggiunto l'associazione al canale bot nella console Amazon Lex.

     - Seleziona Salva modifiche.
4. Nel menu a sinistra, in Settings (Impostazioni), seleziona Basic Information (Informazioni di base). Prendi nota delle seguenti credenziali dell'applicazione:

   - ID client
   - Client Secret
   - Token di verifica

Fase successiva

Fase 4: Integrazione dell'applicazione Slack con Amazon Lex Bot

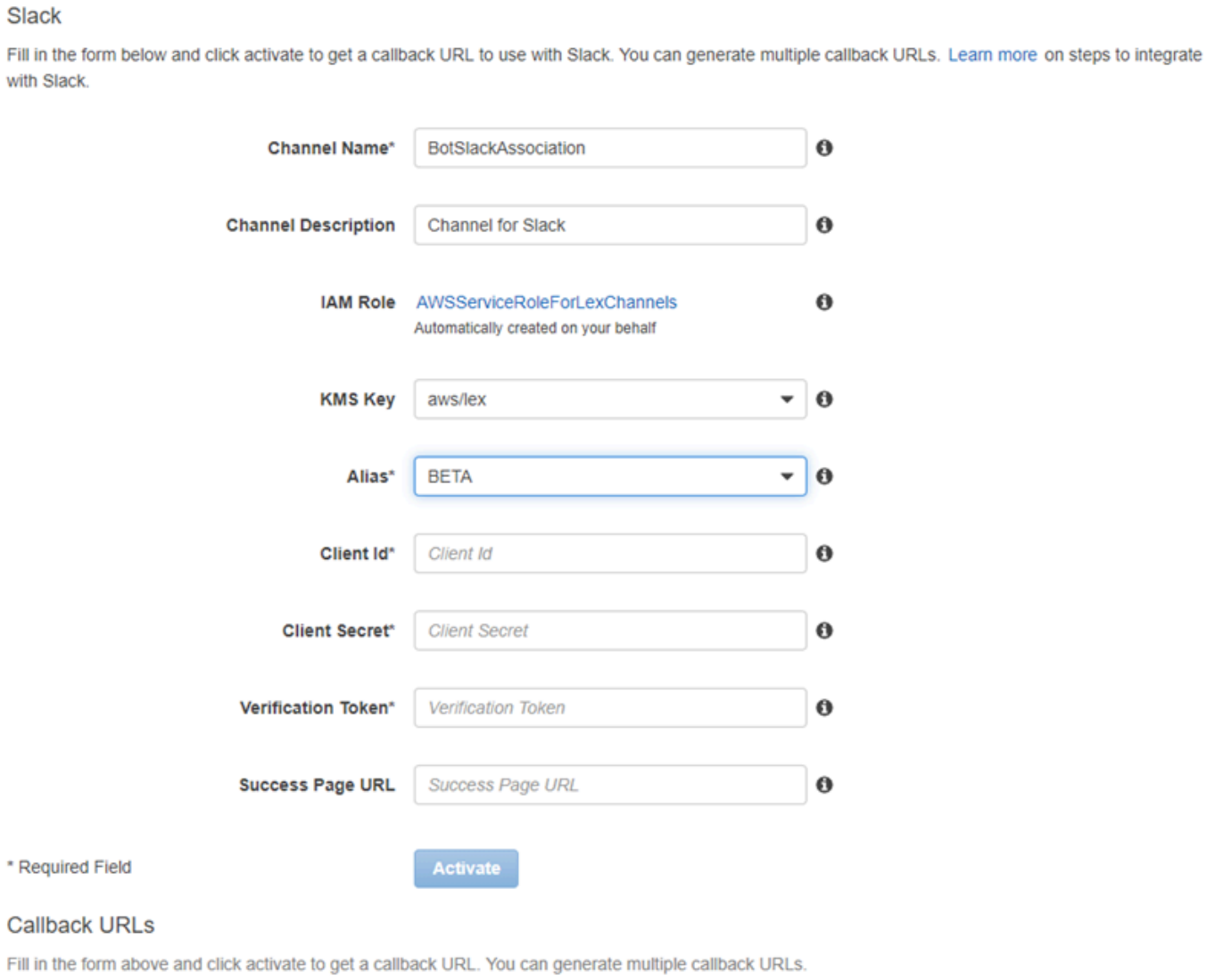
Fase 4: Integrazione dell'applicazione Slack con Amazon Lex Bot

Ora che disponi delle credenziali dell'applicazione Slack, puoi integrare l'applicazione con il tuo bot Amazon Lex. Per associare l'applicazione Slack al tuo bot, aggiungi un'associazione di canali bot in Amazon Lex.

Nella console Amazon Lex, attiva un'associazione di canali bot per associare il bot alla tua applicazione Slack. Quando l'associazione al canale bot è attivata, Amazon Lex restituisce due URL (Postback URL e URL OAuth). Prendi nota di questi URL perché ti serviranno più avanti.

Per integrare l'applicazione Slack con il tuo bot Amazon Lex

1. Accedere alla Console di gestione AWS e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Accedere al bot Amazon Lex creato nella Fase 1.
3. Seleziona la scheda Channels (Canali).
4. Nel menu a sinistra seleziona Slack.
5. Nella pagina Slack fornisci i seguenti dati:
   - Digita un nome. Ad esempio, `BotSlackIntegration`.
   - Seleziona "aws/lex" dall'elenco a discesa KMS key (Chiave KMS).
   - Per l'opzione Alias, seleziona l'alias del bot.
   - Digita l'ID client, il Client Secret e il token di verifica dalla fase precedente, perché sono le credenziali dell'applicazione Slack.



6. Seleziona Activate (Attiva).

   La console crea l'associazione di canali del bot e restituisce due URL (l'URL Postback e l'URL OAuth). Prendine nota. Nella sezione successiva, aggiorna la configurazione dell'applicazione Slack per utilizzare questi endpoint nel modo seguente:

   - L'URL Postback è l'endpoint del bot Amazon Lex che ascolta gli eventi di Slack. Utilizza questo URL:
     - Come URL di richiesta nella funzione Event Subscriptions (Abbonamenti a eventi) dell'applicazione Slack.

- Per sostituire il valore del segnaposto per l'URL di richiesta nella funzione Interactive Messages (Messaggi interattivi) dell'applicazione Slack.
- L'URL OAuth è l'endpoint del bot Amazon Lex per una stretta di mano OAuth con Slack.

Fase successiva

Fase 5: completa l'integrazione con Slack

Fase 5: completa l'integrazione con Slack

In questa sezione, utilizza la console API Slack per completare l'integrazione dell'applicazione Slack.

1. Accedi alla console API Slack su http://api.slack.com. Seleziona l'app che hai creato in Fase 3: crea un'applicazione Slack.
2. Aggiorna la funzione OAuth & Permissions (OAuth e autorizzazioni) nel modo seguente:
   a. Nel menu a sinistra, seleziona OAuth & Permissions (OAuth e autorizzazioni).
   b. Nella sezione URL di reindirizzamento, aggiungi l'URL OAuth fornito da Amazon Lex nel passaggio precedente. Seleziona Add a new Redirect URL (Aggiungi un nuovo URL di reindirizzamento), quindi Save URLs (Salva URL).
   c. Nella sezione Bot Token Scopes, aggiungi due autorizzazioni con il pulsante Aggiungi un ambito OAuth. Filtra l'elenco con il seguente testo:
      - **`chat:write`**
      - **`team:read`**
3. Aggiorna la funzionalità Interattività e scorciatoie aggiornando il valore dell'URL di richiesta all'URL di postback fornito da Amazon Lex nel passaggio precedente. Inserire l'URL del postback; salvato nella fase 4, quindi scegliere Save Changes (Salva cambiamenti).
4. Iscriviti alla funzione Event Subscriptions (Abbonamenti a eventi) nel modo seguente:
   - Abilita gli eventi scegliendo l'opzione On.
   - Imposta il valore dell'URL della richiesta sull'URL di postback fornito da Amazon Lex nel passaggio precedente.
   - Nella sezione Subscribe to Bot Events (Iscriviti a eventi bot), iscriviti all'evento bot `message.im` per abilitare la messaggistica diretta tra l'utente finale e il bot Slack.
   - Salvare le modifiche.

5. Abilita l'invio di messaggi dalla scheda messaggi come segue:

   - Nel menu a sinistra, seleziona App Home.
   - Nella sezione Mostra schede, scegli Consenti agli utenti di inviare comandi e messaggi Slash dalla scheda messaggi.

Fase successiva

Fase 6: verifica l'integrazione

Fase 6: verifica l'integrazione

Ora usa una finestra del browser per testare l'integrazione di Slack con il tuo bot Amazon Lex.

1. Seleziona Manage Distribution (Gestisci la distribuzione) in Settings (Impostazioni). Seleziona Add to Slack (Aggiungi a Slack) per installare l'applicazione. Autorizza il bot a rispondere ai messaggi.
2. Sarai reindirizzato al team Slack. Nel menu a sinistra, nella sezione Direct Messages (Messaggi diretti), seleziona il tuo bot. Se non lo trovi, seleziona il simbolo "più" (+) accanto all'opzione Direct Messages (Messaggi diretti) per cercarlo.
3. Partecipa a una chat con la tua applicazione Slack, che è collegata al bot Amazon Lex. Adesso il tuo bot risponderà ai messaggi.

   Se hai creato il bot utilizzando l'esercizio 1 "Nozioni di base", puoi utilizzare le conversazioni di esempio che trovi nell'esercizio. Per ulteriori informazioni, consulta Passaggio 4: aggiungere la funzione Lambda come Code Hook (console).

Integrazione di un Amazon Lex Bot con SMS programmabili Twilio

Questo esercizio fornisce istruzioni per l'integrazione di un bot Amazon Lex con il servizio di messaggistica semplice (SMS) Twilio. Completa la seguente procedura:

1. Crea un bot Amazon Lex
2. Integra gli SMS programmabili Twilio con il tuo bot Amazon Lex
3. Interagisci con il bot Amazon Lex testando la configurazione utilizzando il servizio SMS sul tuo telefono cellulare
4. Esecuzione del test dell'integrazione

Argomenti

Fase 1: Creazione di un Amazon Lex Bot

Se non disponi già di un bot Amazon Lex, crearne uno. In questo argomento si presuppone che tu stia utilizzando il bot creato nell'esercizio 1 "Nozioni di base". Tuttavia, puoi utilizzare uno qualsiasi dei bot di esempio forniti in questa guida. Per l'esercizio 1 "Nozioni di base", consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).

1. Crea un bot Amazon Lex. Per istruzioni, consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).
2. Distribuisci il bot e crea un alias. Per istruzioni, consulta Esercizio 3. Pubblicazione di una versione e creazione di un alias.

Fase 2: Creazione di un account Twilio SMS

Effettua la registrazione di un account Twilio e registra le seguenti informazioni relative all'account:

- SID ACCOUNT
- TOKEN AUTORIZ

Per le istruzioni sulla registrazione, consulta la pagina https://www.twilio.com/console.

Fase 3: Integrazione dell'endpoint del servizio di messaggistica Twilio con Amazon Lex Bot

Per integrare Twilio con il tuo bot Amazon Lex

1. Per associare il bot Amazon Lex al tuo endpoint SMS programmabile Twilio, attiva l'associazione dei canali del bot nella console Amazon Lex. Quando l'associazione al canale bot è stata attivata, Amazon Lex restituisce un URL di callback. Registra questo URL di callback poiché ne avrai bisogno in un secondo momento.

a. Accedere aAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.

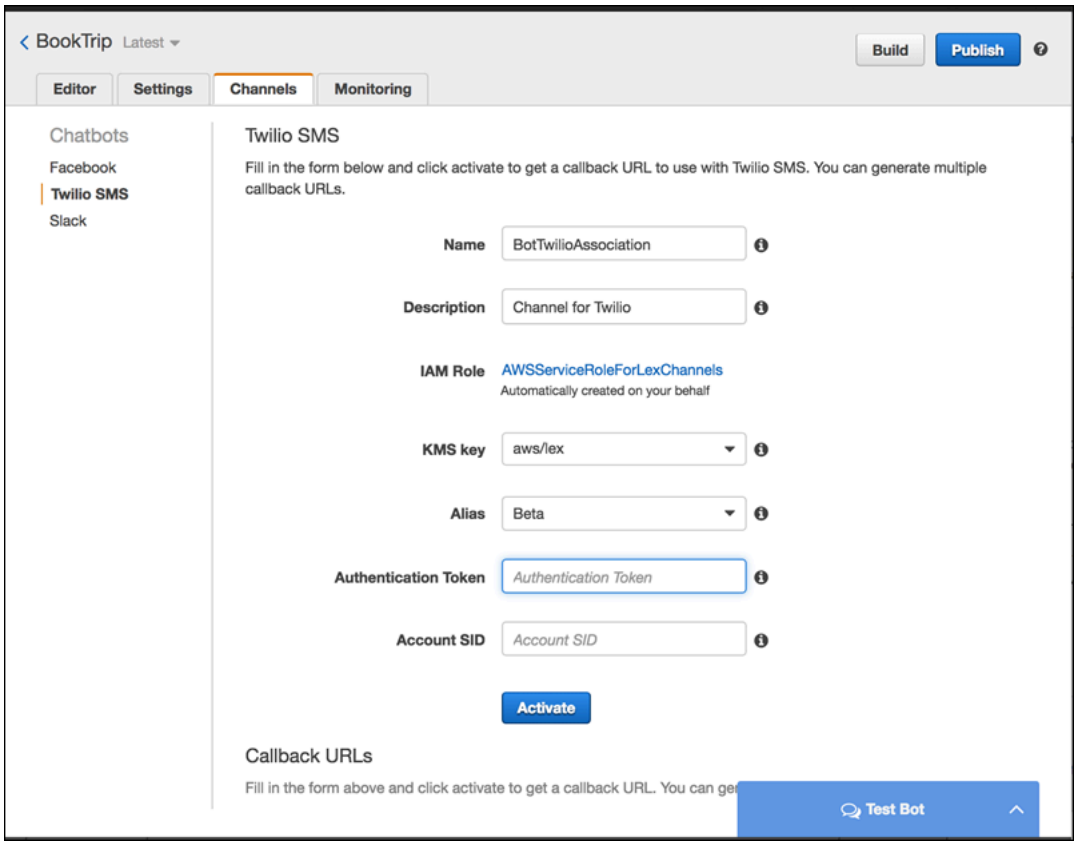
b. Scegli il bot Amazon Lex creato nella Fase 1.

c. Seleziona la scheda Channels (Canali).

d. Nella sezione Chatbots (Chatbot), seleziona Twilio SMS.

e. Nella pagina Twilio SMS, fornisci le seguenti informazioni:

- Digita un nome. Ad esempio, `BotTwilioAssociation`.
- Seleziona "aws/lex" da KMS key (Chiave KMS).
- Per l'opzione Alias, seleziona l'alias del bot.
- Per Authentication Token (Token di autenticazione), digita il TOKEN AUTORIZ per il tuo account Twilio.
- Per Account SID (SID account), digita il SID ACCOUNT per il tuo account Twilio.



f. Seleziona Activate (Attiva).

La console crea l'associazione del canale del bot e restituisce un URL di callback. Registra l'URL.

2. Sulla console Twilio, collega l'endpoint SMS Twilio al bot Amazon Lex.

   a. Accedi alla console di Twilio all'indirizzo https://www.twilio.com/console.

   b. Se non disponi di un endpoint di Twilio SMS, crealo.

   c. Aggiorna la configurazione delle impostazioni in entrata del servizio di messaggistica impostando il valore REQUEST URL sull'URL di callback fornito da Amazon Lex nel passaggio precedente.

Fase 4: testa l'integrazione

Utilizza il tuo cellulare per testare l'integrazione tra Twilio SMS e il tuo bot.

Per eseguire il test dell'integrazione

1. Accedi alla console di Twilio all'indirizzo https://www.twilio.com/console ed effettua quanto segue:

   a. Verificare di avere un numero Twilio associato al servizio di messaggistica nella sezione Manage Numbers (Gestione numeri).

      Invii messaggi a questo numero e interagisci via SMS con il bot Amazon Lex dal tuo telefono cellulare.

   b. Verifica che il tuo telefono cellulare sia elencato come ID chiamante verificato.

      In caso contrario, segui le istruzioni sulla console Twilio per abilitare il telefono cellulare che intendi utilizzare per il test.

      Ora puoi utilizzare il tuo telefono cellulare per inviare messaggi all'endpoint SMS Twilio, che è mappato al bot Amazon Lex.

2. Utilizzando il tuo cellulare, invia messaggi al numero Twilio.

   Il bot Amazon Lex risponde. Se hai creato il bot utilizzando l'esercizio 1 "Nozioni di base", puoi utilizzare le conversazioni di esempio che trovi nell'esercizio. Per ulteriori informazioni, consulta Passaggio 4: aggiungere la funzione Lambda come Code Hook (console).

Distribuzione di un bot Amazon Lex in applicazioni per dispositivi mobili

Utilizzo diAWS Amplify, puoi integrare i tuoi bot Amazon Lex con applicazioni per dispositivi mobili o Web. Per ulteriori informazioni, consultaInterazioni — Guida di base sunellaAWS AmplifyDocumenti.

Importazione ed esportazione di bot, intenti e tipi di slot Amazon Lex

Puoi importare o esportare un bot, un intento o un tipo di slot. Ad esempio, se vuoi condividere un bot con un collega di un altro account AWS, puoi esportarlo e poi inviarlo. Se desideri aggiungere più enunciazioni a un bot, puoi esportarlo, aggiungere le enunciazioni, quindi importarlo di nuovo nel tuo account.

È possibileesportarebot, intenti e tipi di slot in Amazon Lex (per condividerli o modificarli) o in un formato compatibile con una competenza di Alexa. È possibileimportaresolo in formato Amazon Lex.

Quando esegui l'esportazione di una risorsa, devi esportarla in un formato compatibile con il servizio scelto per l'esportazione, Amazon Lex o il kit di competenze Alexa. Se esporti un bot nel formato Amazon Lex, puoi reimportarlo nel tuo account, o un utente Amazon Lex in un altro account può importarlo nel proprio account. Puoi anche esportare un bot in un formato compatibile con una competenza di Alexa. Quindi puoi importare il bot utilizzando il kit di competenze di Alexa per rendere il tuo bot disponibile con Alexa. Per ulteriori informazioni, consultare Esportazione su una competenza di Alexa.

Quando esporti un bot, un intento o un tipo di slot, le risorse vengono scritte su un file JSON. Per esportare un bot, un intento o un tipo di slot, puoi utilizzare la console di Amazon Lex o ilGetExportoperazione. Importazione di un bot, un intento o un tipo di slot tramite StartImport.

Argomenti

- Esportazione e importazione in formato Amazon Lex
- Esportazione su una competenza di Alexa

Esportazione e importazione in formato Amazon Lex

Per esportare bot, intenti e tipi di slot da Amazon Lex con l'intenzione di importarli in Amazon Lex, dovrai creare un file JSON in formato Amazon Lex. Puoi modificare le tue risorse in questo file e importarle di nuovo in Amazon Lex. Ad esempio, puoi aggiungere enunciazioni a un intento, quindi importare l'intento modificato nel tuo account. Puoi inoltre utilizzare il formato JSON per condividere una risorsa. Ad esempio, puoi esportare un bot da una regione AWS e poi importarlo in un'altra regione. In alternativa, puoi inviare il file JSON a un collega per condividere un bot.

Argomenti

Esportazione in formato Amazon Lex

Esporta i tuoi bot, intenti e tipi di slot di Amazon Lex in un formato che puoi importare su unAWSconto. Puoi esportare le seguenti risorse:

- Un bot, inclusi tutti gli intenti e i tipi di slot personalizzati utilizzati dal bot
- Un intento, inclusi tutti i tipi di slot personalizzati utilizzati dall'intento
- Un tipo di slot personalizzato, inclusi tutti i valori per il tipo di slot

Puoi esportare solo una versione numerata di una risorsa. Non è possibile esportare una versione `$LATEST` di una risorsa.

L'esportazione è un processo asincrono. Una volta completata l'esportazione, è possibile ottenere un URL prefirmato da Amazon S3. L'URL fornisce il percorso di un archivio .zip che contiene la risorsa esportata in formato JSON.

Per esportare bot, intenti o tipi di slot personalizzati, puoi utilizzare la console o l'operazione GetExport.

Il processo di esportazione di un bot, di un intento o di un tipo di slot è lo stesso. Nelle seguenti procedure, sostituisci l'intento o il tipo di slot al bot.

Esportazione di un bot

Per esportare un bot

1. Accedere alla Console di gestione AWS e aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegli Bots (Bot), quindi seleziona il bot da esportare.
3. Nel menu Actions (Operazioni), scegli Export (Esporta).
4. Nella finestra di dialogo Export Bot (Esporta Bot), scegli la versione del bot da esportare. Per Platform (Piattaforma), seleziona Amazon Lex.

5. Scegli Export (Esporta).
6. Scarica e salva l'archivio .zip.

Amazon Lex esporta il bot su un file JSON contenuto nell'archivio .zip. Per aggiornare il bot, modifica il testo in formato JSON, quindi importalo di nuovo in Amazon Lex.

Approfondimenti

Importazione in formato Amazon Lex

Importazione in formato Amazon Lex

Dopo aver esportato una risorsa su un file JSON in formato Amazon Lex, puoi importare il file JSON contenente la risorsa in uno o più.AWSconti. Ad esempio, puoi esportare un bot e poi importarlo in un'altra regione AWS. In alternativa, puoi inviare il bot a un collega in modo che possa importarlo nel proprio account.

Quando importi un bot, un intento o un tipo di slot, devi decidere se desideri sovrascrivere la versione `$LATEST` di una risorsa, ad esempio un intento o un tipo di slot, durante l'importazione, oppure se desideri che l'importazione non vada a buon fine, per mantenere la risorsa presente nel tuo account. Ad esempio, se stai caricando una versione modificata di una risorsa sul tuo account, puoi scegliere di sovrascrivere la versione `$LATEST`. Se stai caricando una risorsa ricevuta da un collega, puoi scegliere di far fallire l'importazione se ci sono conflitti di risorse, in modo che le tue risorse non vengano sostituite.

Durante l'importazione di una risorsa, si applicano le autorizzazioni assegnate all'utente che effettua la richiesta di importazione. L'utente deve avere le autorizzazioni per tutte le risorse nell'account su cui avrà effetto l'importazione. L'utente deve anche disporre delle autorizzazioni per le operazioni GetBot PutBot, GetIntent, PutIntent, GetSlotType, PutSlotType. Per ulteriori informazioni sulle autorizzazioni, consultare Come funziona Amazon Lex con IAM.

L'importazione consente di registrare gli errori che si verificano durante l'elaborazione. Alcuni errori sono riportati prima che l'importazione abbia inizio, altri vengono riportati durante il processo di importazione. Ad esempio, se l'account che sta importando un intento non ha l'autorizzazione per chiamare una funzione Lambda utilizzata dall'intento, l'importazione avrà esito negativo prima che le modifiche vengano apportate ai tipi di slot o agli intenti. Se l'importazione ha esito negativo durante il processo, viene modificata la versione `$LATEST` di qualsiasi intento o tipo di slot importati prima del fallimento del processo. Non è possibile eseguire il rollback delle modifiche apportate alla versione `$LATEST`.

Quando importi una risorsa, tutte le risorse dipendenti vengono importate nella versione `$LATEST` della risorsa e ricevono poi una versione numerata. Ad esempio, se un bot impiega un intento, all'intento viene assegnata una versione numerata. Se un intento utilizza un tipo di slot personalizzato, al tipo di slot viene assegnata una versione numerata.

Una risorsa viene importata solo una volta. Ad esempio, se il bot contiene un intento `OrderPizza` e un intento `OrderDrink` ed entrambi si basano sul tipo di slot personalizzato `Size`, il tipo di slot `Size` viene importato una volta e utilizzato per entrambi gli intenti.

> **Note**
>
> Se hai esportato il bot con il`enableModelImprovements`parameter set su`false`, è necessario aprire il file.zip contenente la definizione del bot e modificare il`enableModelImprovements`parameter to`true`nelle seguenti regioni:
>
> - Asia Pacifico (Singapore): ap-southeast-1
> - Asia Pacifico (Tokyo): ap-northeast-1
> - UE (Francoforte): eu-central-1
> - UE (Londra): eu-west-2

Il processo di importazione di un bot, di un intento o di un tipo di slot personalizzato è lo stesso. Nelle seguenti procedure, sostituisci l'intento o il tipo di slot, secondo quanto richiesto.

Importazione di un bot

Per importare un bot

1. Accedere alla Console di gestione AWS e aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegli Bots (Bot), quindi seleziona il bot da importare. Per importare un nuovo bot, salta questa fase.
3. In Actions (Operazioni), seleziona Import (Importa).
4. Per Import Bot (Importa bot), scegli l'archivio .zip che contiene il file JSON in cui si trova il bot da importare. Se desideri visualizzare i conflitti di unione prima che avvenga l'unione, scegli Notify me of merge conflicts (Inviami una notifica per i conflitti di unione). Se disattivi la verifica dei conflitti, la versione `$LATEST` di tutte le risorse utilizzate dal bot viene sovrascritta.

5. Seleziona Import (Importa). Se hai scelto di ricevere una notifica in caso di conflitti di unione, al loro verificarsi, apparirà una finestra di dialogo che li elenca. Per sovrascrivere la versione $LATEST di tutte le risorse in conflitto, scegli Sovrascrivi e continua. Per interrompere l'importazione, scegli Cancel (Annulla).

Ora puoi testare il bot nel tuo account.

Formato JSON per esportazione e importazione

I seguenti esempi mostrano la struttura JSON per l'esportazione e l'importazione di tipi di slot, intenti e bot in formato Amazon Lex.

Struttura dei tipi di slot

La seguente è la struttura JSON per i tipi di slot personalizzati. Utilizza questa struttura per importare o esportare i tipi di slot e quando esporti gli intenti che dipendono da tipi di slot personalizzati.

```
{
  "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"
  }
}
```

Struttura dell'intento

La seguente è la struttura JSON per gli intenti. Utilizza questa struttura per importare o esportare intenti e bot che dipendono da un intento.

```
{
  "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 Struttura dei tipi di slot.
    ]
  }
}
```

Struttura bot

La seguente è la struttura JSON per i bot. Utilizza questa struttura durante l'importazione o l'esportazione di bot.

```
{
  "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 Struttura dell'intento.
    ],
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Struttura dei tipi di slot.
    ],
    "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"
        }
      ]
    }
  }
}
```

Esportazione su una competenza di Alexa

Puoi esportare lo schema del bot in un formato compatibile con una competenza di Alexa. Dopo aver esportato il bot su un file JSON, puoi caricarlo su Alexa utilizzando Alexa Skill Builder.

Per esportare un bot e i relativi schemi (modello di interazione)

1. Accedi allaAWS Management Consolee aprire la console di Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Scegliere il bot da esportare.

3. In Actions (Operazioni), seleziona Export (Esporta).
4. Scegliere la versione del bot che si desidera esportare. Per il formato, seleziona Alexa Skills Kit (Kit competenze Alexa), quindi scegli Export (Esporta).
5. Se viene visualizzata una finestra di dialogo di download, scegliere una posizione in cui salvare il file, quindi Save (Salva).

Il file scaricato è un archivio .zip contenente un file con il nome del bot esportato. Contiene le informazioni necessarie per importare il bot come competenza Alexa.

> **Note**
>
> Amazon Lex e il Kit di competenze di Alexa differiscono nei seguenti modi:
>
> - Gli attributi di sessione, indicati da parentesi quadre ([]), non sono supportati da Alexa Skills Kit. Devi aggiornare i messaggi di richiesta che utilizzano gli attributi di sessione.
> - I segni di interpunzione non sono supportati da Alexa Skills Kit. Devi aggiornare le enunciazioni che utilizzano l'interpunzione.

Per caricare il bot su una competenza Alexa

1. Accedi al portale per gli sviluppatori all'indirizzo https://developer.amazon.com/.
2. Nella pagina Alexa Skills (Competenze di Alexa), scegliere Create Skill (Crea competenza).
3. Nella pagina Create a new skill (Crea una nuova competenza), immettere un nome di competenza e il linguaggio predefinito per essa. Accertarsi che per il modello di competenza sia selezionato Custom (Personalizzato), quindi scegliere Create skill (Crea competenza).
4. Accertarsi che sia selezionato Start from scratch (Inizia da zero), quindi scegliere Choose (Scegli).
5. Nel menu a sinistra, scegliere JSON Editor (Editor JSON) . Trascinare il file JSON esportato da Amazon Lex nell'editor JSON.
6. Scegliere Save Model (Salva modello) per salvare il modello di interazione.

Una volta caricato lo schema nella competenza Alexa, puoi apportare tutte le modifiche necessarie per eseguire la competenza con Alexa. Per ulteriori informazioni su come creare una competenza Alexa, consulta la sezione sull'utilizzo di Skill Builder (beta) in Alexa Skills Kit.

Esempi aggiuntivi: creazione di bot Amazon Lex

Le sezioni seguenti forniscono esercizi Amazon Lex aggiuntivi con step-by-step istruzioni.

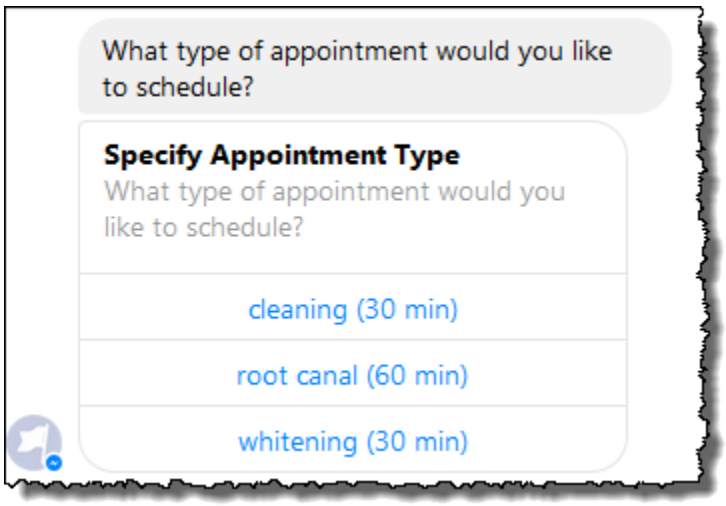
Argomenti

Pianifica un appuntamento

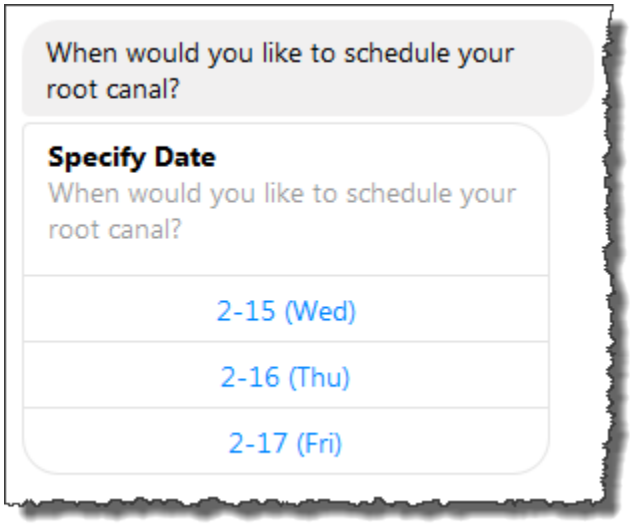
L'esempio di bot in questo esercizio consente di pianificare gli appuntamenti per uno studio dentistico. L'esempio illustra inoltre l'utilizzo di schede di risposta per ottenere l'input utente mediante pulsanti. Più precisamente, l'esempio illustra la generazione dinamica di schede di risposta durante la fase di runtime.

Puoi configurare le schede di risposta durante la creazione (definite anche schede di risposta statiche) oppure generarle dinamicamente in una funzione AWS Lambda. In questo esempio, il bot utilizza le seguenti schede di risposta:

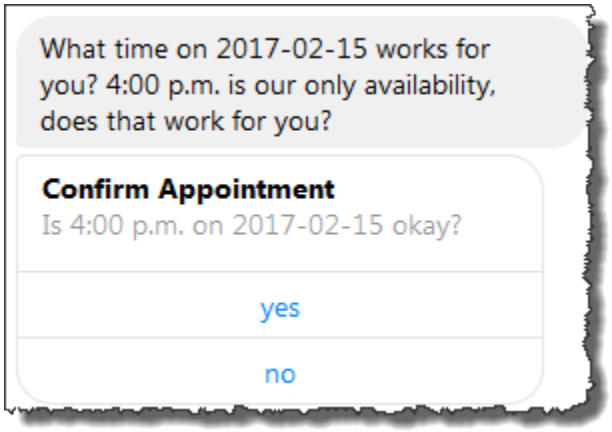
- Una scheda di risposta che elenca i pulsanti per tipo di appuntamento. Guarda l'immagine seguente per un esempio:



- Una scheda di risposta che elenca i pulsanti per data dell'appuntamento. Guarda l'immagine seguente per un esempio:



- Una scheda di risposta che elenca i pulsanti per confermare l'ora suggerita per l'appuntamento. Guarda l'immagine seguente per un esempio:



Le date e gli orari disponibili per l'appuntamento variano, di conseguenza è necessario generare schede di risposta in fase di runtime. Puoi utilizzare una funzione AWS Lambda per generare queste schede di risposta in modo dinamico. La funzione Lambda restituisce le schede di risposta nella sua risposta ad Amazon Lex. Amazon Lex include la scheda di risposta nella risposta al cliente.

Se un client (ad esempio, Facebook Messenger) supporta le schede di risposta, l'utente può scegliere uno dei pulsanti oppure digitare la risposta. In caso contrario, l'utente digita semplicemente la risposta.

Oltre al pulsante illustrato nell'esempio precedente, è anche possibile includere immagini, allegati e altre informazioni utili da visualizzare sulle schede di risposta. Per informazioni sulle schede di risposta, consulta Schede di risposta.

In questo esercizio, devi effettuare le seguenti operazioni:

- Crea e testa un bot (usando il ScheduleAppointment blueprint). Per questo esercizio, utilizzi un piano di bot per configurare ed eseguire il test del bot in tempi brevi. Per un elenco dei piani disponibili, consulta Amazon Lex eAWS LambdaPiani. Questo bot è preconfigurato con un intento (`MakeAppointment`).

- Crea e testa una funzione Lambda (utilizzando il lex-make-appointment-python blueprint fornito da Lambda). È possibile configurare l'`MakeAppointment`intento di utilizzare questa funzione Lambda come code hook per eseguire attività di inizializzazione, convalida ed evasione.

  > **Note**
  >
  > La funzione Lambda di esempio fornita mostra una conversazione dinamica basata sulla disponibilità simulata di un appuntamento dal dentista. In un'applicazione reale, puoi utilizzare un calendario reale per impostare un appuntamento.

- Aggiorna la configurazione dell'`MakeAppointment`intento per utilizzare la funzione Lambda come code hook. e test dell'intera esperienza.
- Pubblica il bot per la pianificazione degli appuntamenti su Facebook Messenger in modo da poter vedere le schede di risposta in azione (il client nella console Amazon Lex attualmente non supporta le schede di risposta).

Le sezioni seguenti forniscono informazioni di riepilogo sui piani utilizzati in questo esercizio.

Argomenti

Panoramica del Bot Blueprint () ScheduleAppointment

Il ScheduleAppointment blueprint utilizzato per creare un bot per questo esercizio è preconfigurato con quanto segue:

- Tipi di slot: un tipo di slot personalizzato denominato `AppointmentTypeValue`, con i valori di enumerazione `root canal`, `cleaning` e `whitening`.
- Intento: un intento (`MakeAppointment`), preconfigurato come segue:
  - Slot: l'intento è configurato con i seguenti slot:
    - Slot `AppointmentType` del tipo personalizzato `AppointmentTypes`.
    - Slot `Date` del tipo integrato `AMAZON.DATE`.
    - Slot `Time` del tipo integrato `AMAZON.TIME`.
  - Enunciazioni: l'intento è preconfigurato con le seguenti enunciazioni:
    - "I would like to book an appointment"
    - "Book an appointment"
    - «Prenota un {AppointmentType}»

    Se l'utente pronuncia una di queste parole, Amazon Lex determina che `MakeAppointment` è l'intento e quindi utilizza le istruzioni per ottenere i dati relativi agli slot.
  - Messaggi di richiesta: l'intento è preconfigurato con i seguenti messaggi di richiesta:
    - Messaggio di richiesta per lo slot `AppointmentType`: "What type of appointment would you like to schedule?"

- Richiedi lo `Date` slot: «Quando devo programmare il tuo {AppointmentType}?»
- Richiedi lo `Time` slot: «A che ora vuoi programmare il {AppointmentType}?» e

  "At what time on {Date}?"
- Messaggio di richiesta di conferma: "{Time} is available, should I go ahead and book your appointment?"
- Messaggio di annullamento: "Okay, I will not schedule an appointment."

Panoramica del Lambda Function Blueprint () lex-make-appointment-python

La funzione Lambda blueprint (lex-make-appointment-python) è un code hook per bot creato utilizzando il blueprint del ScheduleAppointment bot.

Questo codice blueprint della funzione Lambda può eseguire sia attività di inizializzazione/convalida che di adempimento.

- Il codice della funzione Lambda mostra una conversazione dinamica basata sulla disponibilità di esempi per un appuntamento dal dentista (nelle applicazioni reali, è possibile utilizzare un calendario). Per il giorno o la data che l'utente specifica, il codice viene configurato come segue:
  - Se non ci sono appuntamenti disponibili, la funzione Lambda restituisce una risposta che indica ad Amazon Lex di richiedere all'utente un altro giorno o data (impostando il tipo su. `dialogAction` `ElicitSlot`) Per ulteriori informazioni, consulta Formato della risposta.
  - Se è disponibile un solo appuntamento nel giorno o nella data specificati, la funzione Lambda suggerisce l'ora disponibile nella risposta e indica ad Amazon Lex di ottenere la conferma dell'utente impostando il valore `dialogAction` nella risposta a. `ConfirmIntent` Ciò illustra il modo in cui puoi migliorare l'esperienza dell'utente suggerendo in modo proattivo l'ora disponibile per un appuntamento.
  - Se sono disponibili più appuntamenti, la funzione Lambda restituisce un elenco di orari disponibili in risposta ad Amazon Lex. Amazon Lex restituisce una risposta al client con il messaggio della funzione Lambda.
- Come aggancio del codice di evasione, la funzione Lambda restituisce un messaggio di riepilogo che indica che è previsto un appuntamento (ovvero, l'intento è soddisfatto).

> **Note**
>
> In questo esempio, viene descritto come utilizzare le schede di risposta. La funzione Lambda crea e restituisce una scheda di risposta ad Amazon Lex. La scheda di risposta elenca i giorni e le ore disponibili come pulsanti tra cui scegliere. Quando si testa il bot utilizzando il client fornito dalla console Amazon Lex, non è possibile visualizzare la scheda di risposta. Per visualizzarla, devi integrare il bot in una piattaforma di messaggistica come Facebook Messenger. Per istruzioni, consulta Integrazione di un Amazon Lex Bot con Facebook Messenger. Per ulteriori informazioni sulle schede di risposta, consultare Gestione dei messaggi .

Quando Amazon Lex richiama la funzione Lambda, trasmette i dati degli eventi come input. Uno dei campi evento è`invocationSource`, che la funzione Lambda utilizza per scegliere tra un'attività di convalida dell'input e di evasione. Per ulteriori informazioni, consulta Formato dell'evento di input.

Fase successiva

Fase 1: Creare un bot Amazon Lex

Fase 1: Creare un bot Amazon Lex

In questa sezione, crei un bot Amazon Lex utilizzando il ScheduleAppointment blueprint fornito nella console Amazon Lex.

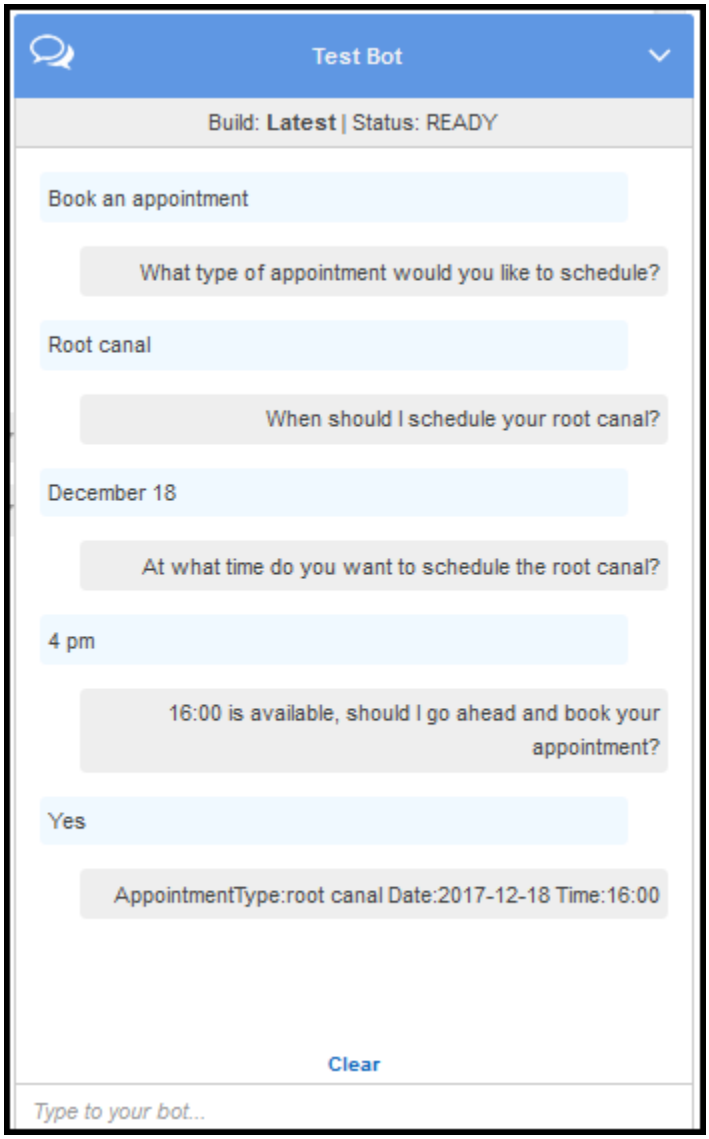
1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Nella pagina Bots (Bot) scegli Create (Crea).
3. Nella pagina Create your Lex bot (Crea bot Lex), procedi come descritto di seguito:
   - Scegli il ScheduleAppointmentprogetto.
   - Lascia il nome del bot predefinito (ScheduleAppointment).
4. Seleziona Create (Crea).

   Questa fase salva e crea il bot. La console invia le seguenti richieste ad Amazon Lex durante il processo di creazione:

- Creazione di una nuova versione dei tipi di slot (a partire dalla versione $LATEST). Per informazioni sui tipi di slot definiti in questo piano di bot, consulta Panoramica del Bot Blueprint () ScheduleAppointment.
- Creazione di una versione dell'intento `MakeAppointment` (a partire dalla versione $LATEST). In alcuni casi, la console invia una richiesta per l'operazione API `update` prima di creare una nuova versione.
- Aggiornamento della versione $LATEST del bot.

  Al momento, Amazon Lex crea un modello di machine learning per il bot. Quando testate il bot nella console, la console utilizza l'API di runtime per inviare l'input dell'utente ad Amazon Lex. Amazon Lex utilizza quindi il modello di machine learning per interpretare l'input dell'utente.

5. La console mostra il ScheduleAppointment bot. Nella scheda Editor, esamina i dettagli relativi all'intento preconfigurato (`MakeAppointment`).
6. Esegui il test del bot nella finestra di prova. Utilizza la seguente schermata per iniziare una conversazione di prova con il bot:



Tieni presente quanto segue:

- A partire dall'input utente iniziale ("Book an appointment"), il bot deduce l'intento (`MakeAppointment`).
- Il bot utilizza quindi i messaggi di richiesta configurati per ottenere i dati dello slot dall'utente.
- Il piano di bot include l'intento `MakeAppointment` configurato con il seguente messaggio di richiesta di conferma:

  ```
  {Time} is available, should I go ahead and book your appointment?
  ```

  Dopo che l'utente ha fornito tutti i dati sugli slot, Amazon Lex restituisce una risposta al client con una richiesta di conferma come messaggio. Il client visualizza il messaggio per l'utente:

```
16:00 is available, should I go ahead and book your appointment?
```

Nota che il bot accetta qualsiasi valore di data e ora per l'appuntamento poiché non disponi di alcun codice per inizializzare o convalidare i dati utente. Nella sezione successiva, aggiungi una funzione Lambda per eseguire questa operazione.

Fase successiva

Fase 2: Creare una funzione Lambda

Fase 2: Creare una funzione Lambda

In questa sezione, si crea una funzione Lambda utilizzando un blueprint (lex-make-appointment-python) fornito nella console Lambda. Puoi anche testare la funzione Lambda richiamandola utilizzando dati di esempio sugli eventi Amazon Lex forniti dalla console.

1. Accedere alla AWS Management Console e aprire la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegliere Create a Lambda function (Crea una funzione Lambda).
3. In Seleziona il blueprint, digita **`lex`** per trovare il blueprint, quindi scegli il lex-make-appointment-pythonblueprint.
4. Configurare la funzione Lambda come segue.
   - Digitare il nome della funzione Lambda (`MakeAppointmentCodeHook`).
   - Per il ruolo, scegli Create a new role from template(s) (Crea un nuovo ruolo da modello), quindi digita un nome di ruolo.
   - Non modificare gli altri valori predefiniti.
5. Selezionare Create function (Crea funzione).
6. Se utilizzi una lingua diversa dall'inglese (USA) (en-US), aggiorna i nomi degli intenti come descritto in. Aggiornamento di un blueprint per una specifica impostazione locale
7. Testate la funzione Lambda.
   a. Seleziona Actions (Operazioni), quindi Configure test event (Configura evento test).
   b. Dall'elenco Sample event template (Modello evento di esempio), scegli Lex-Make Appointment (preview) (Lex-Crea appuntamento (anteprima)). Questo evento di esempio

utilizza il modello di richiesta/risposta di Amazon Lex, con valori impostati in modo da corrispondere a una richiesta del tuo bot Amazon Lex. Per informazioni sul modello di richiesta/risposta di Amazon Lex, consulta. Utilizzo delle funzioni Lambda

c. Seleziona Save and test (Salva ed esegui test).

d. Verifica che la funzione Lambda sia stata eseguita correttamente. La risposta in questo caso corrisponde al modello di risposta di Amazon Lex.

Fase successiva

Fase 3. Aggiornamento dell'intento: configurazione di un hook di codice

Fase 3. Aggiornamento dell'intento: configurazione di un hook di codice

In questa sezione, aggiorni la configurazione dell'`MakeAppointment`intenzione di utilizzare la funzione Lambda come code hook per le attività di convalida e adempimento.

1. Nella console Amazon Lex, seleziona il ScheduleAppointment bot. La console mostra l'MakeAppointmentintento. Modifica la configurazione dell'intento come segue.

> **Note**
>
> Puoi aggiornare solo le versioni $LATEST di qualsiasi risorsa Amazon Lex, comprese le intenzioni. Assicurati che la versione dell'intento sia impostata su $LATEST. Non hai ancora pubblicato una versione del bot, di conseguenza dovrebbe essere ancora la versione $LATEST nella console.

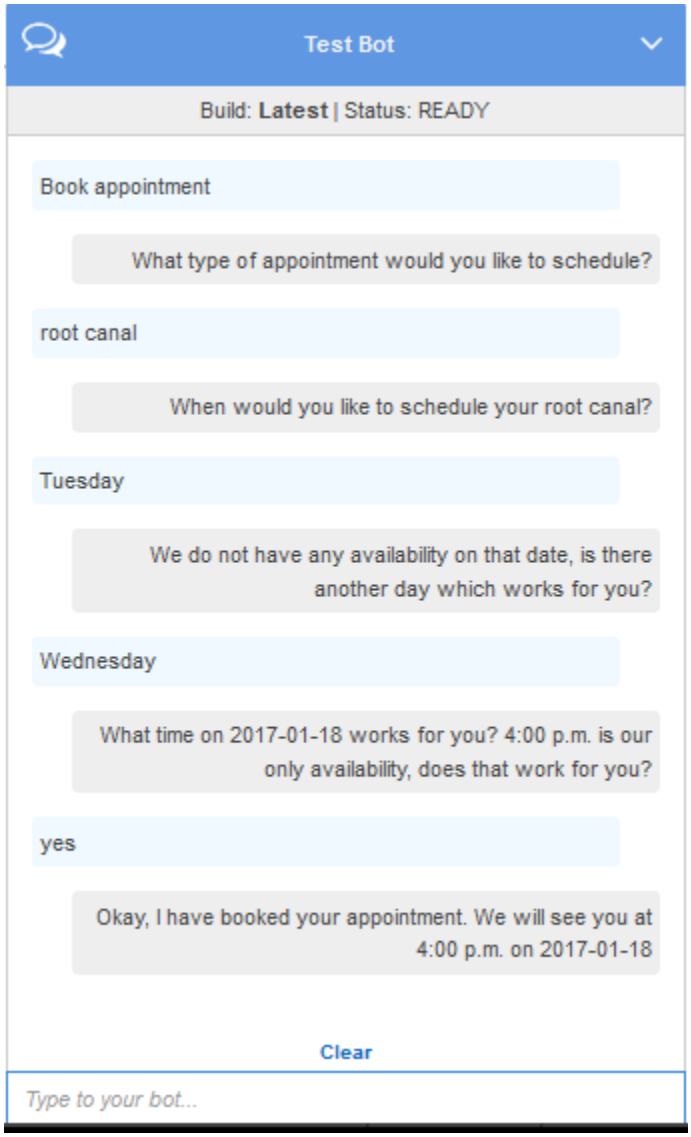
   a. Nella sezione Opzioni, scegli l'hook del codice di inizializzazione e convalida, quindi scegli la funzione Lambda dall'elenco.

   b. Nella sezione Fulfillment, scegli la funzione AWS Lambda, quindi scegli la funzione Lambda dall'elenco.

   c. Seleziona Goodbye message (Messaggio di saluto).

2. Seleziona Save (Salva), quindi Build (Crea).

3. Prova il bot, come nell'immagine seguente:



Fase successiva

Fase 4. Distribuzione del bot sulla piattaforma Facebook Messenger

Fase 4. Distribuzione del bot sulla piattaforma Facebook Messenger

Nella sezione precedente, hai testato il ScheduleAppointment bot utilizzando il client nella console Amazon Lex. Attualmente, la console Amazon Lex non supporta le schede di risposta. Per eseguire il test delle schede di risposta generate dinamicamente che il bot supporta, distribuisci il bot sulla piattaforma Facebook Messenger ed eseguine il test.

Per istruzioni, consulta Integrazione di un Amazon Lex Bot con Facebook Messenger.

Fase successiva

Dettagli del flusso di informazioni

Dettagli del flusso di informazioni

Il piano di bot `ScheduleAppointment` mostra principalmente l'utilizzo di carte di risposta generate dinamicamente. La funzione Lambda in questo esercizio include schede di risposta nella sua risposta ad Amazon Lex. Amazon Lex include le schede di risposta nella risposta al cliente. Questa sezione descrive quanto segue:

- Flusso di dati tra il cliente e Amazon Lex.

  La sezione presuppone che il client invii richieste ad Amazon Lex utilizzando l'API di `PostText` runtime e mostri di conseguenza i dettagli della richiesta/risposta. Per ulteriori informazioni sull'API di runtime `PostText`, consulta PostText.

  > **Note**
  >
  > Per un esempio di flusso di informazioni tra il client e Amazon Lex in cui il client utilizza l'`PostContent`API, vediFase 2a (facoltativo): Revisione dei dettagli del flusso di informazioni parlate (console) .

- Flusso di dati tra Amazon Lex e la funzione Lambda. Per ulteriori informazioni, consulta Formato di evento di input e di risposta della funzione Lambda.

> **Note**
>
> L'esempio presuppone che tu stia utilizzando il client Facebook Messenger, che non trasmette gli attributi di sessione nella richiesta ad Amazon Lex. Di conseguenza, negli esempi di richieste mostrati in questa sezione il campo `sessionAttributes` è vuoto. Se testate il bot utilizzando il client fornito nella console Amazon Lex, il client include gli attributi della sessione.

Questa sezione descrive cosa accade dopo ogni input utente.

1. Utente: tipi **`Book an appointment`**.

   a. Il client (console) invia la seguente richiesta PostContent ad Amazon Lex:

```
POST /bot/ScheduleAppointment/alias/$LATEST/
user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"book appointment",
   "sessionAttributes":{}
}
```

   Sia l'URI della richiesta che il corpo forniscono informazioni ad Amazon Lex:

   - URI di richiesta: fornisce il nome del bot (ScheduleAppointment), l'alias del bot ($LATEST) e l'ID del nome utente. Il codice `text` di fine indica che si tratta di una richiesta API `PostText` (non `PostContent`).
   - Corpo della richiesta: include l'input utente (`inputText`) e il campo `sessionAttributes` vuoto.

   b. `DainputText`, Amazon Lex rileva l'intento ()`MakeAppointment`. Il servizio richiama la funzione Lambda, che è configurata come un code hook, per eseguire l'inizializzazione e la convalida passando il seguente evento. Per informazioni dettagliate, consultare Formato dell'evento di input.

```
{
    "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": {}
}
```

Oltre alle informazioni inviate dal cliente, Amazon Lex include anche i seguenti dati:

- `currentIntent`: fornisce informazioni sull'intento corrente.
- `invocationSource`— Indica lo scopo della chiamata alla funzione Lambda. In questo caso, lo scopo è eseguire l'inizializzazione e la convalida dei dati utente. (Amazon Lex sa che l'utente non ha ancora fornito tutti i dati sugli slot necessari per soddisfare l'intento.)
- `messageVersion`— Attualmente Amazon Lex supporta solo la versione 1.0.

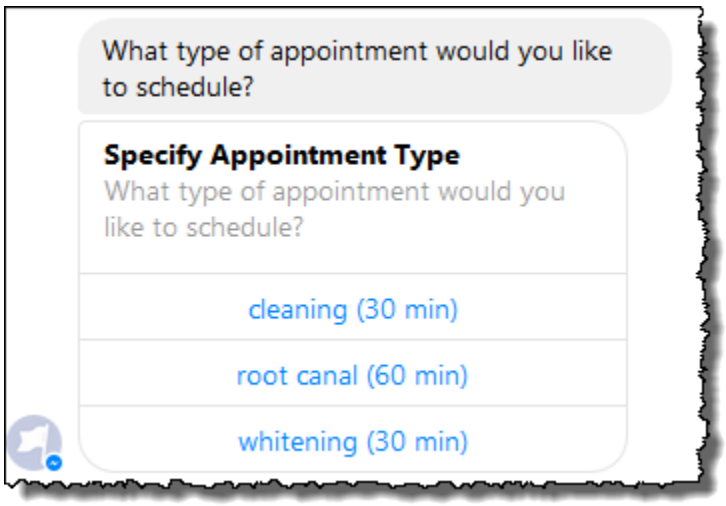
c. A questo punto, tutti i valori di slot sono null (non c'è nulla da convalidare). La funzione Lambda restituisce la seguente risposta ad Amazon Lex, indirizzando il servizio a raccogliere informazioni per lo slot. `AppointmentType` Per informazioni sul formato della risposta, consulta l'argomento Formato della risposta.

```
{
    "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"
                        }
```

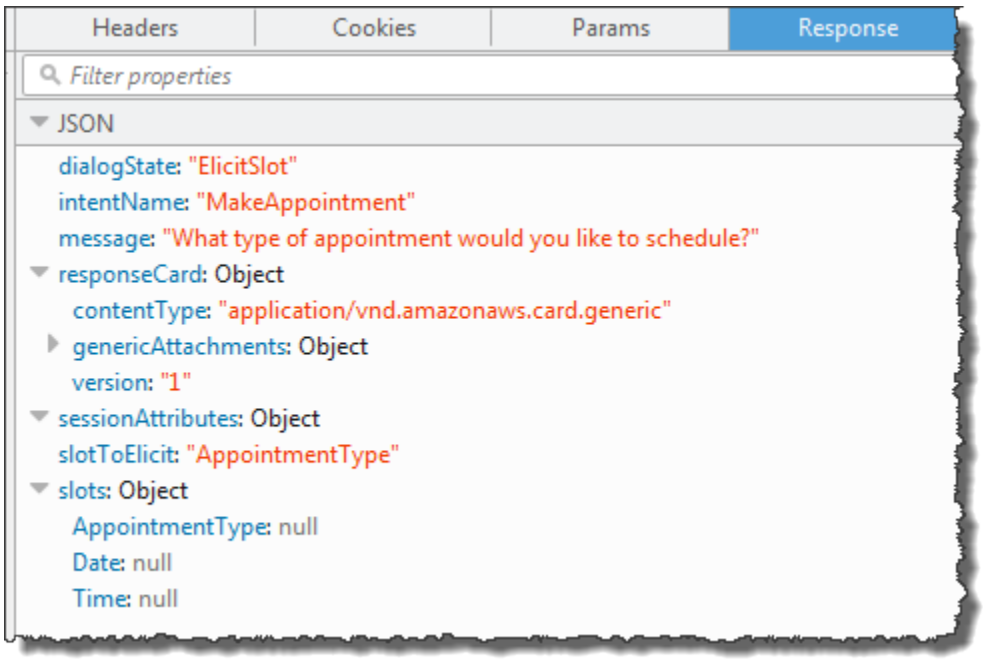
```
                    ],
                    "subTitle": "What type of appointment would you like to
 schedule?",
                    "title": "Specify Appointment Type"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

La risposta include i campi `dialogAction` e `sessionAttributes`. Il campo `dialogAction` restituisce inoltre i seguenti campi:

- `type`— Impostando questo campo su`ElicitSlot`, la funzione Lambda indica ad Amazon Lex di ottenere il valore per lo slot specificato nel campo. `slotToElicit` La funzione Lambda fornisce anche un messaggio `message` all'utente.
- `responseCard`— Identifica un elenco di valori possibili per lo `AppointmentType` slot. Un client che supporta le schede di risposta (ad esempio, Facebook Messenger) visualizza una scheda di risposta per consentire all'utente di scegliere un tipo di appuntamento, come nell'immagine seguente:



d. Come indicato `dialogAction.type` nella risposta della funzione Lambda, Amazon Lex invia al client la seguente risposta:



Il cliente legge la risposta, quindi visualizza il messaggio: «Che tipo di appuntamento desideri fissare?» e la scheda di risposta (se il client supporta le schede di risposta).

2. Utente: a seconda del client, l'utente dispone di due opzioni:

   - Se la scheda di risposta è visualizzata, scegli devitalizzazione (60 min) o digita **`root canal`**.
   - Se il client non supporta le schede di risposta, digita **`root canal`**.

a. Il client invia la seguente `PostText` richiesta ad Amazon Lex (sono state aggiunte interruzioni di riga per la leggibilità):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "root canal",
    "sessionAttributes": {}
}
```

b. Amazon Lex richiama la funzione Lambda per la convalida dei dati utente inviando il seguente evento come parametro:

```
{
    "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": {}
}
```

Nei dati di evento, nota quanto segue:

- `invocationSource` continua a essere `DialogCodeHook`. In questa fase, eseguiamo soltanto la convalida dei dati utente.
- Amazon Lex imposta il `AppointmentType` campo nello `currentIntent.slots` slot su`root canal`.
- Amazon Lex passa semplicemente il `sessionAttributes` campo tra il client e la funzione Lambda.

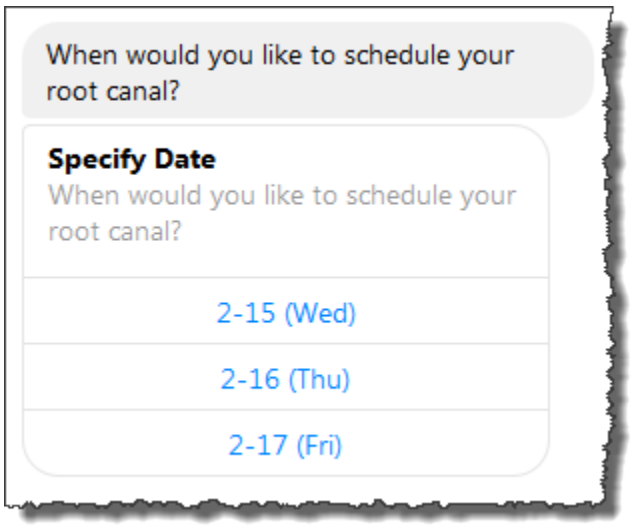
c. La funzione Lambda convalida l'input dell'utente e restituisce la seguente risposta ad Amazon Lex, indicando al servizio di ottenere un valore per la data dell'appuntamento.

```
{
    "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": {}
}
```

La risposta include di nuovo i campi `dialogAction` e `sessionAttributes`. Il campo `dialogAction` restituisce inoltre i seguenti campi:

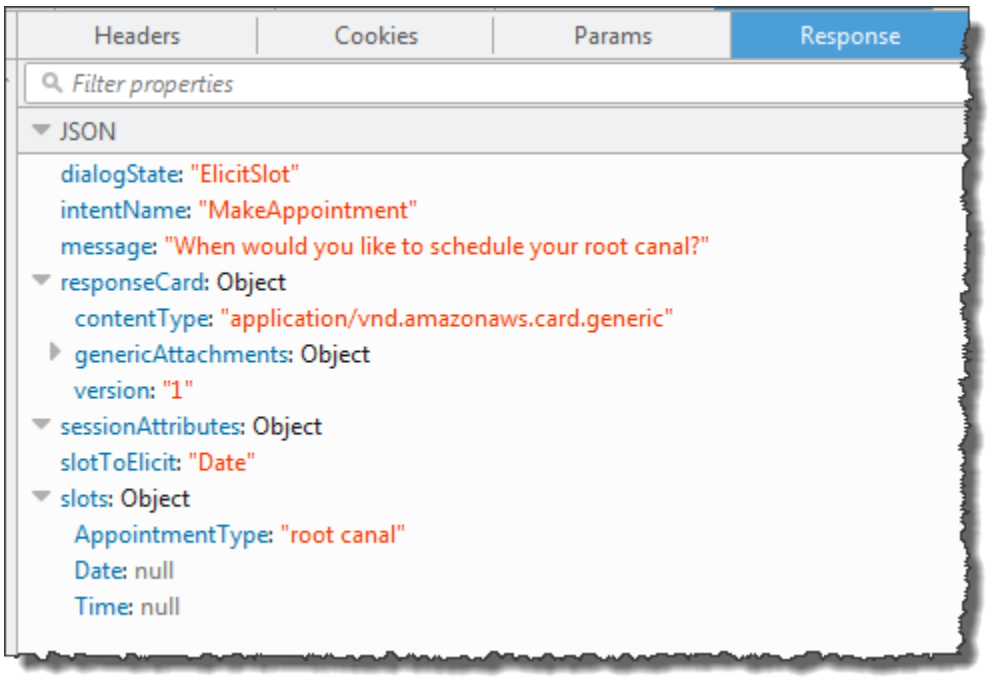
- `type`— Impostando questo campo su`ElicitSlot`, la funzione Lambda indica ad Amazon Lex di ottenere il valore per lo slot specificato nel campo. `slotToElicit` La funzione Lambda fornisce anche un messaggio `message` all'utente.
- `responseCard`— Identifica un elenco di valori possibili per lo `Date` slot. Un client che supporta le schede di risposta (ad esempio, Facebook Messenger) visualizza una scheda di risposta che consente all'utente di scegliere una data per l'appuntamento, come nell'immagine seguente:



Sebbene la funzione Lambda abbia restituito cinque date, il client (Facebook Messenger) ha un limite di tre pulsanti per una scheda di risposta. Pertanto, nello screenshot sono visualizzati solo i primi tre valori.

Queste date sono codificate nella funzione Lambda. In un'applicazione di produzione, puoi utilizzare un calendario per ottenere le date disponibili in tempo reale. Poiché le date sono dinamiche, è necessario generare la scheda di risposta in modo dinamico nella funzione Lambda.

d. Amazon Lex rileva `dialogAction.type` e restituisce al client la seguente risposta che include le informazioni della risposta della funzione Lambda.



Il client visualizza il messaggio: When would you like to schedule your root canal? e la scheda di risposta (se il client supporta le schede di risposta).

3. Utente: tipi **`Thursday`**.

   a. Il client invia la seguente `PostText` richiesta ad Amazon Lex (sono state aggiunte interruzioni di riga per la leggibilità):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   b. Amazon Lex richiama la funzione Lambda per la convalida dei dati utente inviando il seguente evento come parametro:

```
{
    "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": {}
}
```

Nei dati di evento, nota quanto segue:

- `invocationSource` continua a essere `DialogCodeHook`. In questa fase, eseguiamo soltanto la convalida dei dati utente.
- Amazon Lex imposta il `Date` campo nello `currentIntent.slots` slot su`2017-02-16`.
- Amazon Lex passa semplicemente il passaggio `sessionAttributes` tra il client e la funzione Lambda.

c. La funzione Lambda convalida l'input dell'utente. Questa volta la funzione Lambda determina che non ci sono appuntamenti disponibili nella data specificata. Restituisce la seguente risposta ad Amazon Lex, indicando al servizio di richiedere nuovamente un valore per la data dell'appuntamento.

```
{
    "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\": []}"
   }
}
```

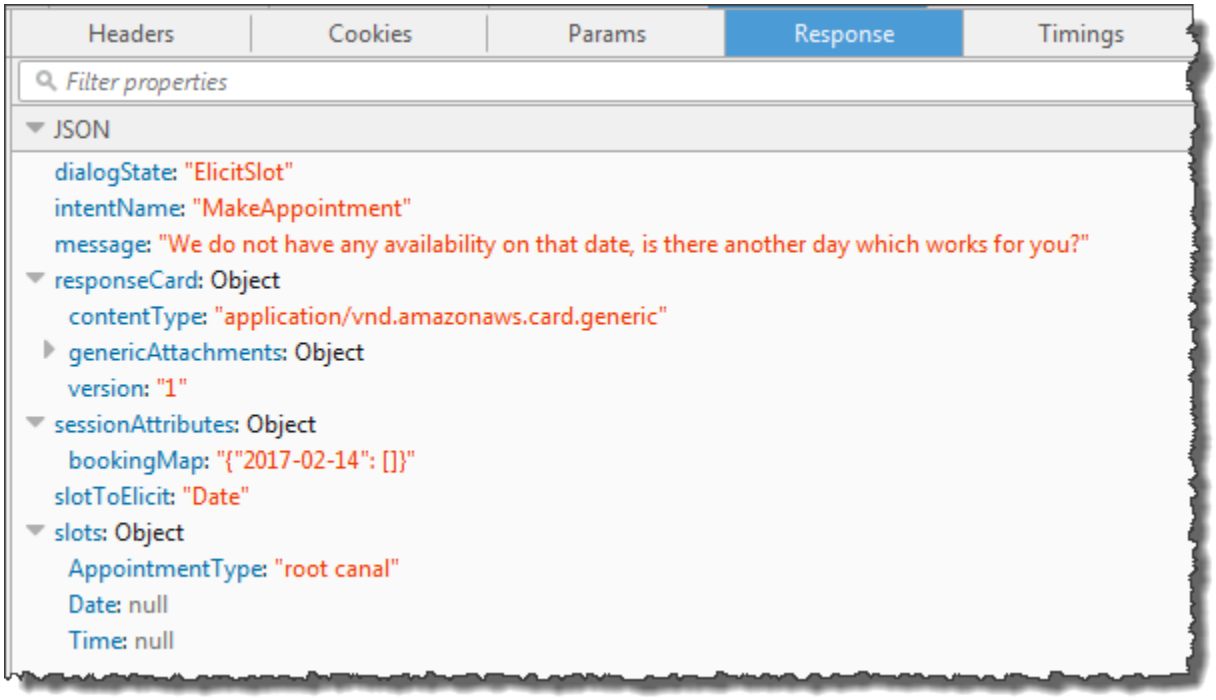
La risposta include di nuovo i campi `dialogAction` e `sessionAttributes`. Il campo `dialogAction` restituisce inoltre i seguenti campi:

- Campo `dialogAction`:

- `type`— La funzione Lambda imposta questo valore su `ElicitSlot` e reimposta il `slotToElicit` campo su. `Date` La funzione Lambda fornisce anche un'appropriata trasmissione `message` all'utente.
- `responseCard`: restituisce un elenco di valori per lo slot `Date`.

- `sessionAttributes`- Questa volta la funzione Lambda include l'attributo `bookingMap` session. Il relativo valore è la data richiesta dell'appuntamento e gli appuntamenti disponibili (un oggetto vuoto indica che non vi sono appuntamenti disponibili).

d. Amazon Lex rileva `dialogAction.type` e restituisce al client la seguente risposta che include le informazioni della risposta della funzione Lambda.



Il client visualizza il messaggio: We do not have any availability on that date, is there another day which works for you? e la scheda di risposta (se il client supporta le schede di risposta).

4. Utente: a seconda del client, l'utente dispone di due opzioni:

- Se la scheda di risposta viene visualizzata, scegli 15-2 (mer.) o digita **`Wednesday`**.
- Se il client non supporta le schede di risposta, digita **`Wednesday`**.

a. Il client invia la seguente `PostText` richiesta ad Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
```

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

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

> **Note**
>
> Il client di Facebook Messenger non imposta alcun attributo di sessione. Se si desidera mantenere gli stati della sessione tra le richieste, è necessario farlo nella funzione Lambda. In un'applicazione reale, puoi avere la necessità di conservare questi attributi di sessione in un database back-end.

b. Amazon Lex richiama la funzione Lambda per la convalida dei dati utente inviando il seguente evento come parametro:

```
{
    "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 è stato aggiornato `currentIntent.slots` impostando lo `Date` slot su`2017-02-15`.

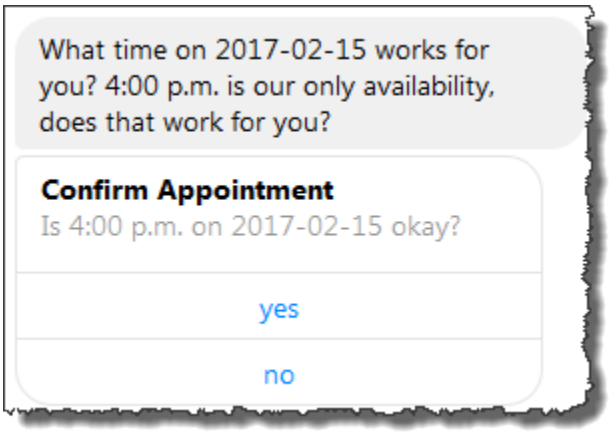
c. La funzione Lambda convalida l'input dell'utente e restituisce la seguente risposta ad Amazon Lex, indirizzandola a ottenere il valore per l'ora dell'appuntamento.

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

```
    }
}
```

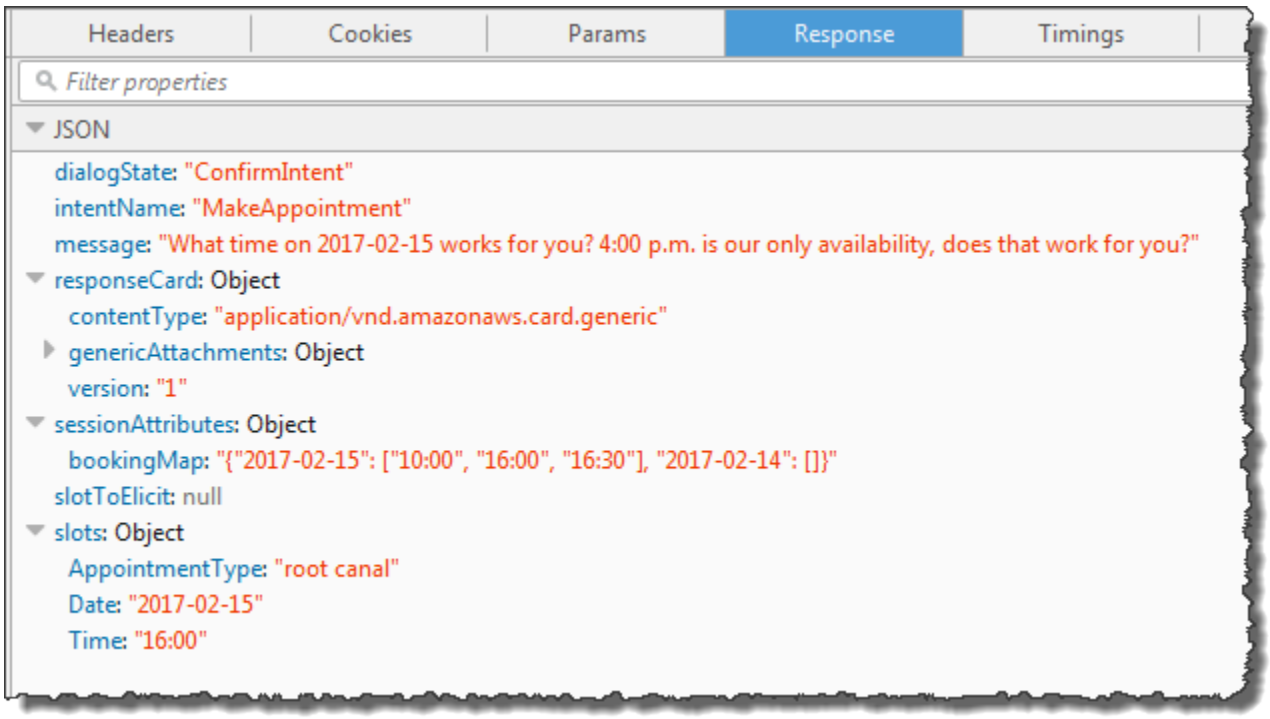
La risposta include di nuovo i campi `dialogAction` e `sessionAttributes`. Il campo `dialogAction` restituisce inoltre i seguenti campi:

- Campo `dialogAction`:
  - `type`— La `Lambda` funzione imposta questo valore su`ConfirmIntent`, indicando ad Amazon Lex di ottenere la conferma da parte dell'utente dell'orario dell'appuntamento suggerito nel`message`.
  - `responseCard`— Restituisce un elenco di valori sì/no tra cui l'utente può scegliere. Se il client supporta le schede di risposta, visualizza la scheda di risposta, come mostrato nel seguente esempio:



  - `sessionAttributes`- La funzione Lambda imposta l'attributo della `bookingMap` sessione con il relativo valore impostato sulla data dell'appuntamento e sugli appuntamenti disponibili in quella data. In questo esempio, si tratta di appuntamenti di 30 minuti. Per una devitalizzazione che richieda un'ora, è possibile scegliere solo le 16:00.

d. Come indicato nella risposta della `dialogAction.type` funzione Lambda, Amazon Lex restituisce al client la seguente risposta:



Il client visualizza il messaggio: a che ora del 2017-02-15 funziona per te? Le 16:00 sono la nostra unica disponibilità, funziona per te?

5. Utente: scegli **`yes`**.

   Amazon Lex richiama la funzione Lambda con i seguenti dati sugli eventi. Poiché l'utente ha rispostoyes, Amazon Lex imposta `confirmationStatus` il `Confirmed` valore e imposta il `Time` campo `currentIntent.slots su4 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": {
   }
}
```

Una volta `confirmationStatus` confermata, la funzione Lambda elabora l'intento (prenota un appuntamento dal dentista) e restituisce la seguente risposta ad Amazon Lex:

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

Tieni presente quanto segue:

- La funzione Lambda ha aggiornato `sessionAttributes`.
- `dialogAction.type`è impostato su`Close`, il che indica ad Amazon Lex di non aspettarsi una risposta da parte dell'utente.
- `dialogAction.fulfillmentState` è impostato su `Fulfilled`, a indicare che l'intento è stato realizzato.

Il cliente visualizza il messaggio: Ok, ho prenotato il tuo appuntamento. Ci vediamo alle 16:00 il 15/02/2017.

Prenota viaggio

Nell'esempio seguente viene illustrata la creazione di un bot configurato per supportare più intenti. L'esempio illustra inoltre come utilizzare gli attributi di sessione per la condivisione di informazioni tra più intenti. Dopo aver creato il bot, usi un client di test nella console Amazon Lex per testare il bot (BookTrip). Il client utilizza l'operazione APIPostText di runtime per inviare richieste ad Amazon Lex per ogni input dell'utente.

Il BookTrip bot in questo esempio è configurato con due intenti (BookHotel e BookCar). Supponiamo ad esempio che un utente prenoti in primo luogo un hotel. Durante l'interazione, l'utente fornisce informazioni quali date di check-in, località e il numero di notti. Dopo che l'intento è realizzato, il client può conservare queste informazioni tramite gli attributi di sessione. Per ulteriori informazioni sugli attributi di sessione, consulta PostText.

Supponiamo a questo punto che l'utente continui con la prenotazione di un'automobile. Utilizzando le informazioni fornite dall'utente nell' BookHotel intento precedente (ovvero città di destinazione e date di check-in e check-out), il code hook (funzione Lambda) configurato per inizializzare e convalidare l' BookCar intento, inizializza i dati degli slot relativi all' BookCar intento (ovvero destinazione, città di ritiro, data di ritiro e data di ritorno). Questo illustra come la condivisione di informazioni tra più intenti consenta di creare bot in grado di iniziare una conversazione dinamica con l'utente.

In questo esempio vengono utilizzati gli attributi di sessione riportati di seguito. Solo il client e la funzione Lambda possono impostare e aggiornare gli attributi della sessione. Amazon Lex li passa solo tra il client e la funzione Lambda. Amazon Lex non mantiene o modifica alcun attributo di sessione.

- `currentReservation`— Contiene i dati relativi agli slot per una prenotazione in corso e altre informazioni pertinenti. Di seguito è riportata una richiesta di esempio dal client ad Amazon Lex, con l'attributo di sessione `currentReservation` mostrato nel corpo della richiesta.

```
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`— Contiene informazioni simili relative a un intento precedente, se del caso. Ad esempio, se l'utente ha prenotato un hotel e poi sta prenotando un'auto, questo attributo di sessione memorizza i dati degli slot per l' BookHotel intento precedente.

- `confirmationContext`— La funzione Lambda lo imposta`AutoPopulate` quando precompila alcuni dati dello slot in base ai dati dello slot della prenotazione precedente (se presente). Ciò consente la condivisione delle informazioni tra più intenti. Ad esempio, se l'utente aveva precedentemente prenotato un hotel e ora desidera prenotare un'auto, Amazon Lex può richiedere all'utente di confermare (o negare) che l'auto è stata prenotata per la stessa città e le stesse date della prenotazione alberghiera

In questo esercizio utilizzerai i blueprint per creare un bot Amazon Lex e una funzione Lambda. Per ulteriori informazioni sui piani, consulta Amazon Lex eAWS LambdaPiani.

Fase successiva

Fase 1. Analisi dei piani utilizzati in questo esercizio

Fase 1. Analisi dei piani utilizzati in questo esercizio

Argomenti

- Panoramica del Bot Blueprint (BookTrip)
- Panoramica del Lambda Function Blueprint (lex-book-trip-python)

Panoramica del Bot Blueprint (BookTrip)

Il blueprint (BookTrip) che usi per creare un bot fornisce la seguente preconfigurazione:

- Tipi di slot: due tipi di slot personalizzati:
  - `RoomTypes` con valori di enumerazione: `king`, `queen` e `deluxe`, per l'uso nell'intento `BookHotel`.
  - `CarTypes` con valori di enumerazione: `economy`, `standard`, `midsize`, `full size`, `luxury` e `minivan`, per l'uso nell'intento `BookCar`.

- Intent 1 (BookHotel) — È preconfigurato come segue:
  - Slot preconfigurati
    - `RoomType`, del tipo di slot personalizzato `RoomTypes`
    - `Location`, del tipo di slot integrato `AMAZON.US_CITY`
    - `CheckInDate`, del tipo di slot integrato `AMAZON.DATE`
    - `Nights`, del tipo di slot integrato `AMAZON.NUMBER`
  - Enunciazioni preconfigurate
    - "Book a hotel"
    - "I want to make hotel reservations"
    - "Book a {Nights} stay in {Location}"

    Se l'utente pronuncia una di queste parole, Amazon Lex stabilisce che questa`BookHotel` è l'intenzione e quindi richiede all'utente i dati dello slot.
  - Messaggi di richiesta preconfigurati
    - Prompt per lo slot `Location`: "What city will you be staying in?"
    - Prompt per lo slot `CheckInDate`: "What day do you want to check in?"
    - Prompt per lo slot `Nights`: "How many nights will you be staying?"
    - Prompt per lo slot `RoomType`: "What type of room would you like, queen, king, or deluxe?"
    - Dichiarazione di conferma: «Ok, ti aspetto per un soggiorno di {7 notti} in {Location} a partire da {CheckInDate}. Shall I book the reservation?"
    - Rifiuto: "Okay, I have cancelled your reservation in progress."

- Intent 2 (BookCar): è preconfigurato come segue:
  - Slot preconfigurati
    - `PickUpCity`, del tipo integrato `AMAZON.US_CITY`

  - `PickUpDate`, del tipo integrato `AMAZON.DATE`
  - `ReturnDate`, del tipo integrato `AMAZON.DATE`
  - `DriverAge`, del tipo integrato `AMAZON.NUMBER`
  - `CarType`, del tipo personalizzato `CarTypes`
- Enunciazioni preconfigurate
  - "Book a car"
  - "Reserve a car"
  - "Make a car reservation"

  Se l'utente pronuncia una di queste parole, Amazon Lex ne determina BookCar l'intenzione e quindi richiede all'utente i dati dello slot.
- Messaggi di richiesta preconfigurati
  - Prompt per lo slot `PickUpCity`: "In what city do you need to rent a car?"
  - Prompt per lo slot `PickUpDate`: "What day do you want to start your rental?""
  - Prompt per lo slot `ReturnDate`: "What day do you want to return this car?"
  - Prompt per lo slot `DriverAge`: "How old is the driver for this rental?"
  - Richiedi lo`CarType` slot: «Che tipo di auto vorresti noleggiare? Le nostre opzioni più richieste sono: economica, media e di lusso"
  - Dichiarazione di conferma: «Ok, ti ho a disposizione per un {CarType} noleggio a {PickUpCity} da {PickUpDate} a {ReturnDate}. Should I book the reservation?"
  - Rifiuto: "Okay, I have cancelled your reservation in progress."

Panoramica del Lambda Function Blueprint (lex-book-trip-python)

Oltre al blueprint del bot,AWS Lambda fornisce un blueprint (lex-book-trip-python) che puoi usare come hook di codice con il blueprint del bot. Per un elenco dei blueprint dei bot e dei relativi blueprint delle funzioni Lambda, consultaAmazon Lex eAWS LambdaPiani.

Quando si crea un bot utilizzando il BookTrip blueprint, si aggiorna la configurazione di entrambi gli intenti (BookCar e BookHotel) aggiungendo questa funzione Lambda come hook di codice sia per l'inizializzazione/convalida dell'input dei dati utente che per la realizzazione degli intenti.

Il codice di funzione Lambda fornito mostra una conversazione dinamica che utilizza le informazioni precedentemente note (conservate negli attributi di sessione) relative a un utente per inizializzare

i valori di slot per un intento. Per ulteriori informazioni, consulta Gestione del contesto di una conversazione.

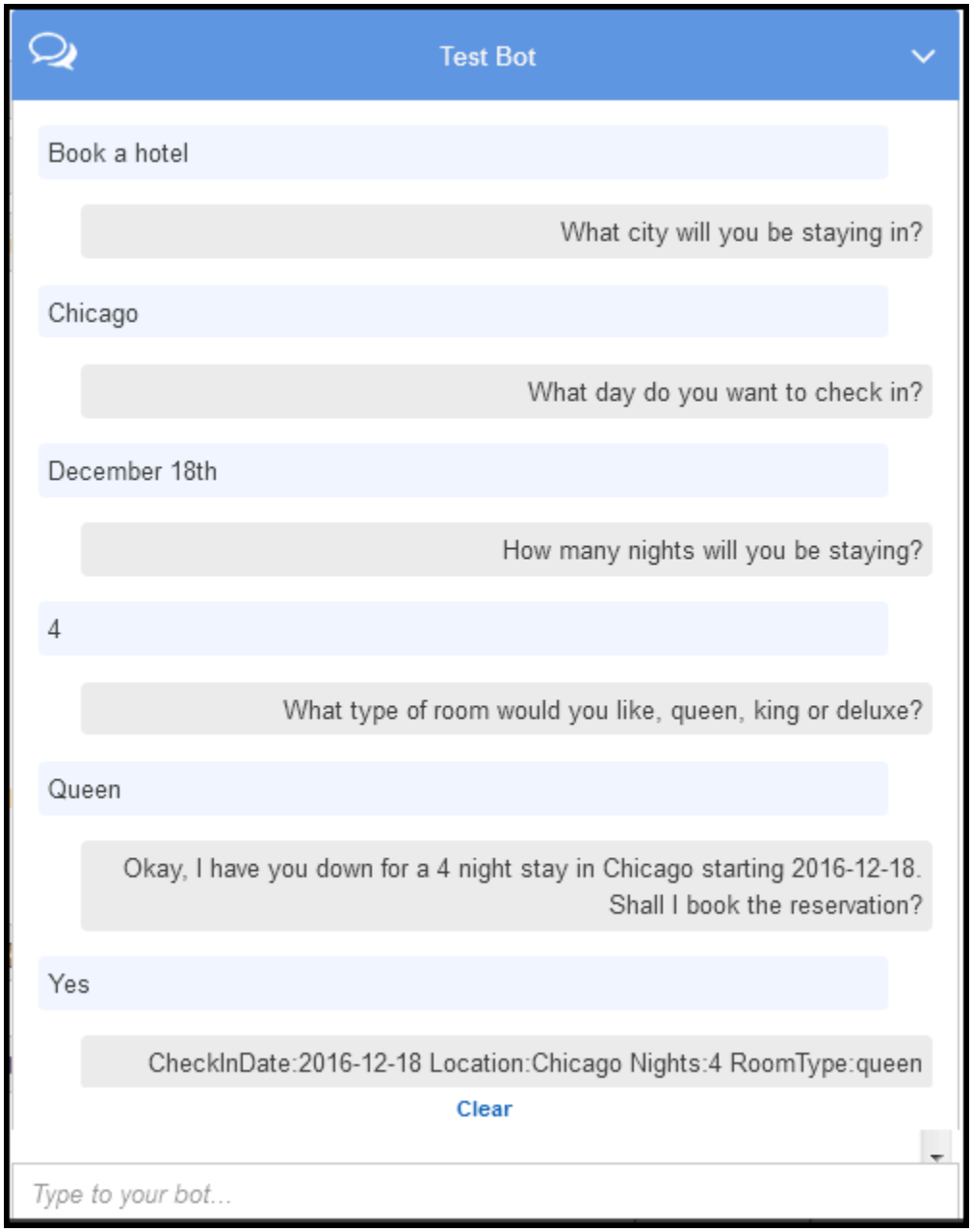
Fase successiva

Fase 2: Creazione di un Amazon Lex Bot

Fase 2: Creazione di un Amazon Lex Bot

In questa sezione, crei un bot Amazon Lex (BookTrip).

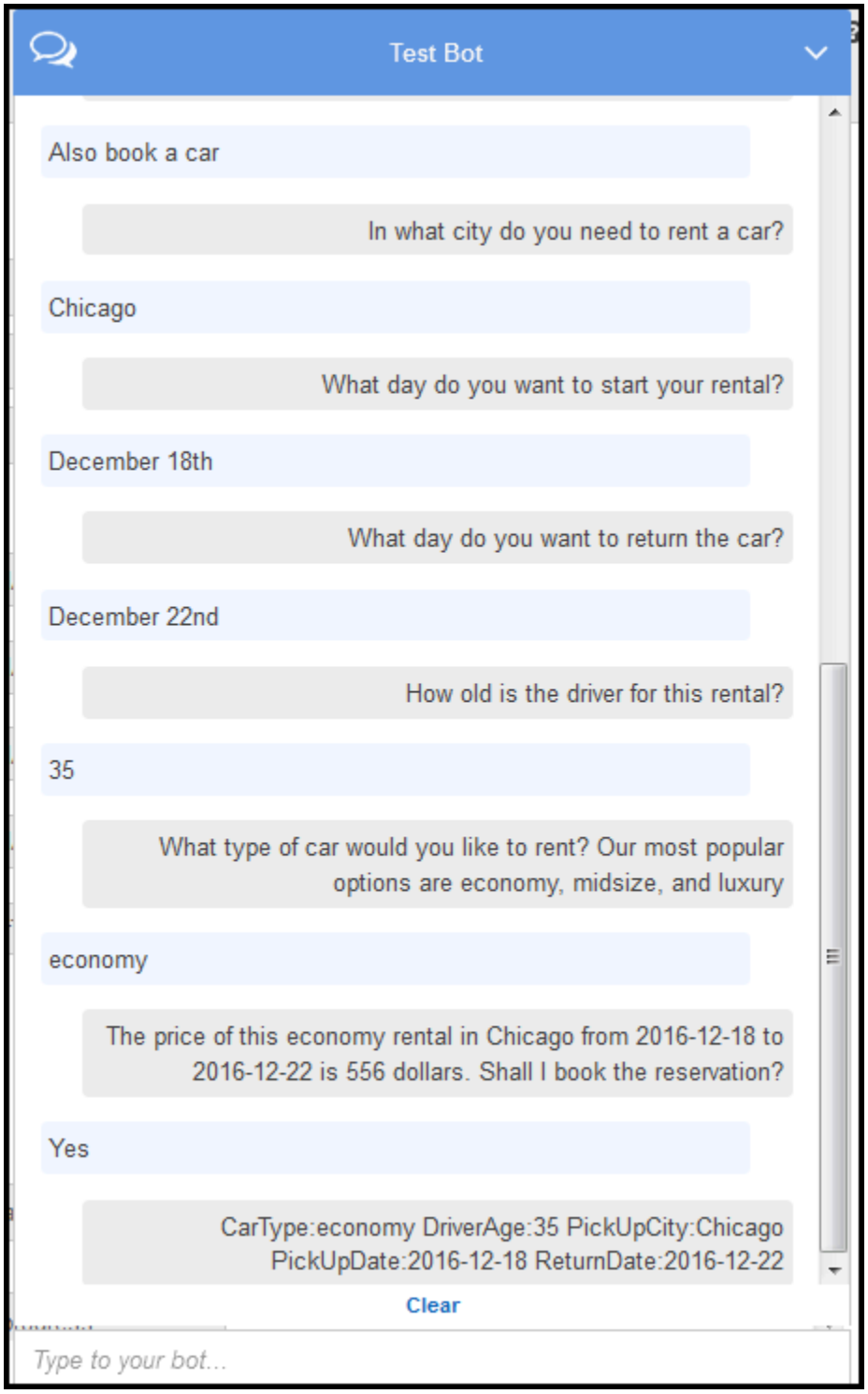
1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Nella pagina Bots (Bot) scegli Create (Crea).
3. Nella pagina Create your Lex bot (Crea bot di Lex):
   - Scegli il BookTripblueprint.
   - Lascia il nome del bot predefinito (BookTrip).
4. Seleziona Create (Crea). La console invia una serie di richieste ad Amazon Lex per creare il bot. Tieni presente quanto segue:
5. La console mostra il BookTrip bot. Nella scheda Editor, rivedi i dettagli degli intenti (BookCar e BookHotel) preconfigurati.
6. Esegui il test del bot nella finestra di prova. Utilizza quanto segue per iniziare una conversazione di test con il bot:



Dall'input iniziale dell'utente («Prenota un hotel»), Amazon Lex deduce l'intento (BookHotel). Il bot utilizza quindi i prompt preconfigurati nell'intento per ottenere i dati dello slot dall'utente. Dopo che l'utente ha fornito tutti i dati dello slot, Amazon Lex restituisce una risposta al client con un messaggio che include tutti gli input dell'utente come messaggio. Il client visualizza il messaggio nella risposta, come mostrato.

```
CheckInDate:2016-12-18 Location:Chicago Nights:5 RoomType:queen
```

Ora continui la conversazione e provi a prenotare un'auto nella conversazione seguente.



Nota,

- Non è attiva alcuna convalida dei dati utente in questo momento. Puoi ad esempio specificare qualsiasi città per prenotare un hotel.

- Alcune delle informazioni fornite per prenotare un'automobile corrispondono a quelle specificate in precedenza (destinazione, città di ritiro, data di ritiro e data di riconsegna). In una conversazione dinamica il bot deve inizializzare alcune di queste informazioni in base all'input utente precedente specificato per la prenotazione dell'hotel.

In questa sezione successiva creerai una funzione Lambda per eseguire alcune attività di convalida e inizializzazione dei dati utente, utilizzando la condivisione delle informazioni tra più intenti tramite gli attributi di sessione. Aggiornerai quindi la configurazione degli intenti aggiungendo la funzione Lambda come un hook di codice per eseguire l'inizializzazione/la convalida dell'input utente e realizzare l'intento.

Fase successiva

Fase 3: Creazione di una funzione Lambda

Fase 3: Creazione di una funzione Lambda

In questa sezione si crea una funzione Lambda utilizzando un blueprint (lex-book-trip-python) fornito nellaAWS Lambda console. È inoltre possibile testare la funzione Lambda richiamandola utilizzando i dati degli eventi di esempio forniti dalla console.

Questa funzione Lambda è scritta in Python.

1. Accedere alla AWS Management Console e aprire la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegli Create function (Crea funzione).
3. Scegliere Use a blueprint (Usa un piano). Digitare **`lex`** per individuare il piano e scegliere il piano `lex-book-trip-python`.
4. Scegliete Configura la funzione Lambda come segue.
   - Digitare il nome di una funzione Lambda (`BookTripCodeHook`).
   - Per il ruolo, scegli Create a new role from template(s) (Crea un nuovo ruolo da modello), quindi digita un nome di ruolo.
   - Non modificare gli altri valori di default.
5. Scegli Create function (Crea funzione).

6. Se utilizzi una lingua diversa dall'inglese (USA) (en-US), aggiorna i nomi degli intenti come descritto inAggiornamento di un blueprint per una specifica impostazione locale.
7. Test della funzione Lambda. Si richiama la funzione Lambda due volte, utilizzando dati di esempio sia per la prenotazione di un'auto che per la prenotazione di un hotel.

   a. Scegliere Configura evento di test dal menu a discesa Seleziona un evento di test.

   b. Seleziona Amazon Lex-Book Hotel nell'elenco Modello evento di esempio.

      Questo evento di esempio corrisponde al modello di richiesta/risposta di Amazon Lex. Per ulteriori informazioni, consulta Utilizzo delle funzioni Lambda.

   c. Seleziona Save and test (Salva ed esegui test).

   d. Eseguire le operazioni per verificare che la funzione Lambda sia stata eseguita nel modo corretto. La risposta in questo caso corrisponde al modello di risposta di Amazon Lex.

   e. Ripeti la fase. Questa volta scegli Amazon Lex-Book Car (Lex-Book Car nell'elenco Modello evento di esempio. La funzione Lambda elabora la prenotazione dell'auto.

Fase successiva

Passaggio 4: aggiungere la funzione Lambda come codice hook

Passaggio 4: aggiungere la funzione Lambda come codice hook

In questa sezione, si aggiornano le configurazioni sia degli BookCar intenti che BookHotel degli intenti aggiungendo la funzione Lambda come hook di codice per le attività di inizializzazione/convalida ed esecuzione. Assicurati di scegliere la versione $LATEST degli intenti perché puoi aggiornare solo la versione $LATEST delle tue risorse Amazon Lex.

1. Nella console Amazon Lex, scegli il BookTripbot.
2. Nella scheda Editor, scegli l'BookHotelintento. Aggiorna la configurazione dell'intento come segue:

   a. Assicurati che la versione dell'intento (accanto al nome dell'intento) sia $LATEST.

   b. Aggiungete la funzione Lambda come codice hook di inizializzazione e convalida come segue:

- In Options (Opzioni) scegli Initialization and validation code hook (Hook di codice di inizializzazione e di convalida).
- Selezionare la funzione Lambda dall'elenco.

c. Aggiungi la funzione Lambda come destinazione per il codice logistico come segue:

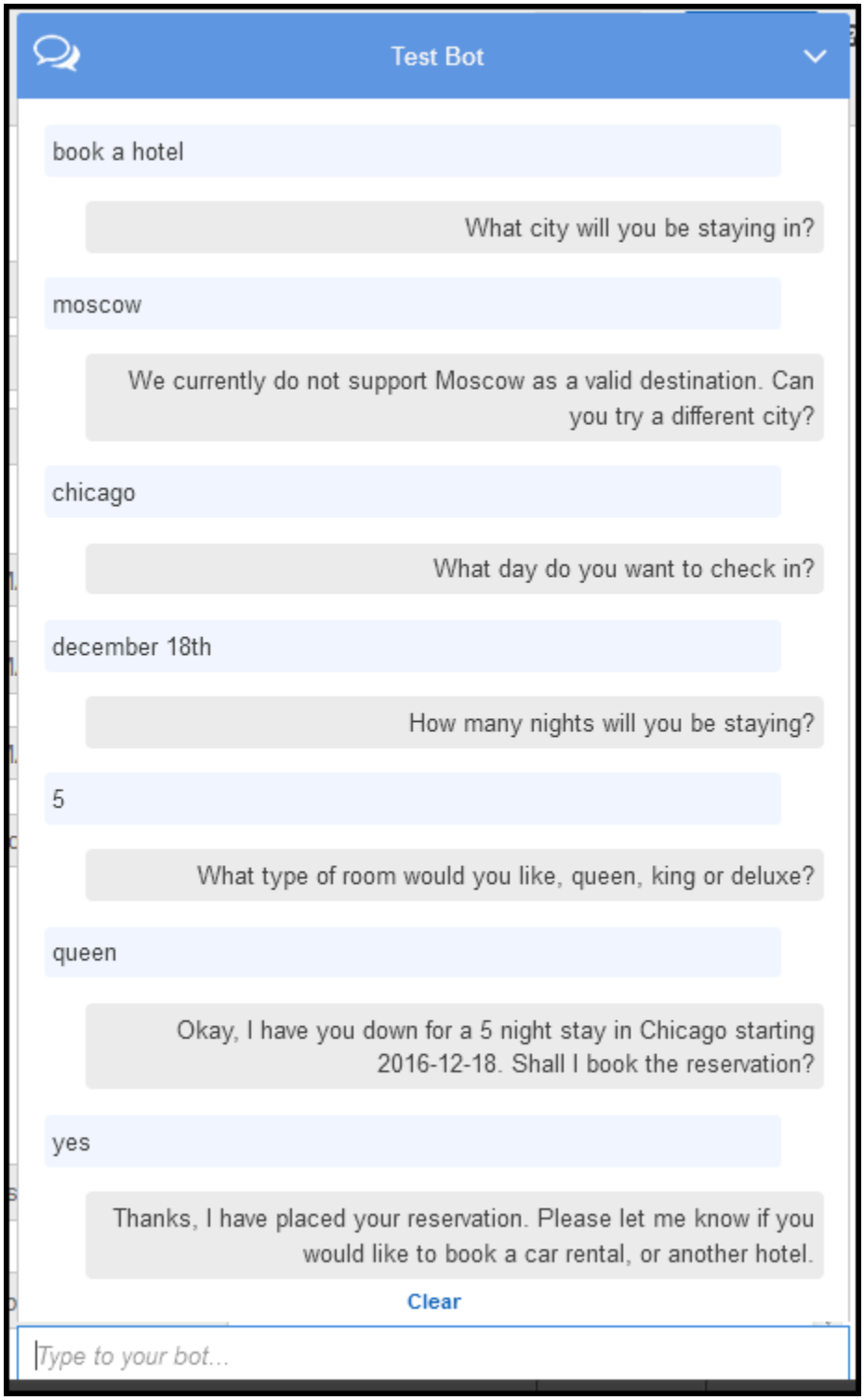
- In Fulfillment (Soddisfazione) scegli AWS Lambda function (Funzione AWS Lambda).
- Selezionare la funzione Lambda dall'elenco.
- Seleziona Goodbye message (Messaggio di saluto).

d. Seleziona Salva.

3. Nella scheda Editor, scegli l' BookCar intento. Segui la fase precedente per aggiungere la funzione Lambda come hook di codice di convalida e soddisfazione.

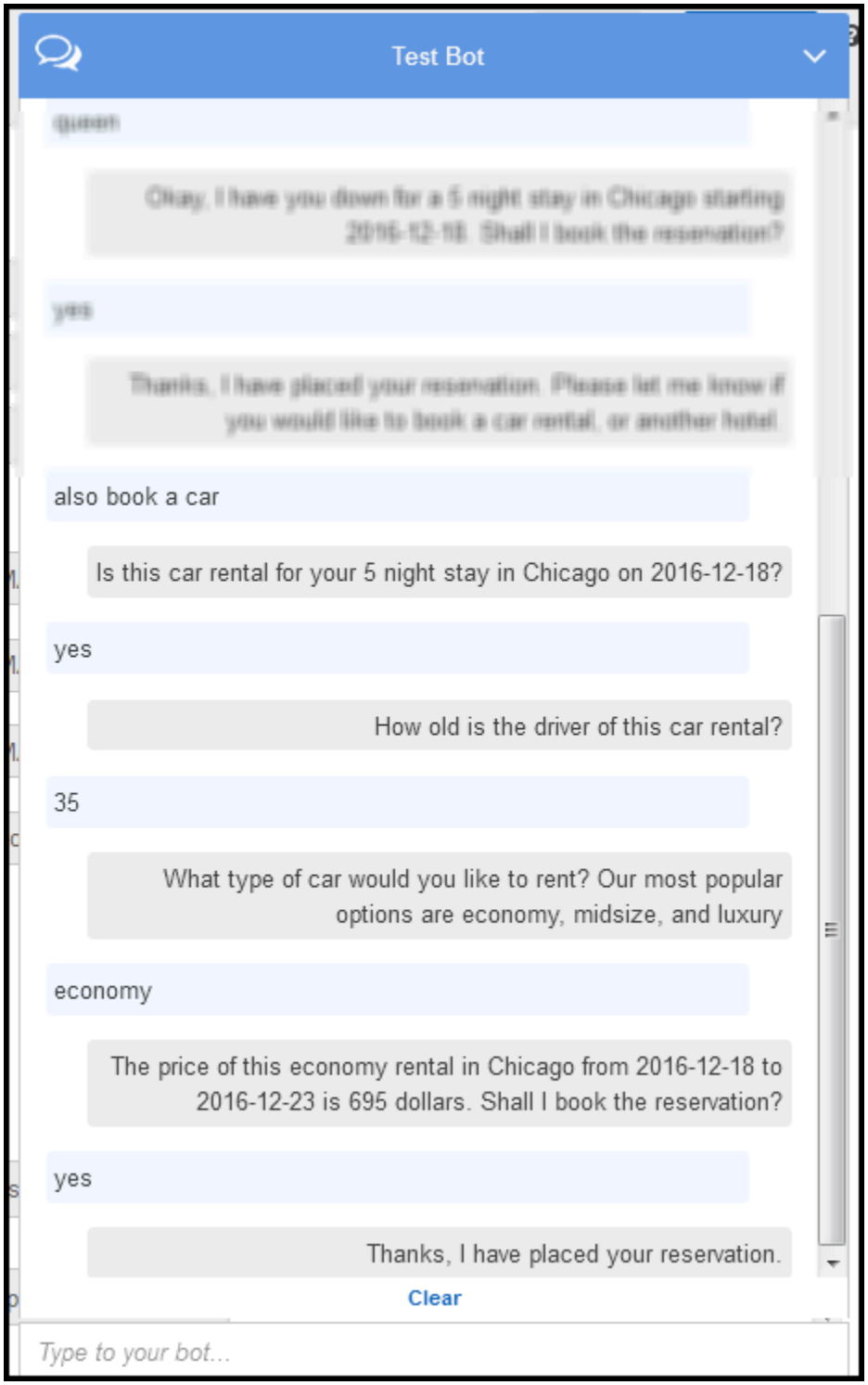
4. Seleziona Build (Crea). La console invia una serie di richieste ad Amazon Lex per salvare le configurazioni.

5. Esegui il test del bot. Ora che disponi di una funzione Lambda che esegue l'inizializzazione, la convalida e l'adempimento dei dati utente, puoi vedere la differenza nell'interazione dell'utente nella seguente conversazione:



Per ulteriori informazioni sul flusso di dati dal client (console) ad Amazon Lex e da Amazon Lex alla funzione Lambda, consultaFlusso di dati: intento di prenotazione hotel.

6. Continua la conversazione e prenota un'auto come mostrato nell'immagine seguente:



Quando scegli di prenotare un'auto, il client (console) invia una richiesta ad Amazon Lex che include gli attributi della sessione (dalla conversazione precedente BookHotel). Amazon Lex passa queste informazioni alla funzione Lambda, che quindi inizializza (ovvero precompila) alcuni dati dello BookCar slot (ovvero, PickUpDate ReturnDate, e PickUpCity).

> **Note**
>
> Viene illustrato come possono essere utilizzati gli attributi di sessione per mantenere il contesto tra gli intenti. Nella finestra di test del client della console è disponibile il collegamento Clear (Cancella), che consente all'utente di cancellare gli attributi di sessione precedenti.

Per ulteriori informazioni sul flusso di dati dal client (console) ad Amazon Lex e da Amazon Lex alla funzione Lambda, consultaFlusso di dati: intento di prenotazione automobile.

Dettagli del flusso di informazioni

In questo esercizio, hai avviato una conversazione con il BookTrip bot Amazon Lex utilizzando il client della finestra di test fornito nella console Amazon Lex. Questa sezione descrive quanto segue:

- Il flusso di dati tra il client e Amazon Lex.

  La sezione presuppone che il client invii richieste ad Amazon Lex utilizzando l'API`PostText` di runtime e mostri i dettagli della richiesta e della risposta di conseguenza. Per ulteriori informazioni sull'API di runtime `PostText`, consulta PostText.

  > **Note**
  >
  > Per un esempio del flusso di informazioni tra il client e Amazon Lex in cui il client utilizza l'`PostContent`API, consultaFase 2a (facoltativo): Revisione dei dettagli del flusso di informazioni parlate (console) .

- Il flusso di dati tra Amazon Lex e la funzione Lambda. Per ulteriori informazioni, consulta Formato di evento di input e di risposta della funzione Lambda.

Argomenti

- Flusso di dati: intento di prenotazione hotel
- Flusso di dati: intento di prenotazione automobile

Flusso di dati: intento di prenotazione hotel

Questa sezione spiega cosa accade dopo ogni input utente.

1. Utente: "book a hotel"

   a. Il client (console) invia la seguente richiesta PostText ad Amazon Lex:

   ```
   POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book a hotel",
      "sessionAttributes":{}
   }
   ```

   Sia l'URI della richiesta che il corpo forniscono informazioni ad Amazon Lex:

   - URI di richiesta: fornisce il nome del bot (BookTrip), l'alias del bot ($LATEST) e il nome utente. Il codice `text` finale indica che si tratta di una richiesta API `PostText` (non `PostContent`).
   - Corpo della richiesta: include l'input utente (`inputText`) e il campo `sessionAttributes` vuoto. Inizialmente, si tratta di un oggetto vuoto e la funzione Lambda imposta innanzitutto gli attributi della sessione.

   b. Da lì`inputText`, Amazon Lex rileva l'intento (BookHotel). Questo intento è configurato con una funzione Lambda come hook di codice per l'inizializzazione/convalida dei dati utente. Pertanto, Amazon Lex richiama quella funzione Lambda passando le seguenti informazioni come parametro dell'evento (vediFormato dell'evento di input):

   ```
   {
      "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"
   }
}
```

Oltre alle informazioni inviate dal cliente, Amazon Lex include anche i seguenti dati aggiuntivi:

- `messageVersion`— Attualmente Amazon Lex supporta solo la versione 1.0.
- `invocationSource`— Indica lo scopo dell'invocazione della funzione Lambda. In questo caso, si tratta di eseguire l'inizializzazione e la convalida dei dati utente (al momento Amazon Lex sa che l'utente non ha fornito tutti i dati dello slot per soddisfare l'intento).
- `currentIntent`: tutti i valori di slot sono impostati su null.

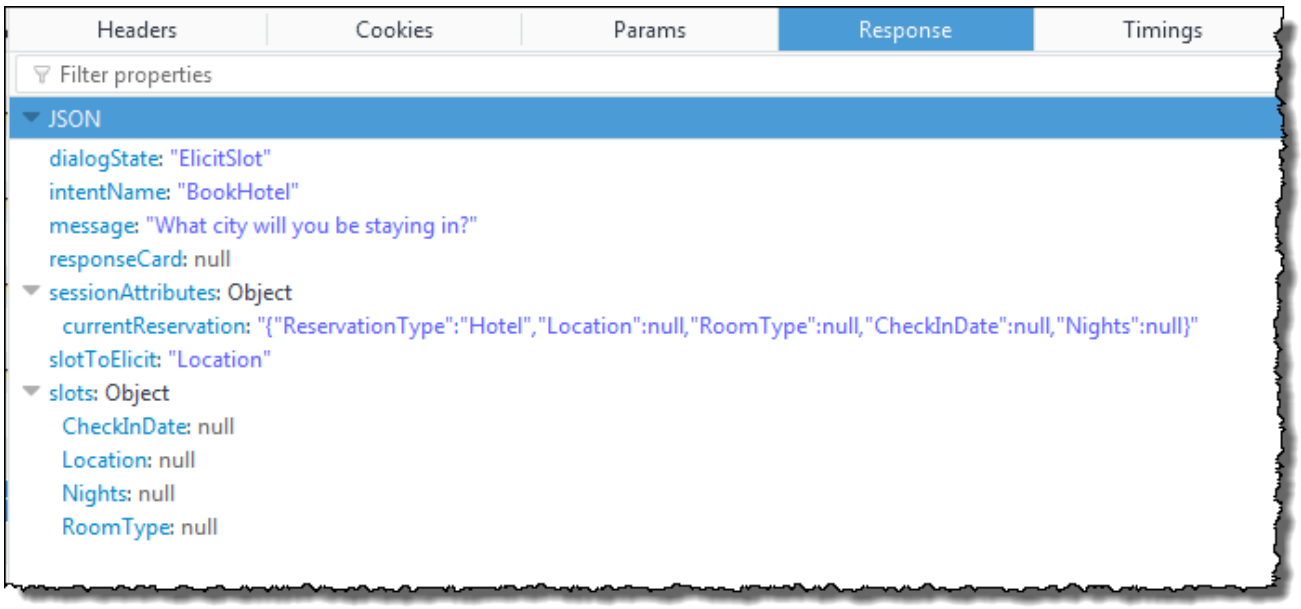
c. Al momento, tutti i valori dello slot sono nulli. Non c'è nulla da convalidare per la funzione Lambda. La funzione Lambda restituisce la seguente risposta ad Amazon Lex. Per informazioni sul formato della risposta, consulta Formato della risposta.

```
{
   "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`— La funzione Lambda include questo attributo di sessione. Il relativo valore è una copia delle informazioni sullo slot e del tipo di prenotazione correnti.
>
>   Solo la funzione Lambda e il client possono aggiornare questi attributi di sessione. Amazon Lex trasmette semplicemente questi valori.
> - `dialogAction.type`— Impostando questo valore su`Delegate`, la funzione Lambda delega la responsabilità della prossima linea d'azione ad Amazon Lex.
>
>   Se la funzione Lambda ha rilevato qualcosa nella convalida dei dati utente, indica ad Amazon Lex cosa fare dopo.

d. In base a`dialogAction.type` ciò, Amazon Lex decide la prossima linea d'azione: raccogliere dati dall'utente per lo`Location` slot. Seleziona uno dei messaggi di richiesta ("What city will you be staying in?") per questo slot in base alla configurazione dell'intento, quindi invia la seguente risposta all'utente:



Gli attributi di sessione vengono passati al client.

Il client legge la risposta e quindi visualizza il messaggio: "What city will you be staying in?"

2. Utente: "Moscow"

   a. Il client invia la seguente`PostText` richiesta ad Amazon Lex (interruzioni di riga aggiunte per la leggibilità):

```
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}"
   }
}
```

   Oltre a `inputText`, il client include gli stessi attributi di sessione `currentReservation` ricevuti.

b. Amazon Lex lo interpreta innanzitutto`inputText` nel contesto dell'intento attuale (il servizio ricorda di aver chiesto all'utente specifico informazioni sullo`Location` slot). Aggiorna il valore dello slot per l'intento corrente e richiama la funzione Lambda utilizzando il seguente evento:

```
{
    "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` continua a essere `DialogCodeHook`. In questa fase, eseguiamo soltanto la convalida dei dati utente.
> - Amazon Lex sta solo passando l'attributo di sessione alla funzione Lambda.
> - Perché`currentIntent.slots`, Amazon Lex ha aggiornato lo`Location` slot a`Moscow`.

c. La funzione Lambda esegue la convalida dei dati utente e determina che`Moscow` si tratta di una posizione non valida.

> **Note**
>
> La funzione Lambda in questo esercizio ha un semplice elenco di città valide e non`Moscow` è inclusa nell'elenco. In un'applicazione di produzione, è possibile utilizzare un database di back-end per ottenere queste informazioni.

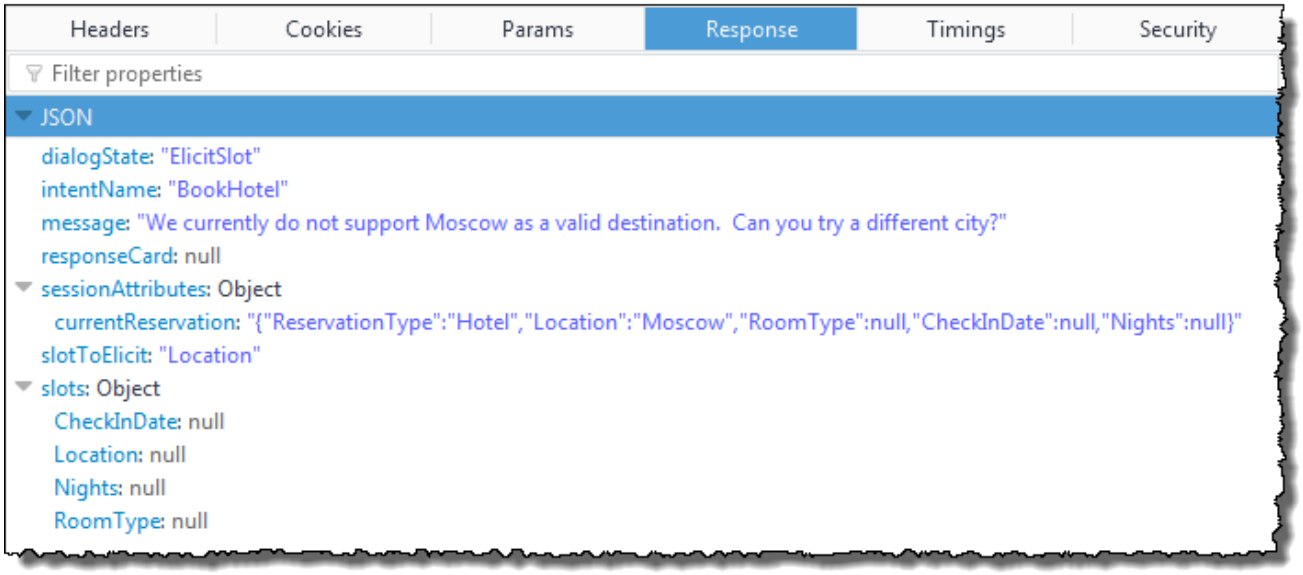
Reimposta il valore dello slot su null e indirizza Amazon Lex a richiedere nuovamente all'utente un altro valore inviando la seguente risposta:

```
{
    "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` viene reimpostato su null.

- `dialogAction.type`è impostato su`ElicitSlot`, il che indica ad Amazon Lex di richiedere nuovamente l'utente fornendo quanto segue:
  - `dialogAction.slotToElicit`: slot per il quale ottenere i dati dall'utente.
  - `dialogAction.message`: elemento `message` da trasmettere all'utente.

d. Amazon Lex rileva`dialogAction.type` e trasmette le informazioni al cliente nella seguente risposta:



Il client visualizza semplicemente il messaggio: "We currently do not support Moscow as a valid destination. Can you try a different city?"

3. Utente: "Chicago"

a. Il client invia la seguente`PostText` richiesta ad 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 conosce il contesto, che stava raccogliendo dati per lo`Location` slot. In questo contesto, sa che il valore `inputText` è per lo slot `Location`. Quindi richiama la funzione Lambda inviando il seguente evento:

```
{
    "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"
    }
}
```

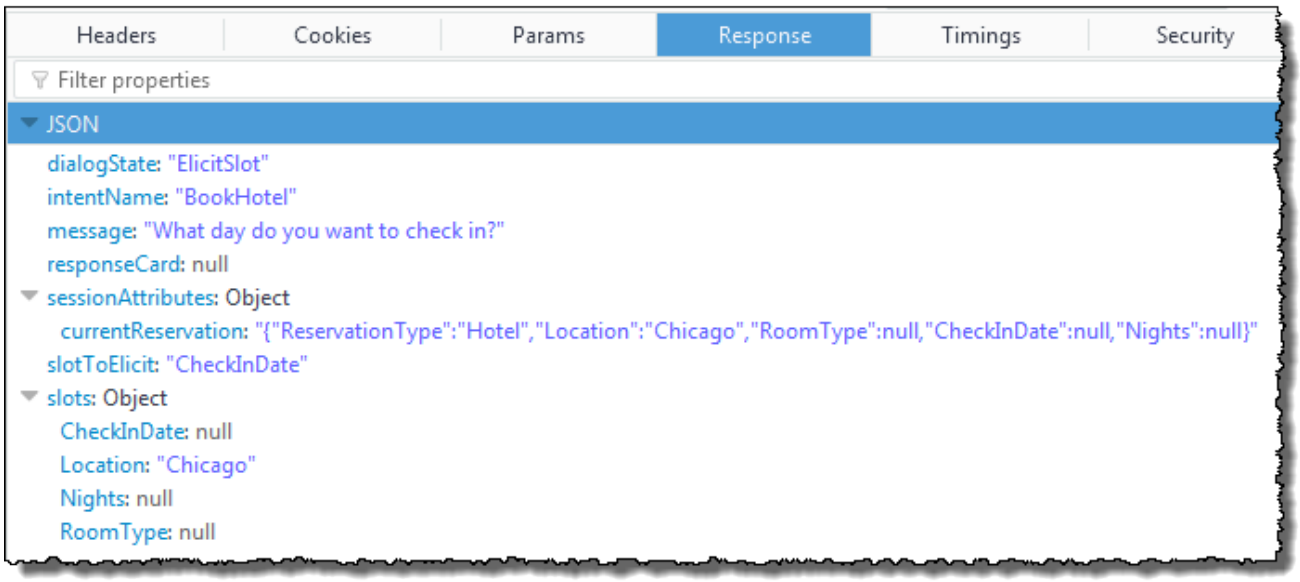
Amazon Lex ha aggiornato il`currentIntent.slots` impostando lo`Location` slot su`Chicago`.

c. In base al`invocationSource` valore di`DialogCodeHook`, la funzione Lambda esegue la convalida dei dati utente. Riconosce`Chicago` come valore di slot valido, aggiorna di conseguenza l'attributo della sessione e quindi restituisce la seguente risposta ad Amazon Lex.

```
{
    "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`— La funzione Lambda aggiorna questo attributo di sessione impostando il valore`Location` to`Chicago`.
> - `dialogAction.type`: è impostato su `Delegate`. I dati utente erano validi e la funzione Lambda indirizza Amazon Lex a scegliere la prossima linea d'azione.

d. Secondo`dialogAction.type` Amazon Lex sceglie la prossima linea d'azione. Amazon Lex sa di aver bisogno di più dati sugli slot e sceglie il successivo slot non riempito (`CheckInDate`) con la massima priorità in base alla configurazione dell'intento. Seleziona uno dei messaggi di richiesta ("What day do you want to check in?") per questo slot in base alla configurazione dell'intento, quindi invia la seguente risposta al client:

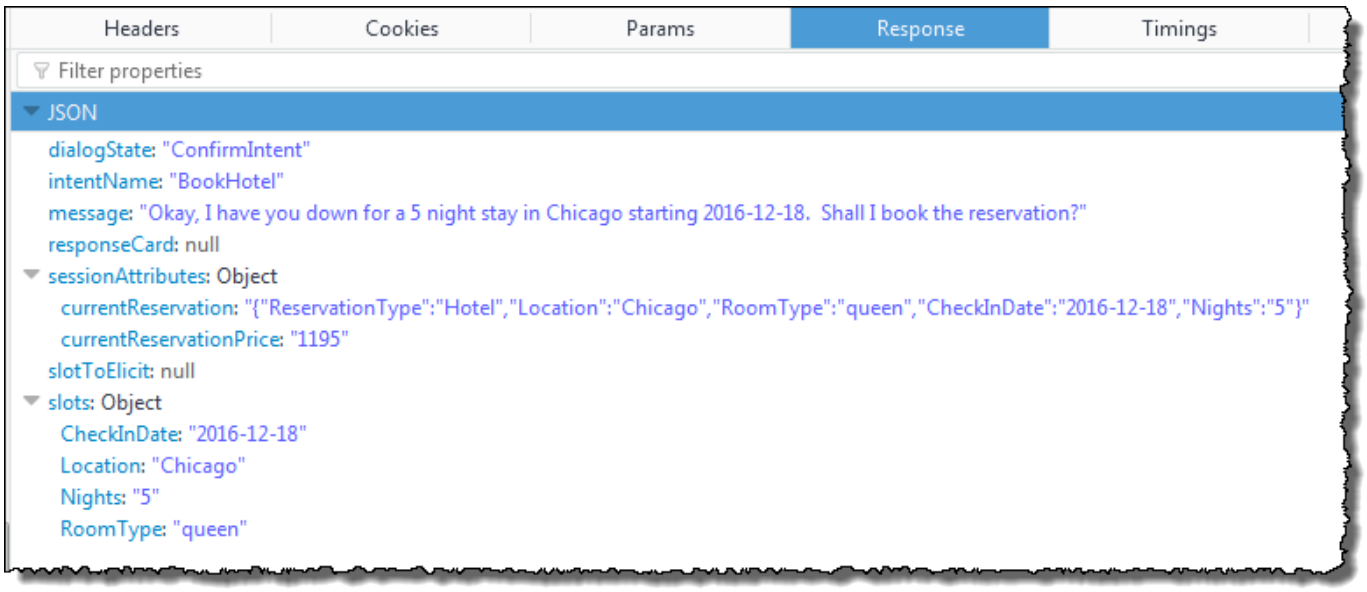


Il client visualizza il messaggio: "What day do you want to check in?"

4. L'interazione con l'utente continua: l'utente fornisce i dati, la funzione Lambda convalida i dati e quindi delega la prossima linea d'azione ad Amazon Lex. Alla fine l'utente fornisce tutti i dati dello slot, la funzione Lambda convalida tutti gli input dell'utente e quindi Amazon Lex riconosce di avere tutti i dati dello slot.

> **Note**
>
> In questo esercizio, dopo che l'utente ha fornito tutti i dati relativi agli slot, la funzione Lambda calcola il prezzo della prenotazione alberghiera e lo restituisce come altro attributo di sessione (`currentReservationPrice`).

A questo punto, l'intento è pronto per essere soddisfatto, ma l' BookHotelintento è configurato con una richiesta di conferma che richiede la conferma dell'utente prima che Amazon Lex possa soddisfare l'intento. Pertanto, Amazon Lex invia il seguente messaggio al cliente richiedendo la conferma prima di prenotare l'hotel:



Il client visualizza il messaggio: "Okay, I have you down for a 5 night in Chicago starting 2016-12-18. Shall I book the reservation?"

5. Utente: "yes"

   a. Il client invia la seguente`PostText` richiesta ad 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 lo interpreta`inputText` nel contesto della conferma dell'intento attuale. Amazon Lex capisce che l'utente desidera procedere con la prenotazione. Questa volta Amazon Lex richiama la funzione Lambda per soddisfare l'intento inviando il seguente evento. Impostando il valore`invocationSource` to`FulfillmentCodeHook` nell'evento,

viene inviato alla funzione Lambda. Amazon Lex imposta anche`confirmationStatus` il`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"
    }
}
```

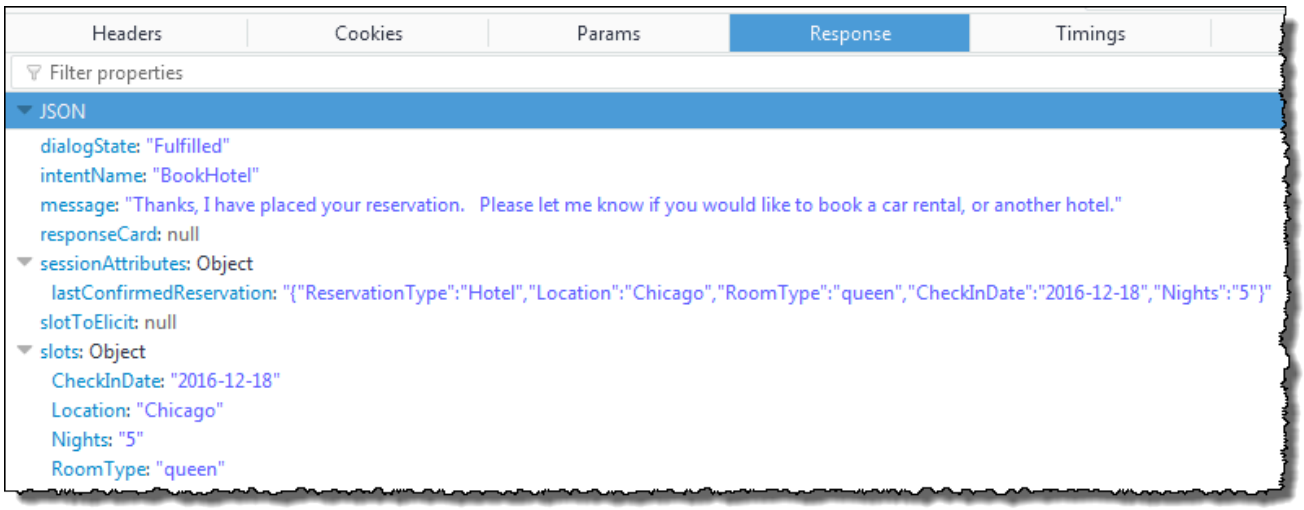
> **Note**
>
> - `invocationSource`— Questa volta, Amazon Lex ha impostato questo valore su`FulfillmentCodeHook`, indirizzando la funzione Lambda a soddisfare l'intento.
> - `confirmationStatus`: è impostato su `Confirmed`.

c. Questa volta, la funzione Lambda soddisfa l' BookHotel intento, Amazon Lex completa la prenotazione e quindi restituisce la seguente risposta:

```
{
    "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`— È un nuovo attributo di sessione aggiunto dalla funzione Lambda (anzichécurrentReservation,currentReservationPrice).
> - `dialogAction.type`— La funzione Lambda imposta questo valore su`Close`, indicando che Amazon Lex non si aspetta una risposta da parte dell'utente.
> - `dialogAction.fulfillmentState`: è impostato su `Fulfilled` (Soddisfatto) e include un elemento `message` appropriato da trasmettere all'utente.

d. Amazon Lex lo esamina`fulfillmentState` e invia la seguente risposta al cliente:



> **Note**
>
> - `dialogState`— Amazon Lex imposta questo valore su`Fulfilled`.
> - `message`— È lo stesso messaggio fornito dalla funzione Lambda.

Il client visualizza il messaggio.

Flusso di dati: intento di prenotazione automobile

Il BookTrip bot in questo esercizio supporta due scopi (BookHotel e BookCar). Dopo aver prenotato un hotel, l'utente può continuare la conversazione per prenotare un'automobile. Purché non si sia verificato il timeout della sessione, in ogni richiesta successiva il client continua a inviare gli attributi di sessione (in questo esempio, `lastConfirmedReservation`). La funzione Lambda può utilizzare queste informazioni per inizializzare i dati dello slot per l' BookCar intento. Ciò illustra come utilizzare gli attributi di sessione nella condivisione dei dati tra più intenti.

In particolare, quando l'utente sceglie l' BookCar intento, la funzione Lambda utilizza le informazioni pertinenti nell'attributo di sessione per precompilare gli slot (PickUpDate ReturnDate, e PickUpCity) relativi all' BookCar intento.

> **Note**
>
> La console Amazon Lex fornisce il link Clear che puoi utilizzare per cancellare gli attributi delle sessioni precedenti.

Seguire le fasi di questa procedura per continuare la conversazione.

1. Utente: "also book a car"

   a. Il client invia la seguente`PostText` richiesta ad 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\"}"
   }
}
```

   Il client include l'attributo di sessione `lastConfirmedReservation`.

   b. Amazon Lex rileva l'intento (BookCar) da`inputText`. Questo intento è anche configurato per richiamare la funzione Lambda per eseguire l'inizializzazione e la convalida dei dati utente. Amazon Lex richiama la funzione Lambda con il seguente evento:

```
{
    "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`— Attualmente Amazon Lex supporta solo la versione 1.0.
> - `invocationSource`: indica che lo scopo dell'invocazione è quello di eseguire l'inizializzazione e la convalida dei dati utente.
> - `currentIntent`— Include il nome dell'intento e gli slot. A questo punto, tutti i valori di slot sono null.

c. La funzione Lambda rileva tutti i valori di slot nulli senza nulla da convalidare. Utilizza tuttavia gli attributi di sessione per inizializzare alcuni valori di slot (`PickUpDateReturnDate` e `PickUpCity`) e restituisce la seguente risposta:

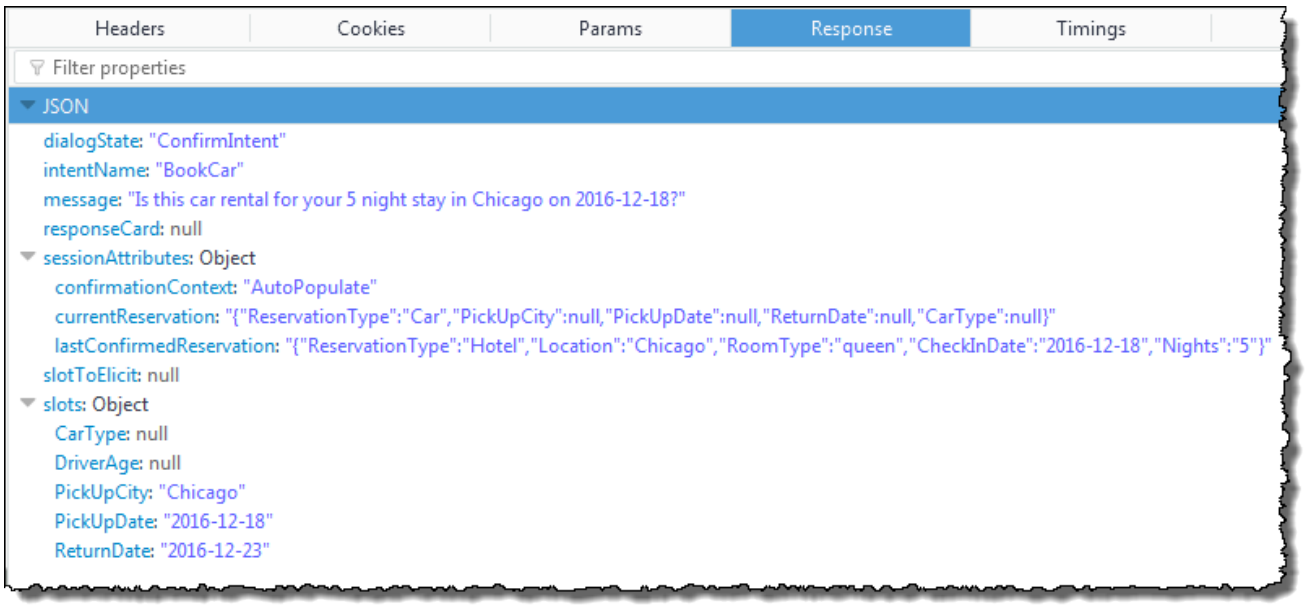
```
{
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
        "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**
>
> - Oltre `alastConfirmedReservation`, la funzione Lambda include più attributi di sessione (`currentReservationeconfirmationContext`).
> - `dialogAction.type`è impostato su`ConfirmIntent`, il che informa Amazon Lex che è prevista una risposta sì, no da parte dell'utente (la funzione ConfirmationContext è impostata su AutoPopulate, la funzione Lambda sa che la risposta utente sì/no serve a ottenere la conferma dell'utente dell'inizializzazione eseguita dalla funzione Lambda (dati slot compilati auto).
>
>
>
> La funzione Lambda include anche nella risposta un messaggio informativo da restituire`dialogAction.message` ad Amazon Lex al client.
>
> > **Note**
> >
> > Il termine `ConfirmIntent` (valore di `dialogAction.type`) non è correlato ad alcun intento di bot. Nell'esempio, la funzione Lambda utilizza

questo termine per indirizzare Amazon Lex a ricevere una risposta sì/no dall'utente.

d. Secondo il`dialogAction.type`, Amazon Lex restituisce al cliente la seguente risposta:



Il client mostra il messaggio: "Is this car rental for your 5 night stay in Chicago on 2016-12-18?"

2. Utente: "yes"

a. Il client invia la seguente`PostText` richiesta ad 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 leggeinputText e conosce il contesto (chiede all'utente di confermare la popolazione auto). Amazon Lex richiama la funzione Lambda inviando il seguente evento:

```
{
    "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"
    }
}
```

Poiché l'utente ha risposto Sì, Amazon Lex impostaconfirmationStatus ilConfirmed.

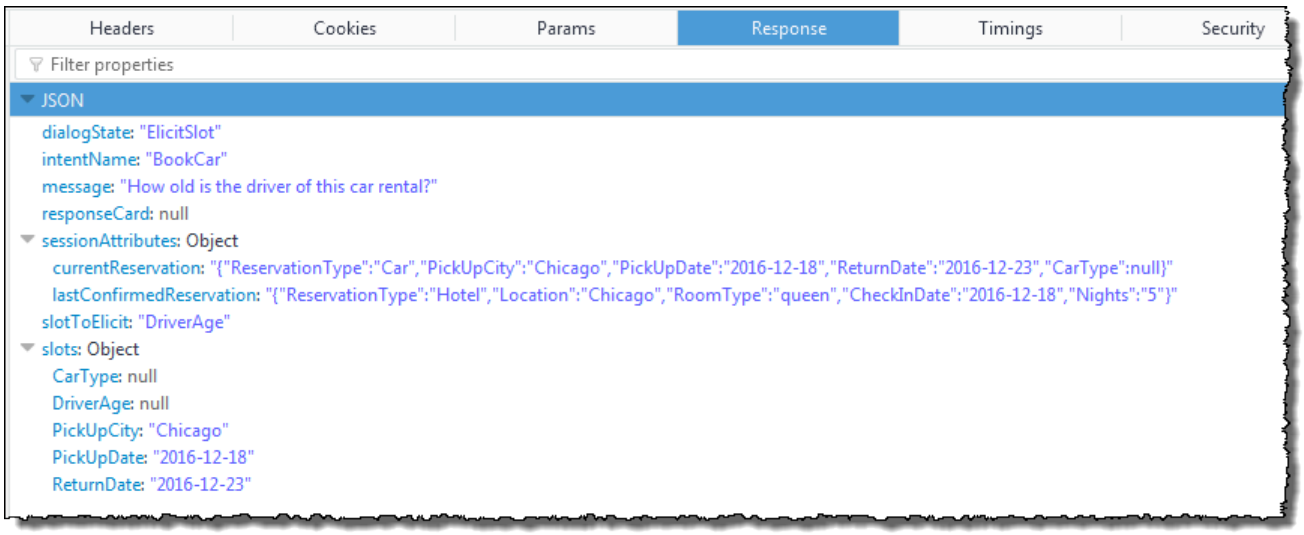
c. Da lì`confirmationStatus`, la funzione Lambda sa che i valori precompilati sono corretti. La funzione Lambda; svolge le operazioni seguenti:

   - Aggiorna l'attributo di sessione `currentReservation` con il valore di slot precompilato.
   - Imposta `dialogAction.type` su `ElicitSlot`.
   - Imposta il valore `slotToElicit` su `DriverAge`.

   Viene inviata la seguente risposta:

```
{
    "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 restituisce la seguente risposta:



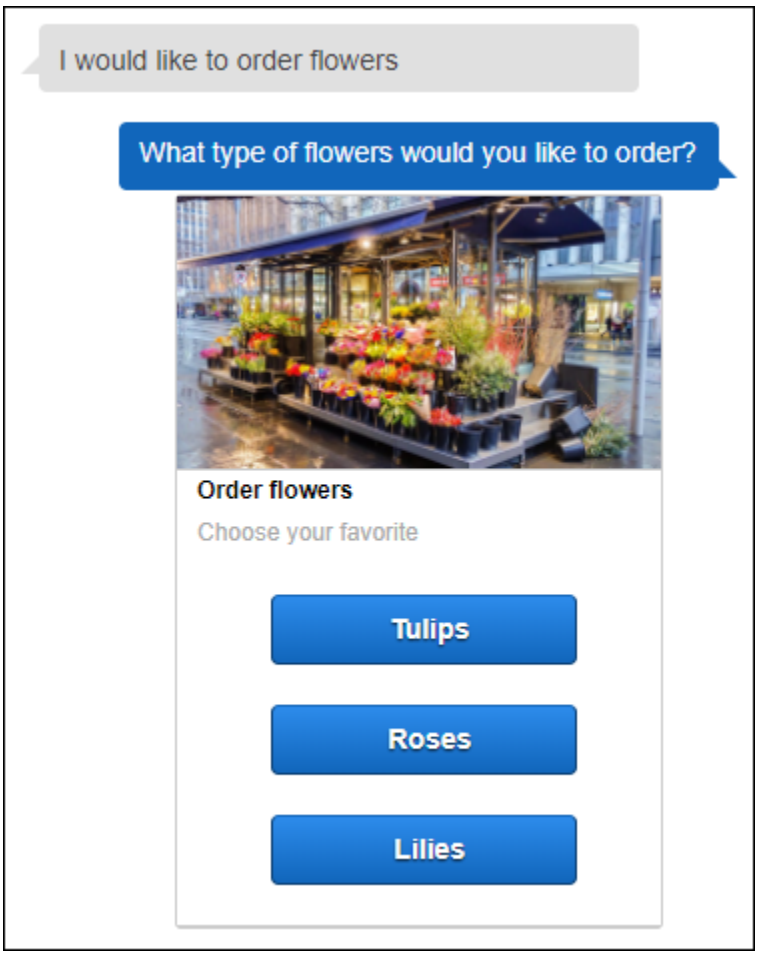
Il cliente visualizza il messaggio «Quanti anni ha il conducente di questo noleggio auto?» e la conversazione continua.

Utilizzo di una scheda di risposta

Questo esercizio è un'estensione dell'esercizio 1 delle Nozioni di base e comporta l'aggiunta di una scheda di risposta. Si crea un bot che supporta l' OrderFlowers intento e quindi si aggiorna l'intento aggiungendo una scheda di risposta per lo`FlowerType` slot. In aggiunta al seguente prompt per lo slot `FlowerType`, l'utente potrà scegliere il tipo di fiori dalla scheda di risposta:

```
What type of flowers would you like to order?
```

La scheda di risposta è la seguente:

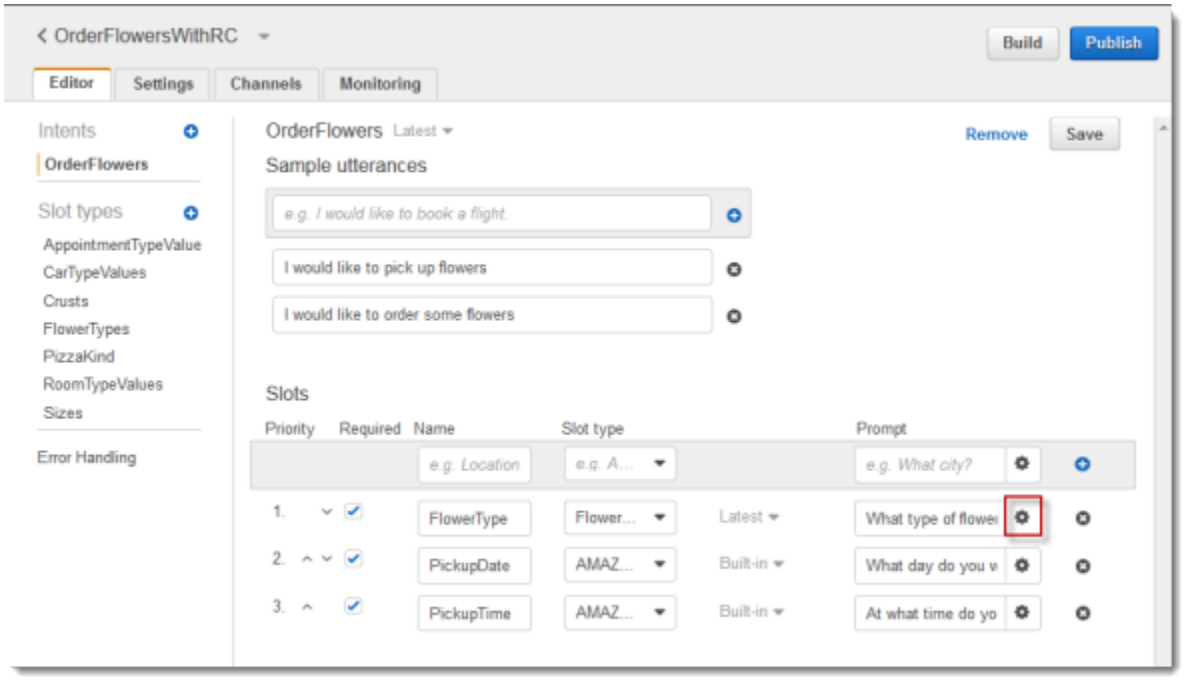


L'utente bot può digitare il testo o effettuare una scelta dall'elenco dei tipi di fiori. La scheda di risposta è configurata con un'immagine, che appare nel client come mostrato. Per ulteriori informazioni sulle schede di risposta, consultare Schede di risposta.

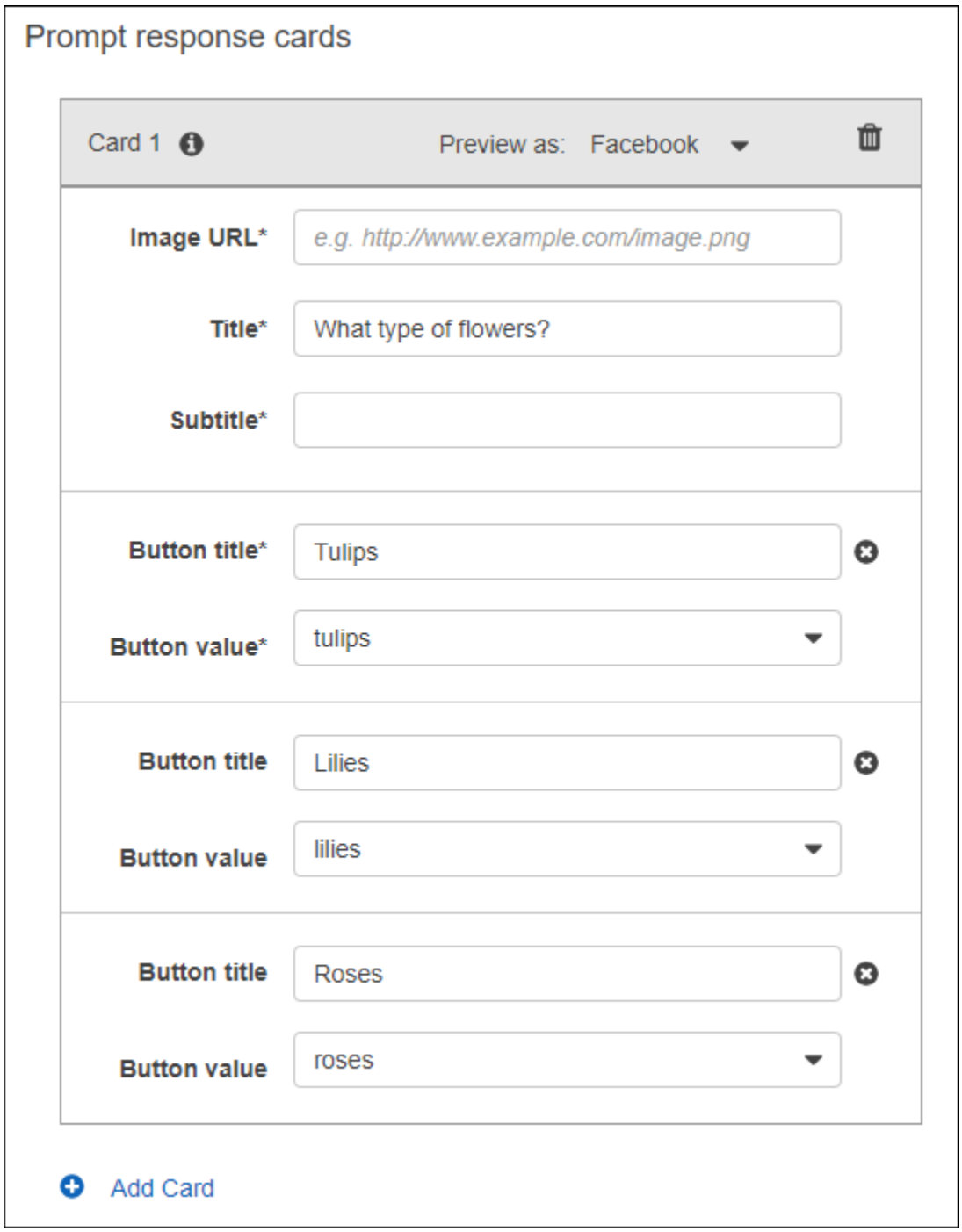
Per creare ed eseguire il test di un bot con una scheda di risposta:

1. Segui l'esercizio 1 introduttivo per creare e testare un OrderFlowers bot. È necessario completare le fasi 1, 2 e 3. Non è necessario aggiungere una funzione Lambda per testare la scheda di risposta. Per istruzioni, consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).
2. Aggiorna il bot aggiungendo la scheda di risposta, quindi pubblica una versione. Quando pubblichi una versione, specifica un alias (BETA) a cui puntare.
   a. Nella console Amazon Lex, scegliere il bot.
   b. Seleziona l'intento `OrderFlowers`.

c. Scegli l'icona a forma di ingranaggio delle impostazioni accanto al prompt «Che tipo di fiori» per configurare una scheda di risposta per il`FlowerType`, come mostrato nell'immagine seguente.



d. Assegna un titolo alla scheda e configura tre pulsanti come mostrato nella schermata che segue. Facoltativamente, puoi anche aggiungere alla scheda di risposta un'immagine, purché tu disponga di un URL dell'immagine. Se stai distribuendo il bot tramite Twilio SMS, devi fornire un URL immagine.



e. Seleziona Save (Salva) per salvare la scheda di risposta.

f. Seleziona Save intent (Salva intento) per salvare la configurazione dell'intento.

g. Per creare il bot, seleziona Build (Crea).

h. Per pubblicare una versione di bot, seleziona Publish (Pubblica). Specifica BETA come un alias che punta alla versione del bot. Per informazioni sulla funzione Versioni multiple, consultare Funzione Versioni multiple e alias.

3. Distribuisci il bot su una piattaforma di messaggistica:

   - Distribuisci il bot sulla piattaforma Facebook Messenger ed esegui il test dell'integrazione. Per istruzioni, consulta Integrazione di un Amazon Lex Bot con Facebook Messenger. Quando l'utente ordina dei fiori, la finestra dei messaggi mostra la scheda di risposta che consente di scegliere un tipo di fiore.
   - Distribuisci il bot sulla piattaforma Slack ed esegui il test dell'integrazione. Per istruzioni, consulta Integrazione di un bot Amazon Lex con Slack . Quando l'utente ordina dei fiori, la finestra dei messaggi mostra la scheda di risposta che consente di scegliere un tipo di fiore.
   - Distribuisci il bot sulla piattaforma Twilio SMS: Per istruzioni, consulta Integrazione di un Amazon Lex Bot con SMS programmabili Twilio . Quando l'utente ordina dei fiori, il messaggio di Twilio mostra l'immagine dalla scheda di risposta. Twilio SMS non supporta i pulsanti nella risposta.

Aggiornamento degli enunciati

In questo esercizio, aggiungerai ulteriori enunciazioni a quelle create nell'esercizio 1 delle Nozioni di base. Utilizza la scheda Monitoraggio nella console Amazon Lex per visualizzare le espressioni che il tuo bot non ha riconosciuto. Per migliorare l'esperienza dei tuoi utenti, puoi aggiungere tali enunciazioni al bot.

Le statistiche sull'enunciato non si verifica una delle seguenti condizioni:

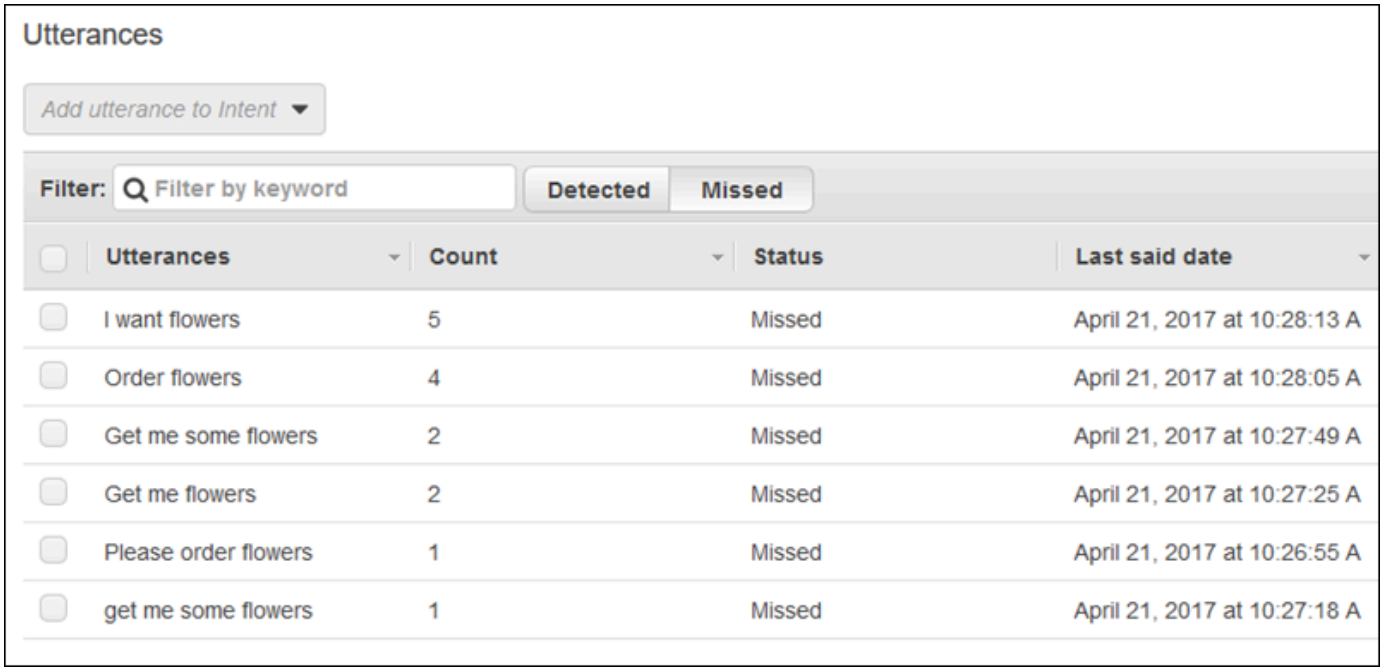
- Il`childDirected` campo è stato impostato su true al momento della creazione del bot.
- Stai utilizzando l'offuscamento degli slot con uno o più slot.
- Hai scelto di non partecipare al miglioramento di Amazon Lex.

> **Note**
>
> Una volta al giorno vengono create le statistiche delle enunciazioni. Puoi visualizzare l'enunciazione che non è stata riconosciuta, quante volte è stata ascoltata e l'ultimo giorno e l'ora in cui è stata ascoltata. Potrebbero essere necessarie fino a 24 ore prima che le enunciazioni mancanti vengano visualizzate nella console.

È possibile visualizzare le espressioni per diverse versioni del tuo bot. Per modificare la versione del bot di cui visualizzi le espressioni, scegliere una versione diversa dall'elenco a discesa accanto al nome del bot.

Per visualizzare e aggiungere le enunciazioni mancanti a un bot:

1. Segui il primo passaggio dell'esercizio 1 nelle Nozioni di base per creare e testare un bot `OrderFlowers`. Per istruzioni, consulta Esercizio 1: Creare un bot Amazon Lex utilizzando un blueprint (console).
2. Esegui il test del bot digitando le seguenti enunciazioni nella finestra Test Bot (Esegui test del bot). Digita ogni enunciazione più volte. Il bot di esempio non riconosce le seguenti enunciazioni:
   - Ordina dei fiori
   - Prendimi dei fiori
   - Per favore, ordina dei fiori
   - Prendimi qualche fiore
3. Attendi che Amazon Lex raccolga i dati di utilizzo relativi agli enunciati persi. I dati sulle enunciazioni vengono generati una volta al giorno, solitamente di notte.
4. Accedi allaAWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
5. Seleziona il bot `OrderFlowers`.
6. Seleziona la scheda Monitoring (Monitoraggio), quindi Utterances (Enunciazioni) nel menu a sinistra, infine scegli il pulsante Missed (Mancanti). Il riquadro seguente mostra un massimo di 100 espressioni mancate.



7. Per scegliere le enunciazioni mancanti che vuoi aggiungere al bot, seleziona la casella di controllo accanto ad esse. Per aggiungere l'enunciazione alla versione `$LATEST` dell'intento, scegli la freccia verso il basso accanto all'elenco a discesa Add utterance to intent (Aggiungi enunciazione a intento), quindi scegli l'intento.
8. Per ricreare il tuo bot, scegli Build (Crea), quindi nuovamente Build (Crea).
9. Per verificare se il tuo bot riconosce le nuove enunciazioni, usa il riquadro Test Bot (Esegui test del bot).

integrazione in un sito Web

In questo esempio si integra un bot in un sito Web utilizzando testo e voce. I servizi JavaScript e AWS ti consentono di creare un'esperienza interattiva per i visitatori del tuo sito Web. È possibile scegliere tra questi esempi documentati nel Blog AWS sull'intelligenza artificiale:

- Distribuisci un'interfaccia utente Web per il chatbotMostra un'interfaccia utente Web completa che offre un client Web per i chatbot Amazon Lex. È possibile utilizzare questi esempi per apprendere informazioni sui client Web o per usarli come elementi di base nelle tue applicazioni.

- "Greetings, visitor!" —Coinvolgi i tuoi utenti Web con Amazon LexMostra l'utilizzo di Amazon Lex, del kit SDK AWS per JavaScript nel browser e di Amazon Cognito per creare un'esperienza di conversazione sul tuo sito Web.

- Acquisizione dell'input vocale in un browser e invio ad Amazon LexMostra l'integrazione di un chatbot vocale in un sito Web utilizzando il kit SDK for JavaScript in the Browser. L'applicazione registra l'audio, lo invia ad Amazon Lex e riproduce la risposta.

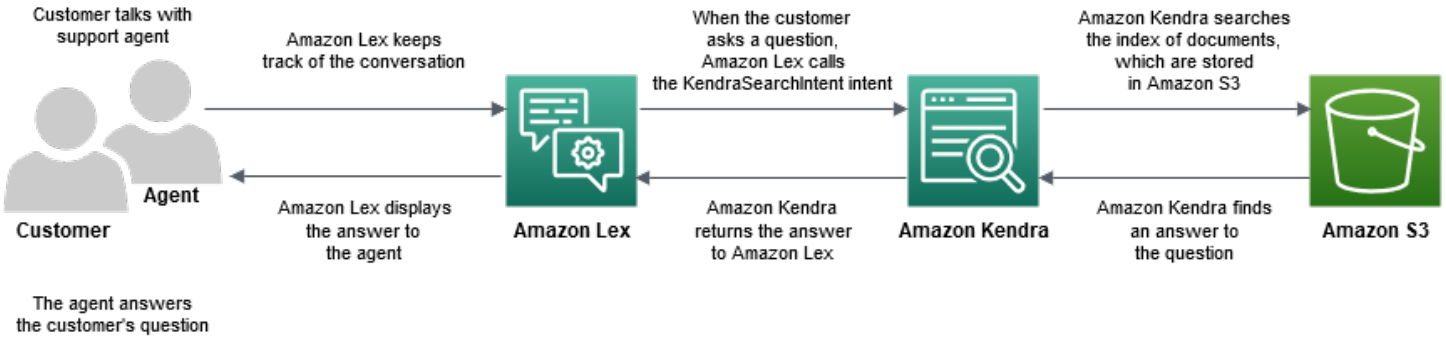
Assistente addetto al call center

In questo tutorial, utilizzi Amazon Lex con Amazon Kendra per creare un bot di assistenza agli agenti che assiste gli agenti dell'assistenza clienti e pubblicarlo come applicazione web. Amazon Kendra è un servizio di ricerca aziendale che utilizza l'apprendimento automatico per cercare tra i documenti e trovare le risposte. Per ulteriori informazioni su Amazon Kendra, consulta la Amazon Kendra Developer Guide.

I bot Amazon Lex sono ampiamente utilizzati nei call center come primo punto di contatto per i clienti. Un bot è spesso in grado di risolvere le domande dei clienti. Quando un bot non è in grado di rispondere a una domanda, trasferisce la conversazione a un dipendente dell'assistenza clienti.

In questo tutorial, creiamo un bot Amazon Lex che gli agenti utilizzano per rispondere alle domande dei clienti in tempo reale. Leggendo le risposte fornite dal bot, l'agente viene risparmiato dalla ricerca manuale delle risposte.

Il bot e l'applicazione web che crei in questo tutorial aiutano gli agenti a rispondere ai clienti in modo efficiente e preciso fornendo rapidamente le risorse giuste. Nel seguente diagramma viene illustrato come funziona l'applicazione Web.



Come mostra il diagramma, l'indice Amazon Kendra è archiviato in un bucket Amazon Simple Storage Service (Amazon S3). Se non hai già un bucket S3, puoi configurarlo quando crei l'indice

Amazon Kendra. Oltre ad Amazon S3, utilizzerai Amazon Cognito per questo tutorial. Amazon Cognito gestisce le autorizzazioni per la distribuzione del bot come applicazione Web.

In questo tutorial, crei un indice Amazon Kendra che fornisce risposte alle domande dei clienti, crei il bot e aggiungi intenti che gli consentano di suggerire risposte in base alla conversazione con il cliente, configura Amazon Cognito per gestire le autorizzazioni di accesso e distribuisci il bot come applicazione web.

Tempo previsto: 75 minuti

Costo stimato: 2,50 USD all'ora per un indice Amazon Kendra e 0,75 USD per 1000 richieste Amazon Lex. L'indice Amazon Kendra continua a funzionare al termine di questo esercizio. Assicurati di eliminarlo per evitare costi inutili.

Nota: assicurati di scegliere la stessa regione AWS per tutti i servizi utilizzati in questo tutorial.

Argomenti

Fase 1: creazione di un indice Amazon Kendra

Inizia creando un indice di documenti Amazon Kendra che risponda alle domande dei clienti. Un indice fornisce un'API di ricerca per le query dei client. L'indice viene creato a partire dai documenti di origine. Amazon Kendra restituisce le risposte trovate nei documenti indicizzati al bot, che le mostra all'agente.

La qualità e l'accuratezza delle risposte suggerite da Amazon Kendra dipendono dai documenti indicizzati. I documenti devono includere file a cui l'agente accede frequentemente e devono essere archiviati in un bucket S3. Puoi indicizzare dati non strutturati e semistrutturati nei formati .html, Microsoft Office (.doc, .ppt), PDF e testo.

Per creare un indice Amazon Kendra, consulta Guida introduttiva a un bucket S3 (console) nella Amazon Kendra Developer Guide.

Per aggiungere domande e risposte (FAQ) che aiutano a rispondere alle domande dei clienti, vedi Aggiungere domande e risposte nella Amazon Kendra Developer Guide. Per questo tutorial, usa il file ML_FAQ.csv su GitHub.

Approfondimenti

Fase 2: creazione di un Amazon Lex Bot

Fase 2: creazione di un Amazon Lex Bot

Amazon Lex fornisce un'interfaccia tra l'agente del call center e l'indice Amazon Kendra. Tiene traccia della conversazione tra l'agente e il cliente e definisce l'`AMAZON.KendraSearchIntent`intento in base alle domande poste dal cliente. Un intento è un'operazione che l'utente vuole eseguire.

Amazon Kendra cerca i documenti indicizzati e restituisce una risposta ad Amazon Lex che mostra nel bot. Questa risposta è visibile solo all'agente.

Per creare un assistente bot

1. Accedere aAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Nel riquadro di navigazione, scegliere Bot.
3. Seleziona Create (Crea).
4. Scegli Bot personalizzato e configura il bot.
    a. Nome bot: inserisci un nome che indichi lo scopo del bot, ad esempio**`AgentAssistBot`**.
    b. Voce in uscita: scegli Nessuno.
    c. Timeout della sessione: Invio**5**.
    d. COPPA — Scegli No.
5. Seleziona Create (Crea). Dopo aver creato il bot, Amazon Lex visualizza la scheda dell'editor dei bot.

Approfondimenti

Fase 3: creazione di un intento personalizzato e integrato

Fase 3: creazione di un intento personalizzato e integrato

Un intento rappresenta un'azione che l'agente del call center desidera che il bot esegua. In questo caso, l'agente desidera che il bot suggerisca risposte e risorse utili in base alla conversazione dell'agente con il cliente.

Amazon Lex ha due tipi di intenti: intenti personalizzati e intenti integrati. `AMAZON.KendraSearchIntent`è un intento integrato. Il bot utilizza l'`AMAZON.KendraSearchIntent`intento di interrogare l'indice e visualizzare le risposte suggerite da Amazon Kendra.

Il bot in questo esempio non ha bisogno di un intento personalizzato. Tuttavia, per creare il bot, devi creare almeno un intento personalizzato con almeno un enunciato di esempio. Questo intento è necessario solo per creare il tuo agent assistant bot. Non svolge nessun'altra funzione. L'espressione dell'intento non deve rispondere a nessuna delle domande che il cliente potrebbe porre. Ciò garantisce che`AMAZON.KendraSearchIntent` sia chiamato a rispondere alle domande dei clienti. Per ulteriori informazioni, consulta AMAZON.KendraSearchIntent.

Per creare l'intento personalizzato richiesto

1. Nella pagina Nozioni di base sul bot scegli Crea intento.
2. Per Aggiungi intento, scegli Crea intento.
3. Nella finestra di dialogo Crea intento, inserisci un nome descrittivo per l'intento, ad esempio**`RequiredIntent`**.
4. Per Espressioni di esempio, inserisci un'enunciazione descrittiva, ad esempio**`Required utterance`**.
5. Scegliere Salva intento.

Per aggiungere AMAZON. KendraSearchIntent intento e messaggio di risposta

1. Nel riquadro di navigazione, scegli il segno più (+) accanto a Intents.
2. Scegli Cerca intenti esistenti.
3. Nella casella Intenti di ricerca, inserisci**`AMAZON.KendraSearchIntent`**, quindi selezionalo dall'elenco.
4. Assegna all'intento un nome descrittivo, ad esempio**`AgentAssistSearchIntent`**, quindi scegli Aggiungi.

5. Nell'editor degli intenti, scegli Query Amazon Kendra per aprire le opzioni delle query.
6. Scegli l'indice in cui vuoi che venga effettuata la ricerca,
7. Nella sezione Risposta, aggiungi i seguenti tre messaggi a un gruppo di messaggi.

```
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. Scegliere Salva intento.
9. Scegli Costruisci per creare il bot.

Approfondimenti

Fase 4: configurazione Amazon Cognito

Fase 4: configurazione Amazon Cognito

Per gestire le autorizzazioni e gli utenti per l'applicazione Web, devi configurare Amazon Cognito. Amazon Cognito garantisce che l'applicazione Web sia sicura e abbia il controllo degli accessi. Amazon Cognito usa pool di identità per fornireAWS le credenziali che consentono agli utenti di accedere ad altriAWS servizi. Per questo tutorial, fornisce l'accesso ad Amazon Lex.

Quando crei un pool di identità, Amazon Cognito ti fornisce ruoliAWS Identity and Access Management (IAM) per utenti autenticati e non autenticati. Puoi modificare i ruoli IAM aggiungendo policy che concedono l'accesso ad Amazon Lex.

Configurazione di Amazon Cognito

1. Accedi aAWS Management Console e apri la console Amazon Cognito all'indirizzo https://console.aws.amazon.com/cognito/.
2. Scegli Manage Identity Pools (Gestisci pool di identità).
3. Scegli Create new identity pool (Crea un nuovo pool di identità).
4. Configurazione del pool di identità.

a. Nome del pool di identità: immettere un nome che indichi lo scopo del pool, ad esempio**`BotPool`**.

b. Nella sezione Identità non autenticate, scegli Abilita l'accesso alle identità non autenticate.

5. Seleziona Create Pool (Crea pool).
6. Nella pagina Identifica i ruoli IAM da utilizzare con il nuovo pool di identità, scegli Visualizza dettagli.
7. Registrazione del nome del ruolo IAM. Le modificherai successivamente.
8. Scegli Allow (Permetti).
9. Nella pagina Guida introduttiva ad Amazon Cognito, per Platform, scegli JavaScript.
10. Nella sezione OttieniAWS credenziali, trova e registra l'ID del pool di identità.
11. Per consentire l'accesso ad Amazon Lex, modifica i ruoli IAM autenticati e non autenticati.

    a. Accedi alla AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.

    b. Nel riquadro di navigazione, in Gestione accessi, scegli Ruoli.

    c. Nella casella di ricerca, inserire il nome del ruolo IAM autenticato e scegliere la casella di controllo accanto ad esso.

        i. Scegli Collega policy.

        ii. Nella casella di ricerca, digitare**`AmazonLexRunBotsOnly`** e scegliere la casella di controllo accanto ad essa.

        iii. Scegli Attach policy (Collega policy).

    d. Immettere il nome del ruolo IAM non autenticato nella casella di ricerca e scegliere la casella di controllo accanto ad esso.

        i. Scegli Collega policy.

        ii. Nella casella di ricerca, digitare**`AmazonLexRunBotsOnly`** e scegliere la casella di controllo accanto ad essa.

        iii. Scegli Attach policy (Collega policy).

Approfondimenti

Fase 5: Implementa il tuo bot come applicazione Web

Fase 5: Implementa il tuo bot come applicazione Web

Per distribuire il bot come applicazione web

1. Scarica il repository all'indirizzo https://github.com/awsdocs/amazon-lex-developer-guide /blob/master/example_apps/agent_assistance_bot/ sul tuo computer.
2. Accedere al repository scaricato e aprire il file index.html in un editor.
3. Apportare le seguenti modifiche:

   a. Nella`AWS.config.credentials` sezione, inserisci il nome della tua regione e l'ID del tuo pool di identità.

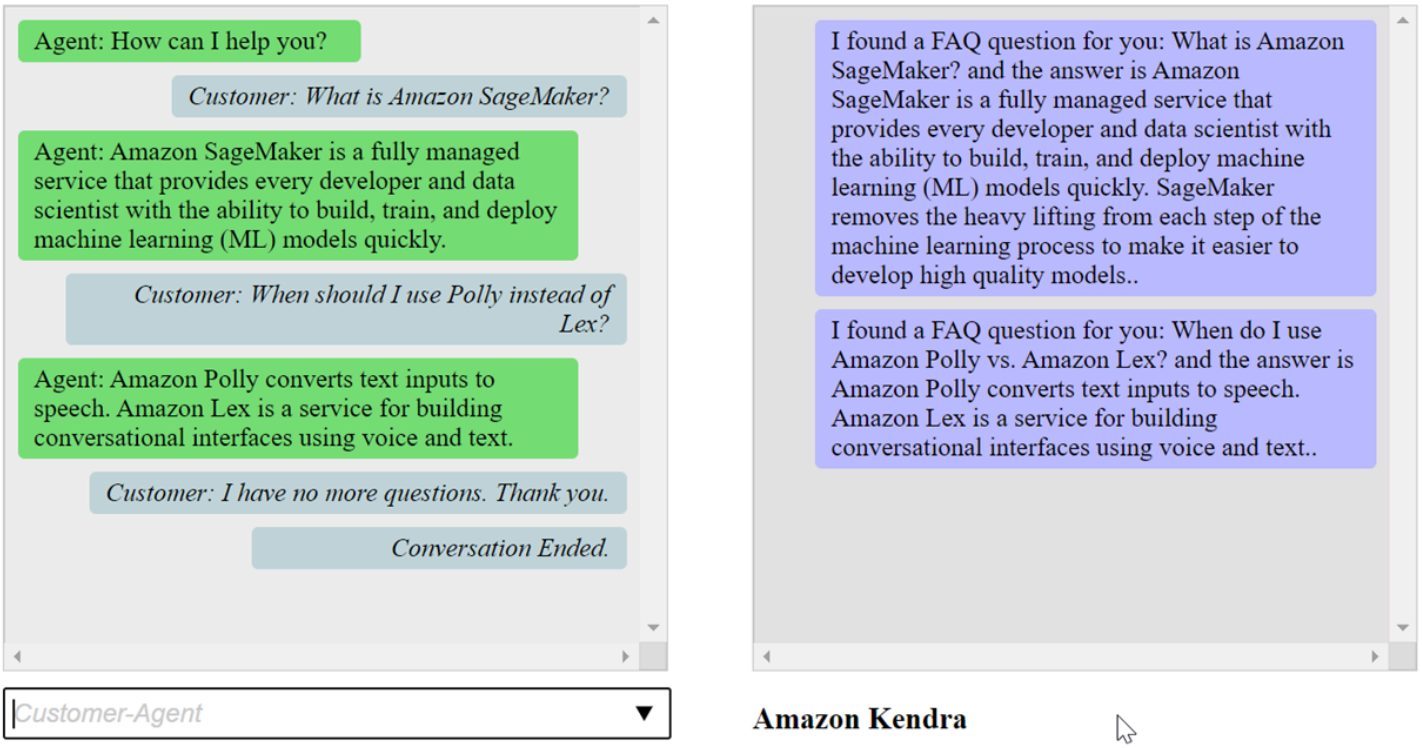
   b. Nella `Amazon Lex runtime parameters` sezione, inserisci il nome del bot.

   c. Salva il file.

Passaggio: utilizzare il bot

A scopo dimostrativo, fornisci input al bot come cliente e come agente. Per distinguere tra le due, le domande poste dal cliente iniziano con «Cliente:» e le risposte fornite dall'agente iniziano con «Agente:». Puoi scegliere da un menu di input suggeriti.

Esegui la tua applicazione web`index.html` aprendola per avviare una conversazione simile alla seguente immagine con il tuo bot:



La funzione `pushChat()` nel file index.html è spiegata di seguito.

```
            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;
```

Quando fornisci input come cliente, l'API di runtime di Amazon Lex li invia ad Amazon Lex.

La`showMessage(daText, senderRequest, displayWindow)` funzione mostra la conversazione tra l'agente e il cliente nella finestra di chat. Le risposte suggerite da Amazon Kendra vengono visualizzate in una finestra adiacente. La conversazione termina quando il cliente dice**“I have no more questions. Thank you.”**

Nota: elimina il tuo indice Amazon Kendra quando non lo usi.

Migrazione di un bot

L'API Amazon Lex V2 utilizza un'architettura delle informazioni aggiornata che consente il controllo semplificato delle versioni delle risorse e il supporto per più lingue in un bot. Per ulteriori informazioni, consulta la Guida alla migrazione nella Guida per gli sviluppatori di Amazon Lex V2.

Per utilizzare queste nuove funzionalità, devi migrare il tuo bot. Quando esegui la migrazione di un bot, Amazon Lex fornisce quanto segue:

- La migrazione copia gli intenti e i tipi di slot personalizzati sul bot Amazon Lex V2.
- Puoi aggiungere più lingue allo stesso bot Amazon Lex V2. In Amazon Lex V1 crei un bot separato per ogni lingua. Puoi migrare più bot Amazon Lex V1, ciascuno con un linguaggio diverso, verso un bot Amazon Lex V2.
- Amazon Lex associa i tipi e gli intenti di slot integrati di Amazon Lex V1 ai tipi e agli intenti di slot integrati di Amazon Lex V2. Se non è possibile migrare un sistema integrato, Amazon Lex restituisce un messaggio che indica cosa fare dopo.

Il processo di migrazione non esegue la migrazione di quanto segue:

- Alias
- Indici Amazon Kendra
- Funzioni AWS Lambda
- Impostazioni del registro di evento
- Canali di messaggistica come Slack
- Tag

Per migrare un bot, il tuo utente o ruolo deve disporre dell'autorizzazione IAM per le operazioni delle API Amazon Lex e Amazon Lex V2. Per le autorizzazioni necessarie, consultaConsenti a un utente di migrare un bot alle API di Amazon Lex V2.

Migrazione di un bot (console)

Usa la console Amazon Lex V1 per migrare la struttura di un bot verso un bot Amazon Lex V2.

Per utilizzare la console per migrare un bot all'API Amazon Lex V2

1. Accedere aAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dal menu di sinistra, seleziona Strumento di migrazione.
3. Dall'elenco dei bot, scegli il bot che desideri migrare, quindi scegli Migra.
4. Scegli la versione del bot che desideri migrare, quindi inserisci il nome del bot a cui effettuare la migrazione. Se inserisci il nome di un bot Amazon Lex V2 esistente, il bot Amazon Lex V1 viene migrato nella lingua indicata nei dettagli e sovrascrive la versione bozza della lingua.
5. Seleziona Successivo.
6. Scegli il ruolo IAM utilizzato da Amazon Lex per eseguire la versione API Amazon Lex V2 del bot. Puoi scegliere di creare un nuovo ruolo con le autorizzazioni minime richieste per eseguire il bot oppure puoi scegliere un ruolo IAM esistente.
7. Seleziona Successivo.
8. Esamina le impostazioni per la migrazione. Se sembrano OK, scegli Avvia migrazione.

Dopo aver avviato il processo di migrazione, si torna alla pagina iniziale dello strumento di migrazione. È possibile monitorare l'avanzamento della migrazione nella tabella di evento nella tabella di evento. Quando la colonna Stato della migrazione dice Completata, la migrazione è terminata.

Amazon Lex utilizza l'`StartImport`operazione nell'API Amazon Lex V2 per importare il bot migrato. Vedrai una voce nella tabella della cronologia delle importazioni della console Amazon Lex V2 per ogni migrazione.

Durante la migrazione, Amazon Lex potrebbe trovare risorse nel bot che non possono essere migrate. Viene visualizzato un messaggio di errore o di avviso per ogni risorsa che non può essere migrata. Ogni messaggio include un link alla documentazione che spiega come risolvere il problema.

Migrazione di una funzione Lambda

Amazon Lex V2 cambia il modo in cui le funzioni Lambda sono definite per un bot. Consente solo una funzione Lambda in un alias per ogni lingua in un bot. Per ulteriori informazioni sulla migrazione delle funzioni Lambda, consultaMigrazione di una funzione Lambda da Amazon Lex V1 ad Amazon Lex V2.

Messaggi di evento

Durante la migrazione, Amazon Lex potrebbe trovare risorse, ad esempio tipi di slot integrati, che non può migrare verso la risorsa equivalente di Amazon Lex V2. Quando ciò accade, Amazon Lex restituisce un messaggio di migrazione che descrive l'accaduto e fornisce un link alla documentazione che spiega come risolvere il problema di migrazione. Le sezioni seguenti descrivono i problemi che potrebbero sorgere durante la migrazione di un bot e come risolvere il problema.

Argomenti

Intento integrato

Quando usi un intento integrato non supportato in Amazon Lex V2, l'intento viene mappato a un intento personalizzato nel tuo bot Amazon Lex V2. L'intento personalizzato non contiene enunciati. Per continuare a usare l'intento, aggiungi esempi di enunciati.

Tipo di slot integrato

A qualsiasi slot migrato che utilizza un tipo di slot non supportato in Amazon Lex V2 non verrà assegnato un valore per il tipo di slot. Per usare questo slot:

- Crea un tipo di slot personalizzato
- Aggiungere i valori del tipo di slot previsti per il tipo di slot
- Aggiorna lo slot per utilizzare il nuovo tipo di slot personalizzato

Registri delle conversazioni

La migrazione non aggiorna le impostazioni del registro delle conversazioni del bot Amazon Lex V2.

Per configurare i registri delle conversazioni

1. Apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2.
2. Dall'elenco dei bot, scegli il bot di cui desideri configurare i registri delle conversazioni.
3. Scegliere Alias (Alias) dal menu di sinistra e quindi scegliere un alias dall'elenco.
4. Nella sezione Registri delle conversazioni, scegli Gestisci i registri delle conversazioni per configurare i registri delle conversazioni per l'alias del bot.

Gruppi di messaggi

Amazon Lex V2 supporta solo un messaggio e due messaggi alternativi per gruppo di messaggi. Se hai più di tre messaggi per gruppo di messaggi in un bot Amazon Lex V1, vengono migrati solo i primi tre messaggi. Per utilizzare più messaggi in un gruppo di messaggi, usa una funzione Lambda per generare vari messaggi.

Istruzioni e frasi

Amazon Lex V2 utilizza un meccanismo diverso per le istruzioni di follow-up, chiarimenti e riagganci.

Per le istruzioni di follow-up, utilizza il riporto del contesto per passare a un intento diverso dopo l'adempimento.

Supponiamo, ad esempio, di avere l'intenzione di prenotare un autonoleggio configurato per restituire un contesto di eventobook_car_fulfilled. Quando l'intento è soddisfatto, Amazon Lex imposta la variabile di contesto di output subook_car_fulfilled. Poichébook_car_fulfilled si tratta di un contesto attivo, un intento conbook_car_fulfilled come contesto di input viene considerato per il riconoscimento, a condizione che l'enunciato dell'utente sia riconosciuto come un tentativo di suscitare tale intento. Puoi utilizzarlo per scopi che hanno senso solo dopo aver prenotato un'auto, come inviare una ricevuta via e-mail o modificare una prenotazione.

Amazon Lex V2 non supporta richieste di chiarimento e frasi di sospensione (dichiarazioni di interruzione). I bot Amazon Lex V2 contengono un intento di fallback predefinito che viene invocato se non corrisponde alcun intento. Per inviare una richiesta di chiarimenti con nuovi tentativi, configura una funzione Lambda e abilita l'hook del codice di dialogo nell'intento di fallback. La funzione Lambda può generare una richiesta di chiarimento come risposta e il valore di tentativo in un attributo di sessione. Se il valore dei tentativi supera il numero massimo di tentativi, puoi emettere una frase di riaggancio e chiudere la conversazione.

Altre funzionalità di Amazon Lex V1

Lo strumento di migrazione supporta solo la migrazione dei bot Amazon Lex V1 e dei relativi intenti, tipi di slot e slot. Per altre funzionalità, consulta gli argomenti seguenti nella documentazione di Amazon Lex V2.

- Alias dei bot: Alias
- Canali bot: implementazione di un bot Amazon Lex V2 su una piattaforma di messaggistica
- Impostazioni del registro delle conversazioni: monitoraggio con registri delle conversazioni
- Indici Amazon Kendra: AMAZON. KendraSearchIntent
- Funzioni Lambda: utilizzo di unaAWS Lambda funzione
- Tag: risorse di etichettatura

Migrazione di una funzione Lambda da Amazon Lex V1 ad Amazon Lex V2

Amazon Lex V2 consente una sola funzione Lambda per ogni lingua in un bot. La funzione Lambda e le relative impostazioni sono configurate per l'alias bot utilizzato in fase di esecuzione.

La funzione Lambda viene richiamata a tutti gli effetti in quella lingua se i dialoghi e gli hook del codice di adempimento sono abilitati a tal fine.

Le funzioni Amazon Lex V2 Lambda hanno un formato di messaggio di input e output diverso da quello di Amazon Lex V1. Queste sono le differenze nel formato di input della funzione Lambda.

- Amazon Lex V2 sostituisce le`alternativeIntents` strutture`currentIntent` e con la`interpretations` struttura. Ogni interpretazione contiene un intento, il punteggio di confidenza NLU per l'intento e un'analisi opzionale del sentiment.
- Amazon Lex V2 sposta il`activeContexts`,`sessionAttributes` in Amazon Lex V1 alla`sessionState` struttura unificata. Questa struttura fornisce informazioni sullo stato corrente della conversazione, incluso l'ID della richiesta di origine.
- Amazon Lex V2 non restituisce il`recentIntentSummaryView`. Utilizza invece le informazioni nella`sessionState` struttura.
- L'ingresso Amazon Lex V2 fornisce l'`botId`e`localeId` nell'`bot`attributo.

- La struttura di input contiene un`inputMode` attributo che fornisce informazioni sul tipo di input: testo, voce o DTMF.

Queste sono le differenze nel formato di output della funzione Lambda:

- `sessionAttributes`Le strutture`activeContexts` e in Amazon Lex V1 vengono sostituite dalla`sessionState` struttura in Amazon Lex V2.
- `recentIntentSummaryView`Non è incluso nell'output.
- La`dialogAction` struttura di Amazon Lex V1 è suddivisa in due strutture,`dialogAction` che fanno parte della`sessionState` struttura e`messages` che sono necessarie quando lo`dialogAction.type` è`ElicitIntent`. Amazon Lex sceglie i messaggi da questa struttura da mostrare all'utente.

Quando crei un bot con le API Amazon Lex V2, esiste solo una funzione Lambda per alias bot per lingua anziché una funzione Lambda per ogni intento. Se si desidera continuare a utilizzare funzioni separate, è possibile creare una funzione router che attiva una funzione separata per ogni intento. Di seguito è riportata una funzione del router che è possibile utilizzare o modificare per l'applicazione.

```
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
```

Elenco dei campi aggiornati

Le tabelle seguenti forniscono informazioni dettagliate sui campi aggiornati nella richiesta e nella risposta di Amazon Lex V2 Lambda. È possibile utilizzare queste tabelle per mappare i campi tra le versioni.

Richiesta

I seguenti campi sono stati aggiornati nel formato di richiesta della funzione Lambda.

Contesti attivi

La`activeContexts` struttura ora fa parte della`sessionState` struttura.

Struttura del sistema V1	Struttura del sistema V2
Contesti attivi	SessionState.ActiveContexts
Contesti attivi [*]. timeToLive	SessionState.ActiveContexts [*]. timeToLive
Contesti attivi [*]. timeToLive. timeToLiv eInSeconds	SessionState.ActiveContexts [*]. timeToLive. timeToLiveInSeconds
Contesti attivi [*]. timeToLive. turnsToLive	SessionState.ActiveContexts [*]. timeToLive. turnsToLive
ActiveContexts [*] .name	SessionState.ActiveContexts [*] .name
ActiveContexts [*] .parametri	SessionState.ActiveContexts [*] .contextA ttributes

Intenti alternativi

L'elenco delle interpretazioni dall'indice 1 a N contiene l'elenco degli intenti alternativi previsti da Amazon Lex V2, insieme ai relativi punteggi di confidenza. `recentIntentSummaryView`Viene

rimosso dalla struttura delle richieste in Amazon Lex V2. Per visualizzare i dettagli di `recentIntentSummaryView`, utilizzare l'GetSessionoperazione.

Struttura del sistema V1	Struttura del sistema V2
Intenti alternativi	interpretazioni [1: *]
recentIntentSummaryVisualizza	N/D

Bot

In Amazon Lex V2, i bot e gli alias dispongono di identificatori. L'ID bot fa parte dell'input del codehook. L'ID alias è incluso, ma non il nome dell'alias. Amazon Lex V2 supporta più impostazioni locali per lo stesso bot, quindi l'ID locale è incluso.

Struttura del sistema V1	Struttura del sistema V2
bot	bot
nome del bot	nome del bot
N/D	bot.id
bot. alias	N/D
N/D	BOT. alias ID
versione bot	versione bot
N/D	ID locale bot.

Intento attuale

La `sessionState.intent` struttura contiene i dettagli dell'intento attivo. Amazon Lex V2 restituisce anche un elenco di tutti gli intenti, inclusi gli intenti alternativi, nella `interpretations` struttura. Il primo elemento nell'elenco delle interpretazioni è sempre lo stesso di `sessionState.intent`.

Struttura del sistema V1	Struttura del sistema V2
currentIntent	SessionState.intent OR interpretazioni [0] .intent
Nome dell'intento corrente	SessionState.intent.name OR interpretazioni [0] .intent.name
currentIntent. nluConfidenceScore	interpretazioni [0] .nluConfidence.score

Azione di dialogo

Il`confirmationStatus` campo fa ora parte della`sessionState` struttura.

Struttura del sistema V1	Struttura del sistema V2
Intenzione corrente. Stato di conferma	SessionState.Intent.confirmationState OR interpretazioni [0] .intent.confirmationState
N/D	SessionState.intent.state OR interpretazioni [*] .intent.state

Amazon Kendra

Il`kendraResponse` campo fa ora parte delle`interpretations` strutture`sessionState` del terreno.

Struttura del sistema V1	Struttura del sistema V2
kendraResponse	SessionState.Intent.KendraResponse OR interpretazioni [0] .intent.KendraResponse

Sentimento

La`sentimentResponse` struttura viene spostata nella nuova`interpretations` struttura.

Struttura del sistema V1	Struttura del sistema V2
sentimentResponse	interpretazioni [0] .sentimentResponse
SentimentResponse.SentimentLabel	interpretazioni [0] .sentimentResponse .sentiment
Risposta sentimentale. SentimentScore	interpretazioni [0] .sentimentResponse .sentimentScore

Slot

Amazon Lex V2 fornisce un singolo`slots` oggetto all'interno della`sessionState.intent` struttura che contiene i valori risolti, il valore interpretato e il valore originale di ciò che l'utente ha detto. Amazon Lex V2 supporta anche slot multivalore impostando l'`slotShape`as`List` e impostando l'`value`elenco. Gli slot a valore singolo sono supportati dal`value` campo, si presume che la loro forma sia`Scalar`.

Struttura del sistema V1	Struttura del sistema V2
Slot per intenzioni correnti	SessionState.intent.slots OR interpretazioni [0] .intent.slots
CurrentIntent.slots [*] .valore	SessionState.Intent.slots [*] .value.Interpreted Value OR interpretazioni [0] .intent.slots [*] .value.InterpretedValue
N/D	SessionState.intent.slots [*] .value.shape OR interpretazioni [0] .intent.slots [*] .shape
N/D	SessionState.intent.slots [*] .valori OR interpret azioni [0] .intent.slots [*] .values
Dettagli dello slot CurrentIntent.Slot	SessionState.intent.slots OR interpretazioni [0] .intent.slots

Struttura del sistema V1	Struttura del sistema V2
CurrentIntent.SlotDetails [*] .risoluzioni	SessionState.Intent.slots [*] .Valori risolti O interpretazioni [0] .intent.slots [*] .Resolved Values
CurrentIntent.SlotDetails [*] .Valore originale	SessionState.Intent.slots [*] .originalValue OR interpretazioni [0] .intent.slots [*] .originalValue

Altri

Il`sessionId` campo Amazon Lex V2 è lo stesso`userId` di Amazon Lex V1. Amazon Lex V2 invia anche il`inputMode` codice del chiamante: testo, DTMF o voce.

Struttura del sistema V1	Struttura del sistema V2
userId	sessionId
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion
sessionAttributes	SessionState.SessionAttributi
requestAttributes	requestAttributes
N/D	Modalità di ingresso
N/D	originatingRequestId

Risposta

I seguenti campi sono stati modificati nel formato del messaggio di risposta della funzione Lambda.

Contesti attivi

La `activeContexts` struttura si è spostata nella `sessionState` struttura.

Struttura del sistema V1	Struttura del sistema V2
Contesti attivi	SessionState.ActiveContexts
Contesti attivi [*]. timeToLive	SessionState.ActiveContexts [*]. timeToLive
Contesti attivi [*]. timeToLive. timeToLiveInSeconds	SessionState.ActiveContexts [*]. timeToLive. timeToLiveInSeconds
Contesti attivi [*]. timeToLive. turnsToLive	SessionState.ActiveContexts [*]. timeToLive. turnsToLive
ActiveContexts [*] .name	SessionState.ActiveContexts [*] .name
ActiveContexts [*] .parametri	SessionState.ActiveContexts [*] .contextAttributes

Azione di dialogo

La `dialogAction` struttura si è spostata nella `sessionState` struttura. È ora possibile specificare più messaggi in un'azione di dialogo e la `genericAttachments` struttura è ora la `imageResponseCard` struttura.

Struttura del sistema V1	Struttura del sistema V2
dialogAction	SessionState.DialogAction
Azione di dialogo. Tipo	SessionState.DialogAction.Type
dialogAction. slotToElicit	SessionState.Intent.DialogAction. slotToElicit
DialogAction.Type.FulfillmentState	SessionState.Intent.State
Azione di dialogo. Messaggio	messaggi
DialogAction.Message.ContentType	messaggi [*] .contentType

Struttura del sistema V1	Struttura del sistema V2
DialogAction.Message.Content	messaggi [*] .content
DialogAction.ResponseCard	messaggi [*]. imageResponseCard
DialogAction.ResponseCard.Version	N/D
DialogAction.ResponseCard.ContentType	messaggi [*] .contentType
DialogAction.ResponseCard.Allegati generici	N/D
DialogAction.ResponseCard.Allegati generici [*] .titolo	messaggi [*]. imageResponseCard.titolo
DialogAction.ResponseCard.Allegati generici [*] .sottotitolo	messaggi [*]. imageResponseCard.sottotitolo
DialogAction.ResponseCard.Allegati generici [*] .imageURL	messaggi [*]. imageResponseCard.URL imageUrl
DialogAction.ResponseCard.Allegati generici [*] .pulsanti	messaggi [*]. imageResponseCard.pulsanti
DialogAction.ResponseCard.GenericAtt achments [*] .value	messaggi [*]. imageResponseCard.pulsanti [*] .valore
DialogAction.ResponseCard.Allegati generici [*] .pulsanti [*] .testo	messaggi [*]. imageResponseCard.pulsanti [*] .testo
dialogAction. kendraQueryRequestCarico utile	dialogAction. kendraQueryRequestCarico utile
dialogAction. kendraQueryFilterCorda	dialogAction. kendraQueryFilterCorda

Intenti e slot di evento

I campi di intento e di slot che facevano parte della`dialogAction` struttura ora fanno parte della`sessionState` struttura.

Struttura del sistema V1	Struttura del sistema V2
Azione di dialogo. IntentName	SessionState.Intent.Name
Azione di dialogo. Slot	SessionState.Intent.Slots
DialogAction.Slots [*] .tasto	SessionState.Intent.Slots [*] .key
DialogAction.slots [*] .valore	SessionState.Intent.Slots [*] .value.Valore interpretato
N/D	SessionState.Intent.Slots [*] .value.shape
N/D	SessionState.Intent.slots [*] .valori

Altri

La`sessionAttributes` struttura ora fa parte della`sessionState` struttura. La`recentIntentSummaryReview` struttura è stata eliminata.

Struttura del sistema V1	Struttura del sistema V2
sessionAttributes	SessionState.SessionAttributi
recentIntentSummaryVisualizza	N/D

Sicurezza in Amazon Lex

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. L'efficacia della nostra sicurezza è regolarmente testata e verificata da revisori di terze parti come parte dei programmi di conformità AWS. Per ulteriori informazioni sui programmi di conformità applicabili ad Amazon Lex, consulta AWS Services in Scope by Compliance Program.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e leggi e normative applicabili.

Questa documentazione ti aiuterà a capire come applicare il modello di responsabilità condivisa quando usi Amazon Lex. I seguenti argomenti mostrano come configurare Amazon Lex per soddisfare i tuoi obiettivi di sicurezza e conformità. Imparerai anche come utilizzare altri servizi AWS che possono aiutarti a monitorare e proteggere le tue risorse Amazon Lex.

Argomenti

Protezione dei dati in Amazon Lex

Amazon Lex raccoglie i contenuti dei clienti per la risoluzione dei problemi e per contribuire a migliorare il servizio. I contenuti dei clienti sono protetti per impostazione predefinita. Puoi eliminare contenuti per singoli clienti utilizzando l'API Amazon Lex.

Amazon Lex archivia quattro tipi di contenuti:

- Esempi di espressioni che sono usate per costruire e addestrare un bot
- Espressioni dei clienti dagli utenti che interagiscono con il bot
- Gli attributi di sessione che forniscono informazioni specifiche dell'applicazione per tutta la durata dell'interazione di un utente con un bot
- Gli attributi di richiesta che contengono informazioni che si applicano a una singola richiesta a un bot

Qualsiasi bot Amazon Lex progettato per essere utilizzato dai bambini è regolato dal Children's Online Privacy Protection Act (COPPA). Comunichi ad Amazon Lex che il bot è soggetto al COPPA utilizzando la console o l'API Amazon Lex su cui impostare il `childDirected` campo. `true` Quando il campo `childDirected` è impostato su `true`, non vengono archiviati espressioni degli utenti.

Argomenti

Crittografia dei dati inattivi

Amazon Lex crittografa gli enunciati degli utenti che memorizza.

Argomenti

Espressioni di esempio

Quando sviluppi un bot, puoi fornire le espressioni di esempio per ogni intento e slot. È anche possibile fornire valori personalizzati e sinonimi per le slot. Queste informazioni sono crittografate quando sono inattive e vengono utilizzate per creare il bot e creare l'esperienza utente.

Espressioni del cliente

Amazon Lex crittografa gli enunciati che gli utenti inviano al bot, a meno che il `childDirected` campo non sia impostato su. `true`

Quando il campo `childDirected` è impostato su `true`, non vengono archiviati espressioni degli utenti.

Quando il campo `childDirected` è impostato su `false` (predefinito), le espressioni degli utenti sono crittografate e memorizzate per 15 giorni per l'uso con l'operazione GetUtterancesView. Per eliminare le espressioni archiviate per un determinato utente, utilizza l'operazione DeleteUtterances.

Quando il bot accetta l'input vocale, l'input viene memorizzato a tempo indeterminato. Amazon Lex lo utilizza per migliorare la capacità del bot di rispondere agli input degli utenti.

Utilizza l'operazione DeleteUtterances per eliminare le espressioni archiviate per un determinato utente.

Attributi di sessione

Gli attributi di sessione contengono informazioni specifiche dell'applicazione che vengono trasferite tra Amazon Lex e le applicazioni client. Amazon Lex trasmette gli attributi di sessione a tutte AWS Lambda le funzioni configurate per un bot. Se una funzione Lambda aggiunge o aggiorna gli attributi della sessione, Amazon Lex restituisce le nuove informazioni all'applicazione client.

Gli attributi di sessione persistono in un archivio crittografato per tutta la durata della sessione. È possibile configurare la sessione in modo che rimanga attiva per un minimo di 1 minuto e fino a 24 ore dopo l'ultima espressione dell'utente. La durata predefinita della sessione è di 5 minuti.

Attributi di richiesta

Gli attributi di richiesta contengono informazioni specifiche sulla richiesta e si applicano solo alla richiesta corrente. Un'applicazione client utilizza gli attributi di richiesta per inviare informazioni ad Amazon Lex in fase di esecuzione.

Utilizza gli attributi di richiesta per inviare informazioni che non devono essere conservate per l'intera sessione. Poiché gli attributi di richiesta non persistono tra le richieste, non vengono archiviati.

Crittografia in transito

Amazon Lex utilizza il protocollo HTTPS per comunicare con l'applicazione client. Utilizza le firme HTTPS e AWS per comunicare con altri servizi, come Amazon Polly, AWS Lambda e per conto della tua applicazione.

Gestione delle chiavi

Amazon Lex protegge i tuoi contenuti dall'uso non autorizzato con chiavi interne.

Identity and Access Management per Amazon Lex

AWS Identity and Access Management (IAM) è uno strumento Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Amazon Lex. IAM è uno Servizio AWS strumento che puoi utilizzare senza costi aggiuntivi.

Argomenti

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Amazon Lex.

Utente del servizio: se utilizzi il servizio Amazon Lex per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità di Amazon Lex per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di Amazon Lex, consultaRisoluzione dei problemi relativi all'identità e all'accesso ad Amazon Lex.

Amministratore del servizio: se sei responsabile delle risorse Amazon Lex presso la tua azienda, probabilmente hai pieno accesso ad Amazon Lex. È tuo compito determinare a quali funzionalità e risorse di Amazon Lex devono accedere gli utenti del servizio. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con Amazon Lex, consultaCome funziona Amazon Lex con IAM.

Amministratore IAM: se sei un amministratore IAM, potresti voler saperne di più su come scrivere policy per gestire l'accesso ad Amazon Lex. Per visualizzare esempi di policy basate sull'identità di Amazon Lex che puoi utilizzare in IAM, consulta. Esempi di policy basate sull'identità per Amazon Lex

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se

non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWSnella Guida per l'utente IAM.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, di un provider di identità Web AWS Directory Service, della directory Identity Center o di qualsiasi utente che accede utilizzando le Servizi AWS credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per ulteriori informazioni su IAM Identity Center, consulta Cos'è IAM Identity Center? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato IAMAdmins e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in AWS Management Console cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in

IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .

- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per conoscere la differenza tra ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta Cross Account Resource Access in IAM nella IAM User Guide.
- Accesso tra servizi: alcuni Servizi AWS utilizzano funzionalità in altri. Servizi AWS Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso diretto (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire azioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWSnella Guida per l'utente IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 e che AWS CLI effettuano richieste API. AWS Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'istanza EC2 e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente IAM.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli

possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano gli ACL. AWS WAF Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzionalità avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità.

Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal`sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente IAM.

- Politiche di controllo dei servizi (SCP): le SCP sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, inclusa ciascuna. Utente root dell'account AWS Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon Lex con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon Lex, scopri quali funzionalità IAM sono disponibili per l'uso con Amazon Lex.

Funzionalità IAM che puoi utilizzare con Amazon Lex

Funzionalità IAM	Supporto Amazon Lex
Policy basate su identità	Sì

Funzionalità IAM	Supporto Amazon Lex
Policy basate su risorse	No
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	Sì

Per avere una visione di alto livello di come Amazon Lex e altri AWS servizi funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Policy basate sull'identità per Amazon Lex

Supporta le policy basate su identità	Sì

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte.

Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Esempi di policy basate sull'identità per Amazon Lex

Per visualizzare esempi di policy basate sull'identità di Amazon Lex, consulta. Esempi di policy basate sull'identità per Amazon Lex

Politiche basate sulle risorse all'interno di Amazon Lex

Supporta le policy basate su risorse	No

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando il principale e la risorsa sono diversi Account AWS, un amministratore IAM dell'account affidabile deve inoltre concedere all'entità principale (utente o ruolo) l'autorizzazione ad accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Cross Account Resource Access in IAM nella IAM User Guide.

Azioni politiche per Amazon Lex

Supporta le operazioni di policy	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action`di una policy JSON descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le azioni politiche in genere hanno lo stesso nome dell'operazione AWS API associata. Ci sono alcune eccezioni, ad esempio le azioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di azioni Amazon Lex, consulta Azioni definite da Amazon Lex nel Service Authorization Reference.

Le azioni politiche in Amazon Lex utilizzano il seguente prefisso prima dell'azione:

```
lex
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "lex:Describe*"
```

Risorse relative alle policy per Amazon Lex

Supporta le risorse di policy	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource`della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

L'ARN di una risorsa bot Amazon Lex ha il seguente formato.

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

Per ulteriori informazioni sul formato degli ARN, consulta Amazon Resource Names (ARNs) e AWS Service Namespaces.

Ad esempio, per specificare il bot `OrderFlowers` nell'istruzione, utilizza il seguente ARN.

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:OrderFlowers"
```

Per specificare tutti i bot che appartengono ad un account specifico, utilizza il carattere jolly (*).

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:*"
```

Alcune azioni di Amazon Lex, come quelle per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare il carattere jolly (*).

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Amazon Lex e dei relativi ARN, consulta Risorse definite da Amazon Lex nel Service Authorization Reference. Per sapere con quali azioni è possibile specificare l'ARN di ogni risorsa, consulta Azioni definite da Amazon Lex.

Chiavi relative alle condizioni delle politiche per Amazon Lex

Supporta le chiavi di condizione delle policy specifiche del servizio	Sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition`(o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition`è facoltativo. Puoi compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition`in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND`logica. Se si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica. `OR` Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Per visualizzare un elenco di chiavi di condizione di Amazon Lex, consulta Chiavi di condizione per Amazon Lex nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Azioni definite da Amazon Lex.

La tabella seguente elenca le chiavi delle condizioni di Amazon Lex che si applicano alle risorse Amazon Lex. Puoi includere queste chiavi negli `Condition` elementi di una policy di autorizzazioni IAM.

ACL in Amazon Lex

Supporta le ACL	No

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Amazon Lex

Supporta ABAC (tag nelle policy)	Parziale

Il controllo dell'accesso basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, questi attributi sono chiamati tag. Puoi allegare tag a entità IAM (utenti o ruoli) e a molte AWS risorse. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni su ABAC, consulta Che cos'è ABAC? nella Guida per l'utente IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Puoi associare tag a determinati tipi di risorse Amazon Lex per l'autorizzazione. Per controllare l'accesso basato su tag, fornisci le informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `lex:ResourceTag/${TagKey}`, `aws:RequestTag/${TagKey}` o `aws:TagKeys`.

Per ulteriori informazioni sull'etichettatura delle risorse Amazon Lex, consultaAssegnazione di tag alle risorse Amazon Lex.

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Usa un tag per accedere a una risorsa.

Nella tabella seguente sono elencate le azioni e i tipi di risorse corrispondenti per il controllo dell'accesso basato su tag. Ogni azione è autorizzata in base ai tag associati al tipo di risorsa corrispondente.

Azione	Tipo di risorsa	Chiavi di condizione	Note
CreateBotVersion	bot	`lex:ResourceTag`	
DeleteBot	bot	`lex:ResourceTag`	
DeleteBotAlias	alias	`lex:ResourceTag`	
DeleteBotChannelAssociation	canale	`lex:ResourceTag`	
DeleteBotVersion	bot	`lex:ResourceTag`	
DeleteSession	bot o alias	`lex:ResourceTag`	Utilizza i tag associati al bot quando l'alias è impostato su `$LATEST`. Utilizza i tag associati all'alias specificato quando usati con altri alias.
DeleteUtterances	bot	`lex:ResourceTag`	
GetBot	bot o alias	`lex:ResourceTag`	Utilizza i tag associati al bot quando `versionOrAlias` è impostato su `$LATEST` o su una versione numerica. Utilizza i tag associati

Azione	Tipo di risorsa	Chiavi di condizione	Note
			all'alias specificato quando usati con alias
GetBotAlias	alias	`lex:ResourceTag`	
GetBotChannelAssociation	canale	`lex:ResourceTag`	
GetBotChannelAssociations	canale	`lex:ResourceTag`	Utilizza i tag associati al bot quando l'alias è impostato su "-". Utilizza i tag associati all'alias specifica to quando viene specificato un alias bot
GetBotVersions	bot	`lex:ResourceTag`	
GetExport	bot	`lex:ResourceTag`	
GetSession	bot o alias	`lex:ResourceTag`	Utilizza i tag associati al bot quando l'alias è impostato su `$LATEST`. Utilizza i tag associati all'alias specificato quando usati con altri alias.
GetUtterancesView	bot	`lex:ResourceTag`	
ListTagsForResource	bot, alias o canale	`lex:ResourceTag`	

Azione	Tipo di risorsa	Chiavi di condizione	Note
PostContent	bot o alias	`lex:ResourceTag`	Utilizza i tag associati al bot quando l'alias è impostato su `$LATEST`. Utilizza i tag associati all'alias specificato quando usati con altri alias.
PostText	bot o alias	`lex:ResourceTag`	Utilizza i tag associati al bot quando l'alias è impostato su `$LATEST`. Utilizza i tag associati all'alias specificato quando usati con altri alias.
PutBot	bot	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutBotAlias	alias	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutSession	bot o alias	`lex:ResourceTag`	Utilizza i tag associati al bot quando l'alias è impostato su `$LATEST`. Utilizza i tag associati all'alias specificato quando usati con altri alias.

Azione	Tipo di risorsa	Chiavi di condizione	Note
StartImport	bot	`lex:ResourceTag`	Si basa sulla policy di accesso per l'operazione `PutBot`. I tag e le autorizzazioni specifiche per l'operazione `StartImport` vengono ignorati.
TagResource	bot, alias o canale	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
UntagResource	bot, alias o canale	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	

Utilizzo di credenziali temporanee con Amazon Lex

Supporta le credenziali temporanee	Sì

Alcune Servizi AWS non funzionano quando accedi utilizzando credenziali temporanee. Per ulteriori informazioni, incluse quelle che Servizi AWS funzionano con credenziali temporanee, consulta la sezione relativa alla Servizi AWS compatibilità con IAM nella IAM User Guide.

Stai utilizzando credenziali temporanee se accedi AWS Management Console utilizzando qualsiasi metodo tranne nome utente e password. Ad esempio, quando accedete AWS utilizzando il link Single Sign-On (SSO) della vostra azienda, tale processo crea automaticamente credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta Cambio di un ruolo (console) nella Guida per l'utente IAM.

È possibile creare manualmente credenziali temporanee utilizzando l'API or. AWS CLI AWS È quindi possibile utilizzare tali credenziali temporanee per accedere. AWS AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza provvisorie in IAM.

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. È possibile ottenere credenziali di sicurezza temporanee chiamando operazioni AWS STS API come AssumeRoleo Token. GetFederation

Autorizzazioni principali multiservizio per Amazon Lex

Supporta l'inoltro delle sessioni di accesso (FAS)	Sì

Quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

Ruoli di servizio per Amazon Lex

Supporta i ruoli di servizio	Sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWSnella Guida per l'utente IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità di Amazon Lex. Modifica i ruoli di servizio solo quando Amazon Lex fornisce indicazioni in tal senso.

Scelta di un ruolo IAM in Amazon Lex

Amazon Lex utilizza ruoli collegati ai servizi per chiamare Amazon Comprehend e Amazon Polly. Utilizza le autorizzazioni a livello di risorsa sulle tue funzioni per richiamarle. AWS Lambda

È necessario fornire un ruolo IAM per abilitare il tagging delle conversazioni. Per ulteriori informazioni, consulta Creazione di un ruolo IAM e delle policy per i log delle conversazioni.

Ruoli collegati ai servizi per Amazon Lex

Supporta i ruoli collegati ai servizi	Sì

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

Per dettagli sulla creazione o la gestione di ruoli collegati ai servizi Amazon Lex, consulta. Utilizzo di ruoli collegati ai servizi per Amazon Lex

Esempi di policy basate sull'identità per Amazon Lex

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Amazon Lex. Inoltre, non possono eseguire attività utilizzando AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS API. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon Lex, incluso il formato degli ARN per ciascun tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon Lex nel Service Authorization Reference.

Argomenti

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon Lex nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le politiche AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWSo Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente IAM.
- Applica le autorizzazioni con privilegio minimo: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente IAM.

- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Amazon Lex

Per accedere alla console Amazon Lex, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Amazon Lex presenti nel tuo Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime di console per gli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

AWS affronta molti casi d'uso comuni fornendo policy IAM autonome create e amministrate da. AWS Queste policy sono denominate policy gestite da AWS. Le policy gestite da AWS rendono più semplice l'assegnazione di autorizzazioni appropriate a utenti, gruppi e ruoli rispetto a come avverrebbe se le policy si dovessero scrivere. Per ulteriori informazioni, consulta Policy gestite AWS nella Guida per gli utenti di IAM.

Le seguenti politiche AWS gestite, che puoi associare a gruppi e ruoli nel tuo account, sono specifiche di Amazon Lex:

- AmazonLexReadOnly— Garantisce l'accesso in sola lettura alle risorse Amazon Lex.
- AmazonLexRunBotsSolo: consente l'accesso all'esecuzione di bot conversazionali Amazon Lex.

- AmazonLexFullAccess— Garantisce l'accesso completo per creare, leggere, aggiornare, eliminare ed eseguire tutte le risorse Amazon Lex. Garantisce inoltre la possibilità di associare funzioni Lambda il cui nome inizia `AmazonLex` con gli intenti di Amazon Lex.

> **Note**
>
> Puoi rivedere queste politiche di autorizzazione accedendo alla console IAM e cercando politiche specifiche.

La AmazonLexFullAccesspolicy non concede all'utente l'autorizzazione a utilizzare l'`KendraSearchIntent`intento di interrogare un indice Amazon Kendra. Per interrogare un indice, devi aggiungere ulteriori autorizzazioni alla policy. Per le autorizzazioni richieste, consulta Politica IAM per Amazon Kendra Search.

Puoi anche creare policy IAM personalizzate per consentire le autorizzazioni per le azioni dell'API Amazon Lex. Puoi collegare queste politiche personalizzate ai ruoli o ai gruppi IAM che richiedono tali autorizzazioni.

Per informazioni dettagliate sulle policy gestite da AWS per Amazon Lex, consultaAWSpolitiche gestite per Amazon Lex.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono cpllegate alla relativa identità utente. Questa policy include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o in modo programmatico. AWS CLI 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": "*"
        }
    ]
}
```

Elimina tutti i bot Amazon Lex

Questa policy di esempio concede a un utente del tuo account AWS l'autorizzazione a eliminare qualsiasi bot nel tuo account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Consenti a un utente di migrare un bot alle API di Amazon Lex V2

La seguente politica di autorizzazione IAM consente a un utente di iniziare a migrare un bot da Amazon Lex alle API Amazon Lex V2 e di visualizzare l'elenco delle migrazioni e il loro progresso.

```
{
    "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": "*"
        }
    ]
}
```

Usa un tag per accedere a una risorsa

Questa politica di esempio concede a un utente o a un ruolo nell' AWS account l'autorizzazione a utilizzare l'`PostText`operazione con qualsiasi risorsa contrassegnata con la chiave **`Department`** e il valore**`Support`**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Action": "lex:PostText",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "lex:ResourceTag/Department": "Support"
                }
            }
        }
    ]
}
```

AWSpolitiche gestite per Amazon Lex

Una policy gestita da AWS è una policy autonoma creata e amministrata da AWS. Le policy gestite da AWS sono progettate per fornire autorizzazioni per molti casi d'uso comuni in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Ricorda che le policy gestite da AWS potrebbero non concedere autorizzazioni con privilegi minimi per i tuoi casi d'uso specifici perché possono essere utilizzate da tutti i clienti AWS. Consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle policy gestite da AWS. Se AWS aggiorna le autorizzazioni definite in una policy gestita da AWS, l'aggiornamento riguarda tutte le identità principali (utenti, gruppi e ruoli) a cui è collegata la policy. È molto probabile che AWS aggiorni una policy gestita da AWS quando viene lanciato un nuovo Servizio AWS o nuove operazioni API diventano disponibili per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

Policy gestita da AWS:AmazonLexReadOnly

È possibile allegare la policy `AmazonLexReadOnly` alle identità IAM.

Questa politica concede autorizzazioni di sola lettura che consentono agli utenti di visualizzare tutte le azioni nel servizio di creazione di modelli Amazon Lex e Amazon Lex V2.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `lex`— Accesso in sola lettura alle risorse Amazon Lex e Amazon Lex V2 nel servizio di creazione di modelli.

```
{
    "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": "*"
        }
    ]
}
```

Policy gestita da AWS:AmazonLexRunBotsOnly

È possibile allegare la policy `AmazonLexRunBotsOnly` alle identità IAM.

Questa politica concede autorizzazioni di sola lettura che consentono l'accesso all'esecuzione di bot conversazionali Amazon Lex e Amazon Lex V2.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `lex`— Accesso in sola lettura a tutte le azioni nel runtime di Amazon Lex e Amazon Lex V2.

```
{
    "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": "*"
        }
    ]
}
```

Policy gestita da AWS:AmazonLexFullAccess

È possibile allegare la policy `AmazonLexFullAccess` alle identità IAM.

Questa politica concede autorizzazioni amministrative che consentono all'utente di creare, leggere, aggiornare ed eliminare le risorse di Amazon Lex e Amazon Lex V2 e di eseguire bot conversazionali Amazon Lex e Amazon Lex V2.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `lex`— Consente ai principali di accedere in lettura e scrittura a tutte le azioni nei servizi di creazione e esecuzione di modelli Amazon Lex e Amazon Lex V2.
- `cloudwatch`— Consente ai responsabili di visualizzare AmazonCloudWatchmetriche e allarmi.
- `iam`— Consente ai responsabili di creare ed eliminare ruoli collegati ai servizi, assegnare ruoli e allegare e scollegare le policy a un ruolo. Le autorizzazioni sono limitate a «lex.amazonaws.com» per le operazioni di Amazon Lex e a «lexv2.amazonaws.com» per le operazioni di Amazon Lex V2.
- `kendra`— Consente ai titolari di elencare gli indici Amazon Kendra.
- `kms`— Consente ai presidi di descrivereAWS KMSchiavi e alias.
- `lambda`— Consente ai presidi di elencareAWS Lambdafunzioni e gestione delle autorizzazioni associate a qualsiasi funzione Lambda.
- `polly`— Consente ai presidi di descrivere le voci di Amazon Polly e sintetizzare il parlato.

```
{
    "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 si aggiorna aAWSpolitiche gestite

Visualizza i dettagli sugli aggiornamenti diAWSpolitiche gestite per Amazon Lex da quando questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS su Amazon LexCronologia dei documenti per Amazon Lexpagina.

Modifica	Descrizione	Data
AmazonLexFullAccess: aggiornamento a una policy esistente	Amazon Lex ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura alle operazioni del servizio di creazione di modelli Amazon Lex V2.	18 agosto 2021
AmazonLexReadOnly: aggiornamento a una policy esistente	Amazon Lex ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura alle operazioni del servizio di creazione di modelli Amazon Lex V2.	18 agosto 2021
AmazonLexRunBotsOnly: aggiornamento a una policy esistente	Amazon Lex ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura alle operazioni del servizio di runtime di Amazon Lex V2.	18 agosto 2021

Modifica	Descrizione	Data
Amazon Lex ha iniziato a monitorare le modifiche	Amazon Lex ha iniziato a monitorare le modifiche relative alAWSpolitiche gestite.	18 agosto 2021

Utilizzo di ruoli collegati ai servizi per Amazon Lex

Amazon Lex utilizza ruoli collegati ai serviziAWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon Lex. I ruoli collegati ai servizi sono definiti automaticamente da Amazon Lex e includono tutte le autorizzazioni richieste dal servizio per tuo conto.AWS

Un ruolo collegato ai servizi semplifica la configurazione di Amazon Lex perché permette di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon Lex definisce le autorizzazioni dei ruoli collegati ai servizi e, salvo diversamente definito, solo Amazon Lex può assumere il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere collegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon Lex perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Autorizzazioni del ruolo collegato ai servizi per Amazon Lex

Amazon Lex utilizza due ruoli collegati ai servizi:

- AWSServiceRoleForLexBots— Amazon Lex utilizza questo ruolo collegato ai servizi per richiamare Amazon Polly per sintetizzare le risposte vocali del bot, chiamare Amazon Comprehend per l'analisi dei sentimenti e, facoltativamente, Amazon Kendra per la ricerca negli indici.
- AWSServiceRoleForLexChannels— Amazon Lex utilizza questo ruolo collegato al servizio per inviare testo al bot durante la gestione dei canali.

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon Lex

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un bot, un canale di bot oppure quando crei un intento collegato ai servizi di Amazon Kendra crea il ruolo collegato ai servizi per tuo conto.AWS Management Console

Se elimini questo ruolo collegato ai servizi, puoi ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando crei un nuovo bot, un'associazione di canali oppure quando crei un Amazon Kendra collegato ai servizi, Amazon Lex crea di nuovo il ruolo collegato ai servizi per tuo conto.

Puoi utilizzare ancheAWS CLI per creare un ruolo collegato ai servizi con il caso AWSServiceRoleForLexBotsd'uso. NellaAWS CLI crea un ruolo collegato ai servizi con il nome di servizio di Amazon `Lex``lex.amazonaws.com`. Per ulteriori informazioni, consulta Fase 1: Creazione di un ruolo collegato ai servizi (AWS CLI). Se elimini il ruolo collegato ai servizi, puoi utilizzare lo stesso processo per crearlo nuovamente.

Modifica di un ruolo collegato ai servizi per Amazon Lex

Amazon Lex non consente di modificare i ruoli collegati ai servizi di Amazon Lex. Dopo aver creato un ruolo collegato ai servizi, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon Lex

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

> **Note**
>
> Se il servizio Amazon Lex utilizza tale ruolo quando provi a eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per eliminare le risorse Amazon Lex utilizzate dai ruoli collegati ai servizi:

1. Elimina tutti i canali bot che stai usando.

2. Elimina eventuali bot nel tuo account.

Per eliminare manualmente il ruolo collegato ai servizi utilizzando IAM

Utilizza la console IAMAWS CLI, la o l'AWSAPI per eliminare i ruoli collegati ai servizi di Amazon Lex. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon Lex

Amazon Lex supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la pagina relativa agli endpoint e quote di Amazon Lex.

Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon Lex

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Amazon Lex e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in Amazon Lex
- Non sono autorizzato a eseguire iam: PassRole
- Desidero consentire a persone esterne Account AWS a me di accedere alle mie risorse Amazon Lex

Non sono autorizzato a eseguire un'azione in Amazon Lex

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `lex:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `lex:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo ad Amazon Lex.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio anziché creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in Amazon Lex. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Desidero consentire a persone esterne Account AWS a me di accedere alle mie risorse Amazon Lex

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo degli accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon Lex supporta queste funzionalità, consultaCome funziona Amazon Lex con IAM.

- Per sapere come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente IAM.
- Per scoprire la differenza tra l'utilizzo di ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta Cross Account Resource Access in IAM nella IAM User Guide.

Monitoraggio in Amazon Lex

Il monitoraggio è importante per mantenere l'affidabilità, la disponibilità e le prestazioni dei chatbot Amazon Lex. Questo argomento descrive come utilizzare Amazon CloudWatch Logs e AWS CloudTrail monitorare Amazon Lex e descrive i parametri di runtime e associazione di canale di Amazon Lex.

Argomenti

- Monitoraggio di Amazon Lex con Amazon CloudWatch
- Monitoraggio delle chiamate API Amazon Lex con AWS CloudTrail log

Monitoraggio di Amazon Lex con Amazon CloudWatch

Per monitorare lo stato dei tuoi bot Amazon Lex, usa Amazon CloudWatch. Con CloudWatch, puoi ottenere parametri per singole operazioni Amazon Lex o per operazioni Amazon Lex globali per il tuo account. Puoi anche impostare CloudWatch allarmi per ricevere notifiche quando una o più metriche superano una soglia da te definita. Ad esempio, puoi monitorare il numero di richieste effettuate a un bot in un determinato periodo di tempo, visualizzare la latenza delle richieste riuscite o attivare un allarme quando gli errori superano una soglia.

CloudWatch Metriche per Amazon Lex

Per ottenere i parametri per le tue operazioni su Amazon Lex, devi specificare le seguenti informazioni:

- La dimensione del parametro. Una dimensione è un insieme di coppie nome-valore utilizzate per identificare una metrica. Amazon Lex ha tre dimensioni:
  - `BotAlias, BotName, Operation`
  - `BotAlias, BotName, InputMode, Operation`
  - `BotName, BotVersion, InputMode, Operation`
- Il nome parametro, ad esempio `MissedUtteranceCount` o `RuntimeRequestCount`.

Puoi ottenere metriche per Amazon Lex con AWS Management ConsoleAWS CLI, the o l' CloudWatch API. Puoi utilizzare l' CloudWatch API tramite uno degli Amazon AWS Software Development Kit (SDK) o gli strumenti CloudWatch API. La console Amazon Lex visualizza grafici basati sui dati grezzi dell' CloudWatch API.

È necessario disporre delle CloudWatch autorizzazioni appropriate per monitorare Amazon Lex. CloudWatch Per ulteriori informazioni, consulta Authentication and Access Control for Amazon CloudWatch nella Amazon CloudWatch User Guide.

Visualizzazione dei parametri di Amazon Lex

Visualizza i parametri di Amazon Lex utilizzando la console Amazon Lex o la CloudWatch console.

Per visualizzare i parametri (console Amazon Lex)

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco di bot, scegli quello di cui intendi visualizzare i parametri.
3. Selezionare Monitoring (Monitoraggio). I parametri sono visualizzati nei grafici.

Per visualizzare i parametri (CloudWatch console)

1. Accedi AWS Management Console e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Seleziona Metrics (Parametri), All Metrics (Tutti i parametri), quindi AWS/Lex.
3. Scegliere la dimensione, selezionare un nome parametro e scegliere Add to graph (Aggiungi a grafico).
4. Seleziona un valore per l'intervallo di date. Il numero di parametri per l'intervallo di date selezionato è visualizzato nel grafico.

Creazione di un allarme

Un CloudWatch allarme controlla una singola metrica in un periodo di tempo specificato ed esegue una o più azioni: inviare una notifica a un argomento di Amazon Simple Notification Service (Amazon SNS) o a una politica di Auto Scaling. L'azione o le azioni si basano sul valore della metrica relativo a una determinata soglia in un certo numero di periodi di tempo specificati. CloudWatchpuò anche inviarti un messaggio Amazon SNS quando l'allarme cambia stato.

CloudWatch gli allarmi richiamano azioni solo quando lo stato cambia e persiste per il periodo specificato.

Impostazione di un allarme

1. Accedi AWS Management Console e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Seleziona Alarms (Allarmi), quindi Create Alarm (Crea allarme).
3. Seleziona AWS/Lex Metrics (Parametri AWS/Lex), quindi un parametro.
4. In Time Range (Intervallo di tempo), scegli un intervallo di tempo durante il quale eseguire il monitoraggio, quindi scegli Next (Successivo).
5. Immetti il Name (Nome) e la Description (Descrizione).
6. Per Whenever (Ogni volta), scegli **>=** e digita un valore massimo.
7. Se desideri CloudWatch inviare un'e-mail quando viene raggiunto lo stato di allarme, nella sezione Azioni, per Ogni volta che si verifica un allarme, scegli Lo stato è ALLARME. Per Send notification to (Invia notifica a), scegli un elenco di indirizzi o New list (Nuovo elenco) e crea un nuovo elenco di indirizzi.
8. Visualizza un'anteprima dell'allarme nella sezione Alarm Preview (Anteprima allarme). Se sei soddisfatto dell'allarme, scegli Create Alarm (Crea allarme).

CloudWatchMetriche per Amazon Lex Runtime

La tabella seguente descrive i parametri di runtime di Amazon Lex.

Parametro	Descrizione
`KendraIndexAccessError`	Il numero di volte in cui Amazon Lex non ha potuto accedere al tuo indice Amazon Kendra.

Parametro	Descrizione
	Dimensione valida per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • `BotName, BotAlias, Operation, InputMode` Dimensione valida per l'operazione `PostText`: • `BotName, BotAlias, Operation` Unità: numero
`KendraLatency`	Il tempo impiegato da Amazon Kendra per rispondere a una richiesta proveniente da. `AMAZON.KendraSearchIntent` Dimensioni valide per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensioni valide per l'operazione `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unità: millisecondi

Parametro	Descrizione
KendraSuccess	Il numero di richieste riuscite dall'indice Amazon AMAZON.KendraSearchIntent Kendra al tuo indice Amazon Kendra. Dimensioni valide per l'operazione PostContent con InputMode Text o Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensioni valide per l'operazione PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Unità: numero
KendraSystemErrors	Il numero di volte in cui Amazon Lex non ha potuto eseguire query sull'indice Amazon Kendra. Dimensione valida per l'operazione PostContent con InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensione valida per l'operazione PostText: • BotName, BotAlias, Operation Unità: numero

Parametro	Descrizione
KendraThrottledEvents	Il numero di volte in cui Amazon Kendra ha limitato le richieste provenienti da. AMAZON.KendraSearchIntent Dimensione valida per l'operazione PostContent con InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensione valida per l'operazione PostText: • BotName, BotAlias, Operation Unità: numero
MissedUtteranceCount	Il numero di enunciazioni non riconosciute nel periodo specificato. Dimensioni valide per l'operazione PostContent con InputMode Text o Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensioni valide per l'operazione PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation

Parametro	Descrizione
`RuntimeConcurrency`	Il numero di connessioni simultanee nel periodo di tempo specificato. `RuntimeConcurrency` è segnalato come`StatisticSet` . Dimensioni valide per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • Operazione BotName, BotVersion, InputMode • Operazione BotName, BotAlias, InputMode Dimensioni valide per altre operazioni: • Funzionamento BotName, BotVersion • Operazione BotName, BotAlias Unità: numero
`RuntimeInvalidLambdaResponses`	Il numero di risposte non valide AWS Lambda (Lambda) nel periodo specificato. Dimensione valida per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • `BotName, BotAlias, Operation, InputMode` Dimensione valida per l'operazione `PostText`: • `BotName, BotAlias, Operation`

Parametro	Descrizione
RuntimeLambdaErrors	Il numero di errori di runtime Lambda nel periodo specificato. Dimensione valida per l'operazione PostContent conText Speecho InputMode : • BotName, BotAlias, Operation, InputMode Dimensione valida per l'operazione PostText: • BotName, BotAlias, Operation
RuntimePollyErrors	Il numero di risposte Amazon Polly non valide nel periodo specificato. Dimensione valida per l'operazione PostContent con InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensione valida per l'operazione PostText: • BotName, BotAlias, Operation
RuntimeRequestCount	Il numero di richieste di runtime nel periodo specificato. Dimensioni valide per l'operazione PostContent con InputMode Text o Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensioni valide per l'operazione PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Unità: numero

Parametro	Descrizione
`RuntimeSucessfulRequestLatency` **Important** Questa metrica lo è e non lo è`RuntimeSuccessfulRequestLatency` . `RuntimeSuccessfulRequestLatency`	La latenza per le richieste eseguite correttamente tra il momento in cui la richiesta è stata effettuata e quello in cui la risposta è stata generata. Dimensioni valide per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensioni valide per l'operazione `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unità: millisecondi
`RuntimeSystemErrors`	Il numero di errori di sistema nel periodo specificato. L'intervallo di codici di risposta per un errore di sistema è compreso tra 500 e 599. Dimensione valida per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • `BotName, BotAlias, Operation, InputMode` Dimensione valida per l'operazione `PostText`: • `BotName, BotAlias, Operation` Unità: numero

Parametro	Descrizione
`RuntimeThrottledEvents`	Il numero di richieste sottoposte a throttling. Amazon Lex limita una richiesta quando riceve un numero di richieste superiore al limite di transazioni al secondo impostato per il tuo account. Se il limite impostato per il tuo account viene superato frequentemente, puoi richiedere un aumento del limite. Per richiedere un aumento, consulta Service Limits per AWS. Dimensione valida per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • BotName, Funzionamento BotAlias, InputMode Dimensione valida per l'operazione `PostText`: • `BotName, BotAlias, Operation` Unità: numero
`RuntimeUserErrors`	Il numero di errori utente nel periodo specificato. L'intervallo di codici di risposta per un errore utente è compreso tra 400 e 499. Dimensione valida per l'operazione `PostContent` con `InputMode` `Text` o `Speech`: • `BotName, BotAlias, Operation, InputMode` Dimensione valida per l'operazione `PostText`: • `BotName, BotAlias, Operation` Unità: numero

I parametri di runtime di Amazon Lex utilizzano lo spazio dei `AWS/Lex` nomi e forniscono parametri nelle seguenti dimensioni. Puoi raggruppare i parametri per dimensioni nella console: CloudWatch

Dimensione	Descrizione
`BotName, BotAlias, Operation, InputMode`	Raggruppa i parametri per alias del bot, nome del bot, operazione (`PostContent`) e tipo di input (testo o voce).
`BotName, BotVersion, Operation, InputMode`	Raggruppa i parametri per nome di bot, versione del bot, operazione (`PostContent`) e tipo di input (testo o voce).
`BotName, BotVersion, Operation`	Raggruppa i parametri per nome di bot, versione dei bot e operazione, `PostText`.
`BotName, BotAlias, Operation`	Raggruppa i parametri per nome di bot, alias del bot e operazione, `PostText`.

CloudWatch Metriche per le associazioni di canale Amazon Lex

Un'associazione di canale è l'associazione tra Amazon Lex e un canale di messaggistica, come Facebook. La tabella seguente descrive i parametri di associazione di canale Amazon Lex.

Parametro	Descrizione
`BotChannelAuthErrors`	Il numero di errori di autenticazione restituiti dal canale di messaggistica nel periodo di tempo specificato. Un errore di autenticazione indica che il token segreto fornito durante la creazione del canale non è valido o è scaduto.
`BotChannelConfigurationErrors`	Il numero di errori di configurazione nel periodo specificato. Un errore di configurazione indica che una o più voci di configurazione per il canale non sono valide.
`BotChannelInboundThrottledEvents`	Il numero di volte in cui i messaggi inviati dal canale di messaggistica sono stati limitati da Amazon Lex nel periodo specificato.
`BotChannelOutboundThrottledEvents`	Il numero di volte in cui gli eventi in uscita da Amazon Lex al canale di messaggistica sono stati limitati nel periodo di tempo specificato.

Parametro	Descrizione
`BotChannelRequestCount`	Il numero di richieste effettuate su un canale nel periodo di tempo specificato.
`BotChannelResponseCardErrors`	Il numero di volte in cui Amazon Lex non ha potuto inviare schede di risposta nel periodo specificato.
`BotChannelSystemErrors`	Il numero di errori interni che si sono verificati in Amazon Lex per un canale nel periodo specificato.

Le metriche di associazione di canale di Amazon Lex utilizzano lo spazio dei `AWS/Lex` nomi e forniscono parametri per la dimensione seguente. Puoi raggruppare le metriche per dimensioni nella console: CloudWatch

Dimensione	Descrizione
`BotAlias, BotChannelName, BotName, Source`	Raggruppa i parametri per alias del bot, nome di canale, nome di bot e origine del traffico.

CloudWatch Metriche per i registri delle conversazioni

Amazon Lex utilizza le seguenti metriche per la registrazione delle conversazioni:

Parametro	Descrizione
`ConversationLogsAudioDeliverySuccess`	Il numero di log audio consegnati correttamente al bucket S3 nel periodo di tempo specificato. Unità: numero
`ConversationLogsAudioDeliveryFailure`	Il numero di log audio che non è stato possibile consegnare al bucket S3 nel periodo di tempo specificato. Un errore di consegna indica un errore con le risorse configurate per i log delle conversazioni. Gli errori possono includere

Parametro	Descrizione
	autorizzazioni IAM insufficienti, una AWS KMS chiave inaccessibile o un bucket S3 inaccessi bile. Unità: numero
`ConversationLogsTextDeliver ySuccess`	Il numero di log di testo inviati correttam ente a Logs nel periodo di tempo specificato CloudWatch . Unità: numero
`ConversationLogsTextDeliver yFailure`	Il numero di log di testo che non sono stati consegnati a CloudWatch Logs nel periodo di tempo specificato. Un errore di consegna indica un errore con le risorse configurate per i log delle conversazioni. Gli errori possono includere autorizzazioni IAM insufficienti, una AWS KMS chiave inaccessibile o un gruppo di log Logs CloudWatch inaccessibile. Unità: numero

Le metriche dei log delle conversazioni di Amazon Lex utilizzano lo spazio dei `AWS/Lex` nomi e forniscono parametri per le seguenti dimensioni. Puoi raggruppare i parametri per dimensione nella console. CloudWatch

Dimensione	Descrizione
`BotAlias`	Raggruppa i parametri in base all'alias del bot.
`BotName`	Raggruppa i parametri in base al nome del bot.
`BotVersion`	Raggruppa i parametri in base alla versione del bot.

Monitoraggio delle chiamate API Amazon Lex con AWS CloudTrail log

Amazon Lex è integrato conAWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in Amazon Lex. CloudTrail acquisisce un sottoinsieme di chiamate API per Amazon Lex come eventi, incluse le chiamate dalla console Amazon Lex e le chiamate di codice alle API Amazon Lex. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon Lex. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata ad Amazon Lex, l'indirizzo IP da cui è stata effettuata, chi ha effettuato la richiesta, quando è stata effettuata e ulteriori dettagli.

Per ulteriori informazioni CloudTrail, incluso come configurarlo e abilitarlo, consulta la Guida per l'AWS CloudTrailutente.

Informazioni su Amazon Lex in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività di evento supportata in Amazon Lex, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Amazon Lex, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon Simple Storage Service (Amazon S3). Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il percorso registra gli eventi di tutte le regioni nella partizione AWS e distribuisce i file di log nel bucket S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail log. Per ulteriori informazioni, consultare:

- Panoramica della creazione di un percorso
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Amazon Lex supporta la registrazione delle seguenti operazioni come eventi nei file di CloudTrail registro:

- 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

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Queste informazioni consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Per informazioni sulle azioni di Amazon Lex registrate nei CloudTrail log, consulta Amazon Lex Model Building Service. Ad esempio, le chiamate alle DeleteBot operazioni PutBotGetBot, e generano voci nel CloudTrail registro. Le operazioni documentate nelle sezioni relative al servizio di runtime di Amazon Lex, a PostContent e a PostText, non vengono registrate.

Esempio: voci dei file di log di Amazon Lex

Un trail è una configurazione che consente la consegna di eventi come file di registro in un bucket S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente di voce di CloudTrail registro mostra il risultato di una chiamata all'`PutBot`operazione.

```
   {
                "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"
            }
        }
```

Convalida della conformità per Amazon Lex

I revisori di terze parti valutano la sicurezza e la conformità di Amazon Lex nell'ambito di diversi programmi di AWS conformità. Amazon Lex è un servizio idoneo alla normativa HIPAA. È conforme a PCI, SOC e ISO. Puoi scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download dei rapporti in AWS Artifact.

La tua responsabilità di conformità quando usi Amazon Lex è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità dell'organizzazione e dalle leggi e dai regolamenti applicabili. Se l'uso di Amazon Lex è soggetto alla conformità a standard come PCI, AWS fornisce le seguenti risorse per aiutarti:

- Guide introduttive su sicurezza e conformità: guide alla distribuzione che illustrano considerazioni architettoniche e forniscono passaggi per la distribuzione di ambienti di base incentrati sulla sicurezza e la conformità su AWS
- Whitepaper sull'architettura per compliance e sicurezza HIPAA: questo whitepaper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni conformi ai requisiti HIPAA.
- AWS Risorse per la conformità: una raccolta di cartelle di lavoro e guide che potrebbero riguardare il settore e la località in cui operi
- AWS Config— Un servizio che valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative
- AWS Security Hub— Una visione completa dello stato di sicurezza dell'utente AWS che consente di verificare la conformità agli standard e alle best practice del settore della sicurezza

Per un elenco di AWS servizi che rientrano nell'ambito di programmi di conformità specifici, consulta AWS Services in Scope by Compliance Program. Per informazioni generali, consulta Programmi di conformità di AWS.

Resilienza in Amazon Lex

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure. AWS

Oltre all'infrastruttura AWS globale, Amazon Lex offre diverse funzionalità per supportare le esigenze di resilienza e backup dei dati.

Sicurezza dell'infrastruttura in Amazon Lex

In quanto servizio gestito, Amazon Lex è protetto dalle procedure di sicurezza di rete AWS globali descritte nel white paper Amazon Web Services: Overview of Security Processes.

Utilizzi chiamate AWS API pubblicate per accedere ad Amazon Lex attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.0. È consigliabile TLS 1.2 o versioni successive. I client devono inoltre supportare le suite di cifratura con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni come Java 7 e versioni successive, supporta tali modalità. Inoltre, le richieste devono essere firmate utilizzando un chiave di accesso ID e una chiave di accesso segreta associata a un account principale IAM. O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Puoi chiamare queste operazioni API da qualsiasi posizione di rete, ma Amazon Lex supporta politiche di accesso a livello di risorsa, che possono includere restrizioni basate sull'indirizzo IP di origine. Puoi anche utilizzare le policy di Amazon Lex per controllare l'accesso da endpoint Amazon Virtual Private Cloud (Amazon VPC) specifici o VPC specifici. In effetti, questo isola l'accesso alla rete a una determinata risorsa Amazon Lex solo dal VPC specifico all'interno AWS della rete.

Linee guida e quote in Amazon Lex

Le seguenti sezioni forniscono linee guida e quote per l'utilizzo di Amazon Lex.

Argomenti

Regioni supportate

Per un elenco delleAWS regioni in cui Amazon Lex è disponibile, consulta Endpoint e Endpoint AWS nella Guida di riferimento generale per Amazon Web Services.

Linee guida generali

Questa sezione descrive le linee guida generali per l'utilizzo di Amazon Lex.

- Richieste di firma: tutte le operazioni API di creazione di modelli e runtime di Amazon Lex inDocumentazione di riferimento delle API uso utilizzano la firma V4 per l'autenticazione delle richieste. Per ulteriori informazioni sulle richieste di autenticazione, consulta la pagina relativa al processo di firma Signature Version 4 nella Riferimenti generali di Amazon Web Services.

  InfattiPostContent, Amazon Lex utilizza l'opzione di payload senza firma descritta in Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk (AWS Signature Version 4) nel riferimento API di Amazon Simple Storage Service (S3).

  Se utilizzi l'opzione di payload senza firma, non includere l'hash di payload nella richiesta canonica. In alternativa, puoi utilizzare la stringa letterale "UNSIGNED-PAYLOAD" come hash di payload. Dovrai inoltre includere un'intestazione denominata `x-amz-content-sha256` e il valore `UNSIGNED-PAYLOAD` nella richiesta `PostContent`.

- Tieni presente quanto segue su come Amazon Lex acquisisce i valori degli slot dalle espressioni degli utenti:

  Amazon Lex utilizza i valori di enumerazione forniti in una definizione del tipo di slot per addestrare i suoi modelli di apprendimento automatico. Supponiamo di definire un intento denominato `GetPredictionIntent` con la seguente enunciazione di esempio:

  ```
  "Tell me the prediction for {Sign}"
  ```

  In cui `{Sign}` è uno slot di tipo personalizzato `ZodiacSign`. Sono presenti 12 valori di enumerazione, da `Aries` a `Pisces`. Dall'espressione dell'utente «Dimmi la previsione per...» Amazon Lex capisce che quello che segue è un segno zodiacale.

  Quando il`valueSelectionStrategy` campo è impostato per`ORIGINAL_VALUE` utilizzare l'PutSlotTypeoperazione o se nella console è selezionato Espandi valori, se l'utente dice «Dimmi la previsione della terra», Amazon Lex deduce che «earth» è un`ZodiacSign` e lo trasmette all'applicazione client o alle funzioni Lambda. Prima di utilizzare i valori di slot per l'operazione di creazione, verifica che contengano valori validi.

  Se imposti il campo `valueSelectionStrategy` su `TOP_RESOLUTION` tramite l'operazione PutSlotType oppure se nella console è selezionata l'opzione Restrict to slot values and synonyms (Limita a valori slot e sinonimi), i valori restituiti vengono limitati in base ai valori definiti per il tipo di slot. Ad esempio, se l'utente dice "Tell me the prediction for earth", questo valore non verrà riconosciuto perché non rientra nei valori definiti per il tipo di slot. Quando definisci i sinonimi per valori di slot, questi vengono riconosciuti esattamente come i valori di slot, tuttavia, al posto del sinonimo viene restituito il valore di slot.

  Quando Amazon Lex chiama una funzione Lambda o restituisce il risultato di un'interazione vocale con l'applicazione client, l'esattezza dei valori degli slot non è garantita. Ad esempio, se stai richiedendo valori per il tipo di slot integrato di Amazon.movie e un utente dice o digita «Via col vento», Amazon Lex può restituire «Via col vento», «andato col vento» o «Via col vento». Nelle

interazioni di testo, la distinzione tra maiuscole e minuscole nei valori di slot corrisponde al testo immesso o al valore di slot, a seconda del valore del campo `valueResolutionStrategy`.

- Quando definisci i valori degli slot che contengono acronimi, usa i seguenti schemi:
  - Lettere maiuscole separate da punti (D.V.D.)
  - Lettere maiuscole separate da spazi (D V D)

- Amazon Lex non supporta il tipo di slot integrato AMAZON.LITERAL supportato dall'Alexa Skills Kit. Tuttavia, Amazon Lex supporta la creazione di tipi di slot personalizzati che puoi utilizzare per implementare questa funzionalità. Come accennato nel punto precedente, puoi acquisire i valori al di fuori della definizione del tipo di slot personalizzato. Per migliorare l'accuratezza del riconoscimento vocale automatico (ASR) e della comprensione del linguaggio naturale (NLU), puoi aggiungere ulteriori valori di enumerazione diversi.

- I tipi di slot incorporati AMAZON.DATE e AMAZON.TIME acquisiscono date e orari sia assoluti che relativi. Le date e le ore relative vengono risolte nella regione in cui Amazon Lex sta elaborando la richiesta.

  Per il tipo di slot`AMAZON.TIME` integrato, se l'utente non specifica che l'ora è prima o dopo mezzogiorno, l'ora è ambigua e Amazon Lex chiederà nuovamente all'utente. Ti consigliamo quindi di utilizzare prompt che consentano di ottenere un orario assoluto. Ad esempio, puoi utilizzare prompt simili al seguente "When do you want your pizza delivered? You can say 6 PM or 6 in the evening."

- Fornire dati di formazione confusi nel bot riduce la capacità di Amazon Lex di comprendere gli input degli utenti. Esamina questi esempi:

  Supponiamo di disporre di due intenti (`OrderPizza` e `OrderDrink`) nel bot, entrambi configurati con enunciazione "I want to order". Questa espressione non corrisponde a un intento specifico da cui Amazon Lex può imparare durante la creazione del modello linguistico per il bot in fase

di compilazione. Di conseguenza, quando un utente inserisce questa espressione in fase di esecuzione, Amazon Lex non può scegliere un intento con un elevato grado di sicurezza.

Consideriamo un altro esempio in cui si definisce un intento personalizzato per ottenere una conferma da parte dell'utente (ad esempio `MyCustomConfirmationIntent`) e si configura l'intento con le enunciazioni "Yes" e "No". Tieni presente che Amazon Lex dispone anche di un modello linguistico per comprendere le conferme degli utenti. la presenza di questo modello può determinare una situazione conflittuale. Infatti, quando l'utente risponde con un "Yes", non è chiaro se si tratta di una conferma relativa all'intento in corso o di una richiesta dell'utente relativa all'intento personalizzato creato.

In generale, le enunciazioni di esempio fornite devono essere mappate a un intento specifico e, facoltativamente, a valori di slot specifici.

- Le operazioni delle API di runtime PostContent e PostText accettano un ID utente come parametro obbligatorio. Gli sviluppatori possono impostare questo parametro su qualsiasi valore che soddisfi i vincoli descritti nell'API. Consigliamo di non utilizzare questo parametro per inviare informazioni riservate quali credenziali di accesso, e-mail o numeri di previdenza sociale dell'utente. Questo ID viene utilizzato principalmente per identificare in modo univoco la conversazione con un bot (potrebbero essere presenti più utenti che ordinano la pizza).

- Se l'applicazione client utilizza Amazon Cognito per l'autenticazione, puoi utilizzare l'ID utente Amazon Cognito come ID utente Amazon Lex. Tieni presente che qualsiasi funzione Lambda configurata per il tuo bot deve disporre di un proprio meccanismo di autenticazione per identificare l'utente per conto del quale Amazon Lex sta richiamando la funzione Lambda.

- Consigliamo di definire un intento che consenta di rilevare l'intenzione dell'utente di interrompere una conversazione. Ad esempio, puoi definire un intent (`NothingIntent`) con espressioni di esempio («I don't want anything», «exit», «bye bye»), senza slot e nessuna funzione Lambda configurata come hook di codice. Questo intento consentirebbe agli utenti di chiudere correttamente una conversazione.

Quote

Questa sezione descrive le quote correnti in Amazon Lex. Tali quote sono raggruppate per categorie.

Le quote di assistenza possono essere modificate o aumentate. Contatta l'assistenzaAWS clienti per aumentare una quota. Potrebbero rebbero userci alcuni giorni per aumentare una quota di servizio. Se stai aumentando la tua quota nell'ambito di un progetto più ampio, assicurati di aggiungere questo periodo al tuo piano.

Argomenti

Quote di servizio runtime

Oltre alle quote descritte nella documentazione di riferimento dell'API, tieni presente quanto segue:

Quote API

- L'input vocale per l'operazione PostContent può avere una durata massima di 15 secondi.

- In entrambe le operazioni delle API di runtime PostContent e PostText le dimensioni del testo di input possono avere un massimo di 1024 caratteri Unicode.

- La dimensione massima delle intestazioni `PostContent` è di 16 KB. Le dimensioni massime delle intestazioni di richiesta e sessione combinate è di 12 KB.

- Quando si utilizzano le`PostText` operazioni`PostContent` or in modalità testo, il numero massimo di conversazioni simultanee con un bot è 2 per l'`$LATEST`alias e 50 per tutti gli altri alias. La quota si applica separatamente per ciascuna API.

- Quando si utilizza l'`PostContent`operazione in modalità vocale, il numero massimo di conversazioni simultanee in modalità testo con un bot è 2 per l'`$LATEST`alias e 125 per tutti gli altri alias. La quota si applica separatamente per ciascuna API.

- Il numero massimo di chiamate simultanee di gestione delle sessioni (PutSessionGetSession, eDeleteSession) è 2 per l'`$LATEST`alias di un bot e 50 per tutti gli altri alias.

- La dimensione massima di input per una funzione Lambda è 12 KB. La dimensione massima dell'output è di 25 KB, di cui 12 KB possono essere attributi di sessione.

Utilizzo della versione **`$LATEST`**

- La`$LATEST` versione del bot deve essere utilizzata solo per i test manuali. Amazon Lex limita il numero di richieste di runtime che puoi effettuare alla`$LATEST` versione del bot.

- Quando aggiorni la`$LATEST` versione del bot, Amazon Lex interrompe tutte le conversazioni in corso per qualsiasi applicazione client che utilizza la`$LATEST` versione del bot. In generale, consigliamo di non utilizzare la versione `$LATEST` di un bot in produzione poiché la versione `$LATEST` può essere aggiornata. In alternativa, consigliamo di pubblicare una versione e utilizzare questa versione.

- Quando aggiorni un alias, Amazon Lex impiega alcuni minuti per rilevare la modifica. Quando si modifica la versione `$LATEST` del bot, la modifica viene rilevata immediatamente.

Timeout della sessione

- Il timeout della sessione impostato al momento della creazione del bot determina il tempo per cui il bot mantiene il contesto della conversazione, ad esempio l'intento dell'utente corrente e i dati dello slot.

- Dopo che un utente inizia la conversazione con il bot e fino alla scadenza della sessione, Amazon Lex utilizza la stessa versione del bot, anche se aggiorni l'alias del bot in modo che punti a un'altra versione.

Quote per la creazione di modelli

La costruzione del modello fa riferimento alla creazione e alla gestione dei bot. Ciò include la creazione e la gestione di bot, intenti, tipi di slot, slot e associazioni di canali bot.

Argomenti

Quote Bot

- Nell'API di creazione del modello è possibile configurare prompt e istruzioni. Ogni prompt o istruzione può contenere fino a cinque messaggi e ogni messaggio può contenere da 1 a 1000 caratteri UTF-8.

- Quando si utilizzano gruppi di messaggi, è possibile definire fino a cinque gruppi di messaggi per ciascun messaggio. Ogni gruppo di messaggi può contenere un massimo di cinque messaggi e il limite è pari a 15 messaggi in tutti i gruppi di messaggi.

- È possibile definire espressioni di esempio per intenti e slot. Si può utilizzare un massimo di 200.000 caratteri per tutte le enunciazioni.

- Ogni tipo di slot può definire un massimo di 10.000 valori e sinonimi. Ogni bot può contenere un massimo di 50.000 sinonimi e valori di tipi di slot.

- Al momento della creazione, nomi di associazioni di canale bot, bot e alias non applicano la distinzione tra maiuscole e minuscole. Se si crea un bot `PizzaBot` e quindi un secondo bot `pizzaBot`, verrà visualizzato un messaggio di errore. Tuttavia, quando si accede a una risorsa, i nomi delle risorse applicano la distinzione tra maiuscole e minuscole e sarà necessario quindi specificare `PizzaBot` e non `pizzaBot`. Questi nomi devono comprendere da 2 e 50 caratteri ASCII.

- Il numero massimo di versioni che è possibile pubblicare per tutti i tipi di risorsa è 100. La funzione Versioni multiple non è disponibile per gli alias.

- All'interno di un bot i nomi di intenti e slot devono essere univoci, non è possibile che un intento e uno slot abbiano lo stesso nome.

- È possibile creare un bot configurato per supportare più intenti. Se due intenti contengono uno slot con lo stesso nome, il tipo di slot corrispondente deve essere lo stesso.

  Ad esempio, supponiamo di creare un bot per supportare due intenti (`OrderPizza` e `OrderDrink`). Se entrambi gli intenti includono lo slot `size`, il tipo di slot deve essere lo stesso per entrambi.

  Inoltre, le enunciazioni di esempio fornite per uno slot in uno degli intenti si applicano a uno slot con lo stesso nome in altri intenti.

- Puoi associare un massimo di 250 intenti a un bot.

- Quando si crea un bot, è necessario specificare un timeout per la sessione. Il timeout della sessione può essere compreso tra 1 minuto e un giorno. L'impostazione predefinita è 5 minuti.

- È possibile creare fino a cinque alias per bot.

- Puoi creare fino a 250 bot per account AWS.

- Non è possibile creare più intenti che si estendono dallo stesso intento incorporato.

Quote di intenti

- Al momento della creazione, nomi di intenti e slot non applicano la distinzione tra maiuscole e minuscole. In altre parole, se si crea l'intento `OrderPizza` e quindi un secondo intento `orderPizza`, verrà visualizzato un messaggio di errore. Tuttavia, quando si accede a queste risorse, i nomi delle risorse applicano la distinzione tra maiuscole e minuscole, sarà necessario quindi specificare `OrderPizza` e non `orderPizza`. Questi nomi devono comprendere da 1 e 100 caratteri ASCII.

- Un intento può includere fino a 1.500 enunciazioni di esempio. È richiesta almeno un'enunciazione di esempio. Ogni enunciazione di esempio può contenere fino a 200 caratteri UTF-8. È possibile utilizzare fino a 200.000 caratteri per tutte le enunciazioni di slot e intenti di un bot. Un'enunciazione di esempio per un intento:
  - Può fare riferimento a zero o più nomi di slot.
  - Può fare riferimento a un nome di slot solo una volta.

  Ad esempio:

```
I want a pizza
I want a {pizzaSize} pizza
I want a {pizzaSize} {pizzaTopping} pizza
```

- Sebbene ogni intento supporti fino a 1.500 enunciati, se utilizzi meno enunciati Amazon Lex potrebbe avere una migliore capacità di riconoscere gli input esterni al set fornito.

- È possibile creare fino a cinque gruppi di messaggi per ciascun messaggio di un intento. Può essere presente un totale di 15 messaggi in tutti i gruppi di messaggi per un messaggio.

- La console può solo creare gruppi di messaggi per messaggi `conclusionStatement` e `followUpPrompt`. Puoi creare gruppi di messaggi per qualsiasi altro messaggio utilizzando l'API Amazon Lex.

- Uno slot può includere fino a 10 enunciazioni di esempio. Ogni enunciazione di esempio deve fare riferimento al nome dello slot esattamente una volta. Ad esempio:

```
{pizzaSize} please
```

- Ogni bot può contenere un massimo di 200.000 caratteri per le enunciazioni di intento e slot combinate.

- Non è possibile fornire enunciazioni per intenti che si estendono da intenti incorporati. Per tutti gli altri intenti è necessario fornire almeno un'enunciazione di esempio. Gli intenti contengono slot, tuttavia, le enunciazioni di esempio a livello di slot sono facoltative.

- Intenti incorporati
  - Attualmente, Amazon Lex non supporta l'elicitazione degli slot per scopi integrati. Non è possibile creare funzioni Lambda per restituire la`ElicitSlot` direttiva nella risposta con un intento derivato da intenti incorporati. Per ulteriori informazioni, consulta Formato della risposta.
  - Il servizio non supporta l'aggiunta di enunciazioni di esempio in intenti incorporati. In modo analogo, non è possibile aggiungere o rimuovere slot in intenti incorporati.

- È possibile creare fino a 1.000 intenti per ogni account AWS. È possibile creare fino a 100 slot in un intento.

Quote del tipo di slot

- Al momento della creazione, nomi del tipo di slot non applicano la distinzione tra maiuscole e minuscole. Se si crea il tipo di slot `PizzaSize` e quindi si crea il tipo di slot `pizzaSize`, verrà visualizzato un messaggio di errore. Tuttavia, quando si accede a queste risorse, i nomi delle risorse applicano la distinzione tra maiuscole e minuscole, sarà necessario quindi specificare `PizzaSize` e non `pizzaSize`. I nomi devono comprendere da 1 e 100 caratteri ASCII.

- Un tipo di slot personalizzato creato può includere un massimo di 10.000 valori di enumerazione e sinonimi. Ogni valore può contenere fino a 140 caratteri UTF-8. I valori di enumerazione e i sinonimi non possono contenere duplicati.

- Per un valore del tipo di slot, dove appropriato, è opportuno specificare sia caratteri maiuscoli che minuscoli. Ad esempio, per un tipo di slot denominato `Procedure`, se il valore è `MRI`, consigliamo di specificare come valore sia "MRI" che "mri".

- Tipi di slot integrati: attualmente, Amazon Lex non supporta l'aggiunta di valori di enumerazione o sinonimi per i tipi di slot integrati.

Documentazione di riferimento delle API

Questa sezione fornisce la documentazione per le operazioni dell'API Amazon Lex. Per un elenco delle regioni AWS in cui è disponibile Amazon Lex, consulta Regioni ed endpoints AWS nell'Amazon Web Services General Reference.

Argomenti

Operazioni

Amazon Lex Model Building Service supporta le seguenti operazioni:

- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents
- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

Amazon Lex Runtime Service supporta le seguenti operazioni:

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Servizio di modellazione Amazon Lex

Le seguenti operazioni sono supportate da Amazon Lex Model Building Service:

- 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

Servizio: Amazon Lex Model Building Service

Crea una nuova versione del bot basata sulla `$LATEST` versione. Se la `$LATEST` versione di questa risorsa non è cambiata da quando hai creato l'ultima versione, Amazon Lex non crea una nuova versione. Restituisce l'ultima versione creata.

> **Note**
>
> Puoi aggiornare solo la `$LATEST` versione del bot. Non è possibile aggiornare le versioni numerate create con l'`CreateBotVersion`operazione.

Quando crei la prima versione di un bot, Amazon Lex imposta la versione su 1. Le versioni successive aumentano di 1. Per ulteriori informazioni, consulta Funzione Controllo delle versioni.

Questa operazione richiede l'autorizzazione per l'operazione `lex:CreateBotVersion`.

Sintassi della richiesta

```
POST /bots/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del bot di cui desideri creare una nuova versione. Il nome distingue tra maiuscole e minuscole.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

checksum

Identifica una revisione specifica della $LATEST versione del bot. Se specifichi un checksum e la $LATEST versione del bot ha un checksum diverso, viene restituita un'PreconditionFailedExceptioneccezione e Amazon Lex non pubblica una nuova versione. Se non specifichi un checksum, Amazon Lex pubblica la $LATEST versione.

- Tipo: stringa

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 201.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

abortStatement

Il messaggio che Amazon Lex utilizza per annullare una conversazione. Per ulteriori informazioni, consulta PutBot.

Tipo: oggetto Statement

checksum

Checksum che identifica la versione del bot che è stata creata.

Tipo: stringa

childDirected

Per ogni bot Amazon Lex creato con Amazon Lex Model Building Service, devi specificare se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al Children's Online

Privacy Protection Act (COPPA) specificando `true` o `false` nel `childDirected` campo. Specificando `true` nel `childDirected` campo, confermi che l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al COPPA. Specificando `false` nel `childDirected` campo, confermi che l'uso di Amazon Lex non è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetti al COPPA. Non puoi specificare un valore predefinito per il `childDirected` campo che non riflette accuratamente se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzata o destinata, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetta al COPPA.

Se l'uso di Amazon Lex si riferisce a un sito Web, programma o altra applicazione rivolta, in tutto o in parte, a bambini di età inferiore ai 13 anni, devi ottenere il consenso genitoriale verificabile richiesto ai sensi del COPPA. Per informazioni sull'uso di Amazon Lex in relazione a siti Web, programmi o altre applicazioni rivolti o destinati, in tutto o in parte, a bambini di età inferiore ai 13 anni, consulta le domande frequenti su Amazon Lex.

Tipo: Booleano

clarificationPrompt

Il messaggio che Amazon Lex utilizza quando non comprende la richiesta dell'utente. Per ulteriori informazioni, consulta PutBot.

Tipo: oggetto Prompt

createdDate

La data in cui è stata creata la versione del bot.

Tipo: Timestamp

description

Una descrizione del bot.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

detectSentiment

Indica se gli enunciati inseriti dall'utente devono essere inviati ad Amazon Comprehend per l'analisi del sentiment.

Tipo: Booleano

enableModelImprovements

Indica se il bot utilizza miglioramenti di precisione. `true`indica che il bot sta utilizzando i miglioramenti, altrimenti`false`.

Tipo: Booleano

failureReason

In caso `status` `FAILED` affermativo, Amazon Lex fornisce il motivo per cui non è riuscita a creare il bot.

- Tipo: stringa

idleSessionTTLInSeconds

Il tempo massimo, in secondi, durante il quale Amazon Lex conserva i dati raccolti in una conversazione. Per ulteriori informazioni, consulta PutBot.

Tipo: integer

Intervallo valido: valore minimo pari a 60. Valore massimo pari a 86400.

intents

Un array di oggetti `Intent`. Per ulteriori informazioni, consulta PutBot.

Tipo: matrice di oggetti Intent

lastUpdatedDate

La data in cui è stata aggiornata la `$LATEST` versione di questo bot.

Tipo: Timestamp

locale

Specifica la lingua di destinazione per il bot.

- Tipo: stringa

Valori validi: `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

Il nome del bot.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

status

Quando invii una richiesta per creare o aggiornare un bot, Amazon Lex imposta l'elemento di `status` risposta su`BUILDING`. Dopo che Amazon Lex ha creato il bot, viene `status` impostato su`READY`. Se Amazon Lex non è in grado di creare il bot, `status` lo imposta`FAILED`. Amazon Lex restituisce il motivo dell'errore nell'elemento di `failureReason` risposta.

- Tipo: stringa

Valori validi: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

La versione del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

voiceId

L'ID vocale Amazon Polly utilizzato da Amazon Lex per le interazioni vocali con l'utente.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

PreconditionFailedException

Il checksum della risorsa che stai cercando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3

- AWS SDK per Python
- AWS SDK per Ruby V3

CreateIntentVersion

Servizio: Amazon Lex Model Building Service

Crea una nuova versione di un intento basata sulla `$LATEST` versione dell'intento. Se la `$LATEST` versione di questo intento non è cambiata dall'ultimo aggiornamento, Amazon Lex non crea una nuova versione. Restituisce l'ultima versione che hai creato.

> **Note**
>
> Puoi aggiornare solo la `$LATEST` versione dell'intento. Non è possibile aggiornare le versioni numerate create con l'`CreateIntentVersion`operazione.

Quando crei una versione di un intento, Amazon Lex imposta la versione su 1. Le versioni successive aumentano di 1. Per ulteriori informazioni, consulta Funzione Controllo delle versioni.

Questa operazione necessita delle autorizzazioni a eseguire l'operazione `lex:CreateIntentVersion`.

Sintassi della richiesta

```
POST /intents/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome dell'intento di cui desideri creare una nuova versione. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

checksum

Checksum della $LATEST versione dell'intento da utilizzare per creare la nuova versione. Se specifichi un checksum e la $LATEST versione dell'intento ha un checksum diverso, Amazon Lex restituisce un'PreconditionFailedExceptioneccezione e non pubblica una nuova versione. Se non specifichi un checksum, Amazon Lex pubblica la $LATEST versione.

- Tipo: stringa

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 201.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

checksum

Checksum della versione dell'intento creata.

Tipo: stringa

conclusionStatement

Dopo che la funzione Lambda specificata nel `fulfillmentActivity` campo soddisfa l'intento, Amazon Lex trasmette questa dichiarazione all'utente.

Tipo: oggetto Statement

confirmationPrompt

Se definito, il prompt utilizzato da Amazon Lex per confermare l'intenzione dell'utente prima di soddisfarla.

Tipo: oggetto Prompt

createdDate

La data di creazione dell'intento.

Tipo: Timestamp

description

Una descrizione dell'intento.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

dialogCodeHook

Se definita, Amazon Lex richiama questa funzione Lambda per ogni input dell'utente.

Tipo: oggetto CodeHook

followUpPrompt

Se definito, Amazon Lex utilizza questo prompt per sollecitare attività utente aggiuntive dopo il raggiungimento dell'intento.

Tipo: oggetto FollowUpPrompt

fulfillmentActivity

Descrive come viene soddisfatto l'intento.

Tipo: oggetto FulfillmentActivity

inputContexts

Una serie di `InputContext` oggetti che elenca i contesti che devono essere attivi affinché Amazon Lex possa scegliere l'intento in una conversazione con l'utente.

Tipo: matrice di oggetti InputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo 5 elementi.

kendraConfiguration

Informazioni di configurazione, se presenti, per connettere un indice Amazon Kendra con l'intento. `AMAZON.KendraSearchIntent`

Tipo: oggetto KendraConfiguration

lastUpdatedDate

La data in cui l'intento è stato aggiornato.

Tipo: Timestamp

name

Il nome dell'intento.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

outputContexts

Una matrice di `OutputContext` oggetti che elenca i contesti che l'intento attiva quando l'intento viene soddisfatto.

Tipo: matrice di oggetti OutputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

parentIntentSignature

Un identificatore univoco per un intento incorporato.

-Tipo: stringa

rejectionStatement

Se l'utente risponde «no» alla domanda definita in`confirmationPrompt`, Amazon Lex risponde con questa dichiarazione per confermare che l'intento è stato annullato.

Tipo: oggetto Statement

sampleUtterances

Una serie di espressioni di esempio configurate per l'intento.

Tipo: matrice di stringhe

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 1500 articoli.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 200.

slots

Una serie di tipi di slot che definisce le informazioni necessarie per soddisfare l'intento.

Tipo: matrice di oggetti Slot

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 100 elementi.

version

Il numero di versione assegnato alla nuova versione dell'intento.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

PreconditionFailedException

Il checksum della risorsa che stai cercando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

CreateSlotTypeVersion

Servizio: Amazon Lex Model Building Service

Crea una nuova versione di un tipo di slot in base alla `$LATEST` versione del tipo di slot specificato. Se la `$LATEST` versione di questa risorsa non è cambiata dall'ultima versione che hai creato, Amazon Lex non crea una nuova versione. Restituisce l'ultima versione che hai creato.

> **Note**
>
> È possibile aggiornare solo la `$LATEST` versione di un tipo di slot. Non è possibile aggiornare le versioni numerate create con l'`CreateSlotTypeVersion`operazione.

Quando crei una versione di un tipo di slot, Amazon Lex imposta la versione su 1. Le versioni successive aumentano di 1. Per ulteriori informazioni, consulta Funzione Controllo delle versioni.

Questa operazione richiede le autorizzazioni per l'operazione `lex:CreateSlotTypeVersion`.

Sintassi della richiesta

```
POST /slottypes/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del tipo di slot per cui desideri creare una nuova versione. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

checksum

Checksum per la $LATEST versione del tipo di slot che desideri pubblicare. Se specifichi un checksum e la $LATEST versione del tipo di slot ha un checksum diverso, Amazon Lex restituisce un'PreconditionFailedExceptioneccezione e non pubblica la nuova versione. Se non specifichi un checksum, Amazon Lex pubblica la $LATEST versione.

- Tipo: stringa

Campo obbligatorio: no

Sintassi della risposta

```
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"
```

```
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 201.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

checksum

Checksum della `$LATEST` versione del tipo di slot.

- Tipo: stringa

createdDate

La data di creazione del tipo di slot.

Tipo: Timestamp

description

Una descrizione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

enumerationValues

Un elenco di `EnumerationValue` oggetti che definisce i valori che il tipo di slot può assumere.

Tipo: matrice di oggetti EnumerationValue

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10000 elementi.

lastUpdatedDate

La data in cui il tipo di slot è stato aggiornato. Quando si crea una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Il nome del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Il tipo di slot integrato utilizzato come elemento principale del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informazioni di configurazione che estendono il tipo di slot integrato principale.

Tipo: matrice di oggetti SlotTypeConfiguration

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

valueSelectionStrategy

La strategia utilizzata da Amazon Lex per determinare il valore dello slot. Per ulteriori informazioni, consulta PutSlotType.

- Tipo: stringa

Valori validi: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

La versione assegnata alla nuova versione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

PreconditionFailedException

Il checksum della risorsa che stai cercando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript

- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteBot

Servizio: Amazon Lex Model Building Service

Elimina tutte le versioni del bot, inclusa la `$LATEST` versione. Per eliminare una versione specifica del bot, usa l'DeleteBotVersionoperazione. L'`DeleteBot`operazione non rimuove immediatamente lo schema del bot. Viene invece contrassegnato per l'eliminazione e rimosso in seguito.

Amazon Lex archivia gli enunciati a tempo indeterminato per migliorare la capacità del bot di rispondere agli input degli utenti. Questi enunciati non vengono rimossi quando il bot viene eliminato. Per rimuovere gli enunciati, utilizzate l'DeleteUtterancesoperazione.

Se un bot ha un alias, non puoi eliminarlo. L'`DeleteBot`operazione restituisce invece un'`ResourceInUseException`eccezione che include un riferimento all'alias che si riferisce al bot. Per rimuovere il riferimento al bot, elimina l'alias. Se si verifica nuovamente la stessa eccezione, eliminate l'alias di riferimento fino al completamento dell'`DeleteBot`operazione.

Questa operazione richiede le autorizzazioni per l'operazione `lex:DeleteBot`.

Sintassi della richiesta

```
DELETE /bots/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del bot. Il nome distingue tra maiuscole e minuscole.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteBotAlias

Servizio: Amazon Lex Model Building Service

Elimina un alias per il bot specificato.

Non puoi eliminare un alias utilizzato nell'associazione tra un bot e un canale di messaggistica. Se un alias viene utilizzato in un'associazione di canali, l'`DeleteBot`operazione restituisce un'`ResourceInUseException`eccezione che include un riferimento all'associazione di canale che si riferisce al bot. È possibile rimuovere il riferimento all'alias eliminando l'associazione di canale. Se si verifica nuovamente la stessa eccezione, eliminate l'associazione di riferimento finché l'`DeleteBotAlias`operazione non avrà esito positivo.

Sintassi della richiesta

```
DELETE /bots/botName/aliases/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botName

Il nome del bot a cui punta l'alias.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

name

Il nome dell'alias da eliminare. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteBotChannelAssociation

Servizio: Amazon Lex Model Building Service

Elimina l'associazione tra un bot Amazon Lex e una piattaforma di messaggistica.

Questa operazione richiede l'autorizzazione per l'operazione `lex:DeleteBotChannelAssociation`.

Sintassi della richiesta

```
DELETE /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

aliasName

Un alias che indica la versione specifica del bot Amazon Lex a cui viene effettuata questa associazione.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

botName

Il nome del bot Amazon Lex.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

name

Il nome dell'associazione. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteBotVersion

Servizio: Amazon Lex Model Building Service

Elimina una versione specifica di un bot. Per eliminare tutte le versioni di un bot, usa l'DeleteBotoperazione.

Questa operazione richiede le autorizzazioni per l'operazione `lex:DeleteBotVersion`.

Sintassi della richiesta

```
DELETE /bots/name/versions/version HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del bot.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

version

La versione del bot da eliminare. Non è possibile eliminare la `$LATEST` versione del bot. Per eliminare la `$LATEST` versione, usa l'DeleteBotoperazione.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteIntent

Servizio: Amazon Lex Model Building Service

Elimina tutte le versioni dell'intento, inclusa la `$LATEST` versione. Per eliminare una versione specifica dell'intento, usa l'operazione. DeleteIntentVersion

È possibile eliminare una versione di un intento solo se non è referenziata. Per eliminare un intento a cui si fa riferimento in uno o più bot (vediAmazon Lex: come funziona: come funziona), devi prima rimuovere tali riferimenti.

> **Note**
>
> Se ottieni l'`ResourceInUseException`eccezione, fornisce un esempio di riferimento che mostra dove viene fatto riferimento all'intento. Per rimuovere il riferimento all'intento, aggiorna il bot o eliminalo. Se ottieni la stessa eccezione quando tenti di eliminare nuovamente l'intento, ripeti finché l'intento non ha più riferimenti e la chiamata a `DeleteIntent` ha esito positivo.

Questa operazione richiede l'autorizzazione per l'operazione `lex:DeleteIntent`.

Sintassi della richiesta

```
DELETE /intents/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome dell'intento. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteIntentVersion

Servizio: Amazon Lex Model Building Service

Elimina una versione specifica di un intento. Per eliminare tutte le versioni di un intento, utilizzate l'DeleteIntentoperazione.

Questa operazione richiede le autorizzazioni per l'operazione `lex:DeleteIntentVersion`.

Sintassi della richiesta

```
DELETE /intents/name/versions/version HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome dell'intento.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

version

La versione dell'intento da eliminare. Non è possibile eliminare la `$LATEST` versione dell'intento. Per eliminare la `$LATEST` versione, utilizzare l'DeleteIntentoperazione.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteSlotType

Servizio: Amazon Lex Model Building Service

Elimina tutte le versioni del tipo di slot, inclusa la `$LATEST` versione. Per eliminare una versione specifica del tipo di slot, utilizzare l'DeleteSlotTypeVersionoperazione.

È possibile eliminare una versione di un tipo di slot solo se non è referenziata. Per eliminare un tipo di slot a cui si fa riferimento in uno o più intenti, è necessario rimuovere prima tali riferimenti.

> **Note**
>
> Se ottenete l'`ResourceInUseException`eccezione, l'eccezione fornisce un esempio di riferimento che mostra l'intento in cui viene fatto riferimento al tipo di slot. Per rimuovere il riferimento al tipo di slot, aggiorna l'intento o eliminalo. Se si verifica la stessa eccezione quando si tenta di eliminare nuovamente il tipo di slot, ripetere l'operazione fino a quando il tipo di slot non contiene riferimenti e la `DeleteSlotType` chiamata ha esito positivo.

Questa operazione richiede l'autorizzazione per l'operazione `lex:DeleteSlotType`.

Sintassi della richiesta

```
DELETE /slottypes/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del tipo di slot. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteSlotTypeVersion

Servizio: Amazon Lex Model Building Service

Elimina una versione specifica di un tipo di slot. Per eliminare tutte le versioni di un tipo di slot, utilizzare l'DeleteSlotTypeoperazione.

Questa operazione richiede le autorizzazioni per l'operazione `lex:DeleteSlotTypeVersion`.

Sintassi della richiesta

```
DELETE /slottypes/name/version/version HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del tipo di slot.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

version

La versione del tipo di slot da eliminare. Non è possibile eliminare la `$LATEST` versione del tipo di slot. Per eliminare la `$LATEST` versione, utilizzare l'DeleteSlotTypeoperazione.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

ResourceInUseException

La risorsa che stai tentando di eliminare viene richiamata da un'altra risorsa. Utilizzate queste informazioni per rimuovere i riferimenti alla risorsa che state tentando di eliminare.

Il corpo dell'eccezione contiene un oggetto JSON che descrive la risorsa.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Codice di stato HTTP: 400

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

DeleteUtterances

Servizio: Amazon Lex Model Building Service

Elimina gli enunciati memorizzati.

Amazon Lex archivia gli enunciati che gli utenti inviano al tuo bot. Gli enunciati vengono archiviati per 15 giorni per essere utilizzati con l'GetUtterancesViewoperazione e quindi archiviati a tempo indeterminato per essere utilizzati per migliorare la capacità del bot di rispondere agli input dell'utente.

Utilizzate l'`DeleteUtterances`operazione per eliminare manualmente gli enunciati memorizzati per un utente specifico. Quando si utilizza l'`DeleteUtterances`operazione, gli enunciati memorizzati per migliorare la capacità del bot di rispondere agli input dell'utente vengono eliminati immediatamente. Gli enunciati memorizzati per essere utilizzati con l'`GetUtterancesView`operazione vengono eliminati dopo 15 giorni.

Questa operazione richiede le autorizzazioni per l'operazione `lex:DeleteUtterances`.

Sintassi della richiesta

```
DELETE /bots/botName/utterances/userId HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botName

Il nome del bot che ha memorizzato gli enunciati.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

userId

L'identificatore univoco dell'utente che ha emesso gli enunciati. Si tratta dell'ID utente inviato nella richiesta di PostTextoperazione PostContento contenente l'enunciato.

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBot

Servizio: Amazon Lex Model Building Service

Restituisce informazioni sui metadati per un bot specifico. È necessario fornire il nome del bot e la versione o l'alias del bot.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetBot`.

Sintassi della richiesta

```
GET /bots/name/versions/versionoralias HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del bot. Il nome distingue tra maiuscole e minuscole.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

versionoralias

La versione o l'alias del bot.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

abortStatement

Il messaggio che Amazon Lex restituisce quando l'utente decide di terminare la conversazione senza completarla. Per ulteriori informazioni, consulta PutBot.

Tipo: oggetto Statement

checksum

Checksum del bot utilizzato per identificare una revisione specifica della versione del bot. `$LATEST`

-Tipo: stringa

childDirected

Per ogni bot Amazon Lex creato con Amazon Lex Model Building Service, devi specificare se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al Children's Online Privacy Protection Act (COPPA) specificando `true` o `false` nel `childDirected` campo. Specificando `true` nel `childDirected` campo, confermi che l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al COPPA. Specificando `false` nel `childDirected` campo, confermi che il tuo utilizzo di Amazon Lex non è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetti al COPPA. Non puoi specificare un valore predefinito per il `childDirected` campo che non riflette accuratamente se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzata o destinata, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetta al COPPA.

Se l'uso di Amazon Lex si riferisce a un sito Web, programma o altra applicazione rivolta, in tutto o in parte, a bambini di età inferiore ai 13 anni, devi ottenere il consenso genitoriale verificabile richiesto ai sensi del COPPA. Per informazioni sull'uso di Amazon Lex in relazione a siti Web, programmi o altre applicazioni rivolti o destinati, in tutto o in parte, a bambini di età inferiore ai 13 anni, consulta le domande frequenti su Amazon Lex.

Tipo: Booleano

clarificationPrompt

Il messaggio che Amazon Lex utilizza quando non comprende la richiesta dell'utente. Per ulteriori informazioni, consulta PutBot.

Tipo: oggetto Prompt

createdDate

La data di creazione del bot.

Tipo: Timestamp

description

Una descrizione del bot.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

detectSentiment

Indica se gli enunciati degli utenti devono essere inviati ad Amazon Comprehend per l'analisi del sentiment.

Tipo: Booleano

enableModelImprovements

Indica se il bot utilizza miglioramenti di precisione. `true`indica che il bot sta utilizzando i miglioramenti, altrimenti`false`.

Tipo: Booleano

failureReason

In caso `status` `FAILED` affermativo, Amazon Lex spiega perché non è riuscito a creare il bot.

-Tipo: stringa

idleSessionTTLInSeconds

Il tempo massimo, in secondi, durante il quale Amazon Lex conserva i dati raccolti in una conversazione. Per ulteriori informazioni, consulta PutBot.

Tipo: integer

Intervallo valido: valore minimo pari a 60. Valore massimo pari a 86400.

intents

Un array di oggetti `intent`. Per ulteriori informazioni, consulta PutBot.

Tipo: matrice di oggetti Intent

lastUpdatedDate

La data in cui il bot è stato aggiornato. Quando crei una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

locale

Il locale di destinazione per il bot.

-Tipo: stringa

Valori validi: `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

Il nome del bot.

-Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

Il punteggio che determina dove Amazon Lex inserisce o entrambi quando restituisce intenti alternativi in una risposta PostContento PostText. `AMAZON.FallbackIntent` `AMAZON.KendraSearchIntent` `AMAZON.FallbackIntent`viene inserito se il punteggio di confidenza a tutti gli effetti è inferiore a questo valore. `AMAZON.KendraSearchIntent`viene inserito solo se è configurato per il bot.

Tipo: double

Intervallo valido: valore minimo di 0. Valore massimo di 1.

status

Lo stato del bot.

Quando lo stato è, `BUILDING` Amazon Lex crea il bot per il test e l'uso.

Se lo stato del bot è ugualeREADY_BASIC_TESTING, puoi testarlo utilizzando le espressioni esatte specificate nelle intenzioni del bot. Quando il bot è pronto per il test completo o per l'esecuzione, lo stato è. `READY`

Se c'è stato un problema con la creazione del bot, lo stato è `FAILED` e il `failureReason` campo spiega perché il bot non ha creato.

Se il bot è stato salvato ma non creato, lo stato è`NOT_BUILT`.

- Tipo: stringa

Valori validi: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

La versione del bot. Per un nuovo bot, la versione è sempre`$LATEST`.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

voiceId

L'ID vocale Amazon Polly utilizzato da Amazon Lex per l'interazione vocale con l'utente. Per ulteriori informazioni, consulta PutBot.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBotAlias

Servizio: Amazon Lex Model Building Service

Restituisce informazioni su un alias bot di Amazon Lex. Per ulteriori informazioni sugli alias, consultare Funzione Versioni multiple e alias.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetBotAlias`.

Sintassi della richiesta

```
GET /bots/botName/aliases/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botName

Il nome del bot.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

name

Il nome dell'alias del bot. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

botName

Il nome del bot a cui punta l'alias.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

botVersion

La versione del bot a cui punta l'alias.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

checksum

Checksum dell'alias del bot.

- Tipo: stringa

conversationLogs

Le impostazioni che determinano in che modo Amazon Lex utilizza i log delle conversazioni per l'alias.

Tipo: oggetto ConversationLogsResponse

createdDate

La data di creazione dell'alias del bot.

Tipo: Timestamp

description

Una descrizione dell'alias del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

lastUpdatedDate

La data in cui l'alias del bot è stato aggiornato. Quando crei una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Il nome dell'alias del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBotAliases

Servizio: Amazon Lex Model Building Service

Restituisce un elenco di alias per un bot Amazon Lex specificato.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetBotAliases`.

Sintassi della richiesta

```
GET /bots/botName/aliases/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botName

Il nome del bot.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

maxResults

Il numero massimo di alias da restituire nella risposta. L'impostazione predefinita è 50.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nameContains

Sottostringa da abbinare nei nomi degli alias dei bot. Verrà restituito un alias se una parte del suo nome corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz».

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

nextToken

Un token di impaginazione per recuperare la pagina successiva di alias. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta.

Per recuperare la pagina successiva di alias, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

BotAliases

Una serie di `BotAliasMetadata` oggetti, ciascuno dei quali descrive un alias bot.

Tipo: matrice di oggetti BotAliasMetadata

nextToken

Un token di impaginazione per recuperare la pagina successiva di alias. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva di alias, specifica il token di impaginazione nella richiesta successiva.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS

- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBotChannelAssociation

Servizio: Amazon Lex Model Building Service

Restituisce informazioni sull'associazione tra un bot Amazon Lex e una piattaforma di messaggistica.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetBotChannelAssociation`.

Sintassi della richiesta

```
GET /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

aliasName

Un alias che indica la versione specifica del bot Amazon Lex a cui viene effettuata questa associazione.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

botName

Il nome del bot Amazon Lex.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

name

Il nome dell'associazione tra il bot e il canale. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

botAlias

Un alias che indica la versione specifica del bot Amazon Lex a cui viene effettuata questa associazione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

botConfiguration

Fornisce le informazioni necessarie alla piattaforma di messaggistica per comunicare con il bot Amazon Lex.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero massimo di 10 elementi.

botName

Il nome del bot Amazon Lex.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

createdDate

La data di creazione dell'associazione tra il bot e il canale.

Tipo: Timestamp

description

Una descrizione dell'associazione tra il bot e il canale.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

failureReason

In caso `status` `FAILED` affermativo, Amazon Lex fornisce il motivo per cui non è riuscita a creare l'associazione.

- Tipo: stringa

name

Il nome dell'associazione tra il bot e il canale.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

status

Lo stato del canale bot.

- `CREATED`- Il canale è stato creato ed è pronto per l'uso.
- `IN_PROGRESS`- La creazione del canale è in corso.
- `FAILED`- Si è verificato un errore durante la creazione del canale. Per informazioni sul motivo dell'errore, consulta il `failureReason` campo.

▬Tipo: stringa

Valori validi: `IN_PROGRESS | CREATED | FAILED`

type

Il tipo di piattaforma di messaggistica.

▬Tipo: stringa

Valori validi: `Facebook | Slack | Twilio-Sms | Kik`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBotChannelAssociations

Servizio: Amazon Lex Model Building Service

Restituisce un elenco di tutti i canali associati al bot specificato.

L'`GetBotChannelAssociations`operazione richiede le autorizzazioni per l'`lex:GetBotChannelAssociations`azione.

Sintassi della richiesta

```
GET /bots/botName/aliases/aliasName/channels/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

aliasName

Un alias che indica la versione specifica del bot Amazon Lex a cui viene effettuata questa associazione.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^(-|^([A-Za-z]_?)+$)$`

Campo obbligatorio: sì

botName

Il nome del bot Amazon Lex nell'associazione.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

maxResults

Il numero massimo di associazioni da restituire nella risposta. L'impostazione predefinita è 50.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nameContains

Sottostringa da abbinare nei nomi delle associazioni di canale. Un'associazione verrà restituita se una parte del suo nome corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz». Per restituire tutte le associazioni dei canali dei bot, utilizzate un trattino («-») come parametro. `nameContains`

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

nextToken

Un token di impaginazione per recuperare la pagina successiva di associazioni. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva di associazioni, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

botChannelAssociations

Una serie di oggetti, uno per ogni associazione, che fornisce informazioni sul bot Amazon Lex e sulla sua associazione con il canale.

Tipo: matrice di oggetti BotChannelAssociation

nextToken

Un token di impaginazione che recupera la pagina successiva di associazioni. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva di associazioni, specifica il token di impaginazione nella richiesta successiva.

Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBots

Servizio: Amazon Lex Model Building Service

Restituisce le informazioni sul bot come segue:

- Se fornite il `nameContains` campo, la risposta include informazioni sulla `$LATEST` versione di tutti i bot il cui nome contiene la stringa specificata.
- Se non specifichi il `nameContains` campo, l'operazione restituisce informazioni sulla `$LATEST` versione di tutti i tuoi bot.

Questa operazione richiede l'autorizzazione per l'operazione `lex:GetBots`.

Sintassi della richiesta

```
GET /bots/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di bot da restituire nella risposta che la richiesta restituirà. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nameContains

Sottostringa da abbinare nei nomi dei bot. Un bot verrà restituito se una parte del suo nome corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz».

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

nextToken

Un token di impaginazione che recupera la pagina successiva dei bot. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta.

Per recuperare la pagina successiva dei bot, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

bots

Una serie di `botMetadata` oggetti, con un ingresso per ogni bot.

Tipo: matrice di oggetti BotMetadata

nextToken

Se la risposta viene troncata, include un token di impaginazione che puoi specificare nella tua prossima richiesta per recuperare la pagina successiva dei bot.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3

- AWS SDK per Python
- AWS SDK per Ruby V3

GetBotVersions

Servizio: Amazon Lex Model Building Service

Ottiene informazioni su tutte le versioni di un bot.

L'`GetBotVersions`operazione restituisce un `BotMetadata` oggetto per ogni versione di un bot. Ad esempio, se un bot ha tre versioni numerate, l'`GetBotVersions`operazione restituisce quattro `BotMetadata` oggetti nella risposta, uno per ogni versione numerata e uno per la `$LATEST` versione.

L'`GetBotVersions`operazione restituisce sempre almeno una versione, la `$LATEST` versione.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetBotVersions`.

Sintassi della richiesta

```
GET /bots/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di versioni del bot da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

name

Il nome del bot per il quale restituire le versioni.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

nextToken

Un token di impaginazione per recuperare la pagina successiva delle versioni dei bot. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione

nella risposta. Per recuperare la pagina successiva di versioni, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

bots

Una serie di `BotMetadata` oggetti, uno per ogni versione numerata del bot più uno per la versione. `$LATEST`

Tipo: matrice di oggetti BotMetadata

nextToken

Un token di impaginazione per recuperare la pagina successiva delle versioni del bot. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione

nella risposta. Per recuperare la pagina successiva di versioni, specifica il token di impaginazione nella richiesta successiva.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2

- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBuiltinIntent

Servizio: Amazon Lex Model Building Service

Restituisce informazioni su un intento integrato.

Questa operazione richiede l'autorizzazione per l'operazione `lex:GetBuiltinIntent`.

Sintassi della richiesta

```
GET /builtins/intents/signature HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

signature

L'identificatore univoco per un intento integrato. Per trovare la firma di un intento, consulta Standard Built-in Intents nell'Alexa Skills Kit.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "signature": "string",
   "slots": [
      {
         "name": "string"
      }
   ],
   "supportedLocales": [ "string" ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

signature

L'identificatore univoco di un intento integrato.

- Tipo: stringa

slots

Una serie di `BuiltinIntentSlot` oggetti, una voce per ogni tipo di slot nell'intento.

Tipo: matrice di oggetti BuiltinIntentSlot

supportedLocales

Un elenco di impostazioni locali supportate dall'intento.

Tipo: matrice di stringhe

Valori validi: `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`

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBuiltinIntents

Servizio: Amazon Lex Model Building Service

Consente di ottenere un elenco di intenti integrati che soddisfano i criteri specificati.

Questa operazione richiede l'autorizzazione per l'operazione `lex:GetBuiltinIntents`.

Sintassi della richiesta

```
GET /builtins/intents/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

locale

Un elenco di impostazioni locali supportate dall'intento.

Valori validi: `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

Il numero massimo di intenti da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nextToken

Un token di impaginazione che recupera la pagina successiva di intenti. Se questa chiamata API viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina di intenti successiva, utilizza il token di impaginazione nella richiesta successiva.

signatureContains

Sottostringa da abbinare alle firme di intento integrate. Verrà restituito un intento se una parte della sua firma corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz». Per trovare la firma di un intento, consulta Standard Built-in Intents nell'Alexa Skills Kit.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ],
   "nextToken": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

intents

Una serie di `builtinIntentMetadata` oggetti, uno per ogni intento della risposta.

Tipo: matrice di oggetti BuiltinIntentMetadata

nextToken

Un token di impaginazione che recupera la pagina successiva di intenti. Se la risposta a questa chiamata API viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina di intenti successiva, specifica il token di impaginazione nella richiesta successiva.

-Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetBuiltinSlotTypes

Servizio: Amazon Lex Model Building Service

Restituisce un elenco di tipi di slot integrati che soddisfano i criteri specificati.

Per un elenco dei tipi di slot integrati, consulta Slot Type Reference nell'Alexa Skills Kit.

Questa operazione richiede l'autorizzazione per l'operazione `lex:GetBuiltInSlotTypes`.

Sintassi della richiesta

```
GET /builtins/slottypes/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

locale

Un elenco di impostazioni locali supportate dal tipo di slot.

Valori validi: `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

Il numero massimo di tipi di slot da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nextToken

Un token di impaginazione che recupera la pagina successiva dei tipi di slot. Se la risposta a questa chiamata API viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva dei tipi di slot, specifica il token di impaginazione nella richiesta successiva.

signatureContains

Sottostringa da abbinare alle firme di tipo slot integrate. Verrà restituito un tipo di slot se una parte della sua firma corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz».

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

nextToken

Se la risposta viene troncata, la risposta include un token di impaginazione che puoi utilizzare nella tua richiesta successiva per recuperare la pagina successiva di tipi di slot.

- Tipo: stringa

slotTypes

Viene restituita una matrice di `BuiltInSlotTypeMetadata` oggetti, una voce per ogni tipo di slot.

Tipo: matrice di oggetti BuiltinSlotTypeMetadata

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetExport

Servizio: Amazon Lex Model Building Service

Esporta il contenuto di una risorsa Amazon Lex in un formato specificato.

Sintassi della richiesta

```
GET /exports/?exportType=exportType&name=name&resourceType=resourceType&version=version
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

exportType

Il formato dei dati esportati.

Valori validi: `ALEXA_SKILLS_KIT | LEX`

Campo obbligatorio: sì

name

Il nome del bot da esportare.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `[a-zA-Z_]+`

Campo obbligatorio: sì

resourceType

Il tipo di risorsa da esportare.

Valori validi: `BOT | INTENT | SLOT_TYPE`

Campo obbligatorio: sì

version

La versione del bot da esportare.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "exportStatus": "string",
   "exportType": "string",
   "failureReason": "string",
   "name": "string",
   "resourceType": "string",
   "url": "string",
   "version": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

exportStatus

Lo stato dell'esportazione.

- `IN_PROGRESS`- L'esportazione è in corso.
- `READY`- L'esportazione è completa.
- `FAILED`- L'esportazione non può essere completata.

➨Tipo: stringa

Valori validi: `IN_PROGRESS | READY | FAILED`

exportType

Il formato dei dati esportati.

- Tipo: stringa

Valori validi: `ALEXA_SKILLS_KIT | LEX`

failureReason

In caso `status` `FAILED` affermativo, Amazon Lex fornisce il motivo per cui non è riuscita a esportare la risorsa.

- Tipo: stringa

name

Il nome del bot che viene esportato.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `[a-zA-Z_]+`

resourceType

Il tipo di risorsa esportata.

- Tipo: stringa

Valori validi: `BOT | INTENT | SLOT_TYPE`

url

Un URL prefirmato S3 che fornisce la posizione della risorsa esportata. La risorsa esportata è un archivio ZIP che contiene la risorsa esportata in formato JSON. La struttura dell'archivio potrebbe cambiare. Il codice non deve basarsi sulla struttura dell'archivio.

- Tipo: stringa

version

La versione del bot che viene esportata.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript

- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetImport

Servizio: Amazon Lex Model Building Service

Ottiene informazioni su un processo di importazione iniziato con l'`StartImport`operazione.

Sintassi della richiesta

```
GET /imports/importId HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

importId

L'identificatore delle informazioni sul processo di importazione da restituire.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "createdDate": number,
   "failureReason": [ "string" ],
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

createdDate

Un timestamp per la data e l'ora di creazione del processo di importazione.

Tipo: Timestamp

failureReason

Una stringa che descrive il motivo per cui un processo di importazione non è stato completato.

Tipo: matrice di stringhe

importId

L'identificatore per il processo di importazione specifico.

- Tipo: stringa

importStatus

Lo stato del processo di importazione. Se lo stato è`FAILED`, è possibile ottenere il motivo dell'errore dal `failureReason` campo.

- Tipo: stringa

Valori validi: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

Azione intrapresa in caso di conflitto tra una risorsa esistente e una risorsa nel file di importazione.

- Tipo: stringa

Valori validi: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

Il nome assegnato al processo di importazione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `[a-zA-Z_]+`

resourceType

Il tipo di risorsa importata.

- Tipo: stringa

Valori validi: `BOT | INTENT | SLOT_TYPE`

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++

- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetIntent

Servizio: Amazon Lex Model Building Service

Restituisce informazioni su un intento. Oltre al nome dell'intento, è necessario specificare la versione dell'intento.

Questa operazione necessita delle autorizzazioni a eseguire l'operazione `lex:GetIntent`.

Sintassi della richiesta

```
GET /intents/name/versions/version HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome dell'intento. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

version

La versione dell'intento.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

checksum

Checksum dell'intento.

- Tipo: stringa

conclusionStatement

Dopo che la funzione Lambda specificata nell'`fulfillmentActivity`elemento soddisfa l'intento, Amazon Lex trasmette questa dichiarazione all'utente.

Tipo: oggetto Statement

confirmationPrompt

Se definito nel bot, Amazon Lex utilizza il prompt per confermare l'intento prima di soddisfare la richiesta dell'utente. Per ulteriori informazioni, consulta PutIntent.

Tipo: oggetto Prompt

createdDate

La data in cui è stato creato l'intento.

Tipo: Timestamp

description

Una descrizione dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

dialogCodeHook

Se definita nel bot, Amazon Amazon Lex richiama questa funzione Lambda per ogni input dell'utente. Per ulteriori informazioni, consulta PutIntent.

Tipo: oggetto CodeHook

followUpPrompt

Se definito nel bot, Amazon Lex utilizza questo prompt per sollecitare ulteriori attività utente dopo che l'intento è stato raggiunto. Per ulteriori informazioni, consulta PutIntent.

Tipo: oggetto FollowUpPrompt

fulfillmentActivity

Descrive come viene soddisfatto l'intento. Per ulteriori informazioni, consulta PutIntent.

Tipo: oggetto FulfillmentActivity

inputContexts

Una serie di `InputContext` oggetti che elenca i contesti che devono essere attivi affinché Amazon Lex possa scegliere l'intento in una conversazione con l'utente.

Tipo: matrice di oggetti InputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo 5 elementi.

kendraConfiguration

Informazioni di configurazione, se presenti, per connettersi a un indice Amazon Kendra con l'intento. `AMAZON.KendraSearchIntent`

Tipo: oggetto KendraConfiguration

lastUpdatedDate

La data in cui l'intento è stato aggiornato. Quando crei una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Il nome dell'intento.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

outputContexts

Una matrice di `OutputContext` oggetti che elenca i contesti che l'intento attiva quando l'intento viene soddisfatto.

Tipo: matrice di oggetti OutputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

parentIntentSignature

Un identificatore univoco per un intento incorporato.

- Tipo: stringa

rejectionStatement

Se l'utente risponde «no» alla domanda definita in `confirmationPrompt`, Amazon Lex risponde con questa dichiarazione per confermare che l'intento è stato annullato.

Tipo: oggetto Statement

sampleUtterances

Una serie di espressioni di esempio configurate per l'intento.

Tipo: matrice di stringhe

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 1500 articoli.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 200.

slots

Una serie di slot di intento configurati per l'intento.

Tipo: matrice di oggetti Slot

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 100 elementi.

version

La versione dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetIntents

Servizio: Amazon Lex Model Building Service

Restituisce le informazioni sull'intento come segue:

- Se si specifica il `nameContains` campo, restituisce la `$LATEST` versione di tutti gli intenti che contengono la stringa specificata.
- Se non si specifica il `nameContains` campo, restituisce informazioni sulla `$LATEST` versione di tutti gli intenti.

L'operazione richiede l'autorizzazione per l'`lex:GetIntents`azione.

Sintassi della richiesta

```
GET /intents/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di intenti da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nameContains

Sottostringa da abbinare nei nomi degli intenti. Verrà restituito un intento se una parte del suo nome corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz».

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

nextToken

Un token di impaginazione che recupera la pagina successiva di intenti. Se la risposta a questa chiamata API viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta.

Per recuperare la pagina di intenti successiva, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

intents

Un array di oggetti `Intent`. Per ulteriori informazioni, consulta PutBot.

Tipo: matrice di oggetti IntentMetadata

nextToken

Se la risposta viene troncata, la risposta include un token di impaginazione che puoi specificare nella tua richiesta successiva per recuperare la pagina successiva di intenti.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetIntentVersions

Servizio: Amazon Lex Model Building Service

Ottiene informazioni su tutte le versioni di un intento.

L'`GetIntentVersions`operazione restituisce un `IntentMetadata` oggetto per ogni versione di un intento. Ad esempio, se un intento ha tre versioni numerate, l'`GetIntentVersions`operazione restituisce quattro `IntentMetadata` oggetti nella risposta, uno per ogni versione numerata e uno per la versione. `$LATEST`

L'`GetIntentVersions`operazione restituisce sempre almeno una versione, la versione. `$LATEST`

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetIntentVersions`.

Sintassi della richiesta

```
GET /intents/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di versioni di intento da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

name

Il nome dell'intento per il quale devono essere restituite le versioni.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

nextToken

Un token di impaginazione per recuperare la pagina successiva delle versioni di intent. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione

nella risposta. Per recuperare la pagina successiva di versioni, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

intents

Una serie di `IntentMetadata` oggetti, uno per ogni versione numerata dell'intento più uno per la versione. `$LATEST`

Tipo: matrice di oggetti IntentMetadata

nextToken

Un token di impaginazione per recuperare la pagina successiva delle versioni di intent. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione

nella risposta. Per recuperare la pagina successiva di versioni, specifica il token di impaginazione nella richiesta successiva.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2

- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetMigration

Servizio: Amazon Lex Model Building Service

Fornisce dettagli su una migrazione continua o completa da un bot Amazon Lex V1 a un bot Amazon Lex V2. Utilizza questa operazione per visualizzare gli avvisi e gli avvisi di migrazione relativi alla migrazione.

Sintassi della richiesta

```
GET /migrations/migrationId HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

migrationId

L'identificatore univoco della migrazione da visualizzare. `migrationID`Viene restituito dall'StartMigrationoperazione.

Vincoli di lunghezza: lunghezza fissa di 10.

Modello: `^[0-9a-zA-Z]+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

alerts

Un elenco di avvisi e avvisi che indicano problemi con la migrazione del bot Amazon Lex V1 ad Amazon Lex V2. Ricevi un avviso quando una funzionalità di Amazon Lex V1 ha un'implementazione diversa in Amazon Lex V2.

Per ulteriori informazioni, consulta Migrazione di un bot nella guida per sviluppatori di Amazon Lex V2.

Tipo: matrice di oggetti MigrationAlert

migrationId

L'identificatore univoco della migrazione. È lo stesso identificatore utilizzato per richiamare l'`GetMigration`operazione.

Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 10.

Modello: `^[0-9a-zA-Z]+$`

migrationStatus

Indica lo stato della migrazione. Quando lo stato è, `COMPLETE` la migrazione è terminata e il bot è disponibile in Amazon Lex V2. Potrebbero esserci avvisi e avvisi che devono essere risolti per completare la migrazione.

- Tipo: stringa

Valori validi: `IN_PROGRESS | COMPLETED | FAILED`

migrationStrategy

La strategia utilizzata per condurre la migrazione.

- `CREATE_NEW`- Crea un nuovo bot Amazon Lex V2 e migra il bot Amazon Lex V1 al nuovo bot.
- `UPDATE_EXISTING`- Sovrascrive i metadati del bot di Amazon Lex V2 esistenti e le impostazioni locali da migrare. Non modifica le altre impostazioni locali nel bot Amazon Lex V2. Se la localizzazione non esiste, viene creata una nuova locale nel bot Amazon Lex V2.

- Tipo: stringa

Valori validi: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

La data e l'ora di inizio della migrazione.

Tipo: Timestamp

v1BotLocale

Le impostazioni locali del bot Amazon Lex V1 sono state migrate su Amazon Lex V2.

- Tipo: stringa

Valori validi: `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

Il nome del bot Amazon Lex V1 è stato migrato ad Amazon Lex V2.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

v1BotVersion

La versione del bot Amazon Lex V1 è stata migrata ad Amazon Lex V2.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

v2BotId

L'identificatore univoco del bot Amazon Lex V2 verso cui viene effettuata la migrazione di Amazon Lex V1.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 10.

Modello: `^[0-9a-zA-Z]+$`

v2BotRole

Il ruolo IAM utilizzato da Amazon Lex per eseguire il bot Amazon Lex V2.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetMigrations

Servizio: Amazon Lex Model Building Service

Ottiene un elenco di migrazioni tra Amazon Lex V1 e Amazon Lex V2.

Sintassi della richiesta

```
GET /migrations?
maxResults=maxResults&migrationStatusEquals=migrationStatusEquals&nextToken=nextToken&sortByAtt
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di migrazioni da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

migrationStatusEquals

Filtra l'elenco per contenere solo le migrazioni nello stato specificato.

Valori validi: `IN_PROGRESS | COMPLETED | FAILED`

nextToken

Un token di impaginazione che recupera la pagina successiva delle migrazioni. Se la risposta a questa operazione viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva delle migrazioni, specifica il token di impaginazione nella richiesta.

sortByAttribute

Il campo in base al quale ordinare l'elenco delle migrazioni. Puoi ordinare in base al nome del bot di Amazon Lex V1 o alla data e all'ora di inizio della migrazione.

Valori validi: `V1_BOT_NAME | MIGRATION_DATE_TIME`

sortByOrder

L'ordine consente di ordinare l'elenco.

Valori validi: `ASCENDING | DESCENDING`

v1BotNameContains

Filtra l'elenco per contenere solo i bot il cui nome contiene la stringa specificata. La stringa trova una corrispondenza in un punto qualsiasi del nome del bot.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

migrationSummaries

Una serie di riepiloghi per le migrazioni da Amazon Lex V1 ad Amazon Lex V2. Per visualizzare i dettagli della migrazione, utilizza il riepilogo `migrationId` contenuto in una chiamata all'operazione. GetMigration

Tipo: matrice di oggetti MigrationSummary

nextToken

Se la risposta viene troncata, include un token di impaginazione che puoi specificare nella tua richiesta successiva per recuperare la pagina successiva di migrazioni.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET

- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetSlotType

Servizio: Amazon Lex Model Building Service

Restituisce informazioni su una specifica versione di un tipo di slot. Oltre a specificare il nome del tipo di slot, è necessario specificare la versione del tipo di slot.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetSlotType`.

Sintassi della richiesta

```
GET /slottypes/name/versions/version HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del tipo di slot. Il nome distingue tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

version

La versione del tipo di slot.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

checksum

Checksum della $LATEST versione del tipo di slot.

Tipo: stringa

createdDate

Data di creazione del tipo di slot.

Tipo: Timestamp

description

Una descrizione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

enumerationValues

Un elenco di `EnumerationValue` oggetti che definisce i valori che il tipo di slot può assumere.

Tipo: matrice di oggetti EnumerationValue

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10000 elementi.

lastUpdatedDate

Data di aggiornamento del tipo di slot. Quando si crea una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Il nome del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Il tipo di slot integrato utilizzato come elemento principale per il tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informazioni di configurazione che estendono il tipo di slot integrato principale.

Tipo: matrice di oggetti SlotTypeConfiguration

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

valueSelectionStrategy

La strategia utilizzata da Amazon Lex per determinare il valore dello slot. Per ulteriori informazioni, consulta PutSlotType.

- Tipo: stringa

Valori validi: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

La versione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetSlotTypes

Servizio: Amazon Lex Model Building Service

Restituisce le informazioni sul tipo di slot come segue:

- Se si specifica il `nameContains` campo, restituisce la `$LATEST` versione di tutti i tipi di slot che contengono la stringa specificata.
- Se non si specifica il `nameContains` campo, restituisce informazioni sulla `$LATEST` versione di tutti i tipi di slot.

L'operazione richiede l'autorizzazione per l'`lex:GetSlotTypes`azione.

Sintassi della richiesta

```
GET /slottypes/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di tipi di slot da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

nameContains

Sottostringa da abbinare nei nomi dei tipi di slot. Verrà restituito un tipo di slot se una parte del suo nome corrisponde alla sottostringa. Ad esempio, «xyz» corrisponde sia a «xyzabc» che a «abcxyz».

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

nextToken

Un token di impaginazione che recupera la pagina successiva dei tipi di slot. Se la risposta a questa chiamata API viene troncata, Amazon Lex restituisce un token di impaginazione nella

risposta. Per recuperare la pagina successiva dei tipi di slot, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

nextToken

Se la risposta viene troncata, include un token di impaginazione che potete specificare nella richiesta successiva per recuperare la pagina successiva dei tipi di slot.

- Tipo: stringa

slotTypes

Una serie di oggetti, uno per ogni tipo di slot, che fornisce informazioni come il nome del tipo di slot, la versione e una descrizione.

Tipo: matrice di oggetti SlotTypeMetadata

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3

- AWS SDK per Python
- AWS SDK per Ruby V3

GetSlotTypeVersions

Servizio: Amazon Lex Model Building Service

Ottiene informazioni su tutte le versioni di un tipo di slot.

L'`GetSlotTypeVersions`operazione restituisce un `SlotTypeMetadata` oggetto per ogni versione di un tipo di slot. Ad esempio, se un tipo di slot ha tre versioni numerate, l'`GetSlotTypeVersions`operazione restituisce quattro `SlotTypeMetadata` oggetti nella risposta, uno per ogni versione numerata e uno per la `$LATEST` versione.

L'`GetSlotTypeVersions`operazione restituisce sempre almeno una versione, la `$LATEST` versione.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetSlotTypeVersions`.

Sintassi della richiesta

```
GET /slottypes/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

maxResults

Il numero massimo di versioni di tipo slot da restituire nella risposta. Il valore predefinito è 10.

Intervallo valido: valore minimo di 1. Valore massimo pari a 50.

name

Il nome del tipo di slot per il quale devono essere restituite le versioni.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

nextToken

Un token di impaginazione per recuperare la pagina successiva delle versioni di tipo slot. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva di versioni, specifica il token di impaginazione nella richiesta successiva.

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

nextToken

Un token di impaginazione per recuperare la pagina successiva delle versioni di tipo slot. Se la risposta a questa chiamata viene troncata, Amazon Lex restituisce un token di impaginazione nella risposta. Per recuperare la pagina successiva di versioni, specifica il token di impaginazione nella richiesta successiva.

- Tipo: stringa

slotTypes

Una serie di `SlotTypeMetadata` oggetti, uno per ogni versione numerata del tipo di slot più uno per la versione. `$LATEST`

Tipo: matrice di oggetti SlotTypeMetadata

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetUtterancesView

Servizio: Amazon Lex Model Building Service

Usa l'`GetUtterancesView`operazione per ottenere informazioni sulle espressioni che i tuoi utenti hanno fatto al tuo bot. Puoi usare questo elenco per ottimizzare gli enunciati a cui risponde il bot.

Ad esempio, supponiamo di aver creato un bot per ordinare fiori. Dopo che gli utenti hanno utilizzato il bot per un certo periodo, utilizzate l'`GetUtterancesView`operazione per vedere le richieste che hanno fatto e se hanno avuto successo. Potresti scoprire che l'espressione «Voglio dei fiori» non viene riconosciuta. È possibile aggiungere questo enunciato all'`OrderFlowers`intento in modo che il bot lo riconosca.

Dopo aver pubblicato una nuova versione di un bot, è possibile ottenere informazioni sulla versione precedente e sulla nuova in modo da poter confrontare le prestazioni tra le due versioni.

Una volta al giorno vengono create le statistiche delle enunciazioni. I dati sono disponibili per gli ultimi 15 giorni. Puoi richiedere informazioni per un massimo di 5 versioni del tuo bot per ogni richiesta. Amazon Lex restituisce gli enunciati più frequenti ricevuti dal bot negli ultimi 15 giorni. La risposta contiene informazioni su un massimo di 100 enunciati per ogni versione.

Le statistiche sugli enunciati non vengono generate nelle seguenti condizioni:

- Il `childDirected` campo era impostato su true al momento della creazione del bot.
- Stai usando l'offuscamento degli slot con uno o più slot.
- Hai scelto di non partecipare al miglioramento di Amazon Lex.

Questa operazione richiede le autorizzazioni per l'operazione `lex:GetUtterancesView`.

Sintassi della richiesta

```
GET /bots/botname/utterances?
view=aggregation&bot_versions=botVersions&status_type=statusType HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botname

Il nome del bot per il quale devono essere restituite le informazioni sull'enunciato.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

botVersions

Una serie di versioni dei bot per le quali devono essere restituite le informazioni sull'enunciato. Il limite è di 5 versioni per richiesta.

Membri dell'array: numero minimo di 1 elemento. Numero massimo 5 elementi.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: sì

statusType

Per restituire enunciati che sono stati riconosciuti e gestiti, usa. `Detected` Per restituire enunciati che non sono stati riconosciuti, usa. `Missed`

Valori validi: `Detected | Missed`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "botName": "string",
   "utterances": [
      {
         "botVersion": "string",
         "utterances": [
```

```
            {
               "count": number,
               "distinctUsers": number,
               "firstUtteredDate": number,
               "lastUtteredDate": number,
               "utteranceString": "string"
            }
         ]
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

botName

Il nome del bot per il quale sono state restituite le informazioni sull'enunciato.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

utterances

Una serie di UtteranceList oggetti, ciascuno contenente un elenco di UtteranceData oggetti che descrivono gli enunciati elaborati dal bot. La risposta contiene un massimo di 100 `UtteranceData` oggetti per ogni versione. Amazon Lex restituisce gli enunciati più frequenti ricevuti dal bot negli ultimi 15 giorni.

Tipo: matrice di oggetti UtteranceList

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

ListTagsForResource

Servizio: Amazon Lex Model Building Service

Ottiene un elenco di tag associati alla risorsa specificata. Solo ai bot, agli alias dei bot e ai canali dei bot possono essere associati tag.

Sintassi della richiesta

```
GET /tags/resourceArn HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

resourceArn

L'Amazon Resource Name (ARN) della risorsa per cui ottenere un elenco di tag.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1011.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

tags

I tag associati a una risorsa.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET

- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PutBot

Servizio: Amazon Lex Model Building Service

Crea un bot conversazionale Amazon Lex o sostituisce un bot esistente. Quando crei o aggiorni un bot, ti viene richiesto solo di specificare un nome, una lingua e se il bot è rivolto a bambini di età inferiore ai 13 anni. Puoi usarlo per aggiungere intenti in un secondo momento o per rimuovere intenti da un bot esistente. Quando crei un bot con le informazioni minime, il bot viene creato o aggiornato ma Amazon Lex restituisce la rispostaFAILED. Puoi creare il bot dopo aver aggiunto uno o più intenti. Per ulteriori informazioni sui bot Amazon Lex, consultaAmazon Lex: come funziona: come funziona.

Se specifichi il nome di un bot esistente, i campi della richiesta sostituiscono i valori esistenti nella $LATEST versione del bot. Amazon Lex rimuove tutti i campi per i quali non vengono forniti valori nella richiesta, ad eccezione dei privacySettings campi idleTTLInSeconds e, che sono impostati sui valori predefiniti. Se non specifichi valori per i campi obbligatori, Amazon Lex genera un'eccezione.

Questa operazione richiede le autorizzazioni per l'operazione lex:PutBot. Per ulteriori informazioni, consulta Identity and Access Management per Amazon Lex.

Sintassi della richiesta

```
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"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del bot. Il nome non fa distinzione tra maiuscole e minuscole.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

abortStatement

Quando Amazon Lex non è in grado di comprendere l'input dell'utente nel contesto, tenta di richiamare le informazioni alcune volte. Successivamente, Amazon Lex invia il messaggio definito `abortStatement` all'utente e quindi annulla la conversazione. Per impostare il numero di tentativi, utilizza il `valueElicitationPrompt` campo per il tipo di slot.

Ad esempio, in un bot per ordinare pizze, Amazon Lex potrebbe chiedere a un utente «Che tipo di crosta preferisci?» Se la risposta dell'utente non è una delle risposte previste (ad esempio, «crosta sottile», «piatto profondo», ecc.), Amazon Lex tenta di ottenere una risposta corretta ancora qualche volta.

Ad esempio, in un'applicazione per ordinare pizze, `OrderPizza` potrebbe essere uno degli intenti. Questo intento potrebbe richiedere lo slot. `CrustType` Il `valueElicitationPrompt` campo viene specificato quando si crea lo `CrustType` slot.

Se è stato definito un intento di fallback, l'istruzione cancel non verrà inviata all'utente, ma verrà utilizzato l'intento di fallback. Per ulteriori informazioni, consulta AMAZON. FallbackIntent.

Tipo: oggetto Statement

Campo obbligatorio: no

checksum

Identifica una revisione specifica della `$LATEST` versione.

Quando crei un nuovo bot, lascia il `checksum` campo vuoto. Se si specifica un checksum si ottiene un'`BadRequestException`eccezione.

Quando vuoi aggiornare un bot, imposta il `checksum` campo sul checksum della revisione più recente della versione. `$LATEST` Se non specifichi il `checksum` campo o se il checksum non corrisponde alla `$LATEST` versione, ottieni un'eccezione. `PreconditionFailedException`

-Tipo: stringa

Campo obbligatorio: no

childDirected

Per ogni bot Amazon Lex creato con Amazon Lex Model Building Service, devi specificare se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al Children's Online Privacy Protection Act (COPPA) specificando `true` o `false` nel `childDirected` campo. Specificando `true` nel `childDirected` campo, confermi che l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al COPPA. Specificando `false` nel `childDirected` campo, confermi che l'uso di Amazon Lex non è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetti al COPPA. Non puoi specificare un valore predefinito per il `childDirected` campo che non riflette accuratamente se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzata o destinata, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetta al COPPA.

Se l'uso di Amazon Lex si riferisce a un sito Web, programma o altra applicazione rivolta, in tutto o in parte, a bambini di età inferiore ai 13 anni, devi ottenere il consenso genitoriale verificabile richiesto ai sensi del COPPA. Per informazioni sull'uso di Amazon Lex in relazione a siti Web, programmi o altre applicazioni rivolti o destinati, in tutto o in parte, a bambini di età inferiore ai 13 anni, consulta le domande frequenti su Amazon Lex.

Tipo: Booleano

Campo obbligatorio: sì

clarificationPrompt

Quando Amazon Lex non comprende l'intenzione dell'utente, utilizza questo messaggio per ottenere chiarimenti. Per specificare quante volte Amazon Lex deve ripetere la richiesta di chiarimento, utilizza il `maxAttempts` campo. Se Amazon Lex continua a non capire, invia il messaggio nel `abortStatement` campo.

Quando crei una richiesta di chiarimento, assicurati che suggerisca la risposta corretta dell'utente. Ad esempio, per un bot che ordina pizza e bevande, potresti creare questo messaggio di chiarimento: «Cosa vorresti fare? Puoi dire 'Ordina una pizza' o 'Ordina un drink. '»

Se hai definito un intento di riserva, questo verrà invocato se la richiesta di chiarimento viene ripetuta il numero di volte definito nel campo. `maxAttempts` Per ulteriori informazioni, consulta AMAZON. FallbackIntent.

Se non definisci una richiesta di chiarimento, in fase di esecuzione Amazon Lex restituirà un'eccezione 400 Bad Request in tre casi:

- Richiesta di follow-up: quando l'utente risponde a una richiesta di follow-up ma non ne fornisce l'intenzione. Ad esempio, in risposta a una richiesta di follow-up che dice «Vuoi qualcos'altro oggi?» l'utente dice «Sì». Amazon Lex restituirà un'eccezione 400 Bad Request perché non dispone di un messaggio di chiarimento da inviare all'utente per determinare l'intento.
- Funzione Lambda: quando si utilizza una funzione Lambda, si restituisce un tipo di dialogo. `ElicitIntent` Poiché Amazon Lex non dispone di una richiesta di chiarimento per ottenere un'intenzione dall'utente, restituisce un'eccezione 400 Bad Request.
- PutSession operazione: quando si utilizza l'`PutSession`operazione, si invia un `ElicitIntent` tipo di dialogo. Poiché Amazon Lex non dispone di una richiesta di chiarimento per ottenere un'intenzione dall'utente, restituisce un'eccezione 400 Bad Request.

Tipo: oggetto Prompt

Campo obbligatorio: no

createVersion

Se impostato su, viene creata `true` una nuova versione numerata del bot. È come chiamare l'`CreateBotVersion`operazione. Se non si specifica`createVersion`, l'impostazione predefinita è`false`.

Tipo: Booleano

Campo obbligatorio: no

description

Una descrizione del bot.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

detectSentiment

Se impostato su `true` utente, gli enunciati vengono inviati ad Amazon Comprehend per l'analisi del sentiment. Se non lo specifichi`detectSentiment`, l'impostazione predefinita è. `false`

Tipo: Booleano

Campo obbligatorio: no

enableModelImprovements

Impostato `true` per consentire l'accesso ai miglioramenti della comprensione del linguaggio naturale.

Quando si imposta il `enableModelImprovements` parametro su, `true` è possibile utilizzarlo per configurare i `nluIntentConfidenceThreshold` punteggi di confidenza. Per ulteriori informazioni, vedere Confidence Scores.

È possibile impostare il `enableModelImprovements` parametro solo in determinate regioni. Se imposti il parametro su`true`, il bot ha accesso a miglioramenti della precisione.

Le regioni in cui è possibile impostare il `enableModelImprovements` parametro `false` per la localizzazione en-US sono:

- Stati Uniti orientali (Virginia settentrionale) (us-east-1)
- Stati Uniti occidentali (Oregon) (us-west-2)
- Asia Pacifico (Sydney) (ap-southeast-2)
- UE (Irlanda): eu-west-1

In altre regioni e impostazioni locali, il `enableModelImprovements` parametro è impostato su come impostazione predefinita`true`. In queste regioni e impostazioni locali, l'impostazione del parametro to `false` genera un'eccezione. `ValidationException`

Tipo: Booleano

Campo obbligatorio: no

idleSessionTTLInSeconds

Il tempo massimo, in secondi, durante il quale Amazon Lex conserva i dati raccolti in una conversazione.

Una sessione di interazione con l'utente rimane attiva per il periodo di tempo specificato. Se durante questo periodo non si verifica alcuna conversazione, la sessione scade e Amazon Lex elimina tutti i dati forniti prima del timeout.

Ad esempio, supponiamo che un utente scelga l' OrderPizza intento, ma venga distratto a metà dell'ordine. Se l'utente non completa l'ordine entro il tempo specificato, Amazon Lex elimina le informazioni sullo slot raccolte e l'utente deve ricominciare da capo.

Se non includi l'`idleSessionTTLInSeconds`elemento in una richiesta di `PutBot` operazione, Amazon Lex utilizza il valore predefinito. Questo vale anche se la richiesta sostituisce un bot esistente.

Il valore predefinito è 300 secondi (5 minuti).

Tipo: integer

Intervallo valido: valore minimo pari a 60. Valore massimo pari a 86400.

Campo obbligatorio: no

intents

Un array di oggetti `Intent`. Ogni intento rappresenta un comando che un utente può esprimere. Ad esempio, un bot per ordinare pizze potrebbe supportare un OrderPizza intento. Per ulteriori informazioni, consulta Amazon Lex: come funziona: come funziona.

Tipo: matrice di oggetti Intent

Campo obbligatorio: no

locale

Specifica il locale di destinazione per il bot. Qualsiasi intento utilizzato nel bot deve essere compatibile con le impostazioni locali del bot.

Il valore predefinito è `en-US`.

-Tipo: stringa

Valori validi: `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`

Campo obbligatorio: sì

nluIntentConfidenceThreshold

Determina la soglia in cui Amazon Lex inserirà o entrambi quando restituisce intenti alternativi in una PostTextrisposta PostContento. `AMAZON.FallbackIntent` `AMAZON.KendraSearchIntent` `AMAZON.FallbackIntente` `AMAZON.KendraSearchIntent` vengono inseriti solo se sono configurati per il bot.

È necessario impostare il `enableModelImprovements` parametro per `true` utilizzare i punteggi di confidenza nelle seguenti regioni.

- Stati Uniti orientali (Virginia settentrionale) (us-east-1)
- Stati Uniti occidentali (Oregon) (us-west-2)
- Asia Pacifico (Sydney) (ap-southeast-2)
- UE (Irlanda): eu-west-1

In altre regioni, il `enableModelImprovements` parametro è impostato come `true` impostazione predefinita.

Ad esempio, supponiamo che un bot sia configurato con la soglia di confidenza di 0,80 e il. `AMAZON.FallbackIntent` Amazon Lex restituisce tre intenti alternativi con i seguenti punteggi di confidenza: intentA (0,70), IntentB (0,60), IntentC (0,50). La risposta dell'operazione sarebbe: `PostText`

- AMAZON. FallbackIntent
- a Tenta
- Intento B
- Intento C

Tipo: double

Intervallo valido: valore minimo di 0. Valore massimo di 1.

Campo obbligatorio: no

processBehavior

Se imposti l'`processBehavior`elemento su`BUILD`, Amazon Lex crea il bot in modo che possa essere eseguito. Se imposti l'elemento `SAVE` su Amazon Lex, il bot viene salvato, ma non lo crea.

Se non specifichi questo valore, il valore predefinito è`BUILD`.

-Tipo: stringa

Valori validi: `SAVE | BUILD`

Campo obbligatorio: no

tags

Un elenco di tag da aggiungere al bot. È possibile aggiungere tag solo quando si crea un bot, non è possibile utilizzare l'`PutBot`operazione per aggiornare i tag su un bot. Per aggiornare i tag, utilizza l'operazione `TagResource`.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Campo obbligatorio: no

voiceId

L'ID vocale Amazon Polly che desideri venga utilizzato da Amazon Lex per le interazioni vocali con l'utente. Le impostazioni locali configurate per la voce devono corrispondere alle impostazioni locali del bot. Per ulteriori informazioni, consulta Voices in Amazon Polly nella Amazon Polly Developer Guide.

-Tipo: stringa

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

abortStatement

Il messaggio che Amazon Lex utilizza per annullare una conversazione. Per ulteriori informazioni, consulta PutBot.

Tipo: oggetto Statement

checksum

Checksum del bot che hai creato.

- Tipo: stringa

childDirected

Per ogni bot Amazon Lex creato con Amazon Lex Model Building Service, devi specificare se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al Children's Online Privacy Protection Act (COPPA) specificando `true` o `false` nel `childDirected` campo. Specificando `true` nel `childDirected` campo, confermi che l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetto al COPPA. Specificando `false` nel `childDirected` campo, confermi che l'uso di Amazon Lex non è correlato a un sito Web, programma o altra applicazione indirizzato o destinato, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetti al COPPA. Non puoi specificare un valore predefinito per il `childDirected` campo che non riflette accuratamente se l'uso di Amazon Lex è correlato a un sito Web, programma o altra applicazione indirizzata o destinata, in tutto o in parte, a bambini di età inferiore ai 13 anni e soggetta al COPPA.

Se l'uso di Amazon Lex si riferisce a un sito Web, programma o altra applicazione rivolta, in tutto o in parte, a bambini di età inferiore ai 13 anni, devi ottenere il consenso genitoriale verificabile richiesto ai sensi del COPPA. Per informazioni sull'uso di Amazon Lex in relazione a siti Web, programmi o altre applicazioni rivolti o destinati, in tutto o in parte, a bambini di età inferiore ai 13 anni, consulta le domande frequenti su Amazon Lex.

Tipo: Booleano

clarificationPrompt

I prompt che Amazon Lex utilizza quando non comprende le intenzioni dell'utente. Per ulteriori informazioni, consulta PutBot.

Tipo: oggetto Prompt

createdDate

La data di creazione del bot.

Tipo: Timestamp

createVersion

`true`se è stata creata una nuova versione del bot. Se il `createVersion` campo non è stato specificato nella richiesta, il `createVersion` campo è impostato su false nella risposta.

Tipo: Booleano

description

Una descrizione del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

detectSentiment

`true`se il bot è configurato per inviare le espressioni degli utenti ad Amazon Comprehend per l'analisi del sentiment. Se il `detectSentiment` campo non è stato specificato nella richiesta, si `detectSentiment` trova `false` nella risposta.

Tipo: Booleano

enableModelImprovements

Indica se il bot utilizza miglioramenti di precisione. `true`indica che il bot sta utilizzando i miglioramenti, altrimenti`false`.

Tipo: Booleano

failureReason

In caso `status` `FAILED` affermativo, Amazon Lex fornisce il motivo per cui non è riuscita a creare il bot.

- Tipo: stringa

idleSessionTTLInSeconds

Il periodo massimo di conservazione dei dati raccolti in una conversazione da parte di Amazon Lex. Per ulteriori informazioni, consulta PutBot.

Tipo: integer

Intervallo valido: valore minimo pari a 60. Valore massimo pari a 86400.

intents

Un array di oggetti `Intent`. Per ulteriori informazioni, consulta PutBot.

Tipo: matrice di oggetti Intent

lastUpdatedDate

La data in cui il bot è stato aggiornato. Quando crei una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

locale

Il locale di destinazione per il bot.

-Tipo: stringa

Valori validi: `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

Il nome del bot.

-Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

Il punteggio che determina dove Amazon Lex inserisce o entrambi quando restituisce intenti alternativi in una risposta PostContento PostText. `AMAZON.FallbackIntent` `AMAZON.KendraSearchIntent` `AMAZON.FallbackIntent`viene inserito se il punteggio di confidenza a tutti gli effetti è inferiore a questo valore. `AMAZON.KendraSearchIntent`viene inserito solo se è configurato per il bot.

Tipo: double

Intervallo valido: valore minimo di 0. Valore massimo di 1.

status

Quando invii una richiesta per creare un bot con `processBehavior` set to`BUILD`, Amazon Lex imposta l'elemento di `status` risposta su`BUILDING`.

`READY_BASIC_TESTING`Nello stato, puoi testare il bot con input utente che corrispondono esattamente alle espressioni configurate per gli intenti e i valori del bot nei tipi di slot.

Se Amazon Lex non è in grado di creare il bot, Amazon Lex decide `status` di farlo`FAILED`. Amazon Lex restituisce il motivo dell'errore nell'elemento di `failureReason` risposta.

Quando lo `processBehavior` impostiSAVE, Amazon Lex imposta il codice di stato su`NOT BUILT`.

Quando il bot è nello `READY` stato, puoi testarlo e pubblicarlo.

- Tipo: stringa

Valori validi: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

tags

Un elenco di tag associati al bot.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

version

La versione del bot. Per un nuovo bot, la versione è sempre`$LATEST`.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

voiceId

L'ID vocale Amazon Polly utilizzato da Amazon Lex per l'interazione vocale con l'utente. Per ulteriori informazioni, consulta PutBot.

- Tipo: stringa

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

PreconditionFailedException

Il checksum della risorsa che stai cercando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2

- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PutBotAlias

Servizio: Amazon Lex Model Building Service

Crea un alias per la versione specificata del bot o sostituisce un alias per il bot specificato. Per cambiare la versione del bot a cui punta l'alias, sostituisci l'alias. Per ulteriori informazioni sugli alias, consultare Funzione Versioni multiple e alias.

Questa operazione richiede le autorizzazioni per l'operazione `lex:PutBotAlias`.

Sintassi della richiesta

```
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"
      }
   ]
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botName

Il nome del bot.

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

name

Nome dell'alias. Il nome non fa distinzione tra maiuscole e minuscole.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

botVersion

La versione del bot.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: sì

checksum

Identifica una revisione specifica della `$LATEST` versione.

Quando crei un nuovo alias bot, lascia il `checksum` campo vuoto. Se si specifica un checksum si ottiene un'`BadRequestException`eccezione.

Quando desideri aggiornare un alias bot, imposta il `checksum` campo sul checksum della revisione più recente della versione. `$LATEST` Se non specifichi il `checksum`

campo o se il checksum non corrisponde alla `$LATEST` versione, ottieni un'eccezione. `PreconditionFailedException`

- Tipo: stringa

Campo obbligatorio: no

conversationLogs

Impostazioni per i registri delle conversazioni per l'alias.

Tipo: oggetto ConversationLogsRequest

Campo obbligatorio: no

description

Una descrizione degli alias.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

tags

Un elenco di tag da aggiungere all'alias del bot. È possibile aggiungere tag solo quando si crea un alias, non è possibile utilizzare l'`PutBotAlias`operazione per aggiornare i tag su un alias bot. Per aggiornare i tag, utilizza l'operazione `TagResource`.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Campo obbligatorio: no

Sintassi della risposta

```
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"
      }
   ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

botName

Il nome del bot a cui punta l'alias.

Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

botVersion

La versione del bot a cui punta l'alias.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

checksum

Il checksum per la versione corrente dell'alias.

- Tipo: stringa

conversationLogs

Le impostazioni che determinano in che modo Amazon Lex utilizza i log delle conversazioni per l'alias.

Tipo: oggetto ConversationLogsResponse

createdDate

La data di creazione dell'alias del bot.

Tipo: Timestamp

description

Una descrizione degli alias.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

lastUpdatedDate

La data in cui l'alias del bot è stato aggiornato. Quando crei una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Nome dell'alias.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

tags

Un elenco di tag associati a un bot.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

PreconditionFailedException

Il checksum della risorsa che stai tentando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PutIntent

Servizio: Amazon Lex Model Building Service

Crea un intento o sostituisce un intento esistente.

Per definire l'interazione tra l'utente e il bot, si utilizzano uno o più intenti. Per un bot che ordina pizze, ad esempio, dovresti creare un `OrderPizza` intento.

Per creare un intento o sostituire un intento esistente, devi fornire quanto segue:

- Nome dell'intento. Ad esempio, `OrderPizza`.
- Esempi di enunciati. Ad esempio, «Posso ordinare una pizza, per favore». e «Voglio ordinare una pizza».
- Informazioni da raccogliere. Specificate i tipi di slot per le informazioni che il bot richiederà all'utente. È possibile specificare tipi di slot standard, come una data o un'ora, o tipi di slot personalizzati come la dimensione e la crosta di una pizza.
- Come verrà realizzato l'intento. È possibile fornire una funzione Lambda o configurare l'intento di restituire le informazioni sull'intento all'applicazione client. Se usi una funzione Lambda, quando tutte le informazioni sull'intento sono disponibili, Amazon Lex richiama la tua funzione Lambda. Se configuri l'intenzione di restituire le informazioni sull'intento all'applicazione client.

È possibile specificare altre informazioni opzionali nella richiesta, ad esempio:

- Una richiesta di conferma per chiedere all'utente di confermare un'intenzione. Ad esempio, «Devo ordinare la tua pizza?»
- Una dichiarazione conclusiva da inviare all'utente dopo che l'intento è stato raggiunto. Ad esempio, «Ho ordinato la tua pizza».
- Una richiesta di follow-up che richiede all'utente attività aggiuntive. Ad esempio, chiedendo «Vuoi ordinare qualcosa da bere con la tua pizza?»

Se specifichi un nome di intento esistente per aggiornare l'intento, Amazon Lex sostituisce i valori nella `$LATEST` versione dell'intento con i valori nella richiesta. Amazon Lex rimuove i campi che non fornisci nella richiesta. Se non specifichi i campi obbligatori, Amazon Lex genera un'eccezione. Quando aggiorni la `$LATEST` versione di un intento, il `status` campo di qualsiasi bot che utilizza la `$LATEST` versione dell'intento viene impostato su. `NOT_BUILT`

Per ulteriori informazioni, consulta Amazon Lex: come funziona: come funziona.

Questa operazione richiede le autorizzazioni per l'operazione `lex:PutIntent`.

Sintassi della richiesta

```
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"
         }
      }
   ]
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome dell'intento. Il nome non fa distinzione tra maiuscole e minuscole.

Il nome non può corrispondere a un nome di intento incorporato o a un nome di intento integrato con «AMAZON». rimosso. Ad esempio, poiché esiste un intento integrato chiamato`AMAZON.HelpIntent`, non è possibile creare un intento personalizzato chiamato. `HelpIntent`

Per un elenco di intenti incorporati, consulta la sezione relativa agli intenti incorporati standard in Alexa Skills Kit.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

checksum

Identifica una revisione specifica della versione. `$LATEST`

Quando crei un nuovo intento, lascia il `checksum` campo vuoto. Se si specifica un checksum si ottiene un'`BadRequestException`eccezione.

Quando desideri aggiornare un intento, imposta il `checksum` campo sul checksum della revisione più recente della versione. `$LATEST` Se non specifichi il `checksum` campo o se il checksum non corrisponde alla `$LATEST` versione, ottieni un'eccezione. `PreconditionFailedException`

-Tipo: stringa

Campo obbligatorio: no

conclusionStatement

L'istruzione che desideri che Amazon Lex trasmetta all'utente dopo che l'intento è stato soddisfatto con successo dalla funzione Lambda.

Questo elemento è rilevante solo se si fornisce una funzione Lambda in. `fulfillmentActivity` Se restituisci l'intento all'applicazione client, non puoi specificare questo elemento.

> **Note**
>
> I `followUpPrompt` e si `conclusionStatement` escludono a vicenda. È possibile specificarne solo uno.

Tipo: oggetto Statement

Campo obbligatorio: no

confirmationPrompt

Richiede all'utente di confermare l'intento. Questa domanda dovrebbe avere una risposta sì o no.

Amazon Lex utilizza questo prompt per garantire che l'utente riconosca che l'intento è pronto per l'adempimento. Ad esempio, con l'`OrderPizza`intento, potresti voler confermare che l'ordine è corretto prima di effettuarlo. Per altri scopi, ad esempio intenti a rispondere semplicemente alle domande degli utenti, potrebbe non essere necessario chiedere conferma all'utente prima di fornire le informazioni.

> **Note**
>
> È necessario fornire sia il che il `rejectionStatement`confirmationPrompt, oppure nessuno dei due.

Tipo: oggetto Prompt

Campo obbligatorio: no

createVersion

Se impostato su `true` una nuova versione numerata dell'intento, viene creata. È lo stesso che chiamare l'`CreateIntentVersion`operazione. Se non si specifica`createVersion`, l'impostazione predefinita è`false`.

Tipo: Booleano

Campo obbligatorio: no

description

Una descrizione dell'intento.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

dialogCodeHook

Specifica una funzione Lambda da richiamare per ogni input dell'utente. È possibile richiamare questa funzione Lambda per personalizzare l'interazione dell'utente.

Ad esempio, supponiamo che il bot determini che l'utente è John. La tua funzione Lambda potrebbe recuperare le informazioni di John da un database di backend e precompilare alcuni valori. Ad esempio, se scopri che John è intollerante al glutine, puoi impostare lo slot di intenti corrispondente, su true. `GlutenIntolerant` Potresti trovare il numero di telefono di John e impostare l'attributo di sessione corrispondente.

Tipo: oggetto CodeHook

Campo obbligatorio: no

followUpPrompt

Amazon Lex utilizza questo prompt per sollecitare attività aggiuntive dopo aver soddisfatto un intento. Ad esempio, una volta raggiunto l'`OrderPizza`intento, potresti richiedere all'utente di ordinare un drink.

L'azione intrapresa da Amazon Lex dipende dalla risposta dell'utente, come segue:

- Se l'utente dice «Sì», risponde con la richiesta di chiarimento configurata per il bot.
- se l'utente dice «Sì» e continua con un'espressione che fa scattare un intento, avvia una conversazione sull'intento.
- Se l'utente dice «No», risponde con la dichiarazione di rifiuto configurata per la richiesta di follow-up.
- Se non riconosce l'enunciato, ripete nuovamente la richiesta di follow-up.

Il `followUpPrompt` campo e il campo si escludono a `conclusionStatement` vicenda. È possibile specificarne solo uno.

Tipo: oggetto FollowUpPrompt

Campo obbligatorio: no

fulfillmentActivity

Obbligatorio. Descrive come viene soddisfatto l'intento. Ad esempio, dopo che un utente ha fornito tutte le informazioni per un ordine di pizza, `fulfillmentActivity` definisce in che modo il bot effettua un ordine presso una pizzeria locale.

Puoi configurare Amazon Lex per restituire tutte le informazioni sull'intento all'applicazione client o indirizzarla a richiamare una funzione Lambda in grado di elaborare l'intento (ad esempio, effettuare un ordine presso una pizzeria).

Tipo: oggetto FulfillmentActivity

Campo obbligatorio: no

inputContexts

Una serie di `InputContext` oggetti che elenca i contesti che devono essere attivi affinché Amazon Lex possa scegliere l'intento in una conversazione con l'utente.

Tipo: matrice di oggetti InputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo 5 elementi.

Campo obbligatorio: no

kendraConfiguration

Informazioni di configurazione necessarie per utilizzare l'`AMAZON.KendraSearchIntent`intento di connessione a un indice Amazon Kendra. Per ulteriori informazioni, consulta AMAZON. KendraSearchIntent.

Tipo: oggetto KendraConfiguration

Campo obbligatorio: no

outputContexts

Una serie di `OutputContext` oggetti che elenca i contesti che l'intento attiva quando l'intento viene soddisfatto.

Tipo: matrice di oggetti OutputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

Campo obbligatorio: no

parentIntentSignature

Un identificativo univoco dell'intento integrato su cui basare questo intento. Per trovare la firma di un intento, consulta Standard Built-in Intents nell'Alexa Skills Kit.

-Tipo: stringa

Campo obbligatorio: no

rejectionStatement

Quando l'utente risponde «no» alla domanda definita in`confirmationPrompt`, Amazon Lex risponde con questa dichiarazione per confermare che l'intento è stato annullato.

> **Note**
>
> Devi fornire sia il che il, oppure nessuno `rejectionStatement` dei `confirmationPrompt` due.

Tipo: oggetto Statement

Campo obbligatorio: no

sampleUtterances

Una serie di enunciati (stringhe) che un utente potrebbe pronunciare per segnalare l'intento. Ad esempio, «I want {PizzaSize} pizza», «Order {Quantity} {PizzaSize} pizze».

In ogni enunciato, il nome di uno slot è racchiuso tra parentesi graffe.

Tipo: matrice di stringhe

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 1500 articoli.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 200.

Campo obbligatorio: no

slots

Una serie di slot di intenti. In fase di esecuzione, Amazon Lex richiede all'utente i valori degli slot richiesti utilizzando i prompt definiti negli slot. Per ulteriori informazioni, consulta Amazon Lex: come funziona: come funziona.

Tipo: matrice di oggetti Slot

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 100 elementi.

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

checksum

Checksum della `$LATEST` versione dell'intento creata o aggiornata.

Tipo: stringa

conclusionStatement

Dopo che la funzione Lambda specificata nell'`fulfillmentActivity`intento soddisfa l'intento, Amazon Lex trasmette questa dichiarazione all'utente.

Tipo: oggetto Statement

confirmationPrompt

Se definito nell'intento, Amazon Lex richiede all'utente di confermare l'intento prima di realizzarlo.

Tipo: oggetto Prompt

createdDate

La data in cui è stato creato l'intento.

Tipo: Timestamp

createVersion

`True`se è stata creata una nuova versione dell'intento. Se il `createVersion` campo non è stato specificato nella richiesta, il `createVersion` campo è impostato su false nella risposta.

Tipo: Booleano

description

Una descrizione dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

dialogCodeHook

Se definita nell'intento, Amazon Lex richiama questa funzione Lambda per ogni input dell'utente.

Tipo: oggetto CodeHook

followUpPrompt

Se definito nell'intento, Amazon Lex utilizza questo prompt per sollecitare ulteriori attività dell'utente dopo il raggiungimento dell'intento.

Tipo: oggetto FollowUpPrompt

fulfillmentActivity

Se definita nell'intento, Amazon Lex richiama questa funzione Lambda per soddisfare l'intento dopo che l'utente ha fornito tutte le informazioni richieste dall'intento.

Tipo: oggetto FulfillmentActivity

inputContexts

Una serie di `InputContext` oggetti che elenca i contesti che devono essere attivi affinché Amazon Lex possa scegliere l'intento in una conversazione con l'utente.

Tipo: matrice di oggetti InputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo 5 elementi.

kendraConfiguration

Informazioni di configurazione, se presenti, necessarie per connettersi a un indice Amazon Kendra e utilizzare l'intento. `AMAZON.KendraSearchIntent`

Tipo: oggetto KendraConfiguration

lastUpdatedDate

La data in cui l'intento è stato aggiornato. Quando crei una risorsa, la data di creazione e le date dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Il nome dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

outputContexts

Una matrice di `OutputContext` oggetti che elenca i contesti che l'intento attiva quando l'intento viene soddisfatto.

Tipo: matrice di oggetti OutputContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

parentIntentSignature

Un identificatore univoco per l'intento incorporato su cui si basa questo intento.

- Tipo: stringa

rejectionStatement

Se l'utente risponde «no» alla domanda definita in `confirmationPrompt` Amazon Lex, risponde con questa dichiarazione per confermare che l'intento è stato annullato.

Tipo: oggetto Statement

sampleUtterances

Una serie di espressioni di esempio configurate per l'intento.

Tipo: matrice di stringhe

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 1500 elementi.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 200.

slots

Una serie di slot di intento configurati per l'intento.

Tipo: matrice di oggetti Slot

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 100 elementi.

version

La versione dell'intento. Per un nuovo intento, la versione è sempre. `$LATEST`

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

PreconditionFailedException

Il checksum della risorsa che stai tentando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PutSlotType

Servizio: Amazon Lex Model Building Service

Crea un tipo di slot personalizzato o sostituisce un tipo di slot personalizzato esistente.

Per creare un tipo di slot personalizzato, specificate un nome per il tipo di slot e un set di valori di enumerazione, che sono i valori che uno slot di questo tipo può assumere. Per ulteriori informazioni, consulta Amazon Lex: come funziona: come funziona.

Se specificate il nome di un tipo di slot esistente, i campi della richiesta sostituiscono i valori esistenti nella `$LATEST` versione del tipo di slot. Amazon Lex rimuove i campi che non fornisci nella richiesta. Se non specifichi i campi obbligatori, Amazon Lex genera un'eccezione. Quando aggiorni la `$LATEST` versione di un tipo di slot, se un bot utilizza la `$LATEST` versione di un intento che contiene il tipo di slot, il `status` campo del bot viene impostato su. `NOT_BUILT`

Questa operazione richiede le autorizzazioni per l'operazione `lex:PutSlotType`.

Sintassi della richiesta

```
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"
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

name

Il nome del tipo di slot. Il nome non fa distinzione tra maiuscole e minuscole.

Il nome non può corrispondere al nome di un tipo di slot integrato o a un nome di tipo di slot integrato con «AMAZON». rimosso. Ad esempio, poiché esiste un tipo di slot integrato chiamato`AMAZON.DATE`, non è possibile creare un tipo di slot personalizzato chiamato`DATE`.

Per un elenco dei tipi di slot integrati, consulta Slot Type Reference nell'Alexa Skills Kit.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

checksum

Identifica una revisione specifica della versione. `$LATEST`

Quando crei un nuovo tipo di slot, lascia il `checksum` campo vuoto. Se si specifica un checksum si ottiene un'`BadRequestException`eccezione.

Quando desideri aggiornare un tipo di slot, imposta il `checksum` campo sul checksum della revisione più recente della versione. `$LATEST` Se non specifichi il `checksum` campo o se il checksum non corrisponde alla `$LATEST` versione, ottieni un'eccezione. `PreconditionFailedException`

- Tipo: stringa

Campo obbligatorio: no

createVersion

Se impostato su `true` una nuova versione numerata dello slot, viene creato un tipo di slot. Equivale a chiamare l'`CreateSlotTypeVersion`operazione. Se non si specifica`createVersion`, l'impostazione predefinita è`false`.

Tipo: Booleano

Campo obbligatorio: no

description

Una descrizione del tipo di slot.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

enumerationValues

Un elenco di `EnumerationValue` oggetti che definisce i valori che il tipo di slot può assumere. Ogni valore può avere un elenco di `synonyms` valori aggiuntivi che aiutano ad addestrare il modello di apprendimento automatico sui valori che risolve per uno slot.

Un tipo di slot per espressioni regolari non richiede valori di enumerazione. Tutti gli altri tipi di slot richiedono un elenco di valori di enumerazione.

Quando Amazon Lex risolve un valore di slot, genera un elenco di risoluzione che contiene fino a cinque valori possibili per lo slot. Se si utilizza una funzione Lambda, questo elenco di risoluzioni viene passato alla funzione. Se non si utilizza una funzione Lambda, è possibile scegliere di restituire il valore immesso dall'utente o il primo valore nell'elenco delle risoluzioni come valore dello slot. Il `valueSelectionStrategy` campo indica l'opzione da utilizzare.

Tipo: matrice di oggetti EnumerationValue

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10000 articoli.

Campo obbligatorio: no

parentSlotTypeSignature

Il tipo di slot integrato utilizzato come elemento principale del tipo di slot. Quando si definisce un tipo di slot principale, il nuovo tipo di slot presenta la stessa configurazione dello slot principale.

Solo `AMAZON.AlphaNumeric`è supportata.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Campo obbligatorio: no

slotTypeConfigurations

Informazioni di configurazione che estendono il tipo di slot integrato principale. La configurazione viene aggiunta alle impostazioni per il tipo di slot principale.

Tipo: matrice di oggetti SlotTypeConfiguration

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

Campo obbligatorio: no

valueSelectionStrategy

Determina la strategia di risoluzione degli slot utilizzata da Amazon Lex per restituire i valori del tipo di slot. Il campo può essere impostato su uno dei seguenti valori:

- `ORIGINAL_VALUE`- Restituisce il valore inserito dall'utente, se il valore utente è simile al valore dello slot.
- `TOP_RESOLUTION`- Se esiste un elenco di risoluzioni per lo slot, restituisce il primo valore nell'elenco delle risoluzioni come valore del tipo di slot. Se non è presente un elenco di risoluzioni, viene restituito null.

Se non si specifica il`valueSelectionStrategy`, l'impostazione predefinita è`ORIGINAL_VALUE`.

- Tipo: stringa

Valori validi: `ORIGINAL_VALUE | TOP_RESOLUTION`

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

checksum

Checksum della $LATEST versione del tipo di slot.

- Tipo: stringa

createdDate

La data di creazione del tipo di slot.

Tipo: Timestamp

createVersion

`True`se è stata creata una nuova versione del tipo di slot. Se il `createVersion` campo non è stato specificato nella richiesta, il `createVersion` campo è impostato su false nella risposta.

Tipo: Booleano

description

Una descrizione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

enumerationValues

Un elenco di `EnumerationValue` oggetti che definisce i valori che il tipo di slot può assumere.

Tipo: matrice di oggetti EnumerationValue

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10000 elementi.

lastUpdatedDate

La data in cui il tipo di slot è stato aggiornato. Quando si crea un tipo di slot, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

name

Il nome del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Il tipo di slot integrato utilizzato come elemento principale del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informazioni di configurazione che estendono il tipo di slot integrato principale.

Tipo: matrice di oggetti SlotTypeConfiguration

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

valueSelectionStrategy

La strategia di risoluzione degli slot utilizzata da Amazon Lex per determinare il valore dello slot. Per ulteriori informazioni, consulta PutSlotType.

- Tipo: stringa

Valori validi: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

La versione del tipo di slot. Per un nuovo tipo di slot, la versione è sempre`$LATEST`.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Errori

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

PreconditionFailedException

Il checksum della risorsa che stai cercando di modificare non corrisponde al checksum della richiesta. Controlla il checksum della risorsa e riprova.

Codice di stato HTTP: 412

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

StartImport

Servizio: Amazon Lex Model Building Service

Avvia un'attività per importare una risorsa in Amazon Lex.

Sintassi della richiesta

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
   "mergeStrategy": "string",
   "payload": blob,
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parametri della richiesta URI:

La richiesta non utilizza parametri URI.

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

mergeStrategy

Speciifica l'azione che l'`StartImport`operazione deve intraprendere quando esiste una risorsa con lo stesso nome.

- FAIL_ON_CONFLICT - L'operazione di importazione viene interrotta al primo conflitto tra una risorsa nel file di importazione e una risorsa esistente. Il nome della risorsa che causa il conflitto si trova nel `failureReason` campo della risposta all'operazione. `GetImport`

  OVERWRITE_LATEST - L'operazione di importazione procede anche in caso di conflitto con una risorsa esistente. La versione $LASTEST della risorsa esistente viene sovrascritta con i dati del file di importazione.

- Tipo: stringa

Valori validi: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

Campo obbligatorio: sì

payload

Un archivio zip in formato binario. L'archivio deve contenere un file, un file JSON contenente la risorsa da importare. La risorsa deve corrispondere al tipo specificato nel `resourceType` campo.

Tipo: oggetto dati binari con codifica Base64

Campo obbligatorio: sì

resourceType

Speciifica il tipo di risorsa da esportare. Ogni risorsa esporta anche tutte le risorse da cui dipende.

- Un bot esporta intenti dipendenti.
- Un intento esporta i tipi di slot dipendenti.

- Tipo: stringa

Valori validi: `BOT | INTENT | SLOT_TYPE`

Campo obbligatorio: sì

tags

Un elenco di tag da aggiungere al bot importato. Puoi aggiungere tag solo quando importi un bot, non puoi aggiungere tag a un intento o a un tipo di slot.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Campo obbligatorio: no

Sintassi della risposta

```
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"
       }
    ]
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 201.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

createdDate

Un timestamp per la data e l'ora in cui è stato richiesto il processo di importazione.

Tipo: Timestamp

importId

L'identificatore per il processo di importazione specifico.

-Tipo: stringa

importStatus

Lo stato del processo di importazione. Se lo stato è`FAILED`, è possibile ottenere il motivo dell'errore utilizzando l'`GetImport`operazione.

-Tipo: stringa

Valori validi: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

L'azione da intraprendere in caso di conflitto di fusione.

-Tipo: stringa

Valori validi: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

Il nome assegnato al processo di importazione.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `[a-zA-Z_]+`

resourceType

Il tipo di risorsa da importare.

Tipo: stringa

Valori validi: `BOT | INTENT | SLOT_TYPE`

tags

Un elenco di tag aggiunti al bot importato.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

StartMigration

Servizio: Amazon Lex Model Building Service

Inizia la migrazione di un bot da Amazon Lex V1 ad Amazon Lex V2. Esegui la migrazione del bot quando desideri sfruttare le nuove funzionalità di Amazon Lex V2.

Per ulteriori informazioni, consulta Migrazione di un bot nella guida per sviluppatori di Amazon Lex.

Sintassi della richiesta

```
POST /migrations HTTP/1.1
Content-type: application/json

{
   "migrationStrategy": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotName": "string",
   "v2BotRole": "string"
}
```

Parametri della richiesta URI:

La richiesta non utilizza parametri URI.

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

migrationStrategy

La strategia utilizzata per condurre la migrazione.

- `CREATE_NEW`- Crea un nuovo bot Amazon Lex V2 e migra il bot Amazon Lex V1 al nuovo bot.
- `UPDATE_EXISTING`- Sovrascrive i metadati del bot di Amazon Lex V2 esistenti e le impostazioni locali da migrare. Non modifica le altre impostazioni locali nel bot Amazon Lex V2. Se la locale non esiste, viene creata una nuova locale nel bot Amazon Lex V2.

➥Tipo: stringa

Valori validi: `CREATE_NEW | UPDATE_EXISTING`

Campo obbligatorio: sì

v1BotName

Il nome del bot Amazon Lex V1 che stai migrando ad Amazon Lex V2.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

v1BotVersion

La versione del bot per la migrazione ad Amazon Lex V2. Puoi migrare la `$LATEST` versione così come qualsiasi versione numerata.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: sì

v2BotName

Il nome del bot Amazon Lex V2 su cui stai migrando il bot Amazon Lex V1.

- Se il bot Amazon Lex V2 non esiste, devi utilizzare la strategia di `CREATE_NEW` migrazione.
- Se il bot Amazon Lex V2 esiste, è necessario utilizzare la strategia di `UPDATE_EXISTING` migrazione per modificare i contenuti del bot Amazon Lex V2.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([0-9a-zA-Z][_-]?)+$`

Campo obbligatorio: sì

v2BotRole

Il ruolo IAM utilizzato da Amazon Lex per eseguire il bot Amazon Lex V2.

- Tipo: stringa

  Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

  Modello: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

  Campo obbligatorio: sì

Sintassi della risposta

```
HTTP/1.1 202
Content-type: application/json

{
   "migrationId": "string",
   "migrationStrategy": "string",
   "migrationTimestamp": number,
   "v1BotLocale": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotId": "string",
   "v2BotRole": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 202.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

migrationId

L'identificatore univoco assegnato da Amazon Lex alla migrazione.

- Tipo: stringa

  Vincoli di lunghezza: lunghezza fissa di 10.

  Modello: `^[0-9a-zA-Z]+$`

migrationStrategy

La strategia utilizzata per condurre la migrazione.

- Tipo: stringa

Valori validi: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

La data e l'ora di inizio della migrazione.

Tipo: Timestamp

v1BotLocale

Le impostazioni locali utilizzate per il bot Amazon Lex V1.

- Tipo: stringa

Valori validi: `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

Il nome del bot Amazon Lex V1 che stai migrando ad Amazon Lex V2.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

v1BotVersion

La versione del bot per la migrazione ad Amazon Lex V2.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

v2BotId

L'identificatore univoco per il bot Amazon Lex V2.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 10.

Modello: `^[0-9a-zA-Z]+$`

v2BotRole

Il ruolo IAM utilizzato da Amazon Lex per eseguire il bot Amazon Lex V2.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Errori

AccessDeniedException

Il tuo utente o ruolo IAM non dispone dell'autorizzazione per chiamare le API Amazon Lex V2 necessarie per migrare il bot.

Codice di stato HTTP: 403

BadRequestException

La richiesta non è ben formulata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

TagResource

Servizio: Amazon Lex Model Building Service

Aggiunge i tag specificati alla risorsa specificata. Se esiste già una chiave di tag, il valore esistente viene sostituito con il nuovo valore.

Sintassi della richiesta

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

resourceArn

L'Amazon Resource Name (ARN) del bot, dell'alias del bot o del canale bot da taggare.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1011.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

tags

Un elenco di chiavi di tag da aggiungere alla risorsa. Se esiste già una chiave di tag, il valore esistente viene sostituito con il nuovo valore.

Tipo: matrice di oggetti Tag

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Campo obbligatorio: sì

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

UntagResource

Servizio: Amazon Lex Model Building Service

Rimuove i tag da un bot, da un alias bot o da un canale bot.

Sintassi della richiesta

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

resourceArn

L'Amazon Resource Name (ARN) della risorsa da cui rimuovere i tag.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1011.

Campo obbligatorio: sì

tagKeys

Un elenco di chiavi di tag da rimuovere dalla risorsa. Se una chiave di tag non esiste sulla risorsa, viene ignorata.

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 200 elementi.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 128 caratteri.

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 204
```

Elementi di risposta

Se l'operazione riesce, il servizio invia una risposta HTTP 204 con un corpo HTTP vuoto.

Errori

BadRequestException

La richiesta non è ben formata. Ad esempio, un valore non è valido o manca un campo obbligatorio. Controlla i valori del campo e riprova.

Codice di stato HTTP: 400

ConflictException

Si è verificato un conflitto nell'elaborazione della richiesta. Riprova la richiesta.

Codice di stato HTTP: 409

InternalFailureException

Si è verificato un errore interno di Amazon Lex. Riprova la richiesta.

Codice di stato HTTP: 500

LimitExceededException

La richiesta ha superato il limite. Riprova la richiesta.

Codice di stato HTTP: 429

NotFoundException

La risorsa specificata nella richiesta non è stata trovata. Controlla la risorsa e riprova.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2

- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

Servizio runtime Amazon Lex

Amazon Lex Runtime Service supporta le seguenti operazioni:

- DeleteSession
- GetSession
- PostContent
- PostText
- PutSession

DeleteSession

Servizio: Amazon Lex Runtime Service

Rimuove le informazioni di sessione per un bot, alias e ID utente specificato.

Sintassi della richiesta

```
DELETE /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botAlias

L'alias in uso per il bot che contiene i dati della sessione.

Campo obbligatorio: sì

botName

Il nome del bot che contiene i dati della sessione.

Campo obbligatorio: sì

userId

L'identificatore dell'utente associato ai dati della sessione.

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Modello: `[0-9a-zA-Z._:-]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "botAlias": "string",
   "botName": "string",
   "sessionId": "string",
   "userId": "string"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

botAlias

L'alias in uso per il bot associato ai dati della sessione.

- Tipo: stringa

botName

Il nome del bot associato ai dati della sessione.

- Tipo: stringa

sessionId

L'identificatore univoco della sessione.

- Tipo: stringa

userId

L'ID dell'utente dell'applicazione client.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Modello: `[0-9a-zA-Z._:-]+`

Errori

BadRequestException

La convalida della richiesta non è riuscita, non è presente alcun messaggio utilizzabile nel contesto oppure la compilazione del bot non è riuscita, è ancora in corso o contiene modifiche non integrate.

Codice di stato HTTP: 400

ConflictException

Due client utilizzano lo stesso account AWS, lo stesso bot Amazon Lex e lo stesso ID utente.

Codice di stato HTTP: 409

InternalFailureException

Errore interno del servizio. Riprova la chiamata.

Codice di stato HTTP: 500

LimitExceededException

È stato superato un limite.

Codice di stato HTTP: 429

NotFoundException

La risorsa (ad esempio il bot Amazon Lex o un alias) a cui si fa riferimento non è stata trovata.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2

- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

GetSession

Servizio: Amazon Lex Runtime Service

Restituisce le informazioni di sessione per un bot, alias e ID utente specificato.

Sintassi della richiesta

```
GET /bot/botName/alias/botAlias/user/userId/session/?
checkpointLabelFilter=checkpointLabelFilter HTTP/1.1
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botAlias

L'alias in uso per il bot che contiene i dati della sessione.

Campo obbligatorio: sì

botName

Il nome del bot che contiene i dati della sessione.

Campo obbligatorio: sì

checkpointLabelFilter

Una stringa usata per filtrare gli intenti restituiti nella `recentIntentSummaryView` struttura.

Quando si specifica un filtro, vengono restituiti solo gli intenti il cui `checkpointLabel` campo è impostato su quella stringa.

Limitazioni di lunghezza: lunghezza minima di 1. Lunghezza massima di 255.

Modello: `[a-zA-Z0-9-]+`

userId

L'ID dell'utente dell'applicazione client. Amazon Lex lo utilizza per identificare la conversazione di un utente con il tuo bot.

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Modello: `[0-9a-zA-Z._:-]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta non ha un corpo della richiesta.

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

activeContexts

Un elenco di contesti attivi per la sessione. Un contesto può essere impostato quando un intento viene soddisfatto o chiamando l'operazione `PostContentPostText`, o. `PutSession`

È possibile utilizzare un contesto per controllare gli intenti che possono far seguito a un intento o per modificare il funzionamento dell'applicazione.

Tipo: matrice di oggetti ActiveContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 20 elementi.

dialogAction

Descrive lo stato attuale del bot.

Tipo: oggetto DialogAction

recentIntentSummaryView

Una serie di informazioni sugli intenti utilizzati nella sessione. L'array può contenere un massimo di tre riepiloghi. Se nella sessione vengono utilizzati più di tre intenti, l'`recentIntentSummaryView`operazione contiene informazioni sugli ultimi tre intenti utilizzati.

Se impostate il `checkpointLabelFilter` parametro nella richiesta, l'array contiene solo gli intenti con l'etichetta specificata.

Tipo: matrice di oggetti IntentSummary

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 3 elementi.

sessionAttributes

Mappa delle coppie chiave/valore che rappresentano le informazioni di contesto specifiche della sessione. Contiene informazioni sull'applicazione trasferite tra Amazon Lex e un'applicazione client.

Tipo: mappatura stringa a stringa

sessionId

Un identificatore univoco per la sessione.

- Tipo: stringa

Errori

BadRequestException

La convalida della richiesta non è riuscita, non è presente alcun messaggio utilizzabile nel contesto o la compilazione del bot non è riuscita, è ancora in corso o contiene modifiche non integrate.

Codice di stato HTTP: 400

InternalFailureException

Errore interno del servizio. Riprova la chiamata.

Codice di stato HTTP: 500

LimitExceededException

È stato superato un limite.

Codice di stato HTTP: 429

NotFoundException

La risorsa (ad esempio il bot Amazon Lex o un alias) a cui si fa riferimento non è stata trovata.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PostContent

Servizio: Amazon Lex Runtime Service

Invia l'input dell'utente (testo o parlato) ad Amazon Lex. I client utilizzano questa API per inviare richieste di testo e audio ad Amazon Lex in fase di esecuzione. Amazon Lex interpreta l'input dell'utente utilizzando il modello di apprendimento automatico creato per il bot.

L'`PostContent`operazione supporta l'ingresso audio a 8 kHz e 16 kHz. È possibile utilizzare l'audio a 8 kHz per ottenere una maggiore precisione di riconoscimento vocale nelle applicazioni audio telefoniche.

In risposta, Amazon Lex restituisce il messaggio successivo da trasmettere all'utente. Considera i seguenti messaggi di esempio:

- Per un input dell'utente «Vorrei una pizza», Amazon Lex potrebbe restituire una risposta con un messaggio che richiama i dati relativi allo slot (ad esempio,`PizzaSize`): «Che dimensione di pizza vorresti?».
- Dopo che l'utente ha fornito tutte le informazioni sull'ordine della pizza, Amazon Lex potrebbe restituire una risposta con un messaggio per ottenere la conferma dell'utente: «Ordina la pizza?».
- Dopo che l'utente ha risposto «Sì» alla richiesta di conferma, Amazon Lex potrebbe restituire una dichiarazione conclusiva: «Grazie, la tua pizza al formaggio è stata ordinata».

Non tutti i messaggi Amazon Lex richiedono una risposta da parte dell'utente. Ad esempio, le dichiarazioni conclusive non richiedono una risposta. Alcuni messaggi richiedono solo una risposta affermativa o negativa. Oltre a`message`, Amazon Lex fornisce un contesto aggiuntivo sul messaggio nella risposta che puoi utilizzare per migliorare il comportamento del client, come la visualizzazione dell'interfaccia utente client appropriata. Considerare i seguenti esempi:

- Se il messaggio serve a richiamare dati sugli slot, Amazon Lex restituisce le seguenti informazioni di contesto:
  - `x-amz-lex-dialog-state`intestazione impostata su `ElicitSlot`
  - `x-amz-lex-intent-name`intestazione impostata sul nome dell'intento nel contesto corrente
  - `x-amz-lex-slot-to-elicit`intestazione impostata sul nome dello slot per il quale sta ottenendo informazioni `message`
  - `x-amz-lex-slots`header impostato su una mappa di slot configurati per l'intento con i loro valori correnti

- Se il messaggio è una richiesta di conferma, l'`x-amz-lex-dialog-state`intestazione viene impostata su `Confirmation` e l'`x-amz-lex-slot-to-elicit`intestazione viene omessa.
- Se il messaggio è una richiesta di chiarimenti configurata per l'intento e indica che l'intento dell'utente non è compreso, l'intestazione viene impostata su e l'`x-amz-dialog-state`intestazione viene omessa. `ElicitIntent` `x-amz-slot-to-elicit`

Inoltre, Amazon Lex restituisce anche prodotti specifici per l'applicazione`sessionAttributes`. Per ulteriori informazioni, consulta Managing Conversation Context.

Sintassi della richiesta

```
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
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

accept

Si passa questo valore come intestazione `Accept` HTTP.

Il messaggio che Amazon Lex restituisce nella risposta può essere di testo o vocale in base al valore dell'intestazione `Accept` HTTP nella richiesta.

- Se il valore è`text/plain; charset=utf-8`, Amazon Lex restituisce il testo nella risposta.
- Se il valore inizia con`audio/`, Amazon Lex restituisce la voce nella risposta. Amazon Lex utilizza Amazon Polly per generare il parlato (utilizzando la configurazione specificata nell'`Accept`intestazione). Ad esempio, se si specifica `audio/mpeg` come valore, Amazon Lex restituisce la voce in formato MPEG.
- Se il valore è`audio/pcm`, la voce restituita è `audio/pcm` in formato little endian a 16 bit.
- I valori accettati sono i seguenti:
  - audio/mpeg

- audio/ogg
- audio/pcm
- testo/semplice; set di caratteri = utf-8
- audio/* (il valore predefinito è mpeg)

activeContexts

Un elenco di contesti attivi per la richiesta. Un contesto può essere attivato quando viene soddisfatto un intento precedente o includendo il contesto nella richiesta,

Se non specifichi un elenco di contesti, Amazon Lex utilizzerà l'elenco corrente di contesti per la sessione. Se specifichi un elenco vuoto, tutti i contesti della sessione vengono cancellati.

botAlias

Alias del bot Amazon Lex.

Campo obbligatorio: sì

botName

Nome del bot Amazon Lex.

Campo obbligatorio: sì

contentType

Passi questo valore come intestazione `Content-Type` HTTP.

Indica il formato audio o il testo. Il valore dell'intestazione deve iniziare con uno dei seguenti prefissi:

- Formato PCM, i dati audio devono essere in ordine di byte little-endian.
  - audio/l16; rate=16000; canali=1
  - audio/x-l16; frequenza di campionamento = 16000; numero di canali=1
  - audio/lpcm; frequenza di campionamento = 8000; =16; conteggio canali = 1; sample-size-bits =falso is-big-endian
- Formato Opus
  - audio/ x-cbr-opus-with -preamble; dimensione del preambolo=0; bit-rate=256000; =4 frame-size-milliseconds
- Formato testo
  - testo/semplice; set di caratteri = utf-8

Campo obbligatorio: sì

requestAttributes

Si passa questo valore come intestazione HTTP. `x-amz-lex-request-attributes`

Informazioni specifiche sulla richiesta trasferite tra Amazon Lex e un'applicazione client. Il valore deve essere una mappa serializzata JSON e codificata in base64 con chiavi e valori di stringa. La dimensione totale delle `sessionAttributes` intestazioni `requestAttributes` and è limitata a 12 KB.

Lo spazio dei nomi `x-amz-lex:` è riservato agli attributi speciali. Non creare alcun attributo di richiesta con il prefisso. `x-amz-lex:`

Per ulteriori informazioni, vedere Impostazione degli attributi della richiesta.

sessionAttributes

Questo valore viene passato come intestazione `x-amz-lex-session-attributes` HTTP.

Informazioni specifiche dell'applicazione trasferite tra Amazon Lex e un'applicazione client. Il valore deve essere una mappa serializzata JSON e codificata in base64 con chiavi e valori di stringa. La dimensione totale delle `requestAttributes` intestazioni `sessionAttributes` and è limitata a 12 KB.

Per ulteriori informazioni, vedere Impostazione degli attributi di sessione.

userId

L'ID dell'utente dell'applicazione client. Amazon Lex lo utilizza per identificare la conversazione di un utente con il tuo bot. In fase di esecuzione, ogni richiesta deve contenere il `userID` campo.

Per decidere l'ID utente da utilizzare per l'applicazione, considera i seguenti fattori.

- Il `userID` campo non deve contenere informazioni di identificazione personale dell'utente, ad esempio nome, numeri di identificazione personale o altre informazioni personali dell'utente finale.
- Se desideri che un utente inizi una conversazione su un dispositivo e continui su un altro dispositivo, utilizza un identificatore specifico dell'utente.
- Se desideri che lo stesso utente possa tenere due conversazioni indipendenti su due dispositivi diversi, scegli un identificatore specifico del dispositivo.
- Un utente non può avere due conversazioni indipendenti con due versioni diverse dello stesso bot. Ad esempio, un utente non può conversare con le versioni PROD e BETA dello stesso bot.

Se prevedi che un utente debba conversare con due versioni diverse, ad esempio durante il test, includi l'alias bot nell'ID utente per separare le due conversazioni.

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Modello: `[0-9a-zA-Z._:-]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati binari.

inputStream

Input dell'utente in formato audio PCM o Opus o in formato di testo come descritto nell'intestazione HTTP. `Content-Type`

Puoi trasmettere dati audio ad Amazon Lex oppure creare un buffer locale che acquisisca tutti i dati audio prima dell'invio. In generale, si ottengono prestazioni migliori se si effettuano lo streaming di dati audio anziché il buffering dei dati localmente.

Campo obbligatorio: sì

Sintassi della risposta

```
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
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

La risposta restituisce le seguenti intestazioni HTTP.

activeContexts

Un elenco di contesti attivi per la sessione. Un contesto può essere impostato quando un intento viene soddisfatto o chiamando l'operazione `PostContentPostText`, o. `PutSession`

È possibile utilizzare un contesto per controllare gli intenti che possono far seguito a un intento o per modificare il funzionamento dell'applicazione.

alternativeIntents

Da uno a quattro intenti alternativi che possono essere applicabili all'intento dell'utente.

Ogni alternativa include un punteggio che indica quanto Amazon Lex sia sicuro che l'intento corrisponda a quello dell'utente. Le intenzioni sono ordinate in base al punteggio di confidenza.

botVersion

La versione del bot che ha risposto alla conversazione. Puoi utilizzare queste informazioni per determinare se una versione di un bot funziona meglio di un'altra versione.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+|\$LATEST`

contentType

Tipo di contenuto come specificato nell'intestazione `Accept` HTTP della richiesta.

dialogState

Identifica lo stato corrente dell'interazione dell'utente. Amazon Lex restituisce uno dei seguenti valori come`dialogState`. Il client può opzionalmente utilizzare queste informazioni per personalizzare l'interfaccia utente.

- `ElicitIntent`- Amazon Lex vuole suscitare l'intento dell'utente. Considerare i seguenti esempi:

  Ad esempio, un utente potrebbe esprimere un intento («Voglio ordinare una pizza»). Se Amazon Lex non è in grado di dedurre l'intento dell'utente da questo enunciato, restituirà questo stato di dialogo.
- `ConfirmIntent`- Amazon Lex prevede una risposta «sì» o «no».

  Ad esempio, Amazon Lex richiede la conferma dell'utente prima di soddisfare un intento. Invece di una semplice risposta «sì» o «no», un utente potrebbe rispondere fornendo informazioni aggiuntive. Ad esempio, «sì, ma preparala come una pizza dalla crosta densa» o «no, voglio ordinare da bere». Amazon Lex può elaborare tali informazioni aggiuntive (in questi esempi, aggiornare lo slot del tipo di crosta o modificare l'intento da OrderPizza a OrderDrink).
- `ElicitSlot`- Amazon Lex prevede il valore di uno slot per l'intento attuale.

  Ad esempio, supponiamo che nella risposta Amazon Lex invii questo messaggio: «Che dimensione di pizza vorresti?». Un utente potrebbe rispondere con il valore dello slot (ad esempio, «medio»). L'utente potrebbe anche fornire informazioni aggiuntive nella risposta (ad esempio, «pizza con crosta di medio spessore»). Amazon Lex può elaborare tali informazioni aggiuntive in modo appropriato.
- `Fulfilled`- Indica che la funzione Lambda ha raggiunto con successo l'intento.
- `ReadyForFulfillment`- Indica che il cliente deve soddisfare la richiesta.
- `Failed`- Indica che la conversazione con l'utente è fallita.

  Ciò può accadere per vari motivi, tra cui il fatto che l'utente non fornisce una risposta appropriata alle richieste del servizio (puoi configurare quante volte Amazon Lex può richiedere a un utente informazioni specifiche) o se la funzione Lambda non riesce a soddisfare l'intento.

Valori validi: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedInputTranscript

Il testo utilizzato per elaborare la richiesta.

In caso di input di flusso audio, il campo `encodedInputTranscript` contiene il testo estratto dal flusso audio. Questo è il testo che viene effettivamente elaborato per riconoscere gli intenti e i valori dello slot. Puoi utilizzare queste informazioni per determinare se Amazon Lex sta elaborando correttamente l'audio che invii.

Il `encodedInputTranscript` campo è codificato in base 64. È necessario decodificare il campo prima di poter utilizzare il valore.

encodedMessage

Il messaggio da trasmettere all'utente. Il messaggio può provenire dalla configurazione del bot o da una funzione Lambda.

Se l'intento non è configurato con una funzione Lambda o se la funzione Lambda viene `Delegate` restituita come tale `dialogAction.type` nella sua risposta, Amazon Lex decide la linea d'azione successiva e seleziona un messaggio appropriato dalla configurazione del bot in base al contesto di interazione corrente. Ad esempio, se Amazon Lex non è in grado di comprendere l'input dell'utente, utilizza un messaggio di richiesta di chiarimento.

Quando crei un intento, puoi assegnare messaggi ai gruppi. Quando i messaggi vengono assegnati a gruppi, Amazon Lex restituisce un messaggio per ogni gruppo nella risposta. Il campo del messaggio è una stringa JSON in escape contenente i messaggi. Per ulteriori informazioni sulla struttura della stringa JSON restituita, consulta. Formati di messaggio supportati

Se la funzione Lambda restituisce un messaggio, Amazon Lex lo trasmette al client nella sua risposta.

Il `encodedMessage` campo è codificato in base 64. È necessario decodificare il campo prima di poter utilizzare il valore.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1366.

inputTranscript

Questa intestazione è obsoleta.

Il testo utilizzato per elaborare la richiesta.

Puoi utilizzare questo campo solo nelle versioni locali de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. `inputTranscript`In tutte le altre lingue, il campo è nullo. Dovresti invece usare il `encodedInputTranscript` campo.

In caso di input di flusso audio, il campo `inputTranscript` contiene il testo estratto dal flusso audio. Questo è il testo che viene effettivamente elaborato per riconoscere gli intenti e i valori dello slot. Puoi utilizzare queste informazioni per determinare se Amazon Lex sta elaborando correttamente l'audio che invii.

intentName

L'attuale intenzione dell'utente di cui Amazon Lex è a conoscenza.

message

Questa intestazione è obsoleta.

Puoi utilizzare questo campo solo nelle versioni locali de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. `messageIn` tutte le altre lingue, il campo è nullo. Dovresti invece usare il `encodedMessage` campo.

Il messaggio da trasmettere all'utente. Il messaggio può provenire dalla configurazione del bot o da una funzione Lambda.

Se l'intento non è configurato con una funzione Lambda o se la funzione Lambda viene `Delegate` restituita come tale `dialogAction.type` nella sua risposta, Amazon Lex decide la linea d'azione successiva e seleziona un messaggio appropriato dalla configurazione del bot in base al contesto di interazione corrente. Ad esempio, se Amazon Lex non è in grado di comprendere l'input dell'utente, utilizza un messaggio di richiesta di chiarimento.

Quando crei un intento, puoi assegnare messaggi ai gruppi. Quando i messaggi vengono assegnati a gruppi, Amazon Lex restituisce un messaggio per ogni gruppo nella risposta. Il campo del messaggio è una stringa JSON in escape contenente i messaggi. Per ulteriori informazioni sulla struttura della stringa JSON restituita, consulta. Formati di messaggio supportati

Se la funzione Lambda restituisce un messaggio, Amazon Lex lo trasmette al client nella sua risposta.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 1024 caratteri.

messageFormat

Il formato del messaggio di risposta. Uno dei seguenti valori:

- `PlainText`- Il messaggio contiene testo UTF-8 semplice.
- `CustomPayload`- Il messaggio è un formato personalizzato per il cliente.
- `SSML`- Il messaggio contiene testo formattato per l'output vocale.
- `Composite`- Il messaggio contiene un oggetto JSON in escape contenente uno o più messaggi provenienti dai gruppi a cui sono stati assegnati i messaggi al momento della creazione dell'intento.

Valori validi: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Fornisce un punteggio che indica quanto Amazon Lex sia sicuro che l'intento restituito corrisponda all'intento dell'utente. Il punteggio è compreso tra 0,0 e 1,0.

Il punteggio è relativo, non assoluto. Il punteggio può cambiare in base ai miglioramenti apportati ad Amazon Lex.

sentimentResponse

Il sentimento espresso in un enunciato.

Quando il bot è configurato per inviare enunciati ad Amazon Comprehend per l'analisi del sentiment, questo campo contiene il risultato dell'analisi.

sessionAttributes

Mappa di coppie chiave/valore che rappresentano le informazioni di contesto specifiche della sessione.

sessionId

L'identificatore univoco della sessione.

slots

Mappa di zero o più slot di intenti (coppie nome/valore) che Amazon Lex ha rilevato dall'input dell'utente durante la conversazione. Il campo è codificato in base 64.

Amazon Lex crea un elenco di risoluzioni contenente i valori probabili per uno slot. Il valore restituito è determinato dal valore `valueSelectionStrategy` selezionato al momento della creazione o dell'aggiornamento del tipo di slot. Se `valueSelectionStrategy` è impostato su`ORIGINAL_VALUE`, viene restituito il valore fornito dall'utente, se il valore utente è simile ai valori dello slot. Se `valueSelectionStrategy` è impostato `TOP_RESOLUTION` su Amazon Lex restituisce il primo valore nell'elenco delle risoluzioni o, se non esiste un elenco di risoluzioni, null. Se non specifichi `avalueSelectionStrategy`, l'impostazione predefinita è`ORIGINAL_VALUE`.

slotToElicit

Se il `dialogState` valore è`ElicitSlot`, restituisce il nome dello slot per il quale Amazon Lex sta ottenendo un valore.

La risposta restituisce quanto segue come corpo HTTP.

audioStream

Il prompt (o l'istruzione) da trasmettere all'utente. Si basa sulla configurazione e sul contesto del bot. Ad esempio, se Amazon Lex non comprende l'intento dell'utente, invia il file `clarificationPrompt` configurato per il bot. Se l'intento richiede una conferma prima di intraprendere l'azione di adempimento, invia il. `confirmationPrompt` Un altro esempio: supponiamo che la funzione Lambda abbia soddisfatto con successo l'intento e abbia inviato un messaggio da trasmettere all'utente. Quindi Amazon Lex invia quel messaggio nella risposta.

Errori

BadGatewayException

Il bot Amazon Lex è ancora in fase di creazione oppure uno dei servizi dipendenti (Amazon Polly, AWS Lambda) non è riuscito a causa di un errore interno del servizio.

Codice di stato HTTP: 502

BadRequestException

La convalida della richiesta non è riuscita, non è presente alcun messaggio utilizzabile nel contesto oppure la compilazione del bot non è riuscita, è ancora in corso o contiene modifiche non integrate.

Codice di stato HTTP: 400

ConflictException

Due client utilizzano lo stesso account AWS, lo stesso bot Amazon Lex e lo stesso ID utente.

Codice di stato HTTP: 409

DependencyFailedException

Una delle dipendenze, come AWS Lambda o Amazon Polly, generava un'eccezione. Ad esempio,

- Se Amazon Lex non dispone di autorizzazioni sufficienti per chiamare una funzione Lambda.
- Se l'esecuzione di una funzione Lambda richiede più di 30 secondi.
- Se una funzione Lambda di adempimento restituisce un'azione di `Delegate` dialogo senza rimuovere alcun valore dello slot.

Codice di stato HTTP: 424

InternalFailureException

Errore interno del servizio. Riprova la chiamata.

Codice di stato HTTP: 500

LimitExceededException

È stato superato un limite.

Codice di stato HTTP: 429

LoopDetectedException

Questa eccezione non viene utilizzata.

Codice di stato HTTP: 508

NotAcceptableException

L'intestazione accept nella richiesta non ha un valore valido.

Codice di stato HTTP: 406

NotFoundException

La risorsa (ad esempio il bot Amazon Lex o un alias) a cui si fa riferimento non è stata trovata.

Codice di stato HTTP: 404

RequestTimeoutException

L'input vocale è troppo lungo.

Codice di stato HTTP: 408

UnsupportedMediaTypeException

L'intestazione Content-Type (`PostContent`API) ha un valore non valido.

Codice di stato HTTP: 415

Esempi

Esempio 1

In questa richiesta, l'URI identifica un bot (Traffic), una versione del bot ($LATEST) e un nome utente finale (someuser). L'`Content-Type`intestazione identifica il formato dell'audio nel corpo. Amazon

Lex supporta anche altri formati. Per convertire l'audio da un formato all'altro, se necessario, puoi utilizzare il software open source SoX. È possibile specificare il formato in cui si desidera ottenere la risposta aggiungendo l'intestazione `Accept` HTTP.

Nella risposta, l'`x-amz-lex-message`intestazione mostra la risposta restituita da Amazon Lex. Il client può quindi inviare questa risposta all'utente. Lo stesso messaggio viene inviato in formato audio/MPEG tramite codifica a blocchi (come richiesto).

Richiesta di esempio

```
"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]"
…
```

Risposta di esempio

```
"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
…
```

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua AWS , consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PostText

Servizio: Amazon Lex Runtime Service

Invia l'input dell'utente ad Amazon Lex. Le applicazioni client possono utilizzare questa API per inviare richieste ad Amazon Lex in fase di esecuzione. Amazon Lex interpreta quindi l'input dell'utente utilizzando il modello di apprendimento automatico creato per il bot.

In risposta, Amazon Lex restituisce il successivo `message` per comunicare all'utente un'opzione `responseCard` da visualizzare. Considera i seguenti messaggi di esempio:

- Per un input dell'utente «Vorrei una pizza», Amazon Lex potrebbe restituire una risposta con un messaggio che richiama i dati relativi allo slot (ad esempio, PizzaSize): «Che dimensione di pizza vorresti?»
- Dopo che l'utente ha fornito tutte le informazioni sull'ordine della pizza, Amazon Lex potrebbe restituire una risposta con un messaggio per ottenere la conferma dell'utente «Procedere con l'ordine della pizza?».
- Dopo che l'utente ha risposto a una richiesta di conferma con un «sì», Amazon Lex potrebbe restituire una dichiarazione conclusiva: «Grazie, la tua pizza al formaggio è stata ordinata».

Non tutti i messaggi Amazon Lex richiedono una risposta da parte dell'utente. Ad esempio, una dichiarazione conclusiva non richiede una risposta. Alcuni messaggi richiedono solo una risposta utente «sì» o «no». Oltre a`message`, Amazon Lex fornisce un contesto aggiuntivo sul messaggio nella risposta che è possibile utilizzare per migliorare il comportamento del client, ad esempio per visualizzare l'interfaccia utente client appropriata. Questi sono i `slots` campi `slotToElicit` `dialogState``intentName`,, e della risposta. Considerare i seguenti esempi:

- Se il messaggio serve a richiamare dati sugli slot, Amazon Lex restituisce le seguenti informazioni di contesto:
  - `dialogState`impostato su ElicitSlot
  - `intentName`impostato sul nome dell'intento nel contesto corrente
  - `slotToElicit`impostato sul nome dello slot per il quale `message` sta ottenendo informazioni
  - `slots`impostato su una mappa di slot, configurata per l'intento, con valori attualmente noti
- Se il messaggio è una richiesta di conferma, `dialogState` è impostato su ConfirmIntent e impostato su `SlotToElicit` null.

- Se il messaggio è una richiesta di chiarimenti (configurata per l'intento) che indica che l'intento dell'utente non è compreso, viene impostato su e impostato su `dialogState` null. ElicitIntent `slotToElicit`

Inoltre, Amazon Lex restituisce anche prodotti specifici per l'applicazione`sessionAttributes`. Per ulteriori informazioni, consulta Managing Conversation Context.

Sintassi della richiesta

```
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"
   }
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

botAlias

L'alias del bot Amazon Lex.

Campo obbligatorio: sì

botName

Il nome del bot Amazon Lex.

Campo obbligatorio: sì

userId

L'ID dell'utente dell'applicazione client. Amazon Lex lo utilizza per identificare la conversazione di un utente con il tuo bot. In fase di esecuzione, ogni richiesta deve contenere il `userID` campo.

Per decidere l'ID utente da utilizzare per l'applicazione, considera i seguenti fattori.

- Il `userID` campo non deve contenere informazioni di identificazione personale dell'utente, ad esempio nome, numeri di identificazione personale o altre informazioni personali dell'utente finale.
- Se desideri che un utente inizi una conversazione su un dispositivo e continui su un altro dispositivo, utilizza un identificatore specifico dell'utente.
- Se desideri che lo stesso utente possa tenere due conversazioni indipendenti su due dispositivi diversi, scegli un identificatore specifico del dispositivo.
- Un utente non può avere due conversazioni indipendenti con due versioni diverse dello stesso bot. Ad esempio, un utente non può conversare con le versioni PROD e BETA dello stesso bot. Se prevedi che un utente debba conversare con due versioni diverse, ad esempio durante il test, includi l'alias bot nell'ID utente per separare le due conversazioni.

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Modello: `[0-9a-zA-Z._:-]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

activeContexts

Un elenco di contesti attivi per la richiesta. Un contesto può essere attivato quando viene soddisfatto un intento precedente o includendo il contesto nella richiesta,

Se non specifichi un elenco di contesti, Amazon Lex utilizzerà l'elenco corrente di contesti per la sessione. Se specifichi un elenco vuoto, tutti i contesti della sessione vengono cancellati.

Tipo: matrice di oggetti ActiveContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 20 elementi.

Campo obbligatorio: no

inputText

Il testo inserito dall'utente (Amazon Lex interpreta questo testo).

Quando utilizzi l'AWS CLI, non puoi passare un URL nel `--input-text` parametro. Passa invece l'URL utilizzando il `--cli-input-json` parametro.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 1024 caratteri.

Campo obbligatorio: sì

requestAttributes

Informazioni specifiche sulla richiesta trasferite tra Amazon Lex e un'applicazione client.

Lo spazio dei nomi `x-amz-lex:` è riservato agli attributi speciali. Non creare alcun attributo di richiesta con il prefisso. `x-amz-lex:`

Per ulteriori informazioni, vedere Impostazione degli attributi della richiesta.

Tipo: mappatura stringa a stringa

Campo obbligatorio: no

sessionAttributes

Informazioni specifiche dell'applicazione trasferite tra Amazon Lex e un'applicazione client.

Per ulteriori informazioni, consulta Impostazione degli attributi di sessione.

Tipo: mappatura stringa a stringa

Campo obbligatorio: no

Sintassi della risposta

```
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"
}
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

I dati seguenti vengono restituiti in formato JSON mediante il servizio.

activeContexts

Un elenco di contesti attivi per la sessione. Un contesto può essere impostato quando un intento viene soddisfatto o chiamando l'operazione `PostContentPostText`, o. `PutSession`

È possibile utilizzare un contesto per controllare gli intenti che possono far seguito a un intento o per modificare il funzionamento dell'applicazione.

Tipo: matrice di oggetti ActiveContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 20 elementi.

alternativeIntents

Da uno a quattro intenti alternativi che possono essere applicabili all'intento dell'utente.

Ogni alternativa include un punteggio che indica quanto Amazon Lex sia sicuro che l'intento corrisponda a quello dell'utente. Le intenzioni sono ordinate in base al punteggio di confidenza.

Tipo: matrice di oggetti PredictedIntent

Membri della matrice: numero massimo di 4 elementi.

botVersion

La versione del bot che ha risposto alla conversazione. Puoi utilizzare queste informazioni per determinare se una versione di un bot funziona meglio di un'altra versione.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `[0-9]+|\$LATEST`

dialogState

Identifica lo stato attuale dell'interazione dell'utente. Amazon Lex restituisce uno dei seguenti valori come`dialogState`. Il client può opzionalmente utilizzare queste informazioni per personalizzare l'interfaccia utente.

- `ElicitIntent`- Amazon Lex vuole suscitare l'intenzione dell'utente.

  Ad esempio, un utente potrebbe esprimere un intento («Voglio ordinare una pizza»). Se Amazon Lex non è in grado di dedurre l'intento dell'utente da questo enunciato, restituirà questo DialogState.

- `ConfirmIntent`- Amazon Lex prevede una risposta «sì» o «no».

  Ad esempio, Amazon Lex richiede la conferma dell'utente prima di soddisfare un intento.

  Invece di un semplice «sì» o «no», un utente potrebbe rispondere fornendo informazioni aggiuntive. Ad esempio, «sì, ma preparala una pizza dalla crosta densa» o «no, voglio ordinare da bere». Amazon Lex può elaborare tali informazioni aggiuntive (in questi esempi, aggiornare il valore dello slot del tipo di crosta o modificare l'intento da OrderPizza a OrderDrink).

- `ElicitSlot`- Amazon Lex prevede un valore di slot per l'intento attuale.

Ad esempio, supponiamo che nella risposta Amazon Lex invii questo messaggio: «Che dimensione di pizza vorresti?». Un utente potrebbe rispondere con il valore dello slot (ad esempio, «medio»). L'utente potrebbe anche fornire informazioni aggiuntive nella risposta (ad esempio, «pizza con crosta di medio spessore»). Amazon Lex è in grado di elaborare tali informazioni aggiuntive in modo appropriato.

- `Fulfilled`- Indica che la funzione Lambda configurata per l'intento ha soddisfatto con successo l'intento.
- `ReadyForFulfillment`- Indica che il cliente deve soddisfare l'intento.
- `Failed`- Indica che la conversazione con l'utente è fallita.

  Ciò può accadere per vari motivi, tra cui il fatto che l'utente non ha fornito una risposta appropriata alle richieste del servizio (puoi configurare quante volte Amazon Lex può richiedere a un utente informazioni specifiche) o la funzione Lambda non è riuscita a soddisfare l'intento.

- Tipo: stringa

Valori validi: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

intentName

L'attuale intenzione dell'utente di cui Amazon Lex è a conoscenza.

- Tipo: stringa

message

Il messaggio da trasmettere all'utente. Il messaggio può provenire dalla configurazione del bot o da una funzione Lambda.

Se l'intento non è configurato con una funzione Lambda o se la funzione Lambda viene `Delegate` restituita come `dialogAction.type` risposta, Amazon Lex decide la linea d'azione successiva e seleziona un messaggio appropriato dalla configurazione del bot in base al contesto di interazione corrente. Ad esempio, se Amazon Lex non è in grado di comprendere l'input dell'utente, utilizza un messaggio di richiesta di chiarimento.

Quando crei un intento, puoi assegnare messaggi ai gruppi. Quando i messaggi vengono assegnati a gruppi, Amazon Lex restituisce un messaggio per ogni gruppo nella risposta. Il campo del messaggio è una stringa JSON in escape contenente i messaggi. Per ulteriori informazioni sulla struttura della stringa JSON restituita, vedere. Formati di messaggio supportati

Se la funzione Lambda restituisce un messaggio, Amazon Lex lo trasmette al client nella sua risposta.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 1024 caratteri.

messageFormat

Il formato del messaggio di risposta. Uno dei seguenti valori:

- `PlainText`- Il messaggio contiene testo UTF-8 semplice.
- `CustomPayload`- Il messaggio è un formato personalizzato definito dalla funzione Lambda.
- `SSML`- Il messaggio contiene testo formattato per l'output vocale.
- `Composite`- Il messaggio contiene un oggetto JSON in escape contenente uno o più messaggi provenienti dai gruppi a cui sono stati assegnati i messaggi al momento della creazione dell'intento.

- Tipo: stringa

Valori validi: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Fornisce un punteggio che indica quanto Amazon Lex sia sicuro che l'intento restituito corrisponda all'intento dell'utente. Il punteggio è compreso tra 0,0 e 1,0. Per ulteriori informazioni, vedere Confidence Scores.

Il punteggio è un punteggio relativo, non assoluto. Il punteggio può cambiare in base ai miglioramenti apportati ad Amazon Lex.

Tipo: oggetto IntentConfidence

responseCard

Rappresenta le opzioni a disposizione dell'utente per rispondere alla richiesta corrente. La Response Card può provenire dalla configurazione del bot (nella console Amazon Lex, scegli il pulsante delle impostazioni accanto a uno slot) o da un code hook (funzione Lambda).

Tipo: oggetto ResponseCard

sentimentResponse

Il sentimento espresso in ed enunciato.

Quando il bot è configurato per inviare enunciati ad Amazon Comprehend per l'analisi del sentiment, questo campo contiene il risultato dell'analisi.

Tipo: oggetto SentimentResponse

sessionAttributes

Una mappa di coppie chiave-valore che rappresentano le informazioni di contesto specifiche della sessione.

Tipo: mappatura stringa a stringa

sessionId

Un identificatore univoco per la sessione.

- Tipo: stringa

slots

Gli slot di intenti rilevati da Amazon Lex in base all'input dell'utente nella conversazione.

Amazon Lex crea un elenco di risoluzioni contenente i valori probabili per uno slot. Il valore restituito è determinato dal valore `valueSelectionStrategy` selezionato al momento della creazione o dell'aggiornamento del tipo di slot. Se `valueSelectionStrategy` è impostato su`ORIGINAL_VALUE`, viene restituito il valore fornito dall'utente, se il valore utente è simile ai valori dello slot. Se `valueSelectionStrategy` è impostato `TOP_RESOLUTION` su Amazon Lex restituisce il primo valore nell'elenco delle risoluzioni o, se non esiste un elenco di risoluzioni, null. Se non specifichi `avalueSelectionStrategy`, l'impostazione predefinita è`ORIGINAL_VALUE`.

Tipo: mappatura stringa a stringa

slotToElicit

Se il `dialogState` valore è`ElicitSlot`, restituisce il nome dello slot per il quale Amazon Lex sta ottenendo un valore.

- Tipo: stringa

Errori

BadGatewayException

Il bot Amazon Lex è ancora in fase di creazione oppure uno dei servizi dipendenti (Amazon Polly, AWS Lambda) non è riuscito a causa di un errore interno del servizio.

Codice di stato HTTP: 502

BadRequestException

La convalida della richiesta non è riuscita, non è presente alcun messaggio utilizzabile nel contesto oppure la compilazione del bot non è riuscita, è ancora in corso o contiene modifiche non integrate.

Codice di stato HTTP: 400

ConflictException

Due client utilizzano lo stesso account AWS, lo stesso bot Amazon Lex e lo stesso ID utente.

Codice di stato HTTP: 409

DependencyFailedException

Una delle dipendenze, come AWS Lambda o Amazon Polly, generava un'eccezione. Ad esempio,

- Se Amazon Lex non dispone di autorizzazioni sufficienti per chiamare una funzione Lambda.
- Se l'esecuzione di una funzione Lambda richiede più di 30 secondi.
- Se una funzione Lambda di adempimento restituisce un'azione di `Delegate` dialogo senza rimuovere alcun valore dello slot.

Codice di stato HTTP: 424

InternalFailureException

Errore interno del servizio. Riprova la chiamata.

Codice di stato HTTP: 500

LimitExceededException

È stato superato un limite.

Codice di stato HTTP: 429

LoopDetectedException

Questa eccezione non viene utilizzata.

Codice di stato HTTP: 508

NotFoundException

La risorsa (ad esempio il bot Amazon Lex o un alias) a cui si fa riferimento non è stata trovata.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

PutSession

Servizio: Amazon Lex Runtime Service

Crea una nuova sessione o modifica una sessione esistente con un bot Amazon Lex. Utilizzate questa operazione per consentire all'applicazione di impostare lo stato del bot.

Per ulteriori informazioni, consulta Gestione delle sessioni.

Sintassi della richiesta

```
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"
  }
}
```

Parametri della richiesta URI

La richiesta utilizza i seguenti parametri URI.

accept

Il messaggio che Amazon Lex restituisce nella risposta può essere testuale o vocale a seconda del valore di questo campo.

- Se il valore è`text/plain; charset=utf-8`, Amazon Lex restituisce il testo nella risposta.
- Se il valore inizia con`audio/`, Amazon Lex restituisce la voce nella risposta. Amazon Lex utilizza Amazon Polly per generare il parlato nella configurazione specificata. Ad esempio, se si specifica `audio/mpeg` come valore, Amazon Lex restituisce la voce in formato MPEG.
- Se il valore è`audio/pcm`, la voce viene restituita `audio/pcm` nel formato little endian a 16 bit.
- I valori accettati sono i seguenti:
  - `audio/mpeg`
  - `audio/ogg`
  - `audio/pcm`
  - `audio/*`(il valore predefinito è mpeg)
  - `text/plain; charset=utf-8`

botAlias

L'alias in uso per il bot che contiene i dati della sessione.

Campo obbligatorio: sì

botName

Il nome del bot che contiene i dati della sessione.

Campo obbligatorio: sì

userId

L'ID dell'utente dell'applicazione client. Amazon Lex lo utilizza per identificare la conversazione di un utente con il tuo bot.

Vincoli di lunghezza: lunghezza minima di 2. Lunghezza massima di 100.

Modello: `[0-9a-zA-Z._:-]+`

Campo obbligatorio: sì

Corpo della richiesta

La richiesta accetta i seguenti dati in formato JSON.

activeContexts

Un elenco di contesti attivi per la richiesta. Un contesto può essere attivato quando viene soddisfatto un intento precedente o includendo il contesto nella richiesta,

Se non specifichi un elenco di contesti, Amazon Lex utilizzerà l'elenco corrente di contesti per la sessione. Se specifichi un elenco vuoto, tutti i contesti della sessione vengono cancellati.

Tipo: matrice di oggetti ActiveContext

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 20 elementi.

Campo obbligatorio: no

dialogAction

Imposta l'azione successiva che il bot deve intraprendere per completare la conversazione.

Tipo: oggetto DialogAction

Campo obbligatorio: no

recentIntentSummaryView

Un riepilogo delle intenzioni recenti del bot. È possibile utilizzare la visualizzazione di riepilogo degli intenti per impostare un'etichetta di checkpoint su un intento e modificare gli attributi degli intenti. Puoi anche usarla per rimuovere o aggiungere oggetti di riepilogo degli intenti all'elenco.

Un intento che modifichi o aggiungi all'elenco deve avere senso per il bot. Ad esempio, il nome dell'intento deve essere valido per il bot. È necessario fornire valori validi per:

- `intentName`
- nomi di slot
- `slotToElict`

Se si invia il `recentIntentSummaryView` parametro in una `PutSession` richiesta, il contenuto della nuova visualizzazione di riepilogo sostituisce la vecchia visualizzazione di riepilogo. Ad esempio, se una `GetSession` richiesta restituisce tre intenti nella visualizzazione di riepilogo e si chiama `PutSession` con un intento nella visualizzazione di riepilogo, la chiamata successiva `GetSession` restituirà solo un intento.

Tipo: matrice di oggetti IntentSummary

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 3 elementi.

Campo obbligatorio: no

sessionAttributes

Mappa delle coppie chiave/valore che rappresentano le informazioni di contesto specifiche della sessione. Contiene informazioni sull'applicazione trasferite tra Amazon Lex e un'applicazione client.

Tipo: mappatura stringa a stringa

Campo obbligatorio: no

Sintassi della risposta

```
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
```

Elementi di risposta

Se l'operazione riesce, il servizio restituisce una risposta HTTP 200.

La risposta restituisce le seguenti intestazioni HTTP.

activeContexts

Un elenco di contesti attivi per la sessione.

contentType

Tipo di contenuto specificato nell'intestazione `Accept` HTTP della richiesta.

dialogState

- `ConfirmIntent`- Amazon Lex si aspetta una risposta «sì» o «no» per confermare l'intento prima di realizzarlo.
- `ElicitIntent`- Amazon Lex vuole suscitare l'intento dell'utente.
- `ElicitSlot`- Amazon Lex prevede il valore di uno slot per l'intento attuale.
- `Failed`- Indica che la conversazione con l'utente è fallita. Ciò può accadere per vari motivi, tra cui l'utente non fornisce una risposta appropriata alle richieste del servizio o se la funzione Lambda non riesce a soddisfare l'intento.
- `Fulfilled`- Indica che la funzione Lambda ha raggiunto con successo l'intento.
- `ReadyForFulfillment`- Indica che il cliente deve soddisfare l'intento.

Valori validi: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedMessage

Il messaggio successivo che deve essere presentato all'utente.

Il `encodedMessage` campo è codificato in base 64. È necessario decodificare il campo prima di poter utilizzare il valore.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1366.

intentName

Il nome dell'intento corrente.

message

Questa intestazione è obsoleta.

Il messaggio successivo che dovrebbe essere presentato all'utente.

Puoi utilizzare questo campo solo nelle versioni locali de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. `messageIn` tutte le altre lingue, il campo è nullo. Dovresti invece usare il `encodedMessage` campo.

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 1024 caratteri.

messageFormat

Il formato del messaggio di risposta. Uno dei seguenti valori:

- `PlainText`- Il messaggio contiene testo UTF-8 semplice.
- `CustomPayload`- Il messaggio è un formato personalizzato per il cliente.
- `SSML`- Il messaggio contiene testo formattato per l'output vocale.
- `Composite`- Il messaggio contiene un oggetto JSON in escape contenente uno o più messaggi provenienti dai gruppi a cui sono stati assegnati i messaggi al momento della creazione dell'intento.

Valori validi: `PlainText | CustomPayload | SSML | Composite`

sessionAttributes

Mappa di coppie chiave/valore che rappresentano informazioni di contesto specifiche della sessione.

sessionId

Un identificatore univoco per la sessione.

slots

Mappa degli slot con zero o più intenti rilevati da Amazon Lex in base all'input dell'utente durante la conversazione.

Amazon Lex crea un elenco di risoluzioni contenente i valori probabili per uno slot. Il valore restituito è determinato dal valore `valueSelectionStrategy` selezionato al momento della creazione o dell'aggiornamento del tipo di slot. Se `valueSelectionStrategy` è impostato su`ORIGINAL_VALUE`, viene restituito il valore fornito dall'utente, se il valore utente è simile ai valori dello slot. Se `valueSelectionStrategy` è impostato `TOP_RESOLUTION` su Amazon Lex restituisce il primo valore nell'elenco delle risoluzioni o, se non esiste un elenco di risoluzioni, null. Se non specifichi a, `valueSelectionStrategy` l'impostazione predefinita è`ORIGINAL_VALUE`.

slotToElicit

Se `dialogState` è`ElicitSlot`, restituisce il nome dello slot per il quale Amazon Lex sta ottenendo un valore.

La risposta restituisce quanto segue come corpo HTTP.

audioStream

La versione audio del messaggio da trasmettere all'utente.

Errori

BadGatewayException

Il bot Amazon Lex è ancora in fase di creazione oppure uno dei servizi dipendenti (Amazon Polly, AWS Lambda) non è riuscito a causa di un errore interno del servizio.

Codice di stato HTTP: 502

BadRequestException

La convalida della richiesta non è riuscita, non è presente alcun messaggio utilizzabile nel contesto oppure la compilazione del bot non è riuscita, è ancora in corso o contiene modifiche non integrate.

Codice di stato HTTP: 400

ConflictException

Due client utilizzano lo stesso account AWS, lo stesso bot Amazon Lex e lo stesso ID utente.

Codice di stato HTTP: 409

DependencyFailedException

Una delle dipendenze, come AWS Lambda o Amazon Polly, generava un'eccezione. Ad esempio,

- Se Amazon Lex non dispone di autorizzazioni sufficienti per chiamare una funzione Lambda.
- Se l'esecuzione di una funzione Lambda richiede più di 30 secondi.
- Se una funzione Lambda di adempimento restituisce un'azione di `Delegate` dialogo senza rimuovere alcun valore dello slot.

Codice di stato HTTP: 424

InternalFailureException

Errore interno del servizio. Riprova la chiamata.

Codice di stato HTTP: 500

LimitExceededException

È stato superato un limite.

Codice di stato HTTP: 429

NotAcceptableException

L'intestazione accept nella richiesta non ha un valore valido.

Codice di stato HTTP: 406

NotFoundException

La risorsa (ad esempio il bot Amazon Lex o un alias) a cui si fa riferimento non è stata trovata.

Codice di stato HTTP: 404

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- Interfaccia a riga di comando AWS
- AWS SDK per.NET
- AWS SDK per C++
- AWS SDK per Go v2
- AWS SDK per Java V2
- AWS SDK per V3 JavaScript
- AWS SDK per PHP V3
- AWS SDK per Python
- AWS SDK per Ruby V3

Tipi di dati

I tipi di dati seguenti sono supportati da Amazon Lex Model Building Service:

- 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

Sono supportati i tipi di dati seguenti da Amazon Lex Runtime Service:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

Servizio di modellazione Amazon Lex

I tipi di dati seguenti sono supportati da Amazon Lex Model Building Service:

- 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

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su un alias bot.

Indice

botName

Il nome del bot a cui punta l'alias.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

botVersion

La versione del bot Amazon Lex a cui punta l'alias.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

checksum

Checksum dell'alias del bot.

- Tipo: stringa

Campo obbligatorio: no

conversationLogs

Impostazioni che determinano in che modo Amazon Lex utilizza i log delle conversazioni per l'alias.

Tipo: oggetto ConversationLogsResponse

Campo obbligatorio: no

createdDate

La data di creazione dell'alias del bot.

Tipo: Timestamp

Campo obbligatorio: no

description

Una descrizione dell'alias del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

lastUpdatedDate

La data in cui l'alias del bot è stato aggiornato. Quando crei una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

Campo obbligatorio: no

name

Il nome dell'alias del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

BotChannelAssociation

Servizio: Amazon Lex Model Building Service

Rappresenta un'associazione tra un bot Amazon Lex e una piattaforma di messaggistica esterna.

Indice

botAlias

Un alias che indica la versione specifica del bot Amazon Lex a cui viene effettuata questa associazione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

botConfiguration

Fornisce le informazioni necessarie per comunicare con la piattaforma di messaggistica.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero massimo di 10 elementi.

Campo obbligatorio: no

botName

Il nome del bot Amazon Lex a cui viene creata questa associazione.

> **Note**
>
> Attualmente, Amazon Lex supporta le associazioni con Facebook e Slack e Twilio.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

createdDate

Data di creazione dell'associazione tra il bot Amazon Lex e il canale.

Tipo: Timestamp

Campo obbligatorio: no

description

Una descrizione testuale dell'associazione che stai creando.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

failureReason

In caso `status` `FAILED` affermativo, Amazon Lex fornisce il motivo per cui non è riuscita a creare l'associazione.

- Tipo: stringa

Campo obbligatorio: no

name

Il nome dell'associazione tra il bot e il canale.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

status

Lo stato del canale bot.

- `CREATED`- Il canale è stato creato ed è pronto per l'uso.
- `IN_PROGRESS`- La creazione del canale è in corso.

- `FAILED`- Si è verificato un errore durante la creazione del canale. Per informazioni sul motivo dell'errore, consulta il `failureReason` campo.

-Tipo: stringa

Valori validi: `IN_PROGRESS | CREATED | FAILED`

Campo obbligatorio: no

type

Speciifica il tipo di associazione indicando il tipo di canale che viene stabilito tra il bot Amazon Lex e la piattaforma di messaggistica esterna.

-Tipo: stringa

Valori validi: `Facebook | Slack | Twilio-Sms | Kik`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

BotMetadata

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su un bot.

Indice

createdDate

La data di creazione del bot.

Tipo: Timestamp

Campo obbligatorio: no

description

Una descrizione del bot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

lastUpdatedDate

La data in cui il bot è stato aggiornato. Quando crei un bot, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

Campo obbligatorio: no

name

Il nome del bot.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

status

Lo stato del bot.

- Tipo: stringa

Valori validi: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

Campo obbligatorio: no

version

La versione del bot. Per un nuovo bot, la versione è sempre`$LATEST`.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

BuiltinIntentMetadata

Servizio: Amazon Lex Model Building Service

Fornisce metadati per un intento integrato.

Indice

signature

Un identificatore univoco per l'intento integrato. Per trovare la firma di un intento, consulta Standard Built-in Intents nell'Alexa Skills Kit.

- Tipo: stringa

Campo obbligatorio: no

supportedLocales

Un elenco di identificatori per le impostazioni locali supportate dall'intento.

Tipo: matrice di stringhe

Valori validi: `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`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue AWS :

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

BuiltinIntentSlot

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su uno slot utilizzato in un intento integrato.

Indice

name

Un elenco degli slot definiti per l'intento.

- Tipo: stringa

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

BuiltinSlotTypeMetadata

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su un tipo di slot integrato.

Indice

signature

Un identificatore univoco per il tipo di slot integrato. Per trovare la firma per un tipo di slot, consulta Slot Type Reference nell'Alexa Skills Kit.

- Tipo: stringa

Campo obbligatorio: no

supportedLocales

Un elenco di localizzazioni di destinazione per lo slot.

Tipo: matrice di stringhe

Valori validi: `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`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

CodeHook

Servizio: Amazon Lex Model Building Service

Specifica una funzione Lambda che verifica le richieste a un bot o soddisfa la richiesta dell'utente a un bot.

Indice

messageVersion

La versione della richiesta-risposta che desideri che Amazon Lex utilizzi per richiamare la tua funzione Lambda. Per ulteriori informazioni, consulta Utilizzo delle funzioni Lambda.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 5.

Campo obbligatorio: sì

uri

Il nome della risorsa Amazon (ARN) della funzione Lambda.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `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-_]+)?`

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ConversationLogsRequest

Servizio: Amazon Lex Model Building Service

Fornisce le impostazioni necessarie per i registri delle conversazioni.

Indice

iamRoleArn

L'Amazon Resource Name (ARN) di un ruolo IAM con il permesso di scrivere nei log per CloudWatch i log di testo e nel bucket S3 per i log audio. Se la crittografia audio è abilitata, questo ruolo fornisce anche l'autorizzazione di accesso per la chiave AWS KMS utilizzata per crittografare i log audio. Per ulteriori informazioni, consulta Creazione di un ruolo e di una policy IAM per i registri di conversazione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Campo obbligatorio: sì

logSettings

Le impostazioni per i registri delle conversazioni. È possibile registrare il testo della conversazione, l'audio della conversazione o entrambi.

Tipo: matrice di oggetti LogSettingsRequest

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ConversationLogsResponse

Servizio: Amazon Lex Model Building Service

Contiene informazioni sulle impostazioni del registro delle conversazioni.

Indice

iamRoleArn

L'Amazon Resource Name (ARN) del ruolo IAM usato per scrivere i log in CloudWatch Logs o in un bucket S3.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Campo obbligatorio: no

logSettings

Le impostazioni per i registri delle conversazioni. Puoi registrare testo, audio o entrambi.

Tipo: matrice di oggetti LogSettingsResponse

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

EnumerationValue

Servizio: Amazon Lex Model Building Service

Ogni tipo di slot può disporre di un set di valori. Ogni valore di enumerazione rappresenta un valore che il tipo di slot può assumere.

Ad esempio, un bot per ordinare pizze potrebbe avere un tipo di slot che specifica il tipo di crosta che deve avere la pizza. Il tipo di slot potrebbe includere i valori

- spessa
- thin
- farciti

Indice

value

Il valore del tipo di slot.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 140.

Campo obbligatorio: sì

synonyms

Valori aggiuntivi relativi al valore del tipo di slot.

Tipo: matrice di stringhe

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 140.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++

- AWS SDK per Java V2
- AWS SDK per Ruby V3

FollowUpPrompt

Servizio: Amazon Lex Model Building Service

Richiesta di attività aggiuntive dopo il raggiungimento di un intento. Ad esempio, una volta soddisfatto l'`OrderPizza`intento, potresti richiedere all'utente di scoprire se desidera ordinare bevande.

Indice

prompt

Richiede informazioni all'utente.

Tipo: oggetto Prompt

Campo obbligatorio: sì

rejectionStatement

Se l'utente risponde «no» alla domanda definita nel `prompt` campo, Amazon Lex risponde con questa dichiarazione per confermare che l'intento è stato annullato.

Tipo: oggetto Statement

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta AWS quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

FulfillmentActivity

Servizio: Amazon Lex Model Building Service

Descrive come l'intento viene soddisfatto dopo che l'utente ha fornito tutte le informazioni necessarie per l'intento. È possibile fornire una funzione Lambda per elaborare l'intento oppure restituire le informazioni sull'intento all'applicazione client. Ti consigliamo di utilizzare una funzione Lambda in modo che la logica pertinente risieda nel cloud e limiti il codice lato client principalmente alla presentazione. Se è necessario aggiornare la logica, è sufficiente aggiornare la funzione Lambda; non è necessario aggiornare l'applicazione client.

Considerare i seguenti esempi:

- In un'applicazione per ordinare pizze, dopo che l'utente ha fornito tutte le informazioni per effettuare un ordine, si utilizza una funzione Lambda per effettuare un ordine presso una pizzeria.
- In un'applicazione di gioco, quando un utente dice «pick up a rock», queste informazioni devono tornare all'applicazione client in modo che possa eseguire l'operazione e aggiornare la grafica. In questo caso, desideri che Amazon Lex restituisca i dati sulle intenzioni al client.

Indice

type

Come deve essere raggiunto l'intento, eseguendo una funzione Lambda o restituendo i dati dello slot all'applicazione client.

- Tipo: stringa

Valori validi: `ReturnIntent | CodeHook`

Campo obbligatorio: sì

codeHook

Una descrizione della funzione Lambda che viene eseguita per soddisfare l'intento.

Tipo: oggetto CodeHook

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

InputContext

Servizio: Amazon Lex Model Building Service

Il nome di un contesto che deve essere attivo affinché un intento possa essere selezionato da Amazon Lex.

Indice

name

Il nome del contesto.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Intent

Servizio: Amazon Lex Model Building Service

Identifica la versione specifica di un intento.

Indice

intentName

Il nome dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

intentVersion

La versione dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

IntentMetadata

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su un intento.

Indice

createdDate

La data in cui è stato creato l'intento.

Tipo: Timestamp

Campo obbligatorio: no

description

Una descrizione dell'intento.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

lastUpdatedDate

La data in cui l'intento è stato aggiornato. Quando crei un intento, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

Campo obbligatorio: no

name

Il nome dell'intento.

Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

version

La versione dell'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

KendraConfiguration

Servizio: Amazon Lex Model Building Service

Fornisce informazioni di configurazione per AMAZON. KendraSearchIntentintento. Quando usi questo intento, Amazon Lex cerca nell'indice Amazon Kendra specificato e restituisce i documenti dall'indice che corrispondono all'enunciato dell'utente. Per ulteriori informazioni, consulta AMAZON. KendraSearchIntent.

Indice

kendraIndex

L'Amazon Resource Name (ARN) dell'indice Amazon Kendra su cui desideri inserire AMAZON. KendraSearchIntent intenzione di cercare. L'indice deve trovarsi nello stesso account e nella stessa regione del bot Amazon Lex. Se l'indice Amazon Kendra non esiste, ottieni un'eccezione quando chiami l'operazione. `PutIntent`

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9][a-zA-Z0-9_-]*`

Campo obbligatorio: sì

role

L'Amazon Resource Name (ARN) di un ruolo IAM autorizzato a effettuare ricerche nell'indice Amazon Kendra. Il ruolo deve trovarsi nello stesso account e nella stessa regione del bot Amazon Lex. Se il ruolo non esiste, ottieni un'eccezione quando chiami l'`PutIntent`operazione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `arn:aws:iam::[0-9]{12}:role/.*`

Campo obbligatorio: sì

queryFilterString

Un filtro di query che Amazon Lex invia ad Amazon Kendra per filtrare la risposta dalla query. Il filtro è nel formato definito da Amazon Kendra. Per ulteriori informazioni, consulta Filtraggio delle interrogazioni.

È possibile sovrascrivere questa stringa di filtro con una nuova stringa di filtro in fase di esecuzione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

LogSettingsRequest

Servizio: Amazon Lex Model Building Service

Impostazioni utilizzate per configurare la modalità di consegna e la destinazione per i registri delle conversazioni.

Indice

destination

Dove verranno consegnati i log. I log di testo vengono consegnati a un gruppo di CloudWatch log Logs. I log audio vengono inviati a un bucket S3.

- Tipo: stringa

Valori validi: `CLOUDWATCH_LOGS | S3`

Campo obbligatorio: sì

logType

Il tipo di registrazione da abilitare. I log di testo vengono consegnati a un gruppo di CloudWatch log Logs. I log audio vengono inviati a un bucket S3.

- Tipo: stringa

Valori validi: `AUDIO | TEXT`

Campo obbligatorio: sì

resourceArn

L'Amazon Resource Name (ARN) del gruppo di log CloudWatch Logs o del bucket S3 in cui devono essere consegnati i log.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 1. La lunghezza massima è 2048 caratteri.

Modello: `^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])$`

Campo obbligatorio: sì

kmsKeyArn

L'Amazon Resource Name (ARN) della chiave gestita dal cliente AWS KMS per la crittografia dei log audio distribuiti a un bucket S3. La chiave non si applica ai CloudWatch log ed è facoltativa per i bucket S3.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue AWS :

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

LogSettingsResponse

Servizio: Amazon Lex Model Building Service

Le impostazioni per i registri delle conversazioni.

Indice

destination

La destinazione in cui vengono consegnati i log.

- Tipo: stringa

Valori validi: `CLOUDWATCH_LOGS | S3`

Campo obbligatorio: no

kmsKeyArn

L'Amazon Resource Name (ARN) della chiave utilizzata per crittografare i log audio in un bucket S3.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Campo obbligatorio: no

logType

Il tipo di registrazione abilitato.

- Tipo: stringa

Valori validi: `AUDIO | TEXT`

Campo obbligatorio: no

resourceArn

L'Amazon Resource Name (ARN) del gruppo di log CloudWatch Logs o del bucket S3 in cui vengono consegnati i log.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 1. La lunghezza massima è 2048 caratteri.

Modello: `^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])$`

Campo obbligatorio: no

resourcePrefix

Il prefisso della risorsa è la prima parte della chiave dell'oggetto S3 all'interno del bucket S3 che hai specificato per contenere i log audio. Per CloudWatch Logs è il prefisso del nome del flusso di log all'interno del gruppo di log specificato.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza massima di 1024.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Message

Servizio: Amazon Lex Model Building Service

L'oggetto messaggio che fornisce il testo del messaggio e il relativo tipo.

Indice

content

Il testo del messaggio.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1000.

Campo obbligatorio: sì

contentType

Il tipo di contenuto della stringa del messaggio.

- Tipo: stringa

Valori validi: `PlainText | SSML | CustomPayload`

Campo obbligatorio: sì

groupNumber

Identifica il gruppo di messaggi a cui appartiene il messaggio. Quando un gruppo viene assegnato a un messaggio, Amazon Lex restituisce un messaggio per ogni gruppo nella risposta.

Tipo: integer

Intervallo valido: valore minimo di 1. Valore massimo di 5.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++

- AWS SDK per Java V2
- AWS SDK per Ruby V3

MigrationAlert

Servizio: Amazon Lex Model Building Service

Fornisce informazioni sugli avvisi e gli avvisi che Amazon Lex invia durante una migrazione. Gli avvisi includono informazioni su come risolvere il problema.

Indice

details

Dettagli aggiuntivi sull'avviso.

Tipo: matrice di stringhe

Campo obbligatorio: no

message

Un messaggio che descrive il motivo per cui è stato emesso l'avviso.

Tipo: stringa

Campo obbligatorio: no

referenceURLs

Un link alla documentazione di Amazon Lex che descrive come risolvere l'avviso.

Tipo: matrice di stringhe

Campo obbligatorio: no

type

Il tipo di avviso. Esistono due tipi di avvisi:

- `ERROR`- Si è verificato un problema con la migrazione che non può essere risolto. La migrazione si interrompe.
- `WARN`- Si è verificato un problema con la migrazione che richiede modifiche manuali al nuovo bot Amazon Lex V2. La migrazione continua.

Tipo: stringa

Valori validi: `ERROR | WARN`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

MigrationSummary

Servizio: Amazon Lex Model Building Service

Fornisce informazioni sulla migrazione di un bot da Amazon Lex V1 ad Amazon Lex V2.

Indice

migrationId

L'identificatore univoco assegnato da Amazon Lex alla migrazione.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 10.

Modello: `^[0-9a-zA-Z]+$`

Campo obbligatorio: no

migrationStatus

Stato dell'operazione. Quando lo stato è, `COMPLETE` il bot è disponibile in Amazon Lex V2. Potrebbero esserci avvisi e avvisi che devono essere risolti per completare la migrazione.

- Tipo: stringa

Valori validi: `IN_PROGRESS | COMPLETED | FAILED`

Campo obbligatorio: no

migrationStrategy

La strategia utilizzata per condurre la migrazione.

- Tipo: stringa

Valori validi: `CREATE_NEW | UPDATE_EXISTING`

Campo obbligatorio: no

migrationTimestamp

La data e l'ora di inizio della migrazione.

Tipo: Timestamp

Campo obbligatorio: no

v1BotLocale

La lingua del bot Amazon Lex V1 che è l'origine della migrazione.

- Tipo: stringa

Valori validi: `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`

Campo obbligatorio: no

v1BotName

Il nome del bot Amazon Lex V1 che è l'origine della migrazione.

- Tipo: stringa

Vincoli di lunghezza: lunghezza minima di 2. La lunghezza massima è 50 caratteri.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

v1BotVersion

La versione del bot Amazon Lex V1 che è l'origine della migrazione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

v2BotId

L'identificatore univoco di Amazon Lex V2 che è la destinazione della migrazione.

- Tipo: stringa

Vincoli di lunghezza: lunghezza fissa di 10.

Modello: `^[0-9a-zA-Z]+$`

Campo obbligatorio: no

v2BotRole

Il ruolo IAM utilizzato da Amazon Lex per eseguire il bot Amazon Lex V2.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 20. La lunghezza massima è 2048 caratteri.

Modello: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

OutputContext

Servizio: Amazon Lex Model Building Service

La specificazione di un contesto di output che viene impostato quando viene soddisfatto un intento.

Indice

name

Il nome del contesto.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

timeToLiveInSeconds

Il numero di secondi in cui il contesto deve essere attivo dopo il primo invio in una risposta `PostContent` or`PostText`. È possibile impostare il valore tra 5 e 86.400 secondi (24 ore).

Tipo: integer

Intervallo valido: valore minimo di 5. Valore massimo pari a 86400.

Campo obbligatorio: sì

turnsToLive

Il numero di conversazioni richiede che il contesto sia attivo. Un turno di conversazione è uno `PostContent` o una `PostText` richiesta e la risposta corrispondente di Amazon Lex.

Tipo: integer

Intervallo valido: valore minimo di 1. Valore massimo di 20.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Prompt

Servizio: Amazon Lex Model Building Service

Ottiene informazioni dall'utente. Per definire un prompt, fornite uno o più messaggi e specificate il numero di tentativi di ottenere informazioni dall'utente. Se fornisci più di un messaggio, Amazon Lex sceglie uno dei messaggi da utilizzare per richiedere conferma all'utente. Per ulteriori informazioni, consulta Amazon Lex: come funziona: come funziona.

Indice

maxAttempts

Il numero di volte in cui vengono richieste informazioni all'utente.

Tipo: integer

Intervallo valido: valore minimo di 1. Valore massimo di 5.

Campo obbligatorio: sì

messages

Una matrice di oggetti, ognuno dei quali fornisce una stringa di messaggio e il relativo tipo. È possibile specificare la stringa del messaggio in testo semplice o in Speech Synthesis Markup Language (SSML).

Tipo: matrice di oggetti Message

Membri dell'array: numero minimo di 1 elemento. Numero massimo di 15 articoli.

Campo obbligatorio: sì

responseCard

Una scheda di risposta. Amazon Lex utilizza questo prompt in fase di esecuzione, nella risposta dell'`PostText`API. Sostituisce gli attributi di sessione e i valori degli slot con i segnaposto nella scheda di risposta. Per ulteriori informazioni, consulta Utilizzo di una scheda di risposta.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 50000.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ResourceReference

Servizio: Amazon Lex Model Building Service

Descrive la risorsa che si riferisce alla risorsa che si sta tentando di eliminare. Questo oggetto viene restituito come parte dell'`ResourceInUseException`eccezione.

Indice

name

Il nome della risorsa che sta utilizzando la risorsa che si sta tentando di eliminare.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `[a-zA-Z_]+`

Campo obbligatorio: no

version

La versione della risorsa che utilizza la risorsa che si sta tentando di eliminare.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Slot

Servizio: Amazon Lex Model Building Service

Identifica la versione di uno slot specifico.

Indice

name

Il nome dello slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z](-|_|.)?)+$`

Campo obbligatorio: sì

slotConstraint

Specifica se lo slot è obbligatorio o facoltativo.

- Tipo: stringa

Valori validi: `Required | Optional`

Campo obbligatorio: sì

defaultValueSpec

Un elenco di valori predefiniti per lo slot. I valori predefiniti vengono utilizzati quando Amazon Lex non ha determinato un valore per uno slot. Puoi specificare valori predefiniti da variabili di contesto, attributi di sessione e valori definiti.

Tipo: oggetto SlotDefaultValueSpec

Campo obbligatorio: no

description

Una descrizione dello slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

obfuscationSetting

Determina se uno slot è offuscato nei registri delle conversazioni e negli enunciati memorizzati. Quando offuscate uno slot, il valore viene sostituito dal nome dello slot tra parentesi graffe ({}). Ad esempio, se il nome dello slot è «full_name», i valori offuscati vengono sostituiti con «{full_name}». Per ulteriori informazioni, consulta Slot Obfuscation.

- Tipo: stringa

Valori validi: `NONE | DEFAULT_OBFUSCATION`

Campo obbligatorio: no

priority

Indica ad Amazon Lex l'ordine in cui ottenere questo valore di slot dall'utente. Ad esempio, se l'intento ha due slot con priorità 1 e 2, AWS Amazon Lex richiama innanzitutto un valore per lo slot con priorità 1.

Se più slot condividono la stessa priorità, l'ordine in cui Amazon Lex ottiene i valori è arbitrario.

Tipo: integer

Intervallo valido: valore minimo di 0. valore massimo pari a 100.

Campo obbligatorio: no

responseCard

Un insieme di possibili risposte per il tipo di slot utilizzato dai client basati su testo. Un utente sceglie un'opzione dalla scheda di risposta, invece di utilizzare il testo per rispondere.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 50000.

Campo obbligatorio: no

sampleUtterances

Se conosci uno schema specifico con cui gli utenti potrebbero rispondere a una richiesta Amazon Lex per il valore di uno slot, puoi fornire tali enunciati per migliorare la precisione. Si tratta di un'opzione facoltativa. Nella maggior parte dei casi, Amazon Lex è in grado di comprendere le espressioni degli utenti.

Tipo: matrice di stringhe

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 200.

Campo obbligatorio: no

slotType

Il tipo di slot, un tipo di slot personalizzato definito dall'utente o uno dei tipi di slot incorporati.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Campo obbligatorio: no

slotTypeVersion

La versione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

valueElicitationPrompt

Il prompt utilizzato da Amazon Lex per ottenere il valore dello slot dall'utente.

Tipo: oggetto Prompt

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

SlotDefaultValue

Servizio: Amazon Lex Model Building Service

Un valore predefinito per uno slot.

Indice

defaultValue

Il valore predefinito per lo slot. È possibile specificare una delle seguenti opzioni:

- `#context-name.slot-name`- Il valore dello slot «slot-name» nel contesto «context-name».
- `{attribute}`- Il valore dello slot dell'attributo di sessione «attribute».
- `'value'`- Il valore discreto «value».

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 202.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

SlotDefaultValueSpec

Servizio: Amazon Lex Model Building Service

Contiene i valori predefiniti per uno slot. I valori predefiniti vengono utilizzati quando Amazon Lex non ha determinato un valore per uno slot.

Indice

defaultValueList

I valori predefiniti per uno slot. È possibile specificare più di un valore predefinito. Ad esempio, è possibile specificare un valore predefinito da utilizzare da una variabile di contesto corrispondente, un attributo di sessione o un valore fisso.

Il valore predefinito scelto viene selezionato in base all'ordine in cui vengono specificati nell'elenco. Ad esempio, se specifichi una variabile di contesto e un valore fisso in quell'ordine, Amazon Lex utilizza la variabile di contesto se è disponibile, altrimenti utilizza il valore fisso.

Tipo: matrice di oggetti SlotDefaultValue

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

SlotTypeConfiguration

Servizio: Amazon Lex Model Building Service

Fornisce informazioni di configurazione per un tipo di slot.

Indice

regexConfiguration

Un'espressione regolare utilizzata per convalidare il valore di uno slot.

Tipo: oggetto SlotTypeRegexConfiguration

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

SlotTypeMetadata

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su un tipo di slot.

Indice

createdDate

La data di creazione del tipo di slot.

Tipo: Timestamp

Campo obbligatorio: no

description

Una descrizione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. Lunghezza massima di 200.

Campo obbligatorio: no

lastUpdatedDate

La data in cui il tipo di slot è stato aggiornato. Quando si crea una risorsa, la data di creazione e la data dell'ultimo aggiornamento coincidono.

Tipo: Timestamp

Campo obbligatorio: no

name

Il nome del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: no

version

La versione del tipo di slot.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

SlotTypeRegexConfiguration

Servizio: Amazon Lex Model Building Service

Fornisce un'espressione regolare utilizzata per convalidare il valore di uno slot.

Indice

pattern

Un'espressione regolare utilizzata per convalidare il valore di uno slot.

Usa un'espressione regolare standard. Amazon Lex supporta i seguenti caratteri nell'espressione regolare:

- A-Z, a-z
- 0-9
- Caratteri Unicode («\ u <Unicode>«)

Rappresenta caratteri Unicode con quattro cifre, ad esempio «\ u0041" o «\ u005A».

I seguenti operatori di espressioni regolari non sono supportati:

- Ripetitori infiniti: *, + o {x,} senza limite superiore.
- Wild card (.)

➨Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue: AWS

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Statement

Servizio: Amazon Lex Model Building Service

Una raccolta di messaggi che trasmettono informazioni all'utente. In fase di esecuzione, Amazon Lex seleziona il messaggio da trasmettere.

Indice

messages

Una raccolta di oggetti di messaggio.

Tipo: matrice di oggetti Message

Membri dell'array: numero minimo di 1 elemento. Numero massimo di 15 elementi.

Campo obbligatorio: sì

responseCard

In fase di esecuzione, se il client utilizza l'PostTextAPI, Amazon Lex include la scheda di risposta nella risposta. Sostituisce tutti gli attributi di sessione e i valori degli slot con i segnaposto nella scheda di risposta.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 50000.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Tag

Servizio: Amazon Lex Model Building Service

Un elenco di coppie chiave/valore che identificano un bot, un alias bot o un canale bot. Le chiavi e i valori dei tag possono essere costituiti da lettere Unicode, cifre, spazi bianchi e uno dei seguenti simboli: _.:/= + - @.

Indice

key

La chiave per il tag. Le chiavi non fanno distinzione tra maiuscole e minuscole e devono essere uniche.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 128 caratteri.

Campo obbligatorio: sì

value

Il valore associato a una chiave. Il valore può essere una stringa vuota ma non può essere nullo.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 0. La lunghezza massima è 256 caratteri.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

UtteranceData

Servizio: Amazon Lex Model Building Service

Fornisce informazioni su una singola espressione che è stata fatta al tuo bot.

Indice

count

Il numero di volte in cui l'enunciato è stato elaborato.

Tipo: integer

Campo obbligatorio: no

distinctUsers

Il numero totale di individui che hanno usato l'enunciato.

Tipo: integer

Campo obbligatorio: no

firstUtteredDate

Data in cui l'enunciato è stato registrato per la prima volta.

Tipo: Timestamp

Campo obbligatorio: no

lastUtteredDate

Data in cui l'enunciato è stato registrato l'ultima volta.

Tipo: Timestamp

Campo obbligatorio: no

utteranceString

Il testo inserito dall'utente o la rappresentazione testuale di una clip audio.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 2000.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

UtteranceList

Servizio: Amazon Lex Model Building Service

Fornisce un elenco di espressioni che sono state fatte a una versione specifica del bot. L'elenco contiene un massimo di 100 enunciati.

Indice

botVersion

La versione del bot che ha elaborato l'elenco.

-Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 64 caratteri.

Modello: `\$LATEST|[0-9]+`

Campo obbligatorio: no

utterances

Uno o più UtteranceData oggetti che contengono informazioni sugli enunciati che sono stati fatti a un bot. Il numero massimo di oggetti è 100.

Tipo: matrice di oggetti UtteranceData

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Servizio runtime Amazon Lex

Amazon Lex Runtime Service supporta i tipi di dati seguenti:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

ActiveContext

Servizio: Amazon Lex Runtime Service

Un contesto è una variabile che contiene informazioni sullo stato corrente della conversazione tra un utente e Amazon Lex. Il contesto può essere impostato automaticamente da Amazon Lex quando un intento viene soddisfatto oppure può essere impostato in fase di esecuzione utilizzando l'operazione `PutContentPutText`, o`PutSession`.

Indice

name

Il nome del contesto.

▬Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 100.

Modello: `^([A-Za-z]_?)+$`

Campo obbligatorio: sì

parameters

Variabili di stato per il contesto corrente. È possibile utilizzare questi valori come valori predefiniti per gli slot negli eventi successivi.

Tipo: mappatura stringa a stringa

Voci sulla mappa: numero minimo di 0 elementi. Numero massimo di 10 elementi.

Limitazioni di lunghezza della chiave: la lunghezza minima è 1. Lunghezza massima di 100.

Valore dei vincoli di lunghezza: lunghezza minima di 1. La lunghezza massima è 1024 caratteri.

Campo obbligatorio: sì

timeToLive

Il periodo di tempo o il numero di turni in cui un contesto rimane attivo.

Tipo: oggetto ActiveContextTimeToLive

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ActiveContextTimeToLive

Servizio: Amazon Lex Runtime Service

Il periodo di tempo o il numero di turni in cui un contesto rimane attivo.

Indice

timeToLiveInSeconds

Il numero di secondi in cui il contesto deve essere attivo dopo il primo invio in una `PostText` risposta `PostContent` or. È possibile impostare il valore tra 5 e 86.400 secondi (24 ore).

Tipo: integer

Intervallo valido: valore minimo di 5. Valore massimo pari a 86400.

Campo obbligatorio: no

turnsToLive

Il numero di conversazioni richiede che il contesto sia attivo. Un turno di conversazione è uno `PostContent` o una `PostText` richiesta e la risposta corrispondente di Amazon Lex.

Tipo: integer

Intervallo valido: valore minimo di 1. Valore massimo di 20.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Button

Servizio: Amazon Lex Runtime Service

Rappresenta un'opzione da mostrare sulla piattaforma client (Facebook, Slack, ecc.)

Indice

text

Testo visibile all'utente sul pulsante.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 15 caratteri.

Campo obbligatorio: sì

value

Il valore inviato ad Amazon Lex quando un utente sceglie il pulsante. Ad esempio, considera il testo del pulsante «NYC». Quando l'utente sceglie il pulsante, il valore inviato può essere «New York City».

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 1000.

Campo obbligatorio: sì

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

DialogAction

Servizio: Amazon Lex Runtime Service

Descrive l'azione successiva che il bot deve intraprendere nella sua interazione con l'utente e fornisce informazioni sul contesto in cui avviene l'azione. Utilizzate il tipo di `DialogAction` dati per impostare l'interazione su uno stato specifico o per riportare l'interazione a uno stato precedente.

Indice

type

L'azione successiva che il bot deve intraprendere nella sua interazione con l'utente. I valori possibili sono:

- `ConfirmIntent`- L'azione successiva è chiedere all'utente se l'intento è completo e pronto per essere realizzato. Questa è una domanda sì/no come «Effettua l'ordine?»
- `Close`- Indica che non ci sarà alcuna risposta da parte dell'utente. Ad esempio, l'affermazione «Il tuo ordine è stato effettuato» non richiede una risposta.
- `Delegate`- L'azione successiva è determinata da Amazon Lex.
- `ElicitIntent`- L'azione successiva consiste nel determinare l'intento che l'utente desidera soddisfare.
- `ElicitSlot`- L'azione successiva consiste nel richiedere all'utente un valore di slot.

➖Tipo: stringa

Valori validi: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Campo obbligatorio: sì

fulfillmentState

Lo stato di adempimento dell'intento. I valori possibili sono:

- `Failed`- La funzione Lambda associata all'intento non è riuscita a soddisfare l'intento.
- `Fulfilled`- L'intento è stato raggiunto dalla funzione Lambda associata all'intento.
- `ReadyForFulfillment`- Tutte le informazioni necessarie per l'intento sono presenti e l'intento è pronto per essere soddisfatto dall'applicazione client.

➖Tipo: stringa

Valori validi: `Fulfilled | Failed | ReadyForFulfillment`

Campo obbligatorio: no

intentName

Il nome dell'intento.

- Tipo: stringa

Campo obbligatorio: no

message

Il messaggio che deve essere mostrato all'utente. Se non specifichi un messaggio, Amazon Lex utilizzerà il messaggio configurato per l'intento.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. La lunghezza massima è 1024 caratteri.

Campo obbligatorio: no

messageFormat

- `PlainText`- Il messaggio contiene testo UTF-8 semplice.
- `CustomPayload`- Il messaggio è un formato personalizzato per il cliente.
- `SSML`- Il messaggio contiene testo formattato per l'output vocale.
- `Composite`- Il messaggio contiene un oggetto JSON in escape contenente uno o più messaggi. Per ulteriori informazioni, consulta Gruppi di messaggi.

- Tipo: stringa

Valori validi: `PlainText | CustomPayload | SSML | Composite`

Campo obbligatorio: no

slots

Mappa degli slot che sono stati raccolti e dei relativi valori.

Tipo: mappatura stringa a stringa

Campo obbligatorio: no

slotToElicit

Il nome dello slot che deve essere richiesto all'utente.

- Tipo: stringa

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

GenericAttachment

Servizio: Amazon Lex Runtime Service

Rappresenta un'opzione resa all'utente quando viene visualizzato un prompt. Potrebbe essere un'immagine, un pulsante, un link o un testo.

Indice

attachmentLinkUrl

L'URL di un allegato alla scheda di risposta.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 1. La lunghezza massima è 2048 caratteri.

Campo obbligatorio: no

buttons

L'elenco delle opzioni da mostrare all'utente.

Tipo: matrice di oggetti Button

Membri dell'array: numero minimo di 0 elementi. Numero massimo 5 elementi.

Campo obbligatorio: no

imageUrl

L'URL di un'immagine che viene mostrata all'utente.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 1. La lunghezza massima è 2048 caratteri.

Campo obbligatorio: no

subTitle

Il sottotitolo visualizzato sotto il titolo.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 80.

Campo obbligatorio: no

title

Il titolo dell'opzione.

- Tipo: stringa

Limitazioni di lunghezza: lunghezza minima pari a 1. Lunghezza massima di 80.

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

IntentConfidence

Servizio: Amazon Lex Runtime Service

Fornisce un punteggio che indica la fiducia di Amazon Lex nel fatto che l'intento sia quello che soddisfa l'intento dell'utente.

Indice

score

Un punteggio che indica quanto Amazon Lex sia sicuro che un intento soddisfi l'intento dell'utente. Intervalli compresi tra 0,00 e 1,00. I punteggi più alti indicano una maggiore confidenza.

Tipo: double

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

IntentSummary

Servizio: Amazon Lex Runtime Service

Fornisce informazioni sullo stato di un intento. È possibile utilizzare queste informazioni per ottenere lo stato corrente di un intento in modo da poterlo elaborare o per riportare l'intento allo stato precedente.

Indice

dialogActionType

L'azione successiva che il bot deve intraprendere nell'interazione con l'utente. I valori possibili sono:

- `ConfirmIntent`- L'azione successiva è chiedere all'utente se l'intento è completo e pronto per essere realizzato. Questa è una domanda sì/no, ad esempio «Effettua l'ordine?»
- `Close`- Indica che non ci sarà alcuna risposta da parte dell'utente. Ad esempio, l'affermazione «Il tuo ordine è stato effettuato» non richiede una risposta.
- `ElicitIntent`- L'azione successiva consiste nel determinare l'intento che l'utente desidera soddisfare.
- `ElicitSlot`- L'azione successiva consiste nel richiedere all'utente un valore di slot.

▬Tipo: stringa

Valori validi: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Campo obbligatorio: sì

checkpointLabel

Un'etichetta definita dall'utente che identifica un intento particolare. È possibile utilizzare questa etichetta per tornare a un intento precedente.

Utilizzate il `checkpointLabelFilter` parametro dell'`GetSessionRequest`operazione per filtrare gli intenti restituiti dall'operazione in base a quelli con solo l'etichetta specificata.

▬Tipo: stringa

Limitazioni di lunghezza: lunghezza minima di 1. Lunghezza massima di 255.

Modello: `[a-zA-Z0-9-]+`

Campo obbligatorio: no

confirmationStatus

Lo stato dell'intento dopo che l'utente ha risposto alla richiesta di conferma. Se l'utente conferma l'intento, Amazon Lex imposta questo campo su. `Confirmed` Se l'utente nega l'intenzione, Amazon Lex imposta questo valore su. `Denied` I valori possibili sono:

- `Confirmed`- L'utente ha risposto «Sì» alla richiesta di conferma, confermando che l'intento è completo e che è pronto per essere soddisfatto.
- `Denied`- L'utente ha risposto «No» alla richiesta di conferma.
- `None`- All'utente non è mai stata richiesta la conferma; oppure all'utente è stata richiesta ma non ha confermato o negato la richiesta.

- Tipo: stringa

Valori validi: `None | Confirmed | Denied`

Campo obbligatorio: no

fulfillmentState

Lo stato di adempimento dell'intento. I valori possibili sono:

- `Failed`- La funzione Lambda associata all'intento non è riuscita a soddisfare l'intento.
- `Fulfilled`- L'intento è stato raggiunto dalla funzione Lambda associata all'intento.
- `ReadyForFulfillment`- Tutte le informazioni necessarie per l'intento sono presenti e l'intento è pronto per essere soddisfatto dall'applicazione client.

- Tipo: stringa

Valori validi: `Fulfilled | Failed | ReadyForFulfillment`

Campo obbligatorio: no

intentName

Il nome dell'intento.

- Tipo: stringa

Campo obbligatorio: no

slots

Mappa degli slot che sono stati raccolti e dei relativi valori.

Tipo: mappatura stringa a stringa

Campo obbligatorio: no

slotToElicit

Lo slot successivo da richiedere all'utente. Se non c'è uno slot da ricercare, il campo è vuoto.

-Tipo: stringa

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

PredictedIntent

Servizio: Amazon Lex Runtime Service

Un intento suggerito da Amazon Lex soddisfa l'intento dell'utente. Include il nome dell'intento, la fiducia che Amazon Lex ha nel fatto che l'intento dell'utente sia soddisfatto e gli slot definiti per l'intento.

Indice

intentName

Il nome dell'intento suggerito da Amazon Lex soddisfa l'intento dell'utente.

- Tipo: stringa

Campo obbligatorio: no

nluIntentConfidence

Indica quanto Amazon Lex sia sicuro che un intento soddisfi l'intento dell'utente.

Tipo: oggetto IntentConfidence

Campo obbligatorio: no

slots

Lo slot e i valori degli slot associati all'intento previsto.

Tipo: mappatura stringa a stringa

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

ResponseCard

Servizio: Amazon Lex Runtime Service

Se configuri una scheda di risposta durante la creazione dei bot, Amazon Lex sostituisce gli attributi di sessione e i valori degli slot disponibili, quindi li restituisce. La scheda di risposta può anche provenire da una funzione Lambda (`dialogCodeHooke` `fulfillmentActivity` da un intento).

Indice

contentType

Il tipo di contenuto della risposta.

- Tipo: stringa

Valori validi: `application/vnd.amazonaws.card.generic`

Campo obbligatorio: no

genericAttachments

Una serie di oggetti allegati che rappresentano opzioni.

Tipo: matrice di oggetti GenericAttachment

Membri dell'array: numero minimo di 0 elementi. Numero massimo di 10 elementi.

Campo obbligatorio: no

version

La versione del formato della scheda di risposta.

- Tipo: stringa

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli AWS SDK specifici della lingua, consulta quanto segue:

- AWS SDK per C++

- AWS SDK per Java V2
- AWS SDK per Ruby V3

SentimentResponse

Servizio: Amazon Lex Runtime Service

Il sentimento espresso in un enunciato.

Quando il bot è configurato per inviare enunciati ad Amazon Comprehend per l'analisi del sentiment, questa struttura di campo contiene il risultato dell'analisi.

Indice

sentimentLabel

La valutazione dedotta in cui Amazon Comprehend ha la massima affidabilità.

- Tipo: stringa

Campo obbligatorio: no

sentimentScore

La probabilità che la valutazione sia stata dedotta correttamente.

- Tipo: stringa

Campo obbligatorio: no

Vedi anche

Per ulteriori informazioni sull'utilizzo di questa API in uno degli SDK specifici della lingua, consulta quanto segue AWS :

- AWS SDK per C++
- AWS SDK per Java V2
- AWS SDK per Ruby V3

Cronologia dei documenti per Amazon Lex

- Ultimo aggiornamento della documentazione: 9 settembre 2021

La tabella che segue descrive le modifiche importanti apportate a ogni versione di Amazon Lex. Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile abbonarti a un feed RSS.

Modifica	Descrizione	Data
Nuova caratteristica	Amazon Lex ora supporta la lingua coreana (ko-KR). Per ulteriori informazioni, consulta Lingue supportate da Amazon Lex.	9 settembre 2021
Nuova caratteristica	Amazon Lex ora supporta la lingua inglese (indiana). Per ulteriori informazioni, consulta Lingue supportate in Amazon Lex.	15 luglio 2021
Nuova caratteristica	Amazon Lex ora fornisce uno strumento per migrare un bot all'API Amazon Lex V2. Per ulteriori informazioni, consulta Migrazione di un bot.	13 luglio 2021
Nuova caratteristica	Amazon Lex ora supporta la lingua giapponese (Giappone). Per ulteriori informazioni, consulta Lingue supportate da Amazon Lex.	1 aprile 2021
Nuova caratteristica	Amazon Lex ora supporta le versioni locali tedesco (tedesco) (de-DE) e spagnolo	23 novembre 2020

	(America Latina) (es-419). Per ulteriori informazioni, consulta Lingue supportate da Amazon Lex.	
Nuova caratteristica	Amazon Lex ora supporta l'utilizzo di contesti per gestire gli intenti di attivazione. Per ulteriori informazioni, vedere Setting del contesto intent.	19 novembre 2020
Nuova caratteristica	Amazon Lex ora supporta le versioni locali francese (fr-FR), francese canadese (fr-CA), italiano (it-IT) e spagnolo (es-ES). Per un elenco completo delle lingue supportate, consulta Lingue supportate da Amazon Lex.	11 novembre 2020
Nuova caratteristica	Amazon Lex ora supporta la lingua spagnolo (Stati Uniti) (es-US). Per ulteriori informazioni, consulta Lingue supportate da Amazon Lex.	22 settembre 2020
Nuova caratteristica	Amazon Lex ora supporta la lingua inglese (britannica) (en-GB). Per ulteriori informazioni, consulta Lingue supportate da Amazon Lex.	15 settembre 2020
Nuova caratteristica	Amazon Lex ora supporta la lingua inglese (australiana) (en-AU). Per ulteriori informazioni, consulta Lingue supportate da Amazon Lex.	8 settembre 2020

Nuova caratteristica	Amazon Lex ha ora 7 nuovi intenti integrati e 9 nuovi tipi di slot integrati. Per ulteriori informazioni, consulta Intenti e tipi di slot incorporati.	8 settembre 2020
Nuovo esempio	Scopri come creare un bot Amazon Lex che gli agenti dell'assistenza clienti possano utilizzare per rispondere alle domande dei clienti cercando le risposte con Amazon Kendra. Per ulteriori informazioni, vedere Esempio: Call Center Agent Assistant.	10 agosto 2020
Nuova caratteristica	Amazon Lex può ora restituire fino a quattro intenti alternativi in base ai punteggi di fiducia. Per ulteriori informazioni, vedere Utilizzo dei punteggi di fiducia.	6 agosto 2020
Espansione regionale	Amazon Lex è ora disponibile nelle Asia Pacifico (Tokyo) (ap-northeast-1).	30 giugno 2020
Nuova caratteristica	Amazon Lex ora supporta la ricerca negli indici Amazon Kendra per trovare le risposte alle domande più frequenti. Per ulteriori informazioni, consulta AMAZON.KendraSearchIntent.	11 giugno 2020

Nuova caratteristica	Amazon Lex ora restituisce ulteriori informazioni nei registri delle conversazioni. Per ulteriori informazioni, consulta Visualizzazione dei registri di testo in Amazon CloudWatch Logs.	9 giugno 2020
Espansione regionale	Amazon Lex è ora disponibile nelle Asia Pacifico (Singapore), Europa (Francoforte) (eu-central-1) e Europa (Londra).	23 aprile 2020
Nuova caratteristica	Amazon Lex ora supporta la codifica. Puoi utilizzare il tagging per identificare le risorse, allocare i costi e controllare l'accesso. Per ulteriori informazioni, consulta Tagging delle risorse Amazon Lex.	12 marzo 2020
Nuova caratteristica	Amazon Lex ora supporta le espressioni regolari per AMAZON. AlphaNumeric tipo di slot integrato. Per ulteriori informazioni, consulta AMAZON. AlphaNumeric.	6 febbraio 2020

Nuova caratteristica	Amazon Lex è ora in grado di registrare le informazioni sulle conversazioni e offuscare i valori degli slot in tali registri. Per ulteriori informazioni, consulta Creazione di Conversation Logs e Offuscamento degli slot.	19 dicembre 2019
Espansione regionale	Amazon Lex è ora disponibile nelle Asia Pacifico (Sydney) (ap-southeast-2).	17 dicembre 2019
Nuova caratteristica	Amazon Lex è ora conforme alla normativa HIPAA. Per ulteriori informazioni, consulta Compliance Validation per Amazon Lex.	10 dicembre 2019
Nuova caratteristica	Amazon Lex può ora inviare gli enunciati degli utenti ad Amazon Comprehend per analizzare il sentimento dell'enunciato. Per ulteriori informazioni, consulta Analisi emozione.	21 novembre 2019
Nuova caratteristica	Amazon Lex è ora conforme allo standard SOC. Per ulteriori informazioni, consulta Compliance Validation per Amazon Lex.	19 novembre 2019

Nuova caratteristica	Amazon Lex è ora conforme allo standard PCI. Per ulteriori informazioni, consulta Compliance Validation per Amazon Lex.	17 ottobre 2019
Nuova caratteristica	Aggiunto il supporto per l'aggiunta di un checkpoint a un intento in modo da poter facilmente tornare all'intento durante una conversazione. Per ulteriori informazioni, consulta Gestione delle sessioni.	10 ottobre 2019
Nuova caratteristica	Aggiunto il supporto per `AMAZON.FallbackIntent` in modo che il bot possa gestire situazioni in cui l'input utente non sia come previsto. Per ulteriori informazioni, consulta AMAZON. FallbackIntent.	3 ottobre 2019
Nuova caratteristica	Amazon Lex ti consente di gestire le informazioni di sessione per i tuoi bot. Per ulteriori informazioni, consulta Gestione delle sessioni con l'API Amazon Lex.	8 agosto 2019
Espansione regionale	Amazon Lex è ora disponibile negli Stati Uniti occidentali (Oregon) (us-west-2).	8 maggio 2018

Nuova caratteristica	È stato aggiunto il supporto per l'esportazione e l'importazione in formato Amazon Lex. Per ulteriori informazioni, consulta Importazione ed esportazione di bot, intenti e tipi di slot Amazon Lex.	13 febbraio 2018
Nuova caratteristica	Amazon Lex ora supporta messaggi di risposta aggiuntivi per i bot. Per ulteriori informazioni, consulta la sezione relativa alle risposte.	8 febbraio 2018
Espansione regionale	Amazon Lex è ora disponibile nelle Europa (Irlanda) (eu-west-1).	21 Novembre 2017
Nuova caratteristica	È stato aggiunto il supporto per la distribuzione di bot Amazon Lex su Kik. Per ulteriori informazioni, consulta Integrazione di un bot Amazon Lex con Kik.	20 Novembre 2017
Nuova caratteristica	Aggiunto il supporto per i nuovi tipi di slot integrati e gli attributi di richiesta. Per ulteriori informazioni, consultare Tipi di slot integrati e Impostazione attributi richiesta.	3 Novembre 2017
Nuova caratteristica	Aggiunta l'esportazione alla caratteristica Alexa Skills Kit. Per ulteriori informazioni, consultare Esportazione in una competenza di Alexa.	7 settembre 2017

Nuova caratteristica	Aggiunto il supporto dei sinonimi per i valori dei tipi di slot. Per ulteriori informazioni, consultare Tipi di slot personalizzati.	31 agosto 2017
Nuova caratteristica	Aggiunta l'integrazione AWS CloudTrail. Per ulteriori informazioni, consulta Monitoraggio delle chiamate API Amazon Lex conAWS CloudTrail log.	15 agosto 2017
Documentazione estesa	Aggiunti esempi di nozioni di base per AWS CLI. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/lex/latest/dg/gs-cli.html.	22 maggio 2017
Nuova guida	La prima versione della Guida per l'utente di Amazon Lex.	19 aprile 2017

Glossario per AWS

Per la terminologia AWS più recente, consultare il glossario AWS nella documentazione di riferimento per Glossario AWS.