aws

Guida per sviluppatori V2

Amazon Lex

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Lex: Guida per sviluppatori V2

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Cos'è Amazon Lex V2?

Amazon Lex V2 è un servizio AWS per la creazione di interfacce conversazionali per applicazioni che utilizzano voce e testo. Amazon Lex V2 offre le funzionalità avanzate e la flessibilità della comprensione del linguaggio naturale (NLU) e del riconoscimento vocale automatico (ASR), così puoi creare esperienze utente estremamente coinvolgenti con interazioni conversazionali realistiche e creare nuove categorie di prodotti.

Amazon Lex V2 consente a qualsiasi sviluppatore di creare rapidamente bot conversazionali. Con Amazon Lex V2, non è necessaria alcuna esperienza di deep learning: per creare un bot, devi specificare il flusso di conversazione di base nella console Amazon Lex V2. Amazon Lex V2 gestisce il dialogo e regola dinamicamente 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 V2 fornisce integrazione e può essere integrata con molti altri servizi sulla piattaforma AWS, tra cui Amazon Connect, Amazon Comprehend e Amazon Kendra. AWS Lambda L'integrazione con Lambda fornisce ai bot l'accesso ai connettori aziendali serverless predefiniti per il collegamento ai dati in applicazioni SaaS come Salesforce.

Per i bot creati dopo il 17 agosto 2022, puoi utilizzare la ramificazione condizionale per controllare il flusso di conversazione con il tuo bot. Con la ramificazione condizionale puoi creare conversazioni complesse senza dover scrivere codice Lambda.

Amazon Lex V2 offre i seguenti vantaggi:

- Semplicità: Amazon Lex V2 ti guida nell'uso della console per creare il tuo bot in pochi minuti. Fornisci alcune frasi di esempio e Amazon Lex V2 crea un modello completo di linguaggio naturale attraverso il quale il bot può interagire usando voce e testo per porre domande, ottenere risposte e completare attività sofisticate.

- Tecnologie di deep learning democratizzate: Amazon Lex V2 fornisce tecnologie ASR e NLU per creare un sistema Speech Language Understanding (SLU). Tramite SLU, Amazon Lex V2 utilizza l'input vocale e testuale in linguaggio naturale, comprende l'intento alla base dell'input e soddisfa l'intento dell'utente richiamando la funzione aziendale appropriata.

Il riconoscimento vocale e la comprensione del linguaggio naturale sono alcuni dei problemi più difficili da risolvere in informatica e richiedono sofisticati algoritmi di deep learning da addestrare su enormi quantità di dati e infrastrutture. Amazon Lex V2 mette le tecnologie di deep learning alla portata di tutti gli sviluppatori. I bot Amazon Lex V2 convertono la voce in entrata in testo e comprendono l'intento dell'utente per generare una risposta intelligente, così puoi concentrarti sulla creazione di bot con valore aggiunto per i tuoi clienti e definire categorie di prodotti completamente nuove rese possibili tramite interfacce conversazionali.

- Implementazione e scalabilità senza interruzioni: con Amazon Lex V2, puoi creare, testare e distribuire i tuoi bot direttamente dalla console Amazon Lex V2. Amazon Lex V2 ti consente di pubblicare bot vocali o di testo da utilizzare su dispositivi mobili, app Web e servizi di chat (ad esempio, Facebook Messenger). Amazon Lex V2 si ridimensiona automaticamente. Non devi preoccuparti del provisioning dell'hardware e della gestione dell'infrastruttura per potenziare la tua esperienza con i bot.

- Integrazione integrata con la piattaforma AWS: Amazon Lex V2 funziona in modo nativo con altri servizi AWS, come AWS Lambda Amazon. CloudWatch 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 V2, non sono previsti costi iniziali o tariffe minime. Ti vengono addebitate solo le richieste vocali o di testo che effettui. I pay-as-you-go prezzi e il basso costo per richiesta rendono il servizio un modo conveniente per creare interfacce conversazionali. Con il piano gratuito di Amazon Lex V2, puoi provare Amazon Lex V2 con facilità senza alcun investimento iniziale.

Pagamento per Amazon Lex

Amazon Lex V2 ti addebita solo le richieste di testo o vocali che effettui. Questo modello offre un servizio a costo variabile che può crescere di pari passo con la tua attività, offrendoti al contempo

i vantaggi in termini di costi dell'infrastruttura AWS. Per ulteriori informazioni, consulta i prezzi di Amazon Lex.

Quando ti iscriviAWS, il tuo AWS account viene registrato automaticamente per tutti i serviziAWS, incluso Amazon Lex. Tuttavia, vengono addebitati solo i servizi che utilizzi. Se sei un nuovo cliente Amazon Lex, puoi iniziare a usare Amazon Lex gratuitamente. Per ulteriori informazioni, consulta il piano gratuito di AWS.

Per vedere la tua fattura, vai sul Pannello di controllo Gestione fatturazione e costi nella console AWS Billing and Cost Management. Per ulteriori informazioni sulla fatturazione dell'Account AWS, consulta la Guida per l'utente di AWS Billing. Per domande su fatturazione di AWS e sugli Account AWS contatta il supporto di AWS.

Sei un utente di Amazon Lex V2 per la prima volta?

Se utilizzi Amazon Lex V2 per la prima volta, ti consigliamo di leggere le seguenti sezioni in ordine:

1. Come funziona— Questa sezione presenta Amazon Lex V2 e le funzionalità utilizzate per creare un chatbot.
2. Guida introduttiva ad Amazon Lex V2— In questa sezione, configuri il tuo account e provi Amazon Lex V2.
3. Riferimento API: questa sezione contiene dettagli sulle operazioni API.

Funzionalità più recenti

Scopri le funzionalità più recenti di Amazon Lex V2 di seguito:

Argomenti

Supporto regionale per AWS GovCloud (Stati Uniti occidentali)

Amazon Lex V2 è ora disponibile in AWS GovCloud (Stati Uniti occidentali).

- Endpoint e quote Amazon Lex

Funzionalità di intelligenza artificiale generativa per Amazon Lex V2

Amazon Lex V2 ora ti consente di sfruttare le funzionalità di intelligenza artificiale generativa di Amazon Bedrock per il tuo bot.

- Generatore di bot descrittivo

  - Cos'è il nuovo post
  - Documentazione
- Risoluzione assistita degli slot
  - Qual è il nuovo post
  - Documentazione
- Generazione di enunciati
  - Cos'è il nuovo post
  - Documentazione
- `AMAZON.QnAIntent`(Domande frequenti sulla conversazione)
  - Qual è il nuovo post
  - Documentazione
- AWS Post sul blog di Machine Learning

Slot integrato di Amazon.Confirmation per la disambiguazione Sì/No/Forse/Non so.

Amazon Lex V2 ora offre uno slot `AMAZON.Confimation` integrato per migliorare la precisione della conferma degli slot e delle risposte Sì/No/Forse/Non so.

- Documentazione

Misurazione delle prestazioni aziendali con Analytics

Amazon Lex V2 ora offre agli utenti la possibilità di visualizzare le prestazioni degli intenti e degli slot nella dashboard di Analytics.

- Qual è il nuovo post
- Documentazione

Valutazione delle prestazioni dei bot con Test workbench

Amazon Lex V2 offre ora agli utenti la possibilità di creare ed eseguire set di test per misurare le prestazioni dei bot e migliorarne i parametri.

- Cos'è il nuovo post
- Documentazione
- AWS Post sul blog di Machine Learning

Modelli di bot verticali specifici

Amazon Lex V2 offre ora agli utenti modelli di bot predefiniti con flussi di ready-to-use conversazione insieme a dati di formazione e istruzioni di dialogo, per le modalità voce e chat.

- Qual è il nuovo post
- Documentazione

Rete di bot

Amazon Lex V2 offre ora agli utenti la possibilità di combinare più bot in un'unica rete e la possibilità di indirizzare le richieste al bot appropriato in base all'input dell'utente.

- Cos'è il nuovo post
- Documentazione

Generatore di conversazioni visive

Amazon Lex V2 ora offre un generatore di conversazioni drag and drop per progettare e visualizzare facilmente i percorsi di conversazione utilizzando gli intenti all'interno di un ambiente visivo ricco.

- Cos'è il nuovo post
- Documentazione
- AWS Post sul blog di Machine Learning

Tipo di slot composito

Amazon Lex V2 offre ora agli utenti la possibilità di combinare più slot in uno slot composito utilizzando espressioni logiche.

- Qual è il nuovo post
- Documentazione

Ramificazione condizionale

Amazon Lex V2 ora offre agli utenti la possibilità di scrivere condizioni per controllare meglio il percorso seguito dai clienti durante una conversazione con il bot.

- Qual è il nuovo post
- Documentazione

Designer automatizzato di chatbot

Amazon Lex V2 offre ora agli utenti la possibilità di progettare automaticamente un chatbot a partire dalle trascrizioni delle conversazioni. Leggi gli esempi di utilizzo.

- Qual è il nuovo post
- Documentazione
- AWS Post sul blog di Machine Learning
- Pagina Amazon Lex Automated Chatbot Designer

Suggerimenti per la fase di esecuzione

Amazon Lex V2 offre ora agli utenti la possibilità di configurare i suggerimenti di runtime per migliorare il riconoscimento delle frasi e migliorare l'evocazione dei valori degli slot.

- Qual è il nuovo post
- Documentazione

Vocabolario personalizzato

Amazon Lex V2 offre ora agli utenti la possibilità di creare un vocabolario personalizzato, un elenco di frasi che possono includere nomi propri o parole specifiche del dominio, che Amazon Lex V2 può riconoscere nell'ingresso audio.

- Cos'è il nuovo post
- Documentazione
- AWS Post sul blog di Machine Learning

Tipo di slot grammaticale

Amazon Lex V2 offre ora agli utenti la possibilità di creare grammatiche in formato XML seguendo la Speech Recognition Grammar Specification (SRGS) per raccogliere informazioni durante una conversazione.

- Cos'è il nuovo post
- Documentazione
- Post sul blog di AWS Machine Learning

Come funziona

Amazon Lex V2 consente di creare applicazioni utilizzando un'interfaccia testuale o vocale per una conversazione con un utente. Di seguito sono riportati i passaggi tipici per lavorare con Amazon Lex V2:

1. Crea un bot e aggiungi una o più lingue. Configura il bot in modo che comprenda l'obiettivo dell'utente, inizi una conversazione con l'utente per ottenere informazioni e soddisfi l'intento dell'utente.
2. Esegui il test del bot. Puoi utilizzare il client della finestra di test fornito dalla console Amazon Lex V2.
3. Pubblica una versione e crea un alias.
4. Distribuisci il bot. Puoi implementare il bot sulle tue applicazioni o piattaforme di messaggistica come Facebook Messenger o Slack

Prima di iniziare, acquisisci familiarità con i seguenti concetti e terminologia di base di Amazon Lex V2:

- Bot: un bot esegue attività automatiche come ordinare una pizza, prenotare un hotel, ordinare fiori e così via. Un bot Amazon Lex V2 è basato sulle funzionalità di riconoscimento vocale automatico (ASR) e comprensione del linguaggio naturale (NLU).

  I bot Amazon Lex V2 sono in grado di comprendere l'input dell'utente fornito con testo o voce e conversare in linguaggio naturale.
- Lingua: un bot Amazon Lex V2 può conversare in una o più lingue. Ogni lingua è indipendente dalle altre, puoi configurare Amazon Lex V2 per conversare con un utente utilizzando parole e frasi native. Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.
- Intento: un'intenzione rappresenta un'azione che l'utente desidera eseguire. Si crea un bot per supportare uno o più intenti correlati. Ad esempio, potresti creare un intento per ordinare pizze e bevande. Per ogni intento, fornisci le informazioni obbligatorie seguenti:
  - Nome dell'intento: un nome descrittivo dell'intento. Ad esempio, **`OrderPizza`**.
  - Esempi di enunciati: come un utente potrebbe trasmettere l'intento. Ad esempio, un utente potrebbe dire «Posso ordinare una pizza» o «Voglio ordinare una pizza».

- Come realizzare l'intento: come si desidera soddisfare l'intento dopo che l'utente ha fornito le informazioni necessarie. Ti consigliamo di creare una funzione Lambda per soddisfare l'intento.

  Facoltativamente, puoi configurare l'intento in modo che Amazon Lex V2 restituisca le informazioni all'applicazione client per l'adempimento necessario.

Oltre agli intenti personalizzati, Amazon Lex V2 fornisce intenti integrati per configurare rapidamente il bot. Per ulteriori informazioni, consulta Intenti incorporati.

Amazon Lex include sempre un intento di riserva per ogni bot. L'intento di riserva viene utilizzato ogni volta che Amazon Lex non è in grado di dedurre l'intento dell'utente. Per ulteriori informazioni, consulta AMAZON.FallbackIntent.

- Slot: un'intenzione può richiedere zero o più slot o parametri. Si aggiungono slot come parte della configurazione dell'intento. In fase di esecuzione, Amazon Lex V2 richiede all'utente valori di slot specifici. L'utente deve fornire i valori per tutti gli slot richiesti prima che Amazon Lex V2 possa soddisfare l'intento.

  Ad esempio, l'`OrderPizza`intento richiede slot come dimensione, tipo di crosta e numero di pizze. Per ogni slot, fornisci il tipo di slot e uno o più prompt che Amazon Lex V2 invia al client per ottenere valori dall'utente. Un utente può rispondere con un valore di slot che contiene parole aggiuntive, ad esempio «pizza grande, per favore» o «continuiamo con la versione piccola». Amazon Lex V2 comprende ancora il valore dello slot.
- Tipo di slot: ogni slot ha un tipo. Puoi creare il tuo tipo di slot oppure puoi utilizzare tipi di slot integrati. 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 V2 offre anche tipi di slot integrati. 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.
- Versione: una versione è un'istantanea numerata del tuo lavoro che puoi pubblicare per utilizzarla in diverse parti del flusso di lavoro, come sviluppo, distribuzione beta e produzione. Una volta creata una versione, puoi utilizzare un bot così com'era al momento della creazione della versione. Una volta creata, una versione rimane invariata mentre si continua a lavorare sull'applicazione.
- Alias: un alias indica una versione specifica di un bot. Con un alias, puoi aggiornare la versione utilizzata dalle tue 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.

Per un elenco delle AWS regioni in cui Amazon Lex V2 è disponibile, consulta gli endpoint e le quote di Amazon Lex V2 nella Guida generale di Amazon Web Services.

Lingue e impostazioni locali supportate da Amazon Lex V2

Amazon Lex V2 supporta una varietà di lingue e impostazioni locali. In questo argomento vengono fornite le lingue supportate, le funzionalità che supportano queste lingue e una guida specifica per lingua per migliorare le prestazioni del bot.

Lingue e impostazioni locali supportate

Amazon Lex V2 supporta le lingue e le impostazioni locali seguenti.

Codice	Lingua e impostazioni locali
ar_AE	Arabo del Golfo (Emirati Arabi Uniti)
ca_IT	catalano (Spagna)
it_AT	Tedesco (Austria)
it_IT	Tedesco (Germania)
it_AU	Inglese (Australia)
it_GB	Inglese (Regno Unito)
it_IN	Inglese (India)
en-US	Inglese (Stati Uniti)
it_ZA	Inglese (Sudafrica)
it_419	Spagnolo (America Latina)
it_IT	Spagnolo (Spagna)

Codice	Lingua e impostazioni locali
it_US	Spagnolo (Stati Uniti)
fi_FI	Finlandese (Finlandia)
fr_CA	Francese (Canada)
fr_FR	Francese (Francia)
cia_in	Hindi (India)
it_IT	Italiano (Italia)
fr_JP	Giapponese (Giappone)
it_KR	Coreano (Corea)
nl_IT	Olandese (Paesi Bassi)
No_no	norvegese (Norvegia)
it_PL	polacco (Polonia)
pt_BR	Portoghese (Brasile)
pt_PT	Portoghese (Portogallo)
it_SE	Svedese (Svezia)
zh_CN	Mandarino (RPC)
zh_HK	Cantonese (Hong Kong)

Lingue e impostazioni locali supportate dalle funzionalità di Amazon Lex V2

La tabella seguente elenca le funzionalità di Amazon Lex V2 limitate a determinate lingue e impostazioni locali. Tutte le altre funzionalità di Amazon Lex V2 sono supportate in tutte le lingue e in tutte le lingue.

Caratteristica	Lingue e impostazioni locali supportate
AMAZON. AlphaNumeric	Tutte le lingue e i locali tranne il coreano (ko_KR)
AMAZON.KendraSearchIntent	Inglese (Stati Uniti) (en_US)
Migliorare il riconoscimento vocale con un vocabolario personalizzato	Inglese (Regno Unito) (en_GB) Inglese (Stati Uniti) (en_US)
Progettista automatizzato di chatbot	Inglese (Stati Uniti) (en_US)
Disponibilità nelle Regioni	Le seguenti lingue e località non sono disponibili nelle regioni Asia Pacifico (Singapore) (ap-southeast-1) e Africa (Città del Capo) (ap-south-1): • Paesi arabi del Golfo (Emirati Arabi Uniti) (ar_AE) • Catalano (Spagna) (ca_ES) • Finlandese (Finlandia) (fi_FI) • Hindi (India) (hi_IN) • Olandese (Paesi Bassi) (nl_NL) • Norvegese (Norvegia) (no_NO) • Polacco (pl_PL) • Portoghese (Brasile) (pt_BR) • Portoghese (Portogallo) (pt_PT) • Svedese (sv_SE) • Mandarino (RPC) (zh_CN) • Cantonese (Hong Kong) (zh_HK)
Impostazione del contesto dell'intento	Inglese (Stati Uniti) (en_US)
Tipo di slot grammaticale	Inglese (Australia) (en_AU) Inglese (Regno Unito) (en_GB)

Caratteristica	Lingue e impostazioni locali supportate
	Inglese (Stati Uniti) (en_US)
Utilizzo di più valori in uno slot	Inglese (Stati Uniti) (en_US)
Miglioramento del riconoscimento dei valori degli slot con suggerimenti di runtime	Inglese (Regno Unito) (en_GB) Inglese (Stati Uniti) (en_US)
Acquisizione dei valori degli slot con stili di ortografia	Inglese (Australia) (en_AU) Inglese (Regno Unito) (en_GB) Inglese (Stati Uniti) (en_US)
Utilizzo dei punteggi di confidenza	Inglese (Regno Unito) (en_GB) Inglese (Stati Uniti) (en_US)

Guida linguistica per Amazon Lex V2

Per migliorare le prestazioni del tuo bot, devi attenerti a queste linee guida per le seguenti lingue.

Arabo

La varietà di arabo su cui Amazon Lex V2 è addestrato è l'arabo del Golfo. Tienilo a mente quando fornisci espressioni di esempio per il tuo bot. Nota che la scrittura araba è scritta da destra a sinistra.

Hindi

Amazon Lex V2 è in grado di servire utenti finali hindi che passano liberamente dall'hindi all'inglese. Se hai intenzione di creare un bot che supporti questo cambio di lingua, ti consigliamo le seguenti best practice:

- Nella definizione del bot, scrivi parole inglesi in caratteri latini.
- Almeno il 50% delle espressioni del campione deve rappresentare un cambio di lingua all'interno della stessa frase. In questi enunciati, usa lo script Devanagari per le parole in hindi e l'alfabeto latino per le parole inglesi (ad esempio, «ticket book») «).
- Se prevedi che gli utenti comunichino con il bot utilizzando parole hindi in caratteri latini o parole inglesi in caratteri Devanagari, dovresti includere esempi di parole hindi in caratteri latini (ad

esempio, «mujhe ek ticket book karni hai») e parole inglesi in caratteri Devanagari (ad esempio, «») nei tuoi esempi di enunciati.

- Se prevedi che gli utenti comunichino con il bot utilizzando frasi completamente in hindi o completamente in inglese, dovresti includere frasi di esempio completamente in un'unica lingua (ad esempio, «Voglio prenotare un biglietto»).

Regioni

Per un elenco delle AWS regioni in cui è disponibile Amazon Lex V2, consulta le regioni e gli endpoint AWS nel. Riferimenti generali di AWS

Guida introduttiva ad Amazon Lex V2

Amazon Lex V2 fornisce operazioni API che puoi integrare con le tue applicazioni esistenti. Per un elenco delle operazioni supportate, consulta l'API Reference. È possibile utilizzare una qualsiasi delle seguenti opzioni:

- SDK AWS: quando utilizzi gli SDK, le richieste ad Amazon Lex V2 vengono firmate e autenticate automaticamente utilizzando le credenziali fornite. Ti consigliamo di utilizzare un SDK per creare la tua applicazione.
- AWS CLI — Puoi utilizzarlo AWS CLI per accedere a qualsiasi funzionalità di Amazon Lex V2 senza dover scrivere alcun codice.
- Console AWS: la console è il modo più semplice per iniziare a testare e utilizzare Amazon Lex V2

Se non conosci Amazon Lex V2, ti consigliamo di leggere Come funziona prima.

Argomenti

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

Prima di utilizzare Amazon Lex V2 per la prima volta, completa le seguenti attività:

1. Registrati per AWS
2. Crea un utente IAM

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 V2. Ti vengono addebitati solo i servizi che utilizzi.

Con Amazon Lex V2, paghi solo per le risorse che utilizzi. Se sei un nuovo AWS cliente, puoi iniziare a usare Amazon Lex V2 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 AWS account, utilizza la procedura seguente per crearne uno.

Per creare un AWS account

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.

Crea un utente IAM

I servizi in AWS, come Amazon Lex V2, richiedono l'immissione di credenziali al momento dell'accesso, in modo che il servizio possa determinare se si dispone delle autorizzazioni per accedere alle risorse di proprietà di tale servizio.

Crea un account utente IAM per accedere al tuo account per Amazon Lex V2:

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

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

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 un utente amministratore chiamato `adminuser` nel tuo AWS account. Per istruzioni, consulta Creazione del primo gruppo di utenti e amministratori IAM 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

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se vogliono interagire con l' AWS AWS Management Console esterno di. Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche agli AWS CLI AWS SDK o alle API. AWS	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta Configurazione dell'uso AWS IAM Identity Center nella Guida AWS CLI per l'utente.AWS Command Line Interface • Per AWS SDK, strumenti e AWS API, consulta

Quale utente necessita dell'accesso programmatico?	Per	Come
		l'autenticazione IAM Identity Center nella Guida di riferimento agli AWS SDK e agli strumenti.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche agli SDK o alle API AWS CLI. AWS AWS	Segui le istruzioni in Uso delle credenziali temporanee con AWS risorse nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare le richieste programmatiche agli AWS CLI AWS SDK o alle API. AWS	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli AWS SDK e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimen to agli SDK e agli AWS strumenti. • Per le AWS API, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Approfondimenti

Fase 2: Guida introduttiva (console)

Fase 2: Guida introduttiva (console)

Il modo più semplice per imparare a usare Amazon Lex V2 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 V2 utilizzando un blueprint, un bot predefinito che fornisce tutta la configurazione dei bot necessaria. È necessario solo un minimo di lavoro per testare la configurazione. end-to-end
- Esercizio 2: esamina le strutture JSON inviate tra l'applicazione client e un bot Amazon Lex V2.

Argomenti

Esercizio 1: Creare un bot da un esempio

In questo esercizio, crei il tuo primo bot Amazon Lex V2 e lo testerai nella console Amazon Lex V2. Per questo esercizio, usi l'OrderFlowersesempio.

Panoramica di esempio

Usi l'OrderFlowersesempio per creare un bot Amazon Lex V2. Per ulteriori informazioni sulla struttura di un bot, consultaCome funziona.

- 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`(tipo FlowerTypes 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?"

Per creare un bot Amazon Lex V2 (console)

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli Crea bot.
3. Per il metodo di creazione, scegli Inizia con un esempio.
4. Nella sezione Bot di esempio, scegli OrderFlowersdall'elenco.
5. Nella sezione Configurazione del bot, assegna al bot un nome e una descrizione opzionale. Il nome deve essere unico nel tuo account.
6. Nella sezione Autorizzazioni, scegli Crea un nuovo ruolo con le autorizzazioni Amazon Lex di base. Questo creerà un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni necessarie ad Amazon Lex V2 per eseguire il bot.
7. Nella sezione Children's Online Privacy Protection Act (COPPA), fai la scelta appropriata.
8. Nelle sezioni Timeout della sessione e Impostazioni avanzate, lascia le impostazioni predefinite.
9. Seleziona Successivo. Amazon Lex V2 crea il tuo bot.

Dopo aver creato il bot, devi aggiungere una o più lingue supportate dal bot. Una lingua contiene gli intenti, i tipi di slot e gli slot che il bot utilizza per conversare con gli utenti.

Per aggiungere una lingua a un bot

1. Nella sezione Lingua, scegli una lingua supportata e aggiungi una descrizione.
2. Lascia i campi relativi alla soglia del punteggio di confidenza nell'interazione vocale e nella classificazione degli intenti con i relativi valori predefiniti.
3. Scegli Fine per aggiungere la lingua al bot.

Dopo aver scelto Fine, la console apre l'editor degli intenti. Puoi usare l'editor degli intenti per esaminare gli intenti usati dal bot. Quando hai finito di esaminare il bot, puoi testarlo.

Per testare il bot OrderFlowers

1. Scegli Costruisci nella parte superiore della pagina. Attendi che il bot crei.
2. Quando la build è completa, scegli Test per aprire la finestra di test.
3. Esegui il test del bot. Inizia la conversazione con uno degli esempi di espressioni, ad esempio «Vorrei raccogliere dei fiori».

Fasi successive

Ora che hai creato il tuo primo bot utilizzando un modello, puoi usare la console per creare il tuo bot. Per istruzioni sulla creazione di un bot personalizzato e per ulteriori informazioni sulla creazione di bot, consultaCostruire bot.

Esercizio 2: Rivedi il flusso della conversazione

In questo esercizio esaminerai le strutture JSON inviate tra l'applicazione client e il bot Amazon Lex V2 che hai creato inEsercizio 1: Creare un bot da un esempio. La conversazione utilizza l'RecognizeTextoperazione per generare le strutture JSON. RecognizeUtteranceRestituisce le stesse informazioni delle intestazioni HTTP nella risposta.

Le strutture JSON sono divise per ogni turno della conversazione. Un turno è una richiesta dall'applicazione client e una risposta dal bot.

Turno 1

Durante il primo turno della conversazione, l'applicazione client avvia la conversazione con il bot. Sia l'URI che il corpo della richiesta forniscono informazioni sulla richiesta.

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

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

- L'URI identifica il bot con cui l'applicazione client sta comunicando. Include anche un identificatore di sessione generato dall'applicazione client che identifica una conversazione specifica tra un utente e il bot.
- Il corpo della richiesta contiene il testo che l'utente ha digitato nell'applicazione client. In questo caso, viene inviato solo il testo, tuttavia l'applicazione può inviare informazioni aggiuntive, come gli attributi della richiesta o lo stato della sessione. Per maggiori informazioni, vedi l'operazione RecognizeText.

`Datext`, Amazon Lex V2 rileva l'intenzione dell'utente di ordinare fiori. Amazon Lex V2 sceglie uno degli slot dell'intento (`FlowerType`) e uno dei prompt per lo slot, quindi invia la seguente risposta all'applicazione client. Il client visualizza la risposta all'utente.


```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": null,
                    "PickupDate": null,
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 0.95
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "What type of flowers would you like to order?",
            "contentType": "PlainText"
```

```
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "FlowerType",
            "type": "ElicitSlot"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": null,
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Turno 2

Nel turno 2, l'utente risponde al prompt del bot Amazon Lex V2 nel turno 1 con un valore che riempie lo`FlowerType` slot.

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

La risposta per il turno 2 mostra lo`FlowerType` slot pieno e fornisce un prompt per richiedere il valore dello slot successivo.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
```

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

```
                        "resolvedValues": []
                    }
                },
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Turno 3

Al turno 3, l'utente risponde al prompt del bot Amazon Lex V2 nel turno 2 con un valore che riempie lo`PickupDate` slot.

```
{
    "text": "next monday"
}
```

La risposta per il turno 3 è stata riempita sia gli`FlowerType``PickupDate` slot che gli slot e fornisce un messaggio per richiedere l'ultimo valore dello slot.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
```

```
                        "interpretedValue": "2022-12-28",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "nluConfidence": {
            "score": 1.0
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "At what time do you want the 1 dozen roses to be picked up?",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
"sessionState": {
    "dialogAction": {
        "slotToElicit": "PickupTime",
        "type": "ElicitSlot"
    },
    "intent": {
        "confirmationState": "None",
        "name": "OrderFlowers",
        "slots": {
            "FlowerType": {
                "value": {
                    "interpretedValue": "dozen roses",
                    "originalValue": "dozen roses",
                    "resolvedValues": []
                }
```

```
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f",
        "sessionAttributes": {}
    }
}
```

Turno 4

Al turno 4, l'utente fornisce il valore finale dello slot per l'intento, l'ora in cui i fiori vengono raccolti.

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

Nella risposta, Amazon Lex V2 invia una richiesta di conferma all'utente per confermare che l'ordine è corretto. Il`dialogAction` è impostato su`ConfirmIntent` e il`confirmationState` è`None`.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
```

```
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "InProgress"
        },
        "nluConfidence": {
            "score": 1.0
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "Okay, your dozen roses will be ready for pickup by 17:00 on
2021-01-04.  Does this sound okay?",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
"sessionState": {
```

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

Turno 5

Nel turno finale, l'utente risponde con la richiesta di conferma.

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

Nella risposta, l'invio di Amazon Lex V2 indica che l'intento è stato soddisfatto impostando l'inizio`confirmationStateConfirmed` e l'inizio della`dialogAction` chiusura. Tutti i valori degli slot sono disponibili per l'applicazione client.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "Confirmed",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2021-01-04",
                            "originalValue": "next monday",
                            "resolvedValues": [
                                "2021-01-04"
                            ]
                        }
                    },
                    "PickupTime": {
                        "value": {
                            "interpretedValue": "17:00",
                            "originalValue": "5 evening",
                            "resolvedValues": [
                                "17:00"
                            ]
                        }
                    }
                },
```

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

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

Costruire bot

Crei un bot Amazon Lex V2 per interagire con i tuoi utenti e ottenere informazioni utili a svolgere un'attività. Ad esempio, puoi creare un bot che raccoglie le informazioni necessarie per ordinare un mazzo di fiori o prenotare una camera d'albergo.

Per creare un bot, sono necessarie le seguenti informazioni:

1. Il linguaggio utilizzato dal bot per interagire con il cliente. Puoi scegliere una o più lingue, ogni lingua contiene intenti, slot e tipi di slot indipendenti.
2. Gli intenti, o obiettivi, che il bot aiuta l'utente a raggiungere. Un bot può contenere uno o più scopi, come ordinare fiori o prenotare un hotel e noleggiare un'auto. È necessario decidere quali affermazioni o enunciati l'utente fa per avviare l'intento.
3. Le informazioni, o gli slot, che è necessario raccogliere dall'utente per soddisfare un intento. Ad esempio, potrebbe essere necessario richiedere all'utente il tipo di fiori o la data di inizio di una prenotazione alberghiera. È necessario definire uno o più prompt che Amazon Lex V2 utilizza per ottenere il valore dello slot dall'utente.
4. Il tipo di slot di cui hai bisogno dall'utente. Potrebbe essere necessario creare un tipo di slot personalizzato, ad esempio un elenco di fiori che un utente può ordinare, oppure utilizzare un tipo di slot integrato, ad esempio utilizzare il tipo di `AMAZON.Date` slot per la data di inizio di una prenotazione.
5. Il flusso di interazione dell'utente all'interno e tra gli intenti. È possibile configurare il flusso di conversazione per definire l'interazione tra l'utente e il bot una volta invocato l'intento. Puoi creare una funzione Lambda per convalidare e soddisfare l'intento.

Argomenti

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica ti offre un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi di dialogo tramite codice hook, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Comprendere la gestione del flusso di conversazione

Il 17 agosto 2022 Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione.

Prima della modifica, Amazon Lex V2 gestiva la conversazione suscitando slot in base alle priorità di intenti. È possibile modificare questo comportamento in modo dinamico e modificare il percorso di conversazione in base agli input degli utenti utilizzando la funzione DialogAction in Lambda. Questo può essere fatto tenendo traccia dello stato corrente della conversazione e decidendo a livello di programmazione cosa fare dopo in base allo stato della sessione.

Con questa modifica, puoi creare percorsi conversazionali e diramazioni condizionali utilizzando la console o le API di Amazon Lex V2 senza utilizzare una funzione Lambda. Amazon Lex V2 monitora lo stato della conversazione e controlla cosa fare successivamente in base alle condizioni definite al momento della creazione del bot. Ciò ti consente di creare facilmente conversazioni complesse durante la progettazione del tuo bot.

Queste modifiche ti danno il controllo completo sulla conversazione con il tuo cliente. Tuttavia, non è necessario definire un percorso. Se non specifichi un percorso di conversazione, Amazon Lex V2 crea un percorso predefinito in base alla priorità degli slot nell'intento. Puoi continuare a utilizzare le funzioni Lambda per definire i percorsi di conversazione in modo dinamico. In uno scenario del genere, la conversazione riprende in base allo stato della sessione configurato nella funzione Lambda.

Questo aggiornamento fornisce quanto segue:

- Una nuova esperienza su console per creare bot con flussi di conversazione complessi.

- Aggiornamenti alle API esistenti per la creazione di bot a supporto dei nuovi flussi di conversazione.
- Una risposta iniziale per inviare un messaggio in caso di chiamata intenzionale.
- Nuove risposte per l'elicitazione degli slot, l'invocazione Lambda come hook del codice di dialogo e la conferma.
- Possibilità di specificare i passaggi successivi ad ogni turno della conversazione.
- Valutazione delle condizioni per progettare più percorsi di conversazione.
- Impostazione dei valori degli slot e degli attributi della sessione in qualsiasi momento della conversazione.

Nota quanto segue per i bot più vecchi:

- I bot creati prima del 17 agosto 2022 continuano a utilizzare il vecchio meccanismo per gestire i flussi di conversazione. I bot creati dopo quella data utilizzano il nuovo modo di gestione del flusso di conversazione.
- I nuovi bot creati tramite importazioni dopo il 17 agosto 2022 utilizzano la nuova gestione del flusso di conversazione. Le importazioni su bot esistenti continuano a utilizzare il vecchio metodo di gestione delle conversazioni.
- Per abilitare la nuova gestione del flusso di conversazione per un bot creato prima del 17 agosto 2022, esporta il bot e quindi importa il bot utilizzando un nuovo nome di bot. Il bot appena creato dall'importazione utilizza la nuova gestione del flusso di conversazione.

Nota quanto segue per i nuovi bot creati dopo il 17 agosto 2022:

- Amazon Lex V2 segue il flusso di conversazione definito esattamente come progettato per offrire l'esperienza desiderata. È necessario configurare tutti i rami di flusso per evitare percorsi di conversazione predefiniti durante l'esecuzione.
- I passaggi di conversazione che seguono un code hook devono essere completamente configurati, poiché i passaggi incompleti possono portare al fallimento del bot. Ti consigliamo di convalidare i bot creati prima del 17 agosto 2022, poiché per questi bot non è prevista una convalida automatica dei passaggi di conversazione a seguito di un code hook.

Creazione di un bot

Puoi creare un bot con Amazon Lex V2 nei seguenti modi:

1. Usa la console Amazon Lex V2 per creare un bot utilizzando un'interfaccia Web. Per ulteriori informazioni, consulta Creazione di un bot utilizzando la console Amazon Lex V2.
2. Usa Descriptive Bot Builder per creare un bot utilizzando le funzionalità di intelligenza artificiale generativa di Amazon Bedrock. Per ulteriori informazioni, consulta Utilizzo del bot builder descrittivo.
3. Utilizza i modelli di bot per creare un bot preconfigurato che corrisponda a casi d'uso aziendali comuni. Per ulteriori informazioni, consulta Generazione di bot predefiniti da modelli di bot.
4. Utilizza un AWSSDK per creare un bot utilizzando le operazioni API.
5. Usa lo strumento di progettazione di Chatbot automatizzati per creare un bot utilizzando le trascrizioni delle chat esistenti tra agenti e clienti. Per ulteriori informazioni, consulta Utilizzo del Chatbot Designer automatizzato.
6. Importa una definizione di bot esistente. Per ulteriori informazioni, consulta Importing (Importazione).
7. Usa AWS CloudFormation per creare un bot. Per ulteriori informazioni, consulta Creazione di risorse Amazon Lex V2AWS CloudFormation.

Argomenti

Creazione di un bot utilizzando la console Amazon Lex V2

Inizia a creare il tuo bot definendo il nome, la descrizione e alcune informazioni di base.

Per creare un bot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli Crea bot.
3. Nella sezione Metodo di creazione, scegli Crea.
4. Nella sezione Configurazione del bot, assegna al bot un nome e una descrizione facoltativa.
5. Nella sezione Autorizzazioni IAM, scegli un ruolo AWS Identity and Access Management (IAM) che fornisca ad Amazon Lex V2 l'autorizzazione per accedere ad altri AWS servizi, come

Amazon. CloudWatch Puoi fare in modo che Amazon Lex V2 crei il ruolo oppure puoi scegliere un ruolo esistente con CloudWatch autorizzazioni.

6. Nella sezione Children's Online Privacy Protection Act (COPPA), scegli la risposta appropriata.
7. Nella sezione Timeout della sessione di inattività, scegli la durata in cui Amazon Lex V2 mantiene aperta una sessione con un utente. Amazon Lex V2 mantiene le variabili di sessione per tutta la durata della sessione in modo che il bot possa riprendere una conversazione con le stesse variabili.
8. Nella sezione Impostazioni avanzate, aggiungi tag che aiutano a identificare il bot e possono essere usati per controllare l'accesso e monitorare le risorse.
9. Scegli Avanti per creare il bot e passare all'aggiunta di una lingua.

Generazione di bot predefiniti da modelli di bot

Amazon Lex V2 offre soluzioni predefinite per creare esperienze su larga scala e promuovere il coinvolgimento digitale. I modelli di bot predefiniti automatizzano e standardizzano le esperienze dei clienti. I modelli di bot forniscono flussi di ready-to-use conversazione insieme ai dati di formazione e alle istruzioni di dialogo, sia per la modalità vocale che per quella di chat. Puoi accelerare la fornitura di soluzioni bot ottimizzando al contempo le risorse, in modo da poterti concentrare sulle relazioni con i clienti.

Puoi creare bot predefiniti in base al tuo caso d'uso aziendale. Puoi utilizzare la AWS CloudFormation console per selezionare le opzioni predefinite per i servizi correlati, come Amazon S3, Amazon Connect e DynamoDB.

Attualmente, Amazon Lex V2 supporta i seguenti settori verticali aziendali:

- Servizi finanziari
- Ordini al dettaglio
- Assicurazione auto
- Telecomunicazioni
- Servizi aerei
- Altre novità arriveranno presto...

Puoi creare un bot con il modello di soluzione aziendale fornito e personalizzare il bot in base alle tue esigenze aziendali.

> **Note**
>
> I modelli creano risorse esterne ad Amazon Lex V2 tramite AWS CloudFormation stack. Potrebbe essere necessario modificare lo stack in altre console come Lambda e DynamoDB.

Prerequisiti necessari per creare e implementare il modello di bot:

- Un account AWS
- Accesso ai seguenti AWS servizi:
  - Amazon Lex V2 per creare bot
  - Funzioni di accesso a Lambda for the business
  - DynamoDB per creare le tabelle
  - Accesso IAM per creare policy e ruoli
  - AWSCloudFormationper eseguire lo stack
- Accesso IAM e credenziali a chiave segreta
- Istanza Amazon Connect (opzionale)

> **Note**
>
> L'uso di diversi AWS servizi comporta i rispettivi costi di utilizzo per ciascun servizio.

Per creare un bot a partire dai modelli Amazon Lex V2:

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Seleziona il pulsante arancione che dice Crea bot da un modello.
3. Seleziona il settore verticale aziendale che desideri utilizzare per il tuo modello di bot. NOTA: attualmente sono disponibili 5 modelli di bot. Presto ne arriveranno altri.
4. Seleziona Crea per il modello che desideri utilizzare. Si apre una scheda in AWS CloudFormation cui è possibile modificare i parametri dello AWS CloudFormation stack. Tutte le opzioni per il modello che hai scelto sono già state completate. Puoi anche saperne di più su come funziona il modello di bot selezionando Ulteriori informazioni.

5. Nella AWS CloudFormation console, AWS CloudFormation crea una configurazione predefinita per ciascuno dei valori del modello scelto. Puoi anche selezionare il nome dello stack, AWS CloudFormation i parametri, la tabella Amazon DynamoDB e i parametri (opzionali) di Amazon Connect.
6. Nella parte inferiore della finestra, seleziona Crea pila.
7. AWS CloudFormationelabora la richiesta in background per diversi minuti per configurare il tuo nuovo bot. NOTA: il processo crea automaticamente risorse per una tabella DynamoDB, un flusso di contatti Amazon Connect e un'istanza Amazon Connect. Puoi monitorare i progressi nella AWS CloudFormation console e quindi tornare alla console Amazon Lex V2 una volta completata la creazione CloudFormation dello stack.
8. Se creato con successo, viene visualizzato un messaggio e puoi selezionare Vai all'elenco dei bot per andare alla pagina Bot, dove troverai il tuo nuovo bot pronto per essere testato e utilizzato.

Configurazione del modello di bot

Funzioni Lambda: il modello di bot crea automaticamente le funzioni Lambda necessarie per la distribuzione. Se più bot fanno parte della soluzione modello, nei parametri sono elencate più funzioni Lambda. AWS CloudFormation Se disponi di funzioni Lambda esistenti da implementare con il tuo bot, puoi inserire il nome della tua funzione Lambda personalizzata.

Amazon DynamoDB: il modello di bot crea automaticamente la tabella DynamoDB necessaria per caricare i dati delle policy di esempio. Puoi anche inserire il nome della tua tabella DynamoDB personalizzata. La tua tabella DynamoDB personalizzata deve essere formattata nello stesso modo della tabella predefinita creata dalla distribuzione del modello di bot.

Amazon Connect: puoi configurare la tua istanza Amazon Connect in modo che funzioni con il tuo nuovo modello di bot inserendo l'ConnectInstanceARN e un codice univocoContactFlowName. Con Amazon Connect, puoi testare il tuo bot utilizzando un sistema IVR dall'inizio alla fine.

Risoluzione dei problemi relativi al modello di bot

- Verifica di disporre delle autorizzazioni appropriate per creare il modello che stai scegliendo. Gli utenti necessitano diCloudFormation: CreateStack autorizzazione insieme alle autorizzazioni per le AWS risorse elencate nel modello. Un elenco di risorse che richiedono le autorizzazioni dell'utente si trova nella parte inferiore della pagina Crea modello.
- Se il modello di bot non viene creato, il banner rosso all'interno della console Amazon Lex V2 fornisce un collegamento allo AWS CloudFormation stack responsabile della creazione del

modello. All'interno della AWS CloudFormation console, puoi visualizzare la scheda eventi per visualizzare l'errore specifico che ha causato il fallimento del modello. Dopo aver esaminato l'AWS CloudFormationerrore, consulta Risoluzione dei problemi CloudFormation per ulteriori informazioni.

- I modelli di entrambi funzionano solo con i dati di esempio. Devi compilare la tabella DynamoDB con i tuoi dati per far funzionare i modelli con i tuoi dati personalizzati.

Utilizzo del Chatbot Designer automatizzato

> **Note**
>
> Puoi utilizzare solo le trascrizioni in lingua inglese (USA).

L'Automated Chatbot Designer ti aiuta a progettare bot a partire da trascrizioni di conversazioni esistenti. Analizza le trascrizioni e suggerisce un progetto iniziale con intenti e tipi di slot. Puoi iterare sulla progettazione del bot, aggiungere istruzioni, creare, testare e implementare il bot.

Dopo aver creato un nuovo bot o aggiunto una lingua al bot utilizzando la console o l'API Amazon Lex V2, puoi caricare trascrizioni di conversazioni tra due parti. Il progettista automatico di chatbot analizza le trascrizioni e determina gli intenti e i tipi di slot per il bot. Inoltre, elenca le conversazioni che hanno influenzato la creazione di un particolare intento o tipo di slot per la tua recensione.

Utilizzi la console Amazon Lex V2 o l'API per analizzare le trascrizioni delle conversazioni e suggerire intenti e tipi di slot per un bot.

Puoi esaminare gli intenti e i tipi di slot suggeriti dopo che il progettista del chatbot ha terminato l'analisi. Dopo aver aggiunto un intento o un tipo di slot suggerito, puoi modificarlo o eliminarlo dal design del bot utilizzando la console o l'API.

Il designer automatico di chatbot supporta i file di trascrizione delle conversazioni utilizzando lo schema Contact Lens for Amazon Connect. Se utilizzi un'applicazione di contact center diversa, devi trasformare le trascrizioni delle conversazioni nel formato utilizzato dal progettista del chatbot. Per informazioni, consulta Formato di trascrizione di input.

Per utilizzare il designer automatico di chatbot, devi consentire l'accesso al ruolo IAM che esegue il designer. Per la politica IAM specifica, consultaConsenti agli utenti di utilizzare Automated Chatbot Designer. Per consentire ad Amazon Lex V2 di crittografare i dati di output con una AWS KMS chiave opzionale, è necessario aggiornare la chiave con la politica mostrata in. Consenti agli utenti di utilizzare una AWS KMS chiave per crittografare e decrittografare i file

> **Note**
>
> Se utilizzi unaKMS key, devi fornire una KMS keypolicy, indipendentemente dal IAM ruolo utilizzato.

Argomenti

Importazione delle trascrizioni delle conversazioni

L'importazione delle trascrizioni delle conversazioni è un processo in tre fasi:

1. Prepara le trascrizioni per l'importazione convertendole nel formato corretto. Se utilizzi Contact Lens per Amazon Connect, le trascrizioni sono già nel formato corretto.
2. Carica le trascrizioni in un bucket Amazon S3. Se utilizzi Contact Lens, le tue trascrizioni sono già in un bucket S3.
3. Analizza le trascrizioni utilizzando la console Amazon Lex V2 o le operazioni API. Il tempo necessario per completare la formazione dipende dal volume delle trascrizioni e dalla complessità della conversazione. In genere, vengono analizzate 500 righe di trascrizioni ogni minuto.

Ciascuno di questi passaggi è descritto nelle sezioni seguenti.

Importazione di trascrizioni da Contact Lens per Amazon Connect

Il designer automatico di chatbot di Amazon Lex V2 è compatibile con i file di trascrizione di Contact Lens. Per utilizzare i file di trascrizione di Contact Lens, devi attivare Contact Lens e annotare la posizione dei relativi file di output.

Per esportare le trascrizioni da Contact Lens

1. Attiva le lenti a contatto nella tua istanza Amazon Connect. Per istruzioni, consulta Abilitare le lenti a contatto per Amazon Connect nella guida per amministratori di Amazon Connect.

2. Nota la posizione del bucket S3 che Amazon Connect utilizza per la tua istanza. Per visualizzare la posizione, apri la pagina di archiviazione dei dati nella console Amazon Connect. Per istruzioni, consulta Update istance settings (Aggiornamento delle impostazioni dell'istanza) nella guida per amministratori di Amazon Connect.

Dopo aver attivato Contact Lens e annotato la posizione dei file di trascrizione, consulta le istruzioni Analizza le tue trascrizioni utilizzando la console Amazon Lex V2 per importare e analizzare le trascrizioni.

Prepara le trascrizioni

Prepara le tue trascrizioni creando file di trascrizione.

- Crea un file di trascrizione per conversazione che elenchi l'interazione tra le parti. Ogni interazione nella conversazione può estendersi su più righe. Puoi fornire versioni della conversazione sia redatte che non redatte.
- Il file deve essere nel formato JSON specificato in. Formato di trascrizione di input
- È necessario fornire almeno 1.000 turni di conversazione. Per migliorare la scoperta delle tue intenzioni e dei tipi di slot, dovresti fornire circa 10.000 o più turni di conversazione. Il progettista automatico di chatbot elaborerà solo i primi 700.000 turni.
- Non c'è limite al numero di file di trascrizione che puoi caricare, né ci sono restrizioni sulla dimensione dei file.

Se intendi filtrare le trascrizioni importate per data, i file devono trovarsi nella seguente struttura di directory:

```
<path or bucket root>
   --> yyyy
      --> mm
         --> dd
            --> transcript files
```

Il file di trascrizione deve contenere la data nel formato "yyyy-mm-dd" in un punto qualsiasi del nome del file.

Per esportare trascrizioni da altre applicazioni di contact center

1. Usa gli strumenti dell'applicazione del tuo contact center per esportare le conversazioni. La conversazione deve contenere almeno le informazioni specificate inFormato di trascrizione di input.
2. Trasforma le trascrizioni prodotte dall'applicazione del contact center nel formato descritto in. Formato di trascrizione di input Sei responsabile dell'esecuzione della trasformazione.

Forniamo tre script per la preparazione delle trascrizioni. Questi sono:

- Uno script per combinare le trascrizioni di Contact Lens con i log delle conversazioni di Amazon Lex V2. Le trascrizioni di Contact Lens non includono parti di conversazioni Amazon Connect che interagiscono con i bot di Amazon Lex V2. Lo script richiede l'attivazione dei log delle conversazioni per Amazon Lex V2 e le autorizzazioni appropriate per interrogare i CloudWatch log delle conversazioni e i bucket Contact Lens S3.
- Uno script per trasformare l'analisi delle chiamate di Amazon Transcribe nel formato di input Amazon Lex V2.
- Uno script per trasformare le trascrizioni delle chat di Amazon Connect nel formato di input Amazon Lex V2.

Puoi scaricare gli script da questo GitHub repository: https://github.com/aws-samples/ -integration. amazon-lex-bot-recommendation

Carica le tue trascrizioni in un bucket S3

Se utilizzi Contact Lens, i tuoi file di trascrizione sono già contenuti in un bucket S3. Per la posizione e i nomi dei file di trascrizione, consulta File di output di esempio con lenti a contatto nella guida per amministratori di Amazon Connect.

Se utilizzi un'altra applicazione di contact center e non hai configurato un bucket S3 per i tuoi file di trascrizione, segui questa procedura. Altrimenti, se disponi di un bucket S3 esistente, dopo aver effettuato l'accesso alla console Amazon S3, segui questa procedura a partire dal passaggio 5.

Per caricare i file in un bucket S3

1. Accedi alla AWS Management Console e apri la console di Amazon S3 all'indirizzo https:// console.aws.amazon.com/s3/.
2. Scegliere Create bucket (Crea bucket).

3. Assegna un nome al bucket e scegli una regione. La regione deve essere la stessa utilizzata per Amazon Lex V2. Imposta le altre opzioni come richiesto per il tuo caso d'uso.
4. Seleziona Create bucket (Crea bucket).
5. Dall'elenco dei bucket, scegli un bucket esistente o il bucket che hai appena creato
6. Scegliere Upload (Carica).
7. Aggiungi i file di trascrizione che desideri caricare.
8. Scegliere Upload (Carica).

Analizza le tue trascrizioni utilizzando la console Amazon Lex V2

Puoi utilizzare la progettazione automatica dei bot solo in una lingua vuota. Puoi aggiungere una nuova lingua a un bot esistente o creare un nuovo bot.

Per creare una nuova lingua in un nuovo bot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli Crea bot
3. Scegli Inizia con Automated Chatbot Designer. Compila le informazioni per creare il tuo nuovo bot.
4. Seleziona Next (Successivo).
5. In Aggiungi lingua al bot inserisci le informazioni per la lingua.
6. Nella sezione Posizione del file di trascrizione su S3, scegli il bucket S3 che contiene i file di trascrizione e, se necessario, il percorso locale dei file.
7. Facoltativamente, puoi scegliere quanto segue:
   - Una AWS KMS chiave per crittografare i dati della trascrizione durante l'elaborazione. Se non si seleziona una chiave, viene utilizzata una AWS KMS chiave di servizio.
   - Per filtrare le trascrizioni in base a un intervallo di date specifico. Se scegli di filtrare le trascrizioni, devono trovarsi nella struttura di cartelle corretta. Per ulteriori informazioni, consulta Prepara le trascrizioni.
8. Seleziona Done (Fatto).

Attendi che Amazon Lex V2 elabori la trascrizione. Quando l'analisi è completa, viene visualizzato un messaggio di completamento.

Come smettere di analizzare la trascrizione

Nel caso in cui sia necessario interrompere l'analisi delle trascrizioni caricate, è possibile interrompere un `BotRecommendation` processo in esecuzione, che ha lo stato di elaborazione. `BotRecommendationStatus` Puoi fare clic sul pulsante Interrompi l'elaborazione presente sul banner dopo aver inviato un lavoro dalla console o utilizzando CLI SDK per l'API. `StopBotRecommendation` Per ulteriori informazioni, consulta StopBotRecommendation

Dopo aver chiamato il`StopBotRecommendation`, la connessione interna `BotRecommendationStatus` viene impostata su `Stopping` e non viene addebitato alcun costo. Per assicurarti che il processo sia stato interrotto, puoi chiamare l'`DescribeBotRecommendation`API e verificare che lo `BotRecommendationStatus` sia`Stopped`. Di solito ci vogliono 3-4 minuti.

Non ti viene addebitato alcun costo per l'elaborazione dopo la chiamata all'`StopBotRecommendation`API.

Creazione di intenti e tipi di slot

Dopo che il progettista del chatbot ha creato gli intenti e i tipi di slot, selezioni gli intenti e i tipi di slot da aggiungere al bot. Puoi esaminare i dettagli di ogni intento e tipo di slot per aiutarti a decidere quali consigli sono più pertinenti al tuo caso d'uso.

Puoi fare clic sul nome di un intento consigliato per visualizzare gli enunciati e gli slot di esempio suggeriti dal designer del chatbot. Se selezioni Mostra trascrizioni associate, puoi anche scorrere le conversazioni che hai fornito. Queste trascrizioni influiscono sulla raccomandazione del designer del chatbot in merito. Se fai clic su un enunciato di esempio, puoi rivedere la conversazione principale e il relativo turno di dialogo, che ha influito su quell'enunciato specifico.

È possibile fare clic sul nome di un tipo di slot specifico per visualizzare i valori di slot consigliati. Se selezioni Mostra trascrizioni associate, puoi rivedere le conversazioni che hanno influenzato questo tipo di slot, evidenziando il prompt dell'agente che viene visualizzato per il tipo di slot. Se fai clic sul valore di un tipo di slot specifico, puoi rivedere la conversazione principale e il relativo turno di dialogo che ha influenzato questo valore.

Per rivedere e aggiungere intenti e tipi di slot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli il bot con cui vuoi lavorare.

3. Scegli Visualizza le lingue.
4. Dall'elenco delle lingue, scegli la lingua con cui lavorare.
5. Nella struttura della conversazione, scegli Revisione.
6. Nell'elenco degli intenti e dei tipi di slot, scegli quelli da aggiungere al bot. Puoi scegliere un intento o un tipo di slot per visualizzare i dettagli e le trascrizioni associate.

Gli intenti vengono ordinati in base alla fiducia che Amazon Lex V2 ha nel fatto che l'intento sia associato alle trascrizioni elaborate.

Formato di trascrizione di input

Di seguito è riportato il formato del file di input per generare intenti e tipi di slot per il tuo bot. Il file di input deve contenere questi campi. Gli altri campi vengono ignorati.

Il formato di input è compatibile con il formato di output di Contact Lens per Amazon Connect. Se utilizzi Contact Lens, non è necessario modificare i file di trascrizione. Per ulteriori informazioni, consulta File di output di esempio per lenti a contatto. Se si utilizza un'altra applicazione di contact center, è necessario trasformare il file di trascrizione in questo formato.

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {
        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string"
    },
    "Transcript": [
        {
            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
```

```
        }
    ]
}
```

Nel file di input devono essere presenti i seguenti campi:

- Partecipanti Identifica i partecipanti alla conversazione e il ruolo che svolgono.
- Versione La versione del formato del file di input. Sempre «1.1.0".
- ContentMetadataIndica se sono state rimosse informazioni sensibili dalla trascrizione. Imposta il `Output` campo su «Raw» se la trascrizione contiene informazioni sensibili.
- CustomerMetadataUn identificatore univoco per la conversazione.
- Trascrizione Il testo della conversazione tra le parti coinvolte nella conversazione. Ogni turno della conversazione è identificato da un identificatore univoco.

Formato di trascrizione di output

Il formato di trascrizione di output è quasi lo stesso del formato di trascrizione di input. Tuttavia, include anche alcuni metadati relativi ai clienti e un campo che elenca i segmenti che hanno influito sulla definizione degli intenti e dei tipi di slot. Puoi scaricare la trascrizione dell'output dalla pagina Review nella console o utilizzando l'API Amazon Lex V2. Per ulteriori informazioni, consulta Formato di trascrizione di input.

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {

        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string",
        "FileName": "string",
```

```
        "InputFormat": "Lex"
    },
    "InfluencingSegments": [
        {
            "Id": "string",
            "StartTurnIndex": number,
            "EndTurnIndex": number,
            "Intents": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SampleUtteranceIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ],
            "SlotTypes": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SlotValueIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ]
        }
    ],
    "Transcript": [
        {

            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
        }
    ]
}
```

- CustomerMetadata— Ci sono due campi aggiunti al `CustomerMetadata` campo, il nome del file di input che contiene la conversazione e il formato di input, che è sempre «Lex».
- InfluencingSegments— Identifica i segmenti della conversazione che hanno influenzato il suggerimento di un intento o di un tipo di slot. L'ID dell'intento o del tipo di slot identifica quello specifico influenzato dalla conversazione.

Aggiungere una lingua

Aggiungi una o più lingue e impostazioni locali al tuo bot per consentirgli di comunicare con gli utenti nella loro lingua. Definite gli intenti, gli slot e i tipi di slot separatamente per ogni lingua, in modo che i valori delle espressioni, dei prompt e degli slot siano specifici per la lingua.

Il tuo bot deve contenere almeno una lingua.

Per aggiungere una lingua al tuo bot

1. Nella sezione Nuova lingua, scegli la lingua che desideri utilizzare. Puoi aggiungere una descrizione per identificare la lingua negli elenchi.
2. Se il tuo bot supporta l'interazione vocale, nella sezione Interazione vocale, scegli la voce di Amazon Polly che Amazon Lex V2 utilizza per comunicare con l'utente. Se il tuo bot non supporta la voce, scegli Nessuno.
3. Per la soglia del punteggio di confidenza della classificazione degli intenti, imposta il valore utilizzato da Amazon Lex V2 per determinare se un intento è l'intento corretto. Puoi modificare questo valore dopo aver testato il tuo bot.
4. Scegli Add (Aggiungi).

Aggiungere intenti

Gli intenti sono gli obiettivi che i tuoi utenti vogliono raggiungere, come ordinare fiori o prenotare un hotel. Il tuo bot deve avere almeno un intento.

Per impostazione predefinita, tutti i bot contengono un unico intento integrato, l'intento di riserva. Questo intento viene utilizzato quando Amazon Lex V2 non riconosce nessun altro intento. Ad esempio, se un utente dice «Voglio ordinare fiori» all'intento di prenotare un hotel, viene attivato l'intento di riserva.

Per aggiungere un intento

1. Accedi alAWS Management Consolee apri la console Amazon Lex all'indirizzohttps://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli il bot a cui vuoi aggiungere l'intento, quindi daAggiungi linguesceglieреVisualizza lingue.
3. Scegli la lingua a cui aggiungere l'intento, quindi scegliIntenti.
4. ScegliAggiungi intento, dai un nome alla tua intenzione, quindi scegliInserisci.
5. Nell'editor degli intenti, aggiungi i dettagli del tuo intento.

   - Flusso di conversazione— Usa il diagramma del flusso della conversazione per vedere come potrebbe apparire una finestra di dialogo con il tuo bot. Puoi scegliere diverse sezioni della conversazione per passare a quella sezione dell'editor di intenti.
   - Dettagli dell'intento— Assegna all'intento un nome e una descrizione per aiutare a identificare lo scopo dell'intento. Puoi anche vedere l'identificatore univoco assegnato da Amazon Lex V2 all'intento.
   - Contesti— Imposta i contesti di input e output per l'intento. Un contesto è una variabile di stato associata a un intento. Un contesto di output viene impostato quando viene soddisfatto un intento. Un intento con un contesto di input può essere riconosciuto solo se il contesto è attivo. Un intento senza contesti di input può sempre essere riconosciuto.
   - Esempi di enunciati— Dovresti fornire 10 o più frasi che ti aspetti che i tuoi utenti utilizzino per avviare un intento. Amazon Lex V2 generalizza partendo da queste frasi per riconoscere che l'utente desidera avviare l'intento.
   - Risposta iniziale— Il messaggio iniziale inviato all'utente dopo l'invocazione dell'intento. Puoi fornire risposte, inizializzare valori e definire il passaggio successivo che Amazon Lex V2 compie per rispondere all'utente all'inizio dell'intento.
   - Slot— Definire gli slot, o i parametri, necessari per soddisfare l'intento. Ogni slot ha un tipo che definisce i valori che possono essere inseriti nello slot. Puoi scegliere tra i tuoi tipi di slot personalizzati oppure puoi scegliere un tipo di slot integrato.
   - Conferma— Queste richieste e risposte vengono utilizzate per confermare o rifiutare l'adempimento dell'intento. La richiesta di conferma chiede all'utente di rivedere i valori degli slot. Ad esempio, «Ho prenotato una camera d'albergo per venerdì. È corretto?» La risposta di declinazione viene inviata all'utente quando rifiuta la conferma. Puoi fornire risposte, impostare valori e definire il passaggio successivo che Amazon Lex V2 compie in base a una risposta di conferma o declinazione da parte dell'utente.

- Adempimento— Risposta inviata all'utente nel corso dell'adempimento. Puoi impostare aggiornamenti sullo stato di avanzamento dell'evasione all'inizio dell'evasione e periodicamente mentre l'evasione è in corso. Ad esempio, «Sto cambiando la tua password, potrebbero volerci alcuni minuti» e «Sto ancora elaborando la tua richiesta». Gli aggiornamenti relativi all'evasione degli ordini vengono utilizzati solo per le conversazioni in streaming. Puoi anche impostare un messaggio di successo successivo all'evasione, un messaggio di errore e un messaggio di timeout. Puoi inviare messaggi dopo l'evasione sia per le conversazioni in streaming che per quelle regolari. Ad esempio, se l'evasione ha esito positivo, puoi inviare «Ho cambiato la tua password». Se l'evasione non va a buon fine, puoi inviare una risposta con ulteriori informazioni, ad esempio «Non sono riuscito a cambiare la tua password, contatta l'help desk per ricevere assistenza». Se l'evasione richiede più tempo del periodo di timeout configurato, puoi inviare un messaggio per informare l'utente, ad esempio «I nostri server sono molto occupati in questo momento. Riprova la richiesta più tardi.» Puoi fornire risposte, impostare valori e definire il passaggio successivo che Amazon Lex V2 compie per rispondere all'utente.
- Risposte di chiusura— Risposta inviata all'utente dopo che l'intento è stato soddisfatto e tutti gli altri messaggi sono stati riprodotti. Ad esempio, un ringraziamento per aver prenotato una camera d'albergo. Oppure può richiedere all'utente di avviare un intento diverso, ad esempio «Grazie per aver prenotato una camera, desideri prenotare un'auto a noleggio?» Puoi fornire risposte e configurare le azioni successive dopo aver soddisfatto l'intento e aver risposto con la risposta di chiusura.
- ganci di codice— Indicare se si sta utilizzando unAWS Lambdafunzione per inizializzare l'intento e convalidare l'input dell'utente. La funzione Lambda viene specificata nell'alias utilizzato per eseguire il bot.

6. ScegliSalva l'intentoper salvare l'intento.

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Configurazione dei prompt in un ordine specifico

Puoi configurare il bot per riprodurre i messaggi in un ordine predefinito selezionando la casella perRiproduci i messaggi in ordine. Altrimenti, il bot riproduce il messaggio e le variazioni in ordine casuale.

I prompt ordinati consentono di riprodurre il messaggio e le varianti di un gruppo di messaggi in ordine tra i tentativi. È possibile utilizzare la riformulazione alternativa di un messaggio quando l'utente fornisce una risposta non valida alla richiesta o per la conferma dell'intento. In ogni slot possono essere impostate fino a due varianti del messaggio originale. Puoi scegliere se riprodurre i messaggi in ordine o in modo casuale.

Il prompt ordinato supporta tutti e quattro i tipi di messaggi: testo, risposta personalizzata al payload, SSML e gruppo di carte. Le risposte vengono ordinate all'interno dello stesso gruppo di messaggi. I diversi gruppi di messaggi sono indipendenti.

Argomenti

Esempi di enunciati

Crei esempi di enunciati che sono variazioni di frasi che ti aspetti che gli utenti utilizzino per avviare un intento. Ad esempio, per un **`BookFlight`** intento, potreste includere enunciati come i seguenti:

1. Voglio prenotare un volo
2. aiutami a prendere un volo.
3. biglietti aerei, per favore!
4. volo da {*`DepartureCity`*} a {*`DestinationCity`*}

È necessario fornire 10 o più enunciati di esempio. Fornisci esempi che rappresentino un'ampia gamma di strutture di frasi e parole che gli utenti possono pronunciare. Prendi in considerazione anche le frasi incomplete, come negli esempi 3 e 4 precedenti. Puoi anche usare gli slot che hai

definito per l'intento in un enunciato di esempio avvolgendo parentesi graffe attorno al nome dello slot, come in {} nell'esempio 4. *`DepartureCity`* Se includi nomi di slot in un enunciato di esempio, Amazon Lex V2 riempie gli slot dell'intento con i valori che l'utente fornisce nell'enunciato.

Una serie di esempi di enunciati aiuta Amazon Lex V2 a generalizzare per riconoscere efficacemente che l'utente desidera avviare l'intento.

Puoi aggiungere esempi di enunciati nell'editor degli intenti, nel generatore di conversazioni visive o con le nostre operazioni API. CreateIntentUpdateIntent Puoi anche generare enunciati di esempio automaticamente sfruttando le funzionalità di intelligenza artificiale generativa di Amazon Bedrock. Per ulteriori informazioni, consulta Generazione di enunciati.

Usa l'editor di intenti o il generatore di conversazioni visive

1. Nell'editor Intent, vai alla sezione Sample utterances. Nel generatore di conversazioni visive, trova la sezione Sample utterances nel blocco Start.
2. Nella casella con il testo trasparente**`I want to book a flight`**, digita un enunciato di esempio. Seleziona Aggiungi enunciato per aggiungere l'enunciato.
3. Visualizza gli enunciati di esempio che hai aggiunto in modalità Anteprima o Testo normale. In Testo normale, ogni riga è un enunciato separato. In modalità Anteprima, passa il mouse su un enunciato per visualizzare le seguenti opzioni:
   - Seleziona la casella di testo per modificare l'enunciato.
   - Seleziona il pulsante x a destra della casella di testo per eliminare l'enunciato.
   - Trascina il pulsante a sinistra della casella di testo per modificare l'ordine degli enunciati di esempio.
4. Usa la barra di ricerca in alto per cercare tra gli enunciati di esempio e il menu a discesa accanto ad essa per ordinarli in base all'ordine in cui hai aggiunto gli enunciati o in ordine alfabetico.

Usa un'operazione API

1. Crea un nuovo intento con l'CreateIntentoperazione o aggiornane uno esistente con l'UpdateIntentoperazione.
2. La richiesta API include un `sampleUtterances` campo, che corrisponde a una serie di SampleUtteranceoggetti.
3. Per ogni enunciato di esempio che desiderate aggiungere, aggiungete un `SampleUtterance` oggetto all'array. Aggiungete l'enunciato di esempio come valore del campo. `utterance`

4. Per modificare ed eliminare gli enunciati di esempio, inviate una `UpdateIntent` richiesta. L'elenco di enunciati fornito nel `sampleUtterances` campo sostituisce gli enunciati esistenti.

> **Important**
>
> Qualsiasi campo lasciato vuoto nella `UpdateIntent` richiesta causerà l'eliminazione delle configurazioni esistenti nell'intento. Utilizzate l'DescribeIntentoperazione per restituire la configurazione del bot e copiare nella richiesta tutte le configurazioni che non desiderate eliminare. `UpdateIntent`

Struttura dell'intento

Gli argomenti seguenti descrivono i diversi passaggi che un bot compie per realizzare un intento e come configurare ciascuno di questi passaggi:

Argomenti

- Risposta iniziale
- Slot
- Conferma
- Compimento
- Risposta di chiusura

Risposta iniziale

La risposta iniziale viene inviata all'utente dopo che Amazon Lex V2 ha determinato l'intento e prima che inizi a generare i valori degli slot. Puoi utilizzare questa risposta per informare l'utente dell'intento riconosciuto e per prepararlo alle informazioni che raccogli per adempiere all'intento.

Ad esempio, se l'intento è quello di fissare un appuntamento di assistenza per un'auto, la risposta iniziale potrebbe essere:

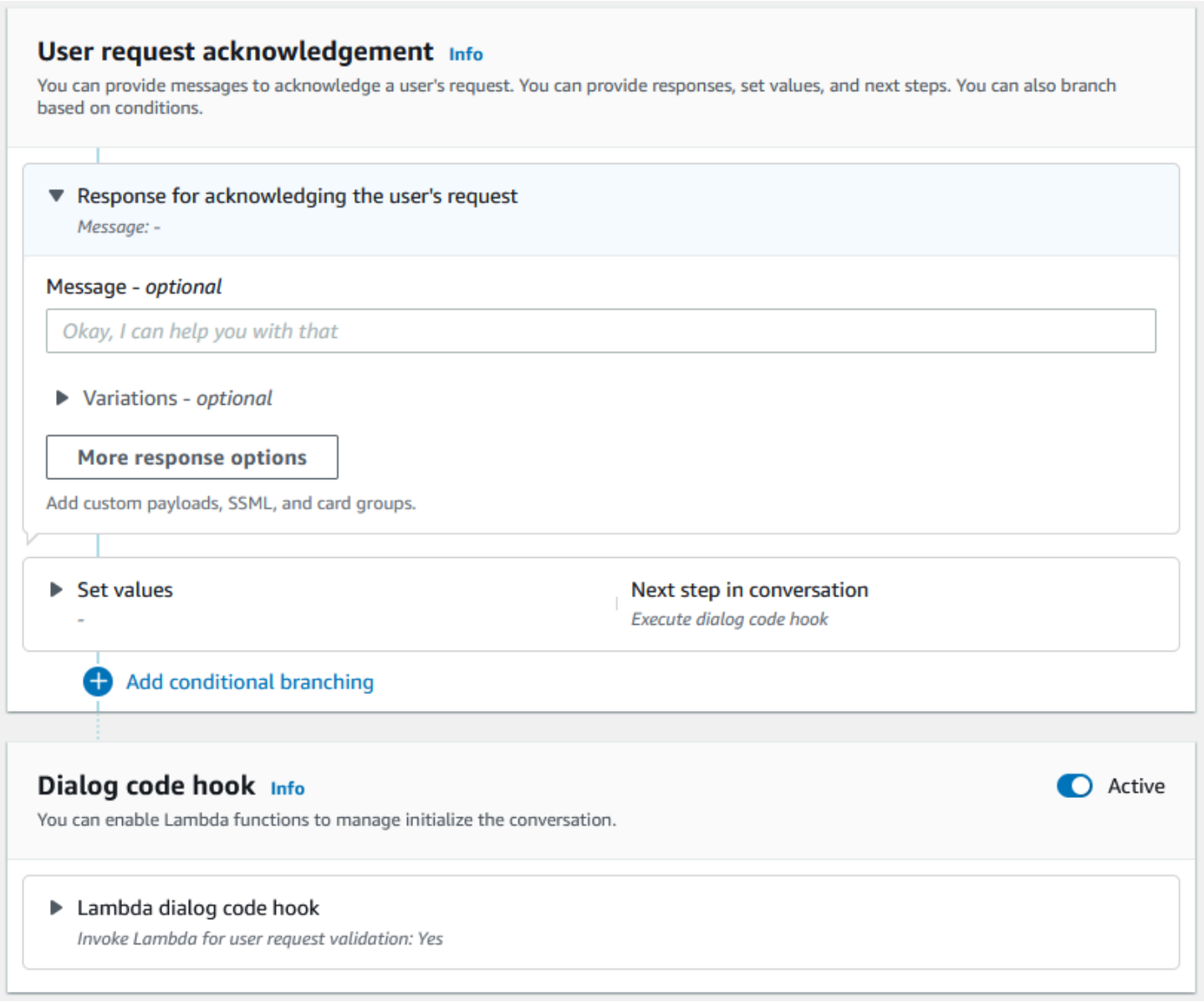
> Posso aiutarti a fissare un appuntamento. Dovrai fornire la marca, il modello e l'anno della tua auto.

Non è richiesto un messaggio di risposta iniziale. Se non ne fornisci uno, Amazon Lex V2 continua a seguire il passaggio successivo della risposta iniziale.

È possibile configurare le seguenti opzioni all'interno della risposta iniziale:

- Configurazione della fase successiva— Puoi indicare il passaggio successivo della conversazione, ad esempio passare a un'azione di dialogo specifica, evocare uno slot particolare o passare a un intento diverso. Per ulteriori informazioni, consulta Configura i passaggi successivi della conversazione.
- Impostare i valori— È possibile impostare valori per gli slot e gli attributi della sessione. Per ulteriori informazioni, consulta Imposta i valori durante la conversazione.
- Aggiungi ramificazione condizionale— Puoi applicare le condizioni dopo aver giocato la risposta iniziale. Quando una condizione restituisce il valore vero, vengono eseguite le azioni definite. Per ulteriori informazioni, consulta Aggiungi condizioni alle conversazioni delle filiali.
- Esegui l'hook del codice di dialogo— È possibile definire un code hook Lambda per inizializzare i dati ed eseguire la logica aziendale. Per ulteriori informazioni, consulta Richiama l'hook del codice di dialogo. Se l'opzione per eseguire la funzione Lambda è abilitata per l'intento, l'hook del codice di dialogo viene eseguito per impostazione predefinita. È possibile disabilitare l'hook del codice di dialogo attivando ilAttivopulsante.

In assenza di una condizione o di un passaggio successivo esplicito, Amazon Lex V2 passa allo slot successivo in ordine di priorità.



> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Slot

Gli slot sono valori forniti dall'utente per soddisfare l'intento. Esistono due tipi di slot:

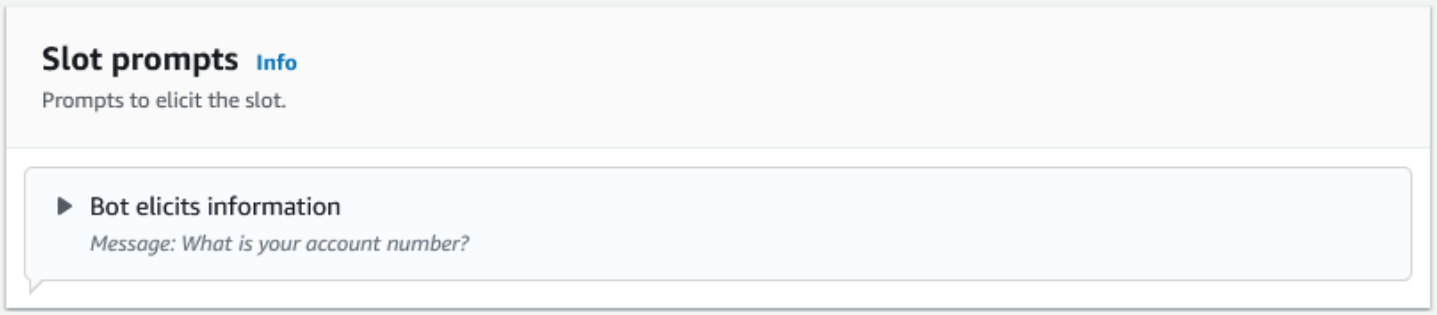
- Tipo di slot integrato— Puoi utilizzare i tipi di slot integrati per acquisire valori standard come numero, nome e città. Per un elenco dei tipi di slot integrati supportati, vedereTipi di slot integrati.
- Tipo di slot personalizzato— È possibile utilizzare tipi di slot personalizzati per acquisire valori personalizzati specifici per l'intento. Ad esempio, puoi utilizzare un tipo di slot personalizzato per acquisire il tipo di conto come «Assegno» o «Risparmio». Per ulteriori informazioni, consulta Tipo di slot personalizzato.

Per definire uno slot in un intento, è necessario configurare quanto segue:

- Informazioni sulle slot— Questo campo contiene un nome e una descrizione opzionale per lo slot. Ad esempio, puoi fornire il nome dello slot come»AccountNumber» per acquisire i numeri di conto. Se lo spazio è richiesto come parte del flusso di conversazione per soddisfare l'intento, deve essere contrassegnato come obbligatorio.
- Tipo di slot— Un tipo di slot definisce l'elenco di valori che uno slot può accettare. È possibile creare un tipo di slot personalizzato o utilizzare un tipo di slot predefinito.
- Slot prompt— Un prompt di slot è una domanda posta all'utente per raccogliere informazioni. È possibile configurare il numero di tentativi utilizzati per raccogliere informazioni e la variazione del prompt utilizzato per ogni tentativo. Puoi anche abilitare una chiamata alla funzione Lambda dopo ogni tentativo per elaborare l'input acquisito e tentare di risolvere con un input valido.
- Attendi e continua (opzionale)— Abilitando questo comportamento, gli utenti possono pronunciare frasi come «aspetta un secondo» per far sì che il bot attenda che trovi le informazioni e le fornisca. Questa opzione è abilitata solo per le conversazioni in streaming. Per ulteriori informazioni, consulta Abilitare il bot ad attendere che l'utente fornisca ulteriori informazioni.
- Risposte di acquisizione tramite slot— È possibile configurare una risposta di successo e una risposta di errore in base al risultato dell'acquisizione del valore dello slot dall'input dell'utente.
- Ramificazione condizionale— Puoi applicare le condizioni dopo aver giocato la risposta iniziale. Quando una condizione restituisce il valore vero, vengono eseguite le azioni definite. Per ulteriori informazioni, consulta Aggiungi condizioni alle conversazioni delle filiali.
- Gancio del codice di dialogo— È inoltre possibile utilizzare un code hook Lambda per convalidare i valori degli slot ed eseguire la logica aziendale. Per ulteriori informazioni, consulta Richiama l'hook del codice di dialogo.

- Tipo di input dell'utente— Puoi configurare il tipo di input in modo che il bot possa accettare una modalità specifica. Per impostazione predefinita, sono accettate sia le modalità audio che DTMF. È possibile impostarlo selettivamente su solo audio o solo DTMF.
- Timeout e lunghezze di ingresso audio— È possibile configurare i timeout audio, inclusi il timeout vocale e il timeout del silenzio. Inoltre, puoi impostare la lunghezza massima dell'audio.
- Timeout, caratteri e lunghezze di input DTMF— È possibile impostare il timeout DTMF insieme al carattere di eliminazione e al carattere finale. Inoltre, puoi impostare la lunghezza massima del DTMF.
- Lunghezza del testo— È possibile impostare la lunghezza massima per la modalità testo.

Dopo aver riprodotto la richiesta dello slot, l'utente fornisce il valore dello slot come input. Se Amazon Lex V2 non rileva il valore di uno slot fornito dall'utente, riprova a richiederlo finché non ne capisce il valore o finché non supera il numero massimo di tentativi configurato per lo slot. Utilizzando le impostazioni avanzate dei tentativi è possibile configurare i timeout, limitare il tipo di input e abilitare o disabilitare l'interruzione per la richiesta iniziale e i nuovi tentativi. Dopo ogni tentativo di acquisizione dell'input, Amazon Lex V2 può chiamare la funzione Lambda configurata per il bot con un'etichetta di chiamata fornita per i nuovi tentativi. È possibile utilizzare la funzione Lambda, ad esempio, per applicare la logica aziendale e tentare di risolverla con un valore valido. Questa funzione Lambda può essere abilitata all'internoOpzioni avanzateper i prompt degli slot.



Puoi definire le risposte che il bot deve inviare all'utente una volta inserito il valore dello slot o se viene superato il numero massimo di tentativi. Ad esempio, per un bot per la pianificazione del servizio per un'auto, puoi inviare un messaggio all'utente quando viene inserito il numero di identificazione del veicolo (VIN):

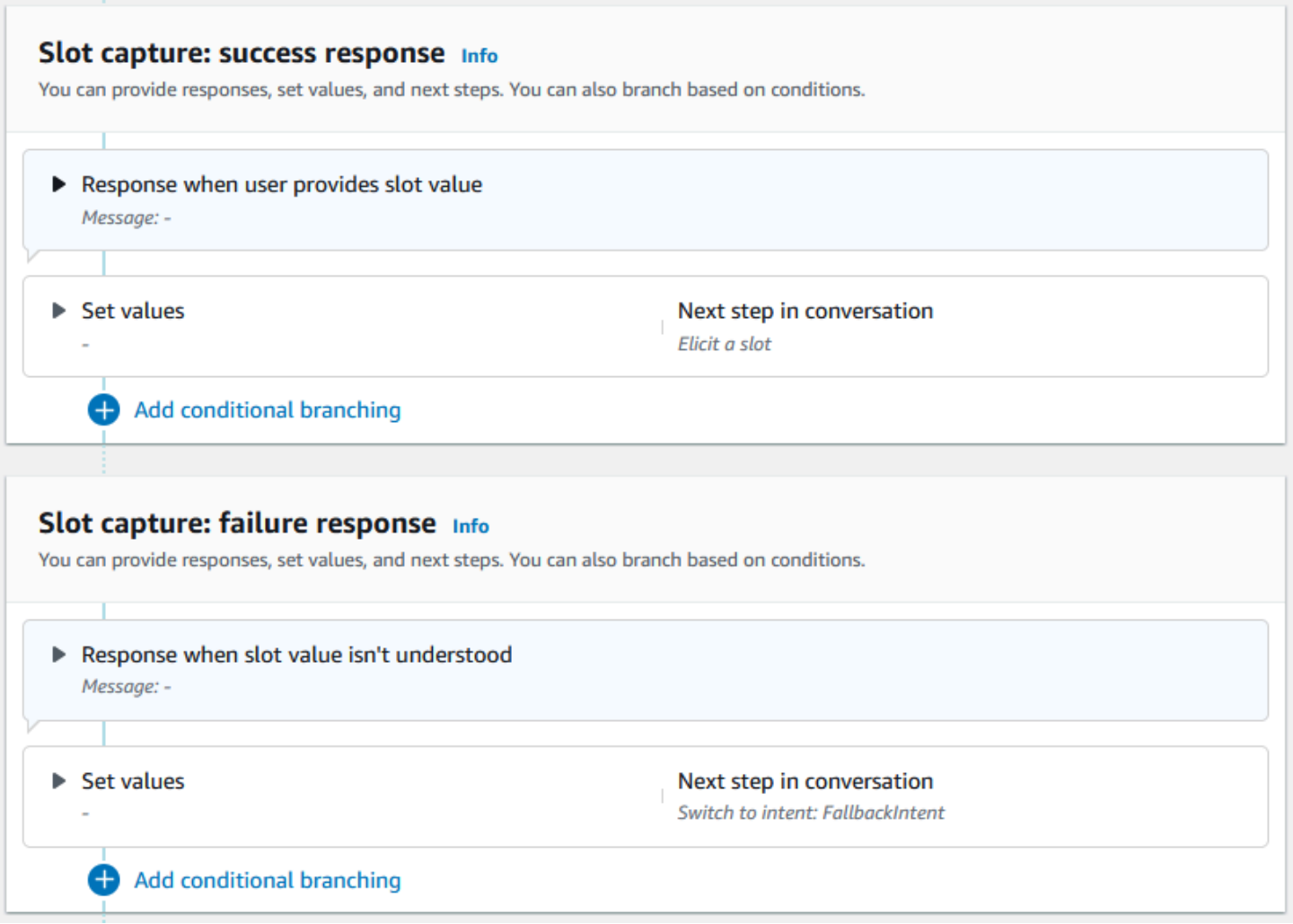
```
Grazie per aver fornito il numero VIN della tua auto. Procederò ora a fissare un appuntamento.
```

Puoi creare due risposte:

- Risposta di successo— inviato quando Amazon Lex V2 rileva il valore di uno slot.
- Risposta al fallimento— inviato quando Amazon Lex V2 non riesce a comprendere il valore di uno slot da parte dell'utente dopo il numero massimo di tentativi.

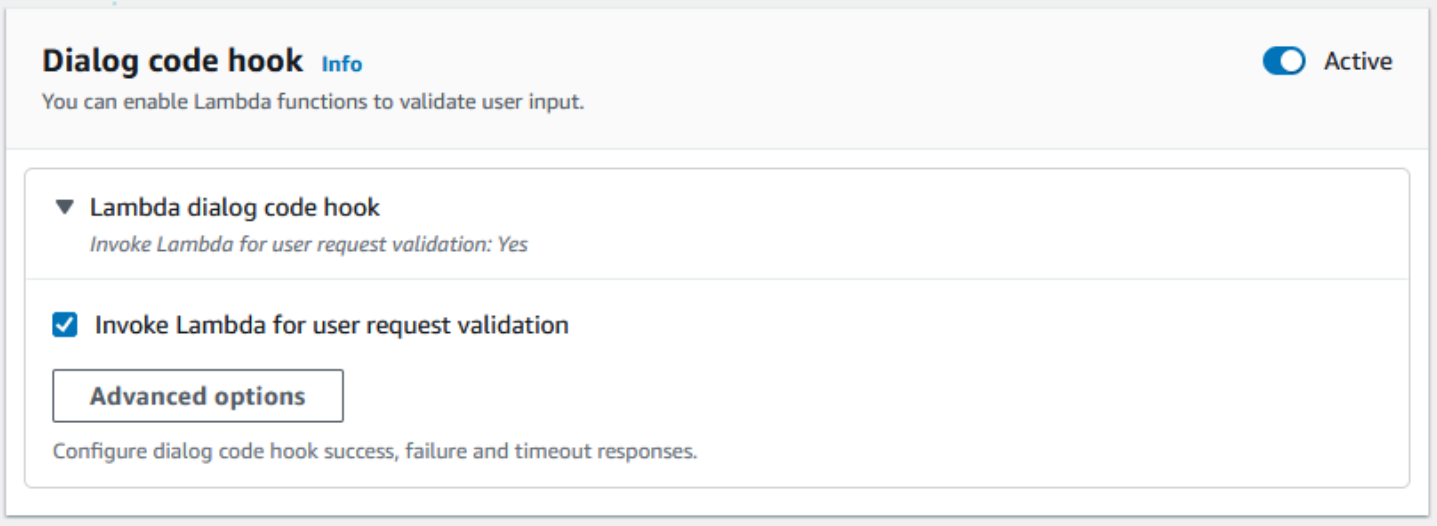
Puoi impostare valori, configurare i passaggi successivi e applicare condizioni che corrispondono a ciascuna risposta per progettare il flusso di conversazione.

In assenza di una condizione o di un passaggio successivo esplicito, Amazon Lex V2 passa allo slot successivo in ordine di priorità.



È possibile utilizzare una funzione Lambda per convalidare un valore di slot immesso da un utente e determinare quale dovrebbe essere l'azione successiva. Ad esempio, puoi utilizzare la funzione di convalida per assicurarti che il valore inserito rientri nell'intervallo corretto o che sia formattato correttamente. Per attivare la funzione Lambda, scegliInvoca la funzione Lambdacasella di controllo eAttivopulsante nelGancio del codice di dialogosezione. È possibile specificare un'etichetta di

chiamata per l'hook del codice di dialogo. Questa etichetta di chiamata può essere utilizzata nella funzione Lambda per scrivere la logica aziendale corrispondente all'elicitazione dello slot.



Gli slot non necessari per l'intento non fanno parte del flusso di conversazione principale. Tuttavia, se un'espressione utente contiene un valore che il bot identifica come corrispondente a uno slot opzionale, può popolare lo slot con quel valore. Ad esempio, se configuri un bot di business intelligence per avere un bot opzionale`City`slot e l'enunciato dell'utente**`What is the sales for April in San Diego?`**, il bot riempie lo slot opzionale con**`San Diego`**. È possibile configurare la logica aziendale per utilizzare il valore dello slot opzionale, se presente.

Gli slot non necessari per l'intento non possono essere ottenuti utilizzando i passaggi successivi. Questi passaggi possono essere compilati solo durante la richiesta di intenti (come nell'esempio precedente) o possono essere attivati impostando lo stato della finestra di dialogo all'interno della funzione Lambda. Se lo slot viene attivato utilizzando la funzione Lambda, è necessario utilizzare la funzione Lambda per decidere il passaggio successivo della conversazione dopo il completamento dell'elicitazione dello slot. Per abilitare il supporto per la fase successiva durante la creazione del bot, devi contrassegnare lo slot come richiesto per l'intento.

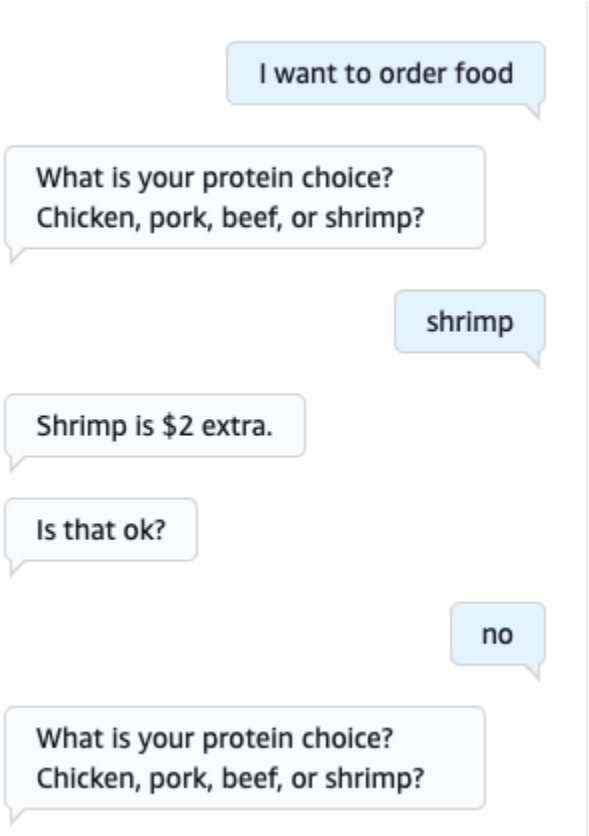
> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Gli argomenti seguenti descrivono come configurare un bot per rievocare un valore di slot che è già stato riempito e come creare uno slot composto da più valori:

Argomenti

Riattivazione degli slot

Puoi configurare il tuo bot in modo che richieda nuovamente uno slot già occupato impostando il valore di tale slot su **`null`** e impostando il passaggio successivo della conversazione in modo che torni a richiamare quello slot. Ad esempio, potresti voler richiedere nuovamente uno slot dopo che il cliente ha rifiutato la conferma della richiesta di slot sulla base di informazioni aggiuntive, come nella conversazione seguente:



È possibile configurare un ciclo a partire dalla risposta di conferma per riattivare lo slot con l'intent editor o con. Usare Visual Conversation Builder

> **Note**
>
> Puoi tornare indietro per riattivare uno slot in qualsiasi momento della conversazione, a condizione che tu abbia impostato il valore dello slot in anticipo. **`null`**

Riproduzione dell'esempio precedente con l'intent editor

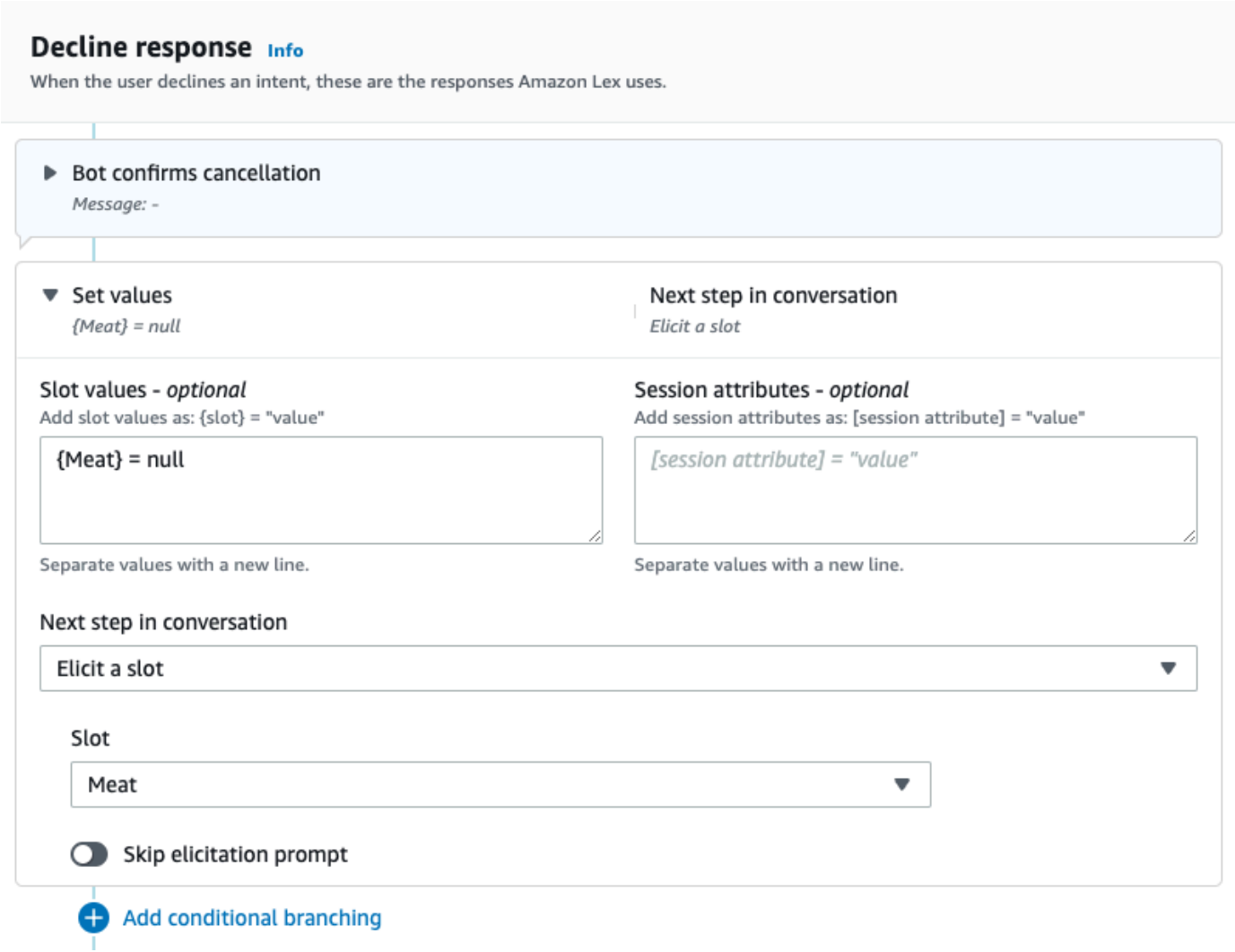
1. Nella sezione Conferma dell'editor degli intenti, seleziona la freccia destra accanto a Prompt per confermare l'intenzione di espandere la sezione.
2. Seleziona Opzioni avanzate in basso.
3. Nella sezione Rifiuta risposta, seleziona la freccia destra accanto a Imposta valori per espandere la sezione. Compila questa sezione con i seguenti passaggi, come nell'immagine qui sotto:

    a. Imposta il valore dello slot che desideri richiamare nuovamente. **`null`** In questo esempio, vogliamo riattivare lo `Meat` slot, quindi inseriamo **`{Meat} = null`** nella sezione Valori Slot.

    b. Nel menu a discesa sotto Fase successiva della conversazione, scegli Richiedi uno slot.

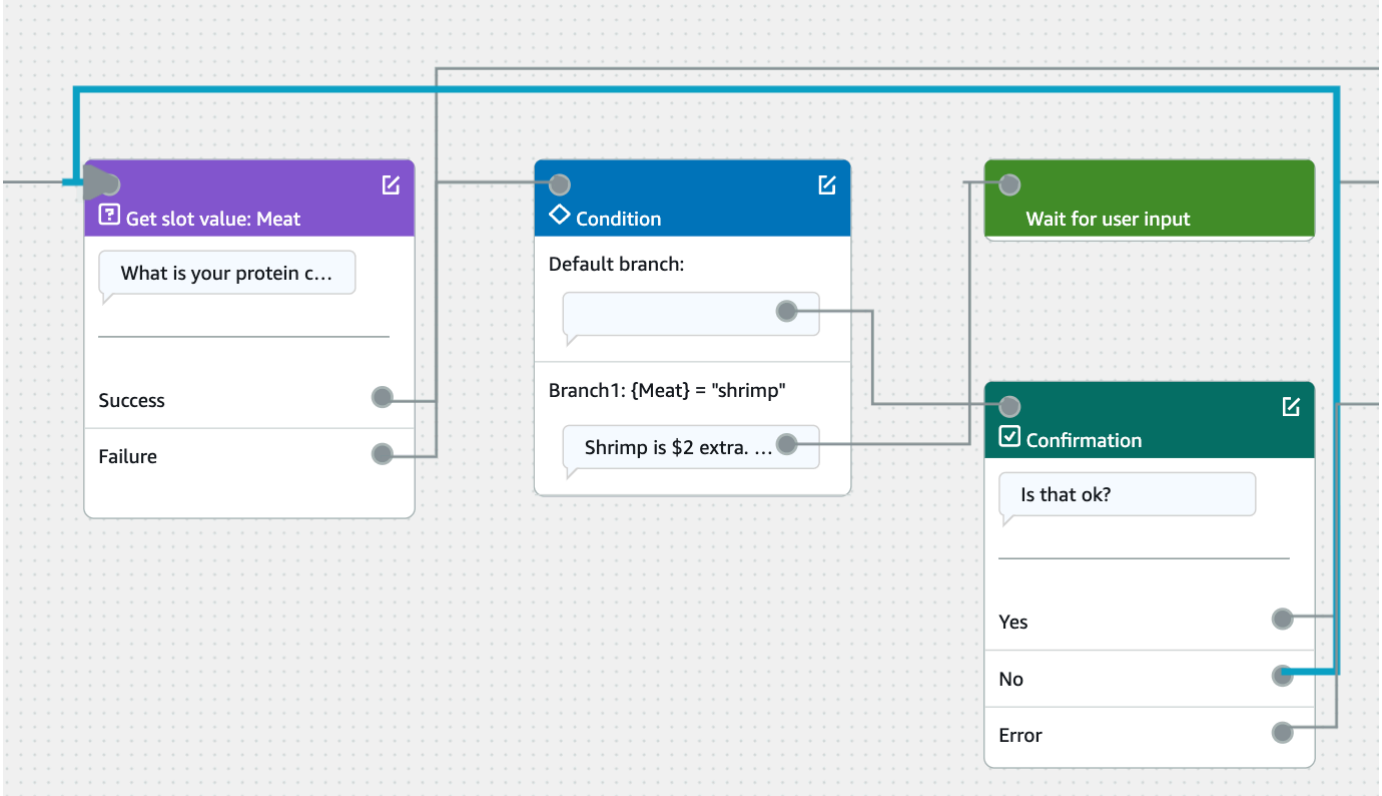
    c. Apparirà una sezione Slot. Nel menu a discesa sottostante, scegli lo slot che desideri riattivare.

    d. Seleziona Opzioni di aggiornamento per confermare le modifiche.

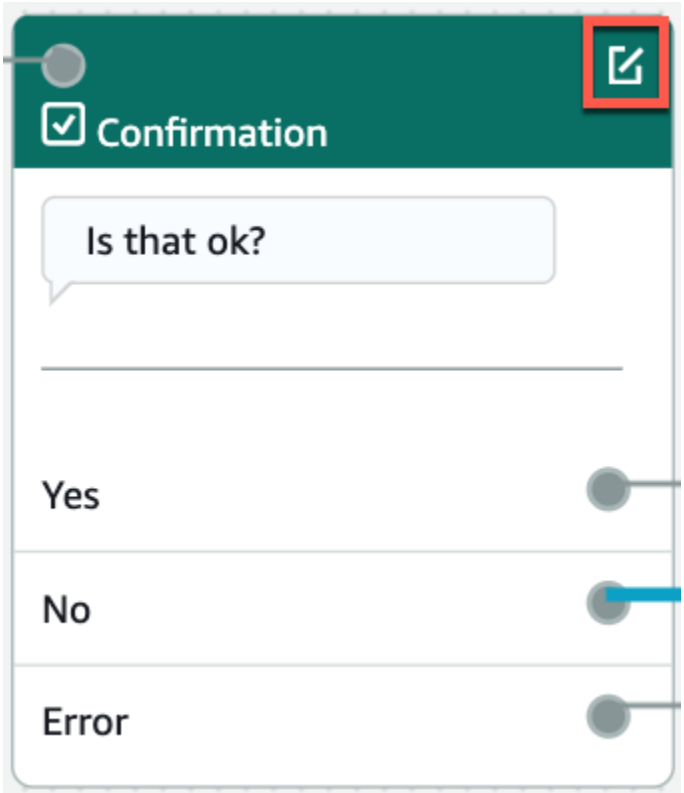


Riproduzione dell'esempio precedente con Visual conversation builder

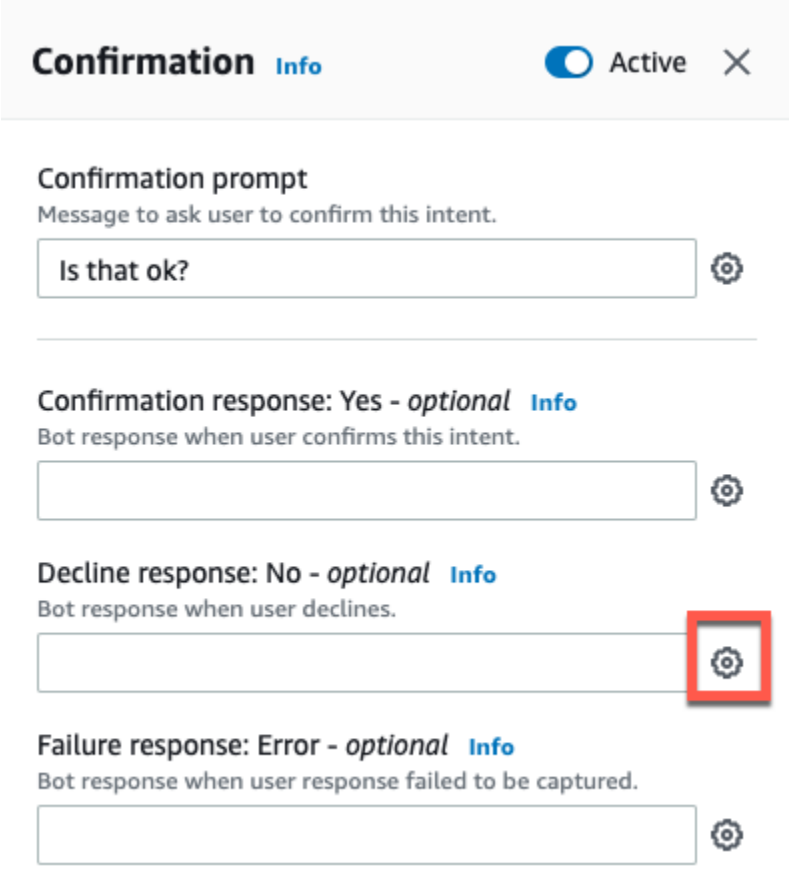
1. Crea una connessione dalla porta No del blocco di conferma alla porta di ingresso del blocco Get slot value: Meat.



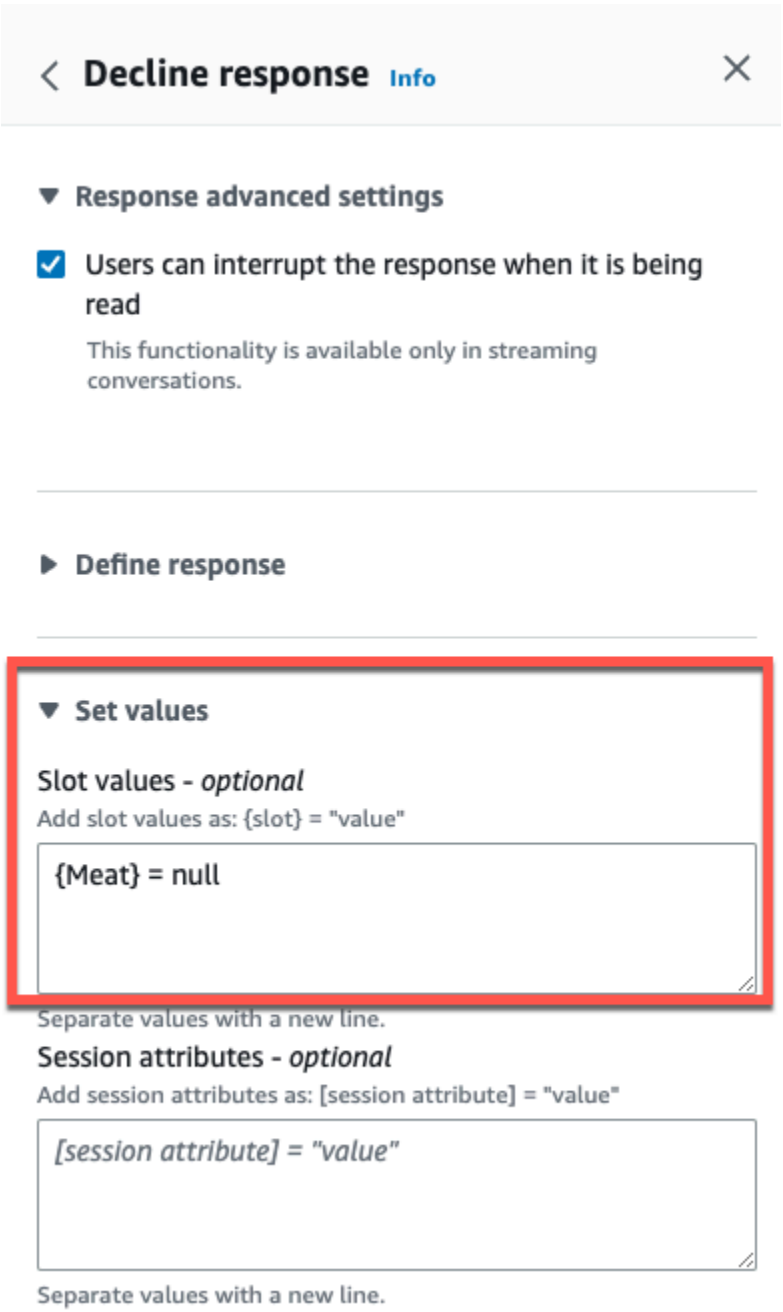
2. Seleziona l'icona Modifica nell'angolo in alto a destra del blocco di conferma.



3. Seleziona l'icona a forma di ingranaggio accanto alla risposta del bot nella sezione Risposta rifiutata.



4. Nella sezione Imposta valori, aggiungi «{Meat} = null» nella casella Valori Slot.



5. Seleziona Salva intento.

Utilizzo di più valori in uno slot

> **Note**
>
> Gli slot con valori multipli sono supportati solo nella lingua inglese (Stati Uniti).

Per alcuni scopi, potresti voler acquisire più valori per un singolo slot. Ad esempio, un bot che ordina pizza potrebbe avere l'intenzione di pronunciare la seguente frase:

```
I want a pizza with {toppings}
```

L'intento prevede che lo `{toppings}` slot contenga un elenco dei condimenti che il cliente desidera sulla propria pizza, ad esempio «peperoni e ananas».

Per configurare uno slot per acquisire più valori, è necessario impostare il `allowMultipleValues` campo sullo slot su true. È possibile impostare il campo utilizzando la console o con l'UpdateSlotoperazione CreateSlotor.

Puoi contrassegnare gli slot con tipi di slot personalizzati solo come slot multivalore.

Per uno slot multivalore, Amazon Lex V2 restituisce un elenco di valori di slot in risposta all'operazione RecognizeTextor RecognizeUtterance. Di seguito sono riportate le informazioni sullo slot restituite dal bot per l'espressione «Voglio una pizza con peperoni e ananas». OrderPizza

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

```
                    "interpretedValue": "pineapple",
                    "originalValue": "pineapple",
                    "resolvedValues": [
                        "pineapple"
                    ]
                }
            }
        ]
    }
}
```

Gli slot multivalore restituiscono sempre un elenco di valori. Quando l'enunciato contiene un solo valore, l'elenco dei valori restituiti contiene solo una risposta.

Amazon Lex V2 riconosce più valori separati da spazi, virgole (,) e dalla congiunzione «e». Gli slot multivalore funzionano con input sia testuali che vocali.

È possibile utilizzare slot multivalore nei prompt. Ad esempio, è possibile impostare la richiesta di conferma per l'intenzione di

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

Quando Amazon Lex V2 invia la richiesta all'utente, invia «Vuoi che ordini la tua pizza con peperoni e ananas?»

Gli slot multivalore supportano valori predefiniti singoli. Se vengono forniti più valori predefiniti, Amazon Lex V2 compila lo slot con solo il primo valore disponibile. Per ulteriori informazioni, consulta Utilizzo dei valori di slot predefiniti.

È possibile utilizzare l'offuscamento degli slot per mascherare i valori di uno slot multivalore nei registri delle conversazioni. Quando offuscate i valori degli slot, il valore di ciascuno dei valori degli slot viene sostituito con il nome dello slot. Per ulteriori informazioni, consulta Oscurare i valori degli slot nei registri delle conversazioni.

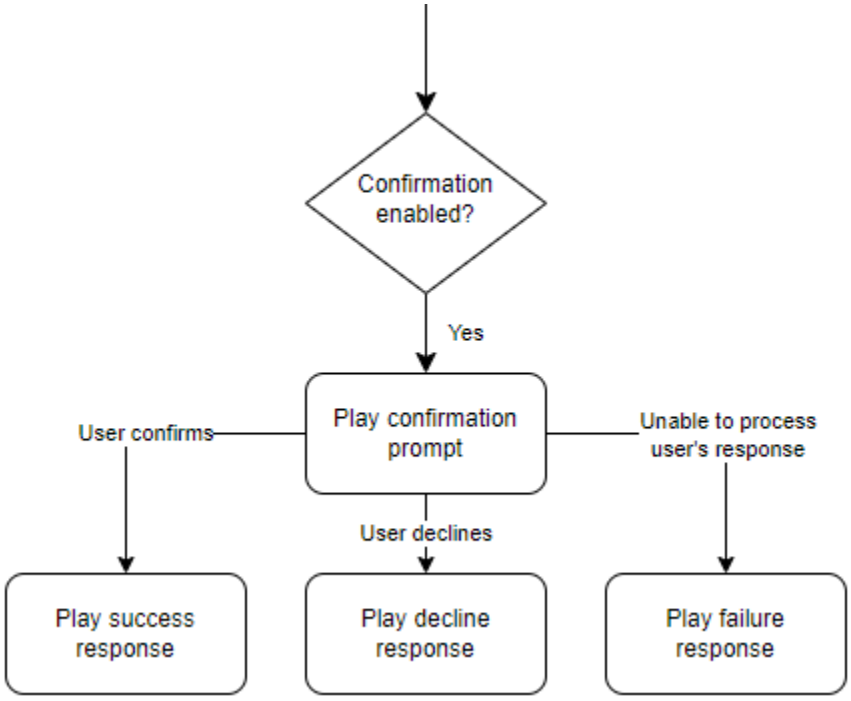
Conferma

Dopo aver completato la conversazione con l'utente e aver inserito i valori degli slot per l'intento, è possibile configurare una richiesta di conferma per chiedere all'utente se i valori degli slot sono corretti. Ad esempio, un bot che pianifica gli appuntamenti di assistenza per le auto potrebbe richiedere all'utente quanto segue:

Ho l'assistenza per la tua Honda Civic 2017 prevista per il 25 marzo alle 15:00. È tutto a posto?

Puoi definire 3 tipi di risposte alla richiesta di conferma:

- Risposta di conferma— Questa risposta viene inviata all'utente quando l'utente conferma l'intento. Ad esempio, dopo che l'utente ha risposto «sì» alla richiesta «vuoi effettuare l'ordine?»
- Rifiuta la risposta— Questa risposta viene inviata all'utente quando l'utente rifiuta l'intento. Ad esempio, dopo che l'utente ha risposto «no» alla richiesta «vuoi effettuare l'ordine?»
- Risposta al fallimento— Questa risposta viene inviata all'utente quando la richiesta di conferma non può essere elaborata. Ad esempio, se la risposta dell'utente non può essere compresa o non può essere risolta con un sì o un no.

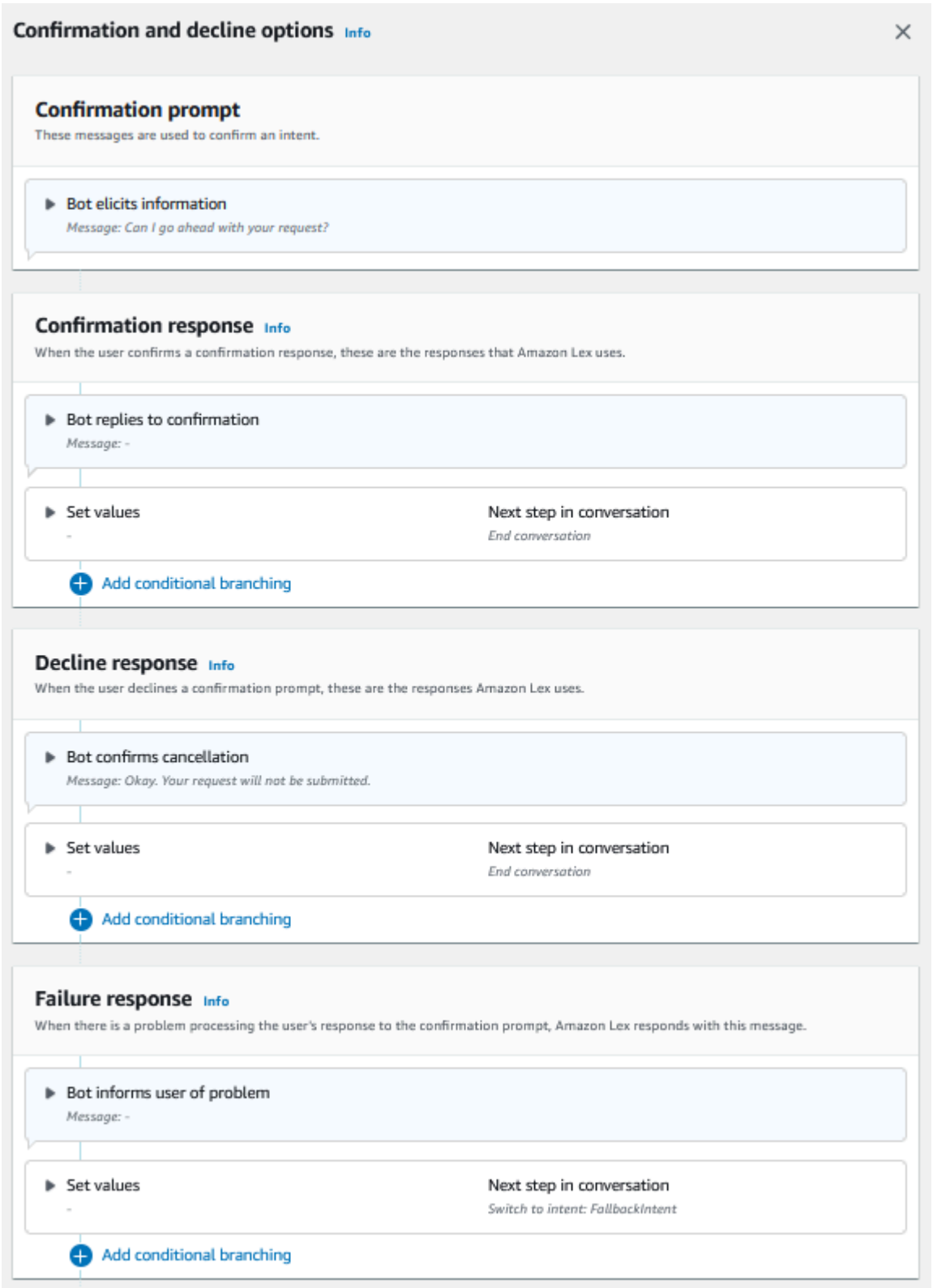


Se non specifichi una richiesta di conferma, Amazon Lex V2 passa alla fase di evasione o alla risposta di chiusura.

Puoi impostare valori, configurare i passaggi successivi e applicare le condizioni corrispondenti a ciascuna risposta per progettare il flusso di conversazione. In assenza di una condizione o di un passaggio successivo esplicito, Amazon Lex V2 passa alla fase di adempimento.

Puoi anche abilitare l'hook del codice di dialogo per convalidare le informazioni acquisite nell'intento prima di inviarlo per l'adempimento. Per utilizzare un code hook, abilita l'hook del codice di dialogo nelle opzioni avanzate del prompt di conferma. Inoltre, configura il passaggio successivo dello stato precedente per eseguire l'hook del codice di dialogo. Per ulteriori informazioni, consulta Richiama l'hook del codice di dialogo.

> **Note**
>
> Se si utilizza un code hook per attivare la fase di conferma in fase di esecuzione, è necessario contrassegnare la fase di conferma comeAttivoin fase di costruzione.


Confirmation and decline options Info
Confirmation prompt
These messages are used to confirm an intent.
Bot elicits information
Message: Can I go ahead with your request?
Confirmation response Info
When the user confirms a confirmation response, these are the responses that Amazon Lex uses.
Bot replies to confirmation
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching
Decline response Info
When the user declines a confirmation prompt, these are the responses Amazon Lex uses.
Bot confirms cancellation
Message: Okay. Your request will not be submitted.
Set values
-
Next step in conversation
End conversation
Add conditional branching
Failure response Info
When there is a problem processing the user's response to the confirmation prompt, Amazon Lex responds with this message.
Bot informs user of problem
Message: -
Set values
-
Next step in conversation
Switch to intent: FallbackIntent
Add conditional branching

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Utilizzo di una funzione Lambda per convalidare un intento.

Puoi definire un code hook Lambda per convalidare l'intento prima di inviarlo per l'adempimento. Per utilizzare un code hook, abilita l'hook del codice di dialogo nelle opzioni avanzate del prompt di conferma.

Quando usi un code hook, puoi definire le azioni che Amazon Lex V2 esegue dopo l'esecuzione del code hook. Puoi creare tre tipi di risposte:

- Risposta di successo— Inviato all'utente quando il code hook viene completato correttamente.
- Risposta al fallimento— Inviato all'utente quando il code hook non viene eseguito correttamente o quando il code hook ritorna`Failure`nella risposta.
- Risposta in timeout— Inviato all'utente quando il code hook non viene completato nel periodo di timeout configurato.

Compimento

Dopo che tutti i valori degli slot sono stati forniti dall'utente per l'intento, Amazon Lex V2 soddisfa la richiesta dell'utente. Puoi configurare le seguenti opzioni di evasione.

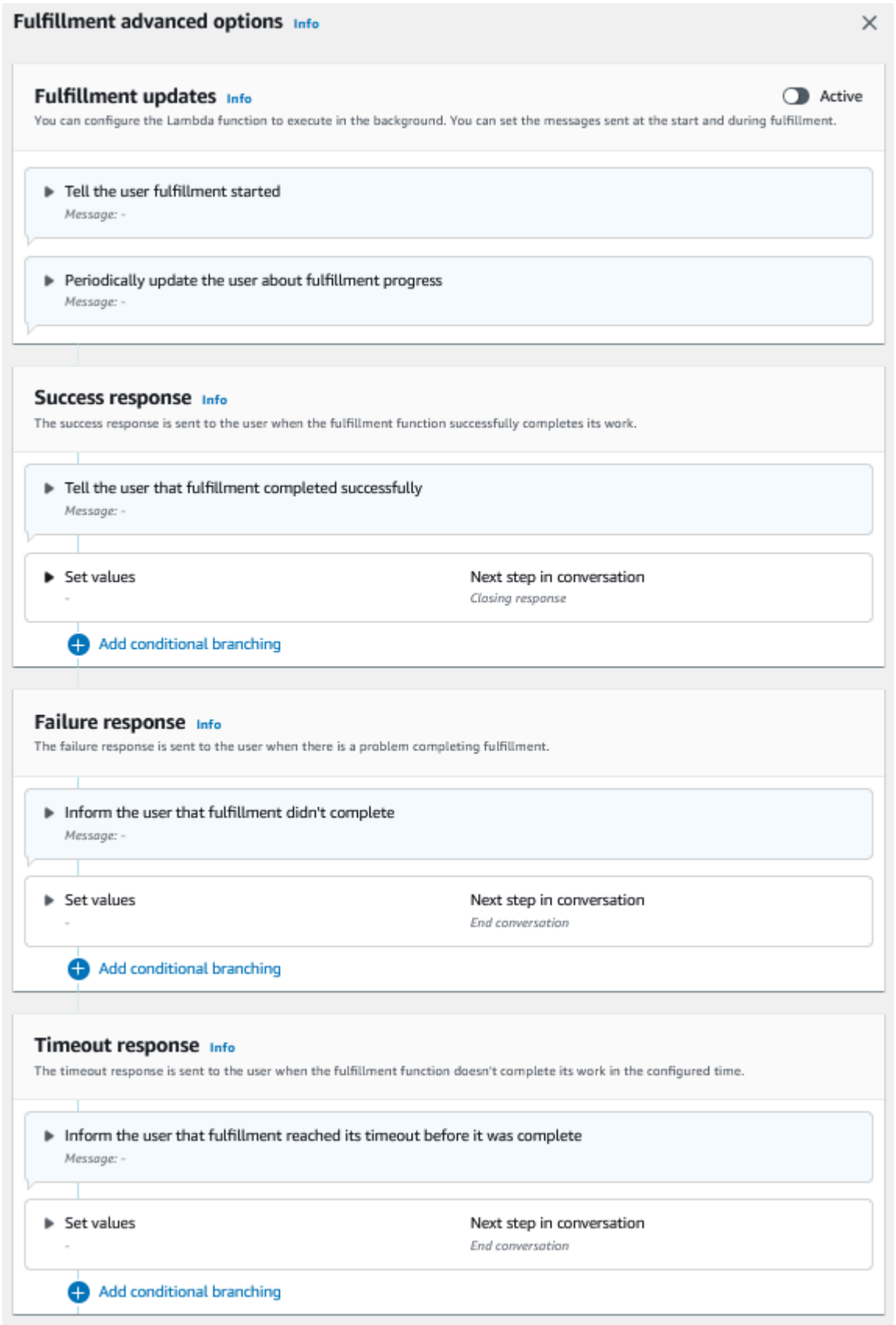
- Codice di adempimento: hook— Puoi usare questa opzione per controllare la chiamata Lambda dell'adempimento. Se l'opzione è disabilitata, l'adempimento ha esito positivo senza richiamare la funzione Lambda.
- Aggiornamenti sull'adempimento— È possibile abilitare gli aggiornamenti di adempimento per le funzioni Lambda il cui completamento richiede più di qualche secondo, in modo che l'utente sappia che il processo è in corso. Per ulteriori informazioni, consulta Configurazione degli aggiornamenti sullo stato di avanzamento dell'evasione. Questa funzionalità è disponibile solo per le conversazioni in streaming.

- Risposte all'adempimento— È possibile configurare una risposta di successo, una risposta di errore e una risposta di timeout. La risposta appropriata viene restituita all'utente in base allo stato della chiamata Lambda di adempimento.

Esistono tre possibili risposte di adempimento:

- Risposta di successo— Un messaggio inviato quando l'adempimento Lambda viene completato con successo.
- Risposta al fallimento— Un messaggio inviato se l'adempimento non è riuscito o Lambda non può essere completato per qualche motivo.
- Risposta in timeout— Un messaggio inviato se la funzione di adempimento Lambda non termina entro il timeout configurato.

Puoi impostare valori, configurare i passaggi successivi e applicare le condizioni corrispondenti a ciascuna risposta per progettare il flusso di conversazione. In assenza di una condizione o di un passaggio successivo esplicito, Amazon Lex V2 passa alla risposta di chiusura.


Fulfillment advanced options Info
Fulfillment updates Info
Active
You can configure the Lambda function to execute in the background. You can set the messages sent at the start and during fulfillment.
Tell the user fulfillment started
Message: -
Periodically update the user about fulfillment progress
Message: -
Success response Info
The success response is sent to the user when the fulfillment function successfully completes its work.
Tell the user that fulfillment completed successfully
Message: -
Set values
-
Next step in conversation
Closing response
Add conditional branching
Failure response Info
The failure response is sent to the user when there is a problem completing fulfillment.
Inform the user that fulfillment didn't complete
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching
Timeout response Info
The timeout response is sent to the user when the fulfillment function doesn't complete its work in the configured time.
Inform the user that fulfillment reached its timeout before it was complete
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

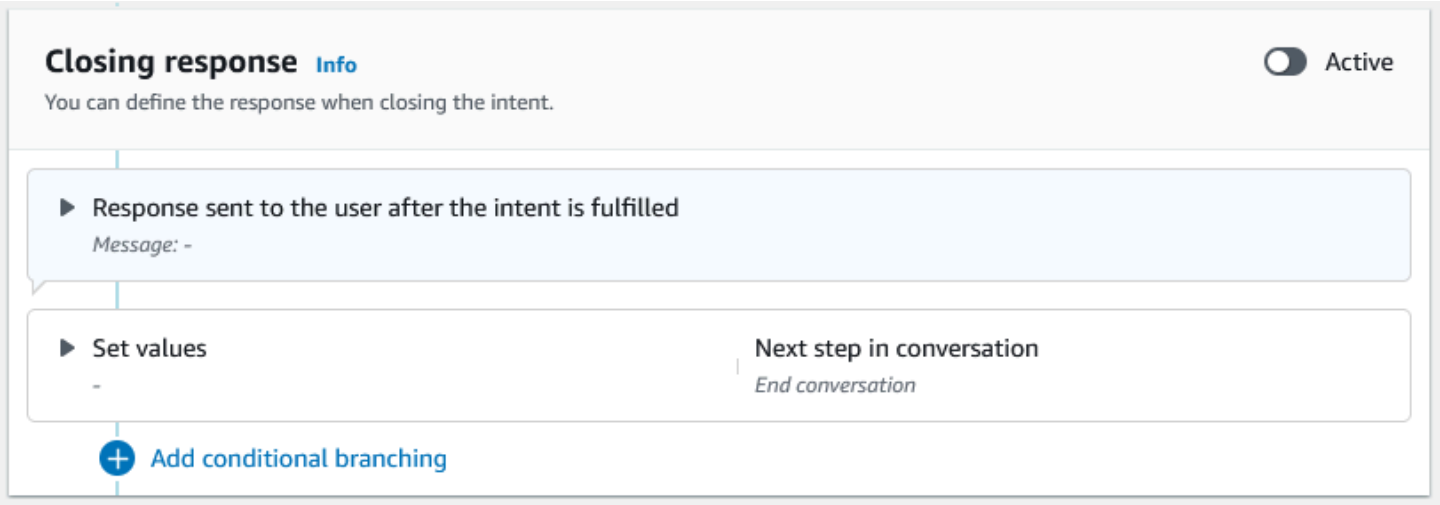
Risposta di chiusura

La risposta di chiusura viene inviata all'utente dopo che il suo intento è stato soddisfatto. Puoi utilizzare la risposta di chiusura per terminare la conversazione oppure puoi utilizzarla per far sapere all'utente che può continuare con un altro intento. Ad esempio, in un bot per la prenotazione di viaggi, puoi impostare la risposta di chiusura per la prenotazione di una camera d'albergo in questo modo:

```
Va bene, ho prenotato la tua camera d'albergo. Posso fare altro per aiutarti?
```

Puoi impostare valori, configurare i passaggi successivi e applicare condizioni dopo la risposta di chiusura alla progettazione del percorso di conversazione. In assenza di una condizione o di un passaggio successivo esplicito, Amazon Lex V2 termina la conversazione.

Se non fornisci una risposta conclusiva o se nessuna delle condizioni risulta vera, Amazon Lex V2 termina la conversazione con il tuo bot.

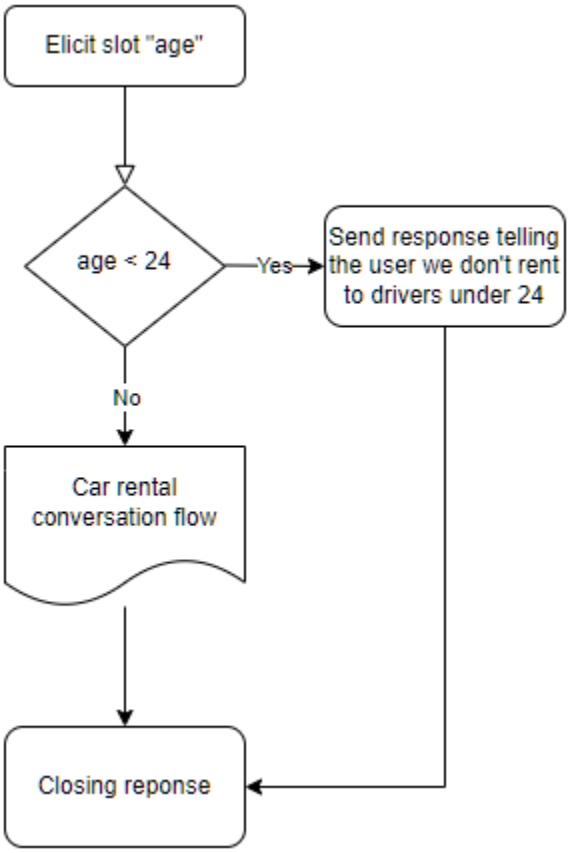


> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Creazione di percorsi di conversazione

In genere, Amazon Lex V2 gestisce il flusso di conversazioni con gli utenti. Per i bot semplici, il flusso predefinito può essere sufficiente per creare un'esperienza positiva per gli utenti. Tuttavia, per i bot più complessi, potresti voler prendere il controllo della conversazione e indirizzare il flusso verso percorsi più complessi.

Ad esempio, in un bot che prenota il noleggio auto, potresti non noleggiarlo a conducenti più giovani. In questo caso, puoi creare una condizione che verifichi se un conducente ha meno di una certa età e, in tal caso, passare alla risposta di chiusura.



Per progettare tali interazioni, puoi configurare la fase successiva in ogni punto della conversazione, valutare le condizioni, impostare valori e richiamare i code hook.

La ramificazione condizionale consente di creare percorsi per gli utenti attraverso interazioni complesse. Puoi utilizzare un ramo condizionale in qualsiasi momento in cui passi il controllo della conversazione al tuo bot. Ad esempio, puoi creare una condizione prima che il bot richieda il primo valore dello slot, puoi creare una condizione tra l'ottenimento del valore di ogni slot o puoi creare una condizione prima che il bot chiuda la conversazione. Per un elenco dei luoghi in cui è possibile aggiungere condizioni, consulta. Aggiungere intenti

Quando crei un bot, Amazon Lex V2 crea un percorso predefinito attraverso la conversazione in base all'ordine di priorità degli slot. Per personalizzare il percorso della conversazione, puoi modificare il passaggio successivo in qualsiasi momento della conversazione. Per ulteriori informazioni, consulta Configura i passaggi successivi della conversazione.

Per creare percorsi alternativi in base alle condizioni, puoi utilizzare un ramo condizionale in qualsiasi punto della conversazione. Ad esempio, puoi creare una condizione prima che il bot richieda il primo valore dello slot. Puoi creare una condizione tra l'acquisizione del valore di ogni slot oppure

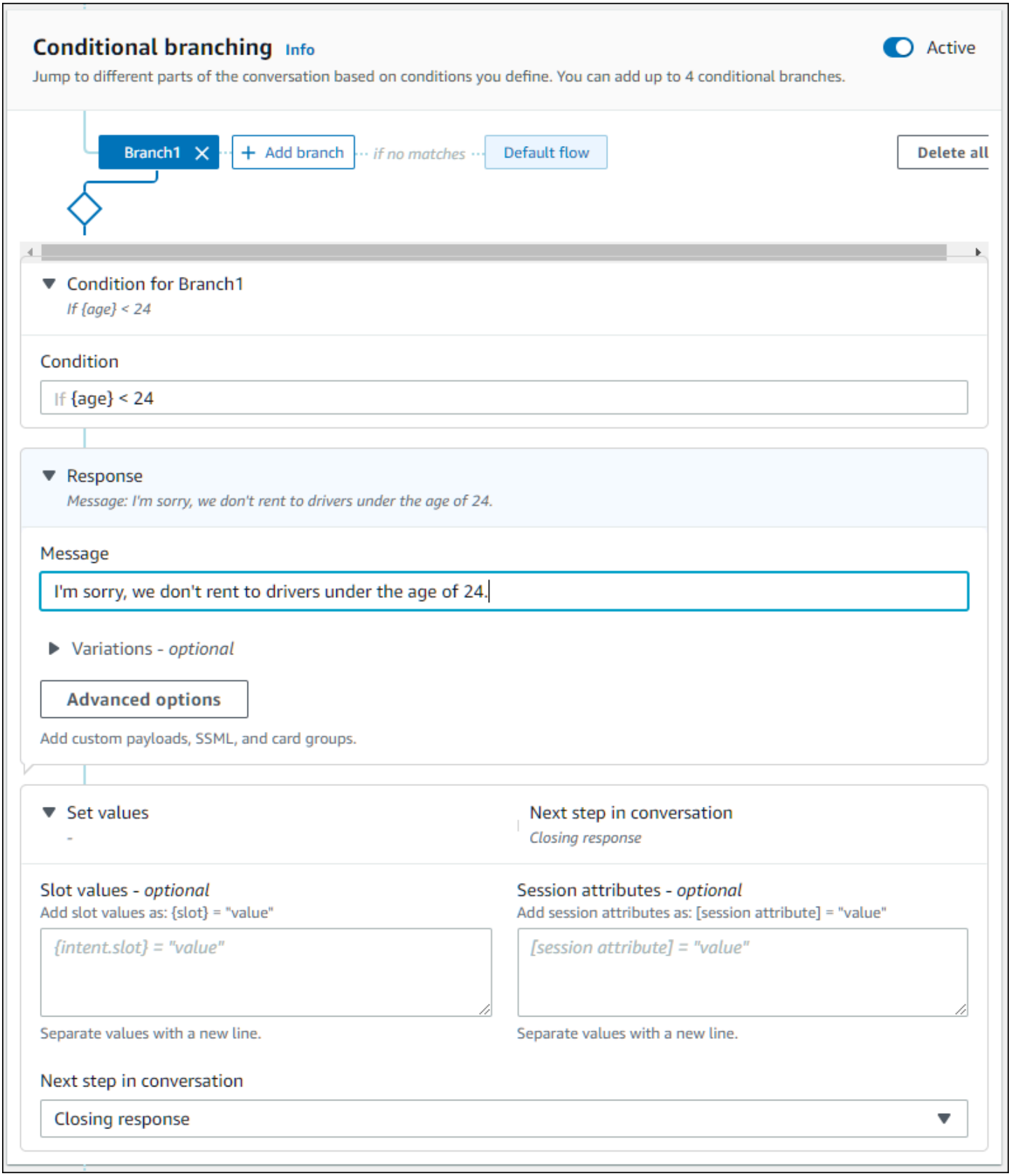
puoi creare una condizione prima che il bot chiuda la conversazione. Per un elenco dei luoghi che consentono di aggiungere condizioni, consulta. Aggiungi condizioni alle conversazioni delle filiali

Puoi impostare condizioni in base ai valori dello slot, agli attributi della sessione, alla modalità di input e alla trascrizione di input o a una risposta di Amazon Kendra.

Puoi impostare i valori degli attributi dello slot e della sessione in ogni punto della conversazione. Per ulteriori informazioni, consulta Imposta i valori durante la conversazione.

Puoi anche impostare l'azione successiva su dialog code hook per eseguire una funzione Lambda. Per ulteriori informazioni, consulta Richiama l'hook del codice di dialogo.

L'immagine seguente mostra la creazione di un percorso per uno slot nella console. In questo esempio, Amazon Lex V2 genererà lo slot «age». Se il valore dello slot è inferiore a 24, Amazon Lex V2 passa alla risposta di chiusura, altrimenti Amazon Lex seguirà il percorso predefinito.


Conditional branching Info
Active
Jump to different parts of the conversation based on conditions you define. You can add up to 4 conditional branches.
Branch1
Add branch
if no matches
Default flow
Delete all
Condition for Branch1
If {age} < 24
Condition
If {age} < 24
Response
Message: I'm sorry, we don't rent to drivers under the age of 24.
Message
I'm sorry, we don't rent to drivers under the age of 24.
Variations - optional
Advanced options
Add custom payloads, SSML, and card groups.
Set values
-
Next step in conversation
Closing response
Slot values - optional
Add slot values as: {slot} = "value"
{intent.slot} = "value"
Separate values with a new line.
Session attributes - optional
Add session attributes as: [session attribute] = "value"
[session attribute] = "value"
Separate values with a new line.
Next step in conversation
Closing response

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica ti offre un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi di dialogo tramite codice hook, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Configura i passaggi successivi della conversazione

Puoi configurare un passaggio successivo in ogni fase della conversazione per progettare le conversazioni. In genere, Amazon Lex V2 configura automaticamente i passaggi successivi predefiniti per ogni fase della conversazione secondo l'ordine seguente.

Risposta iniziale → Slot Elicitation → Conferma (se attiva) → Adempimento (se attiva) → Risposta di chiusura (se attiva) → Termina conversazione

Puoi modificare i passaggi successivi predefiniti e progettare la conversazione in base all'esperienza utente prevista. I seguenti passaggi successivi possono essere configurati in ogni fase della conversazione:

Vai a

- Risposta iniziale: la conversazione viene riavviata dall'inizio dell'intento. Puoi scegliere di saltare la risposta iniziale durante la configurazione di questo passaggio successivo
- Evoca uno slot: puoi suscitare qualsiasi slot nell'intento.
- Valuta le condizioni: puoi valutare le condizioni e avviare la conversazione in qualsiasi fase della conversazione.
- Invoke Dialog Code Hook: puoi richiamare la business logic in qualsiasi momento.
- Conferma l'intento: all'utente verrà richiesto di confermare l'intento.
- Realizza l'intento: l'adempimento dell'intento inizierà nella fase successiva.
- Risposta di chiusura: la risposta di chiusura verrà restituita all'utente.

Passa a

- Intento: puoi passare a un intento diverso e continuare la conversazione con questo intento. Facoltativamente, puoi saltare la risposta iniziale dell'intento durante la transizione.
- Intento: slot specifico: puoi richiamare direttamente uno slot specifico con un intento diverso se hai già acquisito alcuni valori di slot nell'intento corrente.

Attendi l'input dell'utente: il bot attende che l'utente fornisca input per riconoscere qualsiasi nuovo intento. Puoi configurare istruzioni come «C'è qualcos'altro in cui posso aiutarti?» prima di impostare questo passaggio successivo. Il bot sarà in uno stato `ElicitIntent` di dialogo.

Termina conversazione: la conversazione con il bot è chiusa.

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica ti offre un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi di dialogo tramite codice hook, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Imposta i valori durante la conversazione

Amazon Lex V2 offre la possibilità di impostare valori di slot e valori degli attributi di sessione in ogni fase della conversazione. Puoi quindi utilizzare questi valori durante la conversazione per valutare le condizioni o utilizzarli durante la realizzazione degli intenti.

È possibile impostare i valori degli slot per l'intento corrente. Se il passaggio successivo della conversazione consiste nell'invocare un altro intento, puoi impostare i valori degli slot del nuovo intento.

Se lo slot assegnato non viene riempito o se il percorso JSON non può essere analizzato, l'attributo verrà impostato su. `null`

Utilizzate la seguente sintassi quando utilizzate i valori degli slot e gli attributi di sessione:

- Valori dello slot: racchiudono il nome dello slot tra parentesi («{}»). Per i valori dello slot nell'intento corrente, è sufficiente utilizzare il nome dello slot. Ad esempio, `{slot}`. Se state impostando un

valore nell'intento successivo, dovete utilizzare sia il nome dell'intento che il nome dello slot per identificare lo slot. Ad esempio, `{intent.slot}`.

Esempi:

- `{PhoneNumber} = "1234567890"`
- `{CheckBalance.AccountNumber} = "99999999"`
- `{BookingID} = "ABC123"`
- `{FirstName} = "John"`

Il valore di uno slot può essere uno dei seguenti:

- una stringa costante
- un percorso JSON che fa riferimento al blocco delle trascrizioni nella risposta di Amazon Lex (per en-US e en-GB)
- un attributo di sessione

Esempi:

- `{username} = "john.doe"`
- `{username_confidence} = $.transcriptions[0].transcriptionConfidence`
- `{username_slot_value} = [username]`

> **Note**
>
> I valori degli slot possono anche essere impostati su`null`. Se è necessario richiamare nuovamente il valore di uno slot che è stato riempito, è necessario impostare il valore su `null` prima di richiedere nuovamente al cliente il valore dello slot. Se lo slot assegnato non viene riempito o se il percorso JSON non può essere analizzato, l'attributo verrà impostato su. `null`

- Attributi di sessione: racchiudono il nome dell'attributo tra parentesi quadre («[]»). Ad esempio, `[sessionAttribute]`.

  Esempi:

  - `[username] = "john.doe"`
  - `[username_confidence] = $.transcriptions[0].transcriptionConfidence`
  - `[username_slot_value] = {username}`

Il valore dell'attributo di sessione può essere uno dei seguenti:

- una stringa costante
- un percorso JSON che fa riferimento al blocco delle trascrizioni nella risposta di Amazon Lex (per en-US e en-GB)
- un riferimento al valore dello slot

> **Note**
>
> Se lo slot assegnato non viene riempito o se il percorso JSON non può essere analizzato, l'attributo verrà impostato su. `null`

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica ti offre un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi di dialogo tramite codice hook, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Aggiungi condizioni alle conversazioni delle filiali

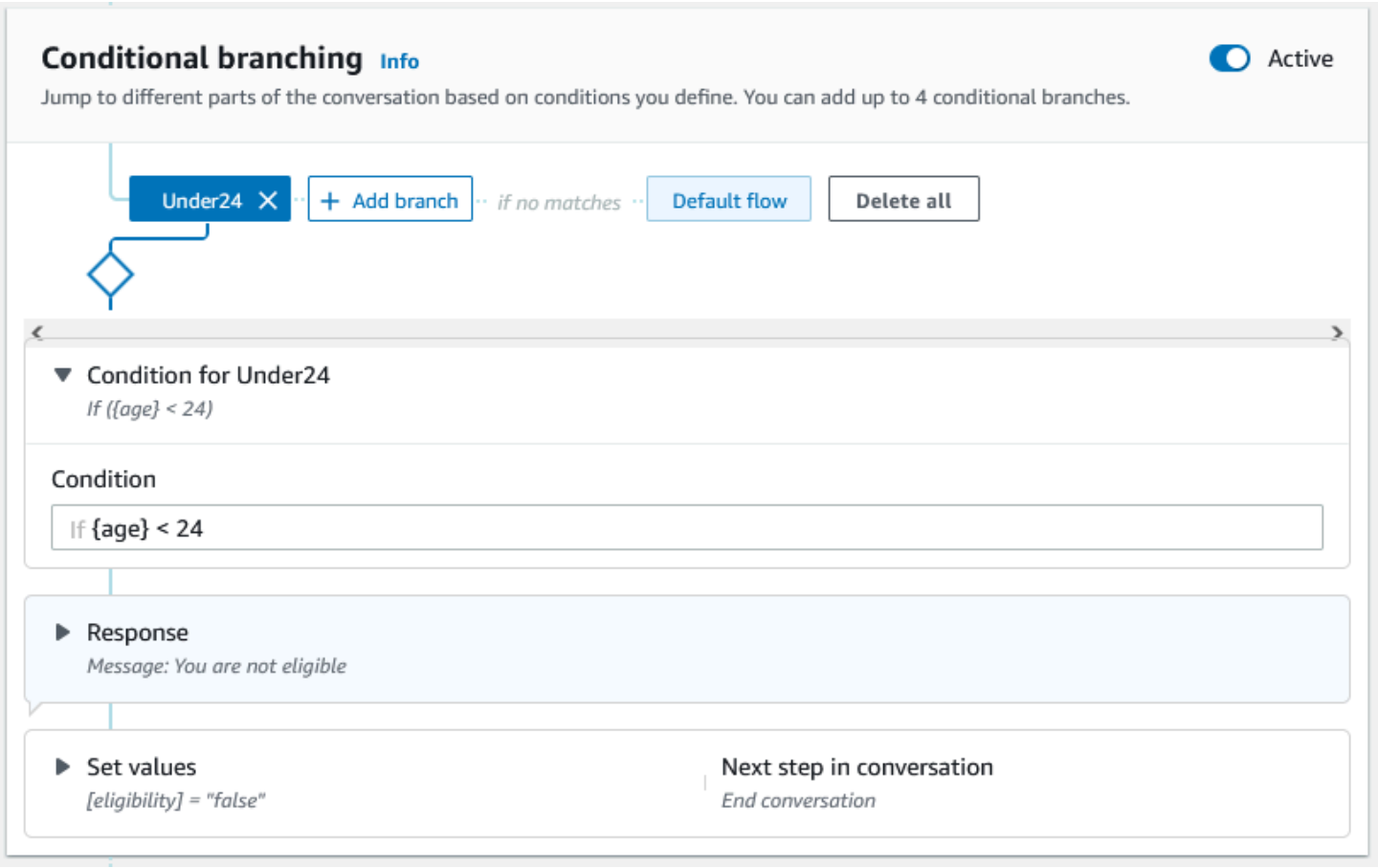
Puoi utilizzare la ramificazione condizionale per controllare il percorso seguito dal cliente durante la conversazione con il bot. Puoi suddividere la conversazione in base ai valori degli slot, agli attributi della sessione, al contenuto della modalità di input e dei campi di trascrizione di input o a una risposta di Amazon Kendra.

Puoi definire fino a quattro rami. Ogni filiale presenta una condizione che deve essere soddisfatta affinché Amazon Lex V2 possa seguire quella filiale. Se nessuna delle filiali soddisfa la sua condizione, viene seguita una diramazione predefinita.

Quando definisci una filiale, definisci l'azione che Amazon Lex V2 deve intraprendere se le condizioni corrispondenti a quel ramo risultano vere. Puoi definire una delle seguenti azioni:

- Una risposta inviata all'utente.

- Valori degli slot da applicare agli slot.
- Valori degli attributi di sessione per la sessione corrente.
- La fase successiva della conversazione. Per ulteriori informazioni, consulta Creazione di percorsi di conversazione.



Ogni ramo condizionale ha un'espressione booleana che deve essere soddisfatta affinché Amazon Lex V2 segua il ramo. Esistono operatori di confronto e booleani, funzioni e operatori quantificatori che puoi utilizzare per le tue condizioni. Ad esempio, la condizione seguente restituisce true se lo slot {age} è inferiore a 24.

```
{age} < 24
```

La seguente condizione restituisce true se lo slot multivalore {toppings} contiene la parola «pineapple».

```
{toppings} CONTAINS "pineapple"
```

È possibile combinare più operatori di confronto con un operatore booleano per condizioni più complesse. Ad esempio, la seguente condizione restituisce true se il valore dello slot {make} è «Honda» e il valore dello slot {model} è «Civic». Utilizzate le parentesi per impostare l'ordine di valutazione.

```
({make} = "Honda") AND ({model} = "Civic")
```

Negli argomenti seguenti vengono forniti dettagli sugli operatori e sulle funzioni condizionali delle filiali.

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica ti offre un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi di dialogo tramite codice hook, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Argomenti

Operatori di confronto

Amazon Lex V2 supporta i seguenti operatori di confronto per le condizioni:

- Uguale a (=)
- Non è uguale a (! =)
- Minore di (<)
- Minore o uguale a (< =)
- Maggiore di (>)

- Maggiore o uguale a (> =)

Quando si utilizza un operatore di confronto, vengono utilizzate le seguenti regole.

- Il lato sinistro deve essere un riferimento. Ad esempio, per fare riferimento a un valore di slot, si usa. `{slotName}` Per fare riferimento al valore di un attributo di sessione, si utilizza`[attribute]`. Per la modalità di input e la trascrizione di input, si utilizza `$.inputMode` e`$.inputTranscript`.
- Il lato destro deve essere costante e dello stesso tipo del lato sinistro.
- Qualsiasi espressione che fa riferimento a un attributo che non è stato impostato viene considerata non valida e non viene valutata.
- Quando si confronta uno slot multivalore, il valore utilizzato è un elenco separato da virgole di tutti i valori interpretati.

I confronti si basano sul tipo di slot del riferimento. Vengono risolti come segue:

- Stringhe: le stringhe vengono confrontate in base alla loro rappresentazione ASCII. Il confronto distingue tra lettere maiuscole e minuscole.
- Numeri: gli slot basati sui numeri vengono convertiti dalla rappresentazione di stringa a un numero e quindi confrontati.
- Data/ora: gli slot basati sull'ora vengono confrontati in base alle serie temporali. La data o l'ora precedenti sono considerate inferiori. Per le durate, i periodi più brevi sono considerati inferiori.

Operatori booleani

Amazon Lex V2 supporta gli operatori booleani per combinare operatori di confronto. Ti consentono di creare istruzioni simili alle seguenti:

```
({number} >= 5) AND ({number} <= 10)
```

È possibile utilizzare i seguenti operatori booleani:

- E (&&)
- O (||)
- NON (!)

Operatori quantificatori

Gli operatori quantificatori valutano gli elementi di una sequenza e determinano se uno o più elementi soddisfano la condizione.

- CONTIENE: determina se il valore specificato è contenuto in uno slot multivalore e restituisce true in caso affermativo. Ad esempio, `{toppings} CONTAINS "pineapple"` restituisce true se l'utente ha ordinato dell'ananas sulla pizza.

Funzioni

Le funzioni devono essere precedute dalla stringa. `fn.` L'argomento della funzione è un riferimento a uno slot, un attributo di sessione o un attributo di richiesta. Amazon Lex V2 offre due funzioni per ottenere informazioni dai valori degli slot, SessionAttribute o RequestAttribute.

- fn.Count (): conta il numero di valori in uno slot multivalore.

  Ad esempio, se lo slot `{toppings}` contiene il valore «pepperoni, ananas»:

  ```
  fn.COUNT({toppings}) = 2
  ```
- fn.is_set () — il valore è vero se uno slot, un attributo di sessione o un attributo di richiesta è impostato nella sessione corrente.

  Sulla base dell'esempio precedente:

  ```
  fn.IS_SET({toppings})
  ```
- fn.length () — value è la lunghezza del valore dell'attributo di sessione, del valore dello slot o dell'attributo slot impostato nella sessione corrente. Questa funzione non supporta slot multivalore o slot compositi.

  Esempio:

  Se lo slot `{credit-card-number}` contiene il valore «123456781234":

  ```
  fn.LENGTH({credit-card-number}) = 12
  ```

Esempi di espressioni condizionali

Ecco alcuni esempi di espressioni condizionali. NOTA: `$.` rappresenta il punto di ingresso alla risposta JSON di Amazon Lex. Il valore seguente `$.` verrà analizzato all'interno della risposta

Amazon Lex per recuperare il valore. Le espressioni condizionali che utilizzano il blocco di riferimento del percorso JSON alle trascrizioni nella risposta Amazon Lex saranno supportate solo nelle stesse lingue che supportano i punteggi di trascrizione ASR.

Value type (Tipo di valore)	Caso d'uso	Espressione condizionale
Slot personalizzato	`pizzaSize` il valore dello slot è uguale a grande	`{pizzaSize} = "large"`
Slot personalizzato	`pizzaSize` è uguale a grande o medio	`{pizzaSize} = "large"`O `{pizzaSize} = "medium"`
Slot personalizzato	Espressioni con `()` e `AND/OR`	`{pizzaType} = "pepperoni"` OPPURE `{pizzaSize} = "medium"` OPPURE `{pizzaSize} = "small"`
Slot personalizzato (slot multivalore)	Controlla se uno dei condimenti è Cipolla	`{toppings} CONTAINS "Onion"`
Slot personalizzato (slot multivalore)	Il numero di condimenti è superiore a 3	`fn.COUNT({topping}) > 2`
`AMAZON.AlphaNumeric`	`bookingID` è ABC123	`{bookingID} = "ABC123"`
`AMAZON.Number`	il valore della fascia di età è superiore a 30	`{age} > 30`
`AMAZON.Number`	il valore della fascia di età è pari a 10	`{age} = 10`
`AMAZON.Date`	`dateOfBirth` valore dello slot prima del 1990	`{dateOfBirth} < "1990-10-01"`

Value type (Tipo di valore)	Caso d'uso	Espressione condizionale
AMAZON.State	destinationState il valore dello slot è uguale a Washington	{destinationState} = "washington"
AMAZON.Country	destinationCountry il valore dello slot non è quello degli Stati Uniti	{destinationCountry} != "united states"
AMAZON.FirstName	firstName il valore dello slot è John	{firstName} = "John"
AMAZON.PhoneNumber	phoneNumber il valore dello slot è 716767891932	{phoneNumer} = 716767891932
AMAZON.Percentage	Controlla se il valore percentuale dello slot è maggiore o uguale a 78	{percentage} >= 78
AMAZON.EmailAddress	emailAddress il valore dello slot è userA@hmail.com	{emailAddress} = "userA@hmail.com"
AMAZON.LastName	lastNameil valore dello slot è Doe	{lastName} = "Doe"
AMAZON.City	Il valore dello slot cittadino è uguale a Seattle	{city} = "Seattle"
AMAZON.Time	L'ora è dopo le 20:00	{time} > "20:00"
AMAZON.StreetName	streetName il valore dello slot è Boren Avenue	{streetName} = "boren avenue"
AMAZON.Duration	travelDuration il valore dello slot è inferiore a 2 ore	{travelDuration} < P2H
modalità di input	La modalità di input è vocale	$.inputMode = "Speech"

Value type (Tipo di valore)	Caso d'uso	Espressione condizionale
Trascrizione di input	La trascrizione di input è uguale a «Voglio una pizza grande»	`$.inputTranscript = "I want a large pizza"`
Attributo di sessione	controlla l'attributo customer_subscription_type	`[customer_subcription_type] = "yearly"`
Attributo di richiesta	controlla il flag retry_enabled	`((retry_enabled)) = "TRUE"`
Risposta di Kendra	La risposta di Kendra contiene domande frequenti	`fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1) ))`
Espressione condizionale con trascrizioni	Espressioni condizionali che utilizzano il percorso JSON delle trascrizioni	`$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5`
Imposta gli attributi della sessione	Imposta gli attributi della sessione utilizzando le trascrizioni, i valori del percorso JSON e dello slot	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`
Imposta i valori degli slot	Imposta i valori degli slot utilizzando gli attributi di sessione e il percorso JSON delle trascrizioni	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName`si riferisce al nome di uno slot nel bot Amazon Lex. Se lo slot non è risolto (nullo) o se lo slot non esiste, le assegnazioni vengono ignorate in fase di esecuzione. `sessionAttributes`si riferisce al nome dell'attributo di sessione impostato dal cliente in fase di compilazione.

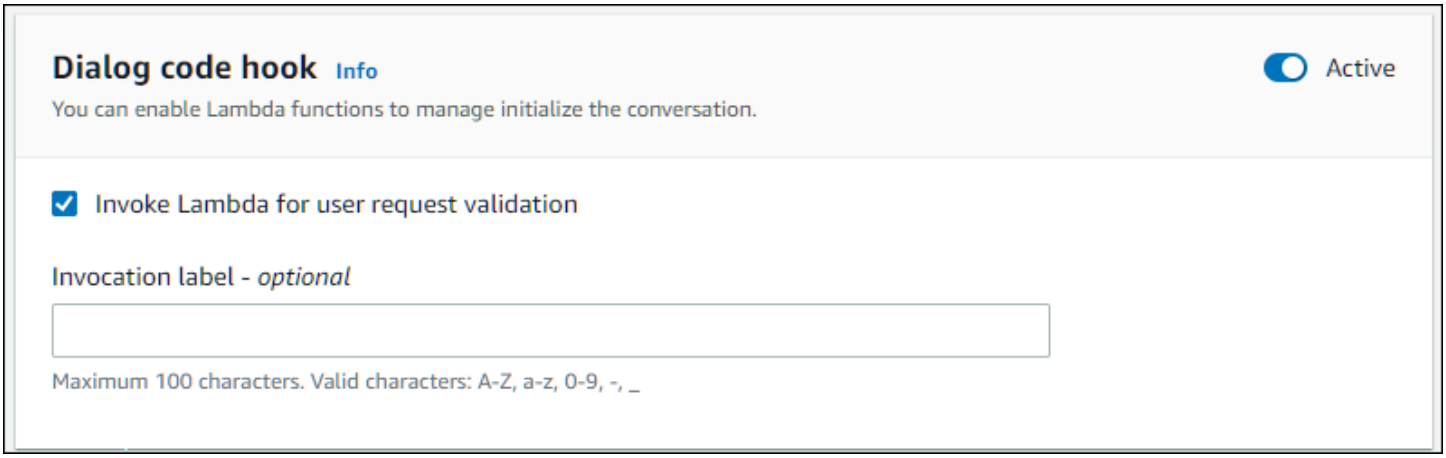
Richiama l'hook del codice di dialogo

In ogni fase della conversazione in cui Amazon Lex invia un messaggio all'utente, puoi utilizzare una funzione Lambda come fase successiva della conversazione. È possibile utilizzare la funzione per implementare la logica aziendale in base allo stato corrente della conversazione.

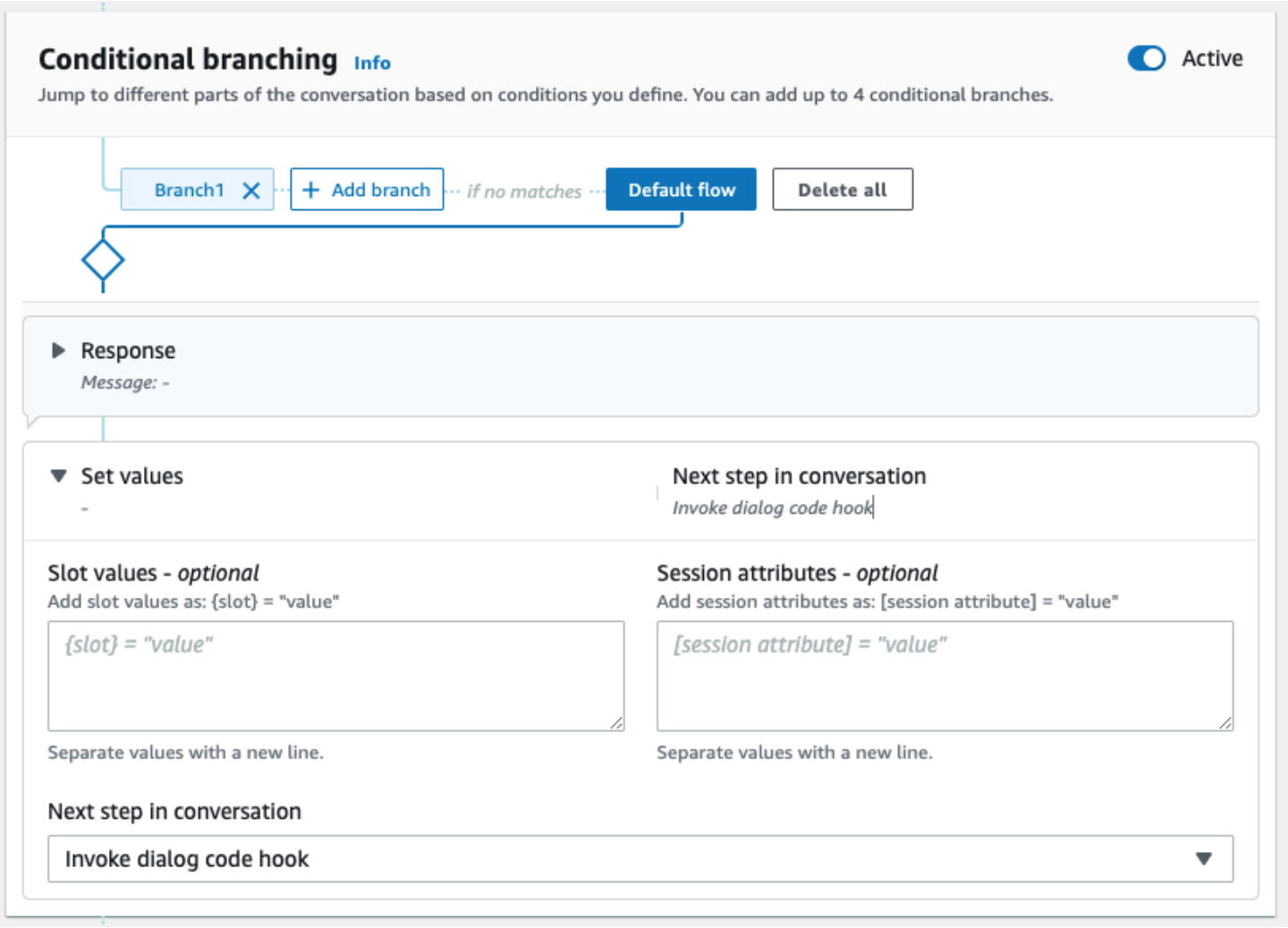
La funzione Lambda in esecuzione è associata all'alias bot in uso. Per richiamare la funzione Lambda in tutti i code hook di dialogo dell'intento, è necessario selezionare Usa una funzione Lambda per l'inizializzazione e la convalida dell'intento. Per ulteriori informazioni sulla scelta di una funzione Lambda, vedere. Creazione e associazione di una funzione Lambda a un alias bot

Esistono due passaggi per utilizzare una funzione Lambda. Innanzitutto, è necessario attivare l'hook dei codici di dialogo in qualsiasi momento della conversazione. In secondo luogo, è necessario impostare la fase successiva della conversazione per utilizzare l'hook di codici di dialogo.

L'immagine seguente mostra l'hook di codici di dialogo attivato.

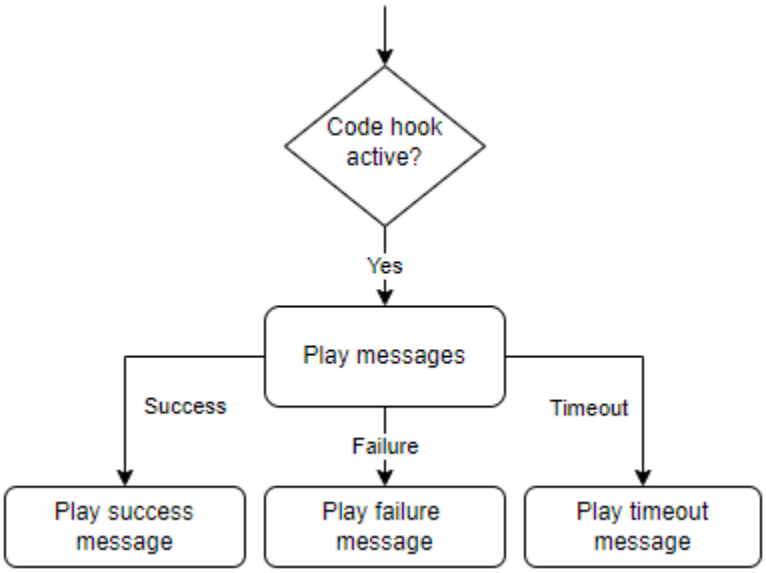


Quindi, imposta l'hook di codice come azione successiva per la fase di conversazione. Puoi farlo configurando il passaggio successivo della conversazione su Invoke dialog code hook. L'immagine seguente mostra un ramo condizionale in cui invocare l'hook del codice di dialogo è il passaggio successivo per il percorso predefinito della conversazione.



Quando i code hook sono attivi, puoi impostare tre risposte da restituire all'utente:

- Operazione riuscita: inviata quando la funzione Lambda è stata completata correttamente.
- Errore: inviato se si è verificato un problema con l'esecuzione della funzione Lambda o se la funzione Lambda ha restituito un valore di. `intent.state` `Failed`
- Timeout: inviato se la funzione Lambda non è stata completata nel periodo di timeout configurato.



Scegli Lambda dialog code hook, quindi scegli Opzioni avanzate per visualizzare le tre opzioni per le risposte che corrispondono alla chiamata della funzione Lambda. Puoi impostare valori, configurare i passaggi successivi e applicare le condizioni corrispondenti a ciascuna risposta per progettare il flusso di conversazione. In assenza di una condizione o di un passaggio successivo esplicito, Amazon Lex V2 decide il passaggio successivo in base allo stato corrente della conversazione.

Nella pagina Opzioni avanzate puoi anche scegliere di abilitare o disabilitare la chiamata della funzione Lambda. Quando la funzione è abilitata, l'hook del codice di dialogo viene richiamato con una chiamata Lambda, seguito dal messaggio di successo, errore o timeout basato sui risultati della chiamata Lambda. Quando la funzione è disabilitata, Amazon Lex V2 non esegue la funzione Lambda e procede come se l'hook del codice di dialogo avesse successo.

È inoltre possibile impostare un'etichetta di chiamata che viene inviata alla funzione Lambda quando viene richiamata da questo messaggio. Puoi usarlo per identificare la sezione della funzione Lambda da eseguire.

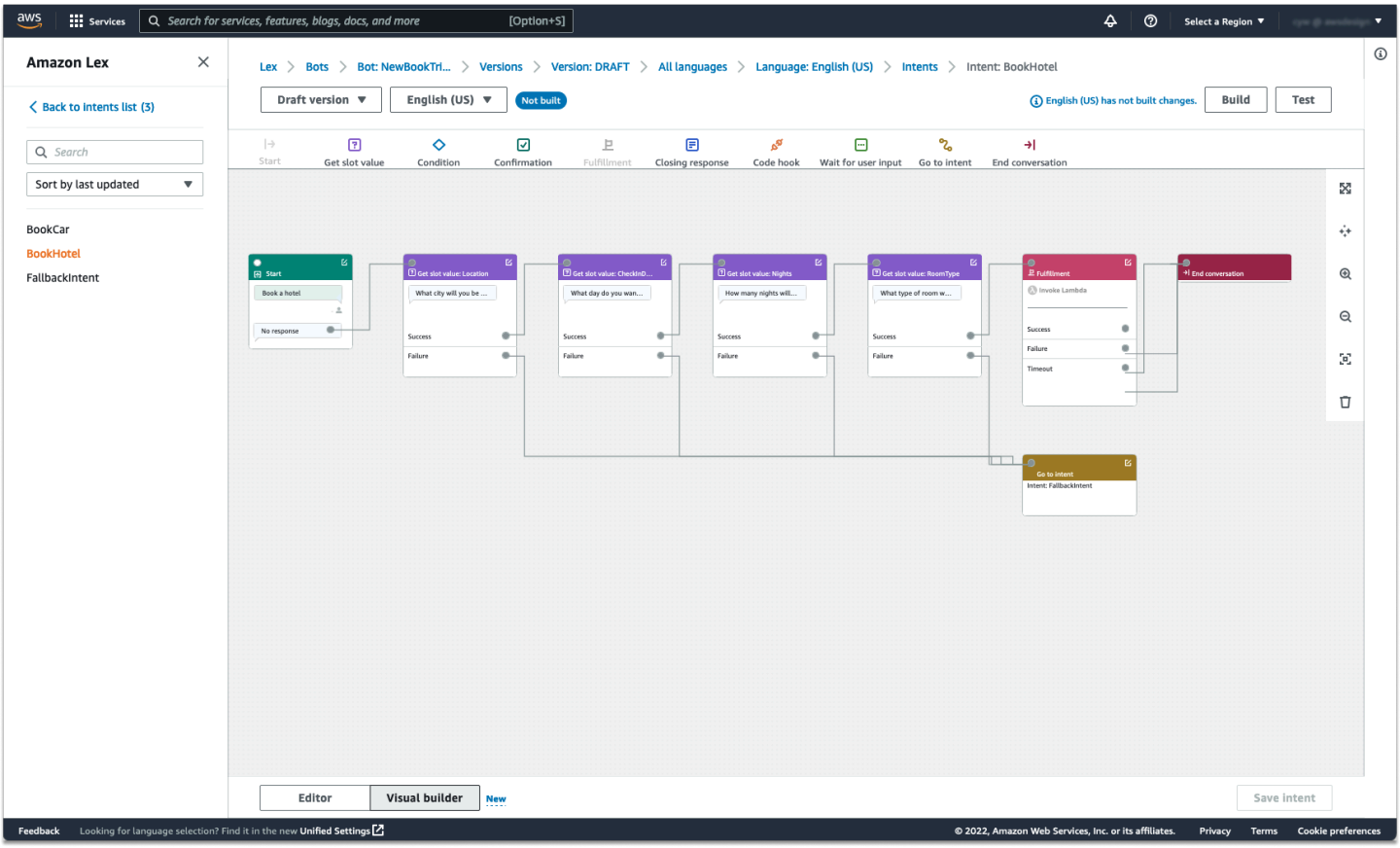
> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica ti offre un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi di dialogo tramite codice hook, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Usare Visual Conversation Builder

Visual Conversation Builder è un generatore di conversazioni drag and drop per progettare e visualizzare facilmente i percorsi di conversazione utilizzando gli intenti all'interno di un ricco ambiente visivo.

Per accedere al generatore di conversazioni visive

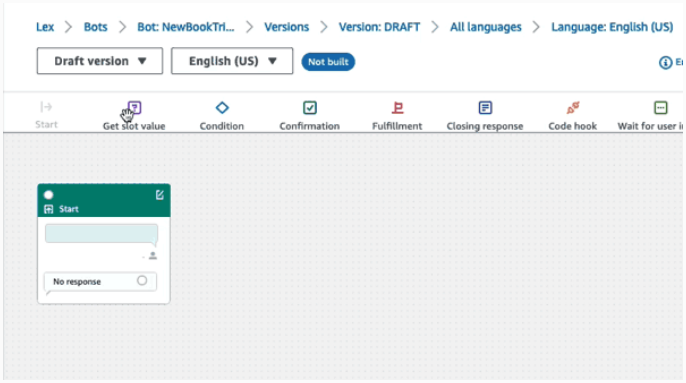
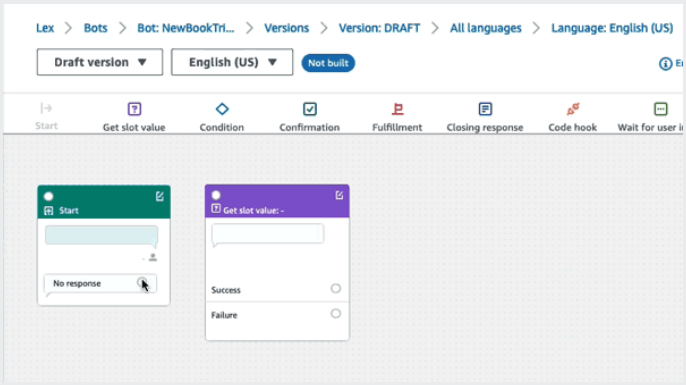
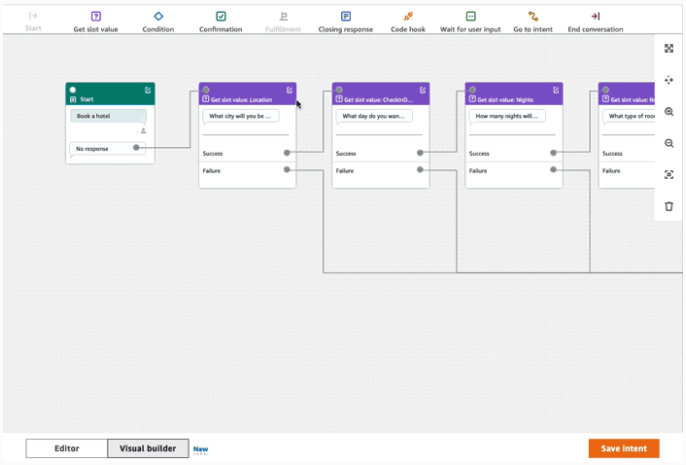
1. Nella console Amazon Lex V2, scegli un bot e selezionaIntentidal riquadro di navigazione a sinistra.
2. Accedere all'editor degli intenti in uno dei seguenti modi:
   - SelezionaAggiungi intentonell'angolo in alto a destra delIntentisezione, quindi scegli di aggiungere un intento vuoto o un intento incorporato.
   - Scegli il nome di un intento dalIntentisezione.
3. Nell'editor degli intenti, selezionaCostruttore visivonel riquadro nella parte inferiore dello schermo per accedere a Visual conversation builder.
4. Per tornare all'interfaccia dell'editor degli intenti del menu, selezionareRedattore.

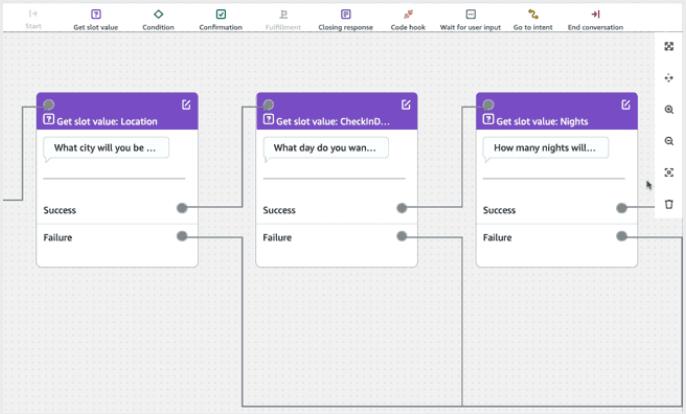
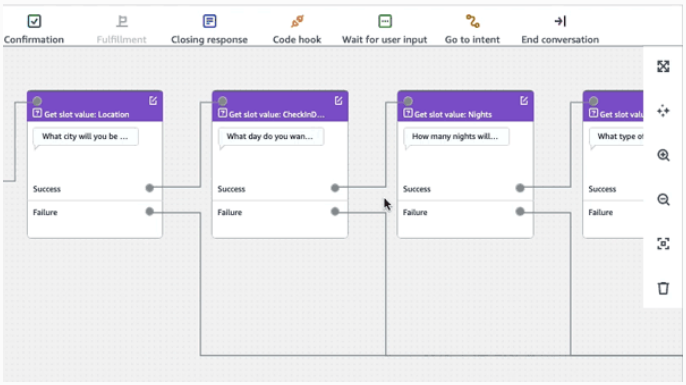
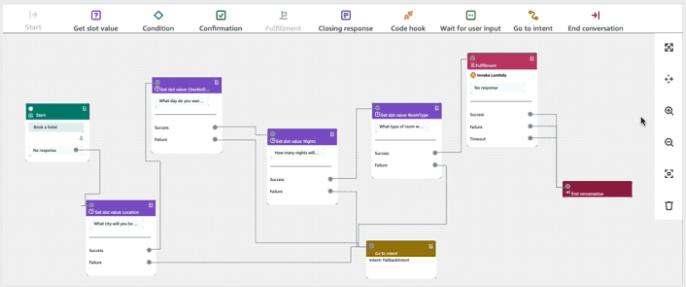


Visual Conversation Builder offre un'interfaccia utente più intuitiva con la possibilità di visualizzare e modificare il flusso della conversazione. Trascinando i blocchi, puoi estendere un flusso esistente o riordinare i passaggi della conversazione. Puoi sviluppare un flusso di conversazione con ramificazioni complesse senza scrivere alcun codice Lambda.

Questa modifica aiuta a separare la progettazione del flusso di conversazione da altre logiche aziendali in Lambda. Il generatore di conversazioni visive può essere utilizzato insieme all'editor di intenti esistente e può essere utilizzato per creare flussi di conversazione. Tuttavia, si consiglia di utilizzare la visualizzazione dell'editor visivo per flussi di conversazione più complessi.

Quando salvi un intento, Amazon Lex V2 può connettere automaticamente le intenzioni quando rileva che ci sono connessioni perse, Amazon Lex V2 suggerisce una connessione oppure puoi selezionare la tua connessione per il blocco.

Operazione	Esempio
Aggiungere un blocco all'area di lavoro	
Effettuare una connessione tra blocchi	
Aprire il pannello di configurazione su un blocco	

Operazione	Esempio
Ingrandisci per adattarlo	
Eliminare un blocco dal flusso di conversazione	
Pulizia automatica dell'area di lavoro	

Terminologia:

Blocco— L'unità costruttiva di base di un flusso di conversazione. Ogni blocco ha una funzionalità specifica per gestire diversi casi d'uso di una conversazione.

Porto— Ogni blocco contiene porte che possono essere utilizzate per collegare un blocco all'altro. I blocchi possono contenere porte di ingresso e porte di uscita. Ogni porta di uscita rappresenta una particolare variazione funzionale di un blocco (ad esempio errori, timeout o esito positivo).

Bordo— Un bordo è una connessione tra la porta di uscita di un blocco alla porta di ingresso di un altro blocco. Fa parte di un ramo di un flusso di conversazione.

Flusso di conversazione— Un insieme di blocchi collegati da bordi che descrive le interazioni a livello di intento con un cliente.

Blocchi

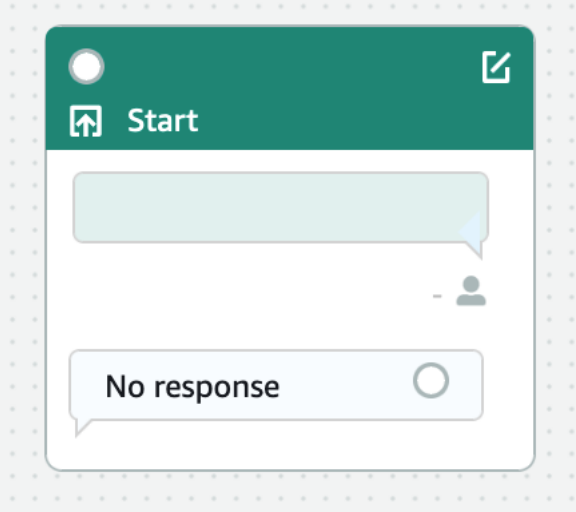
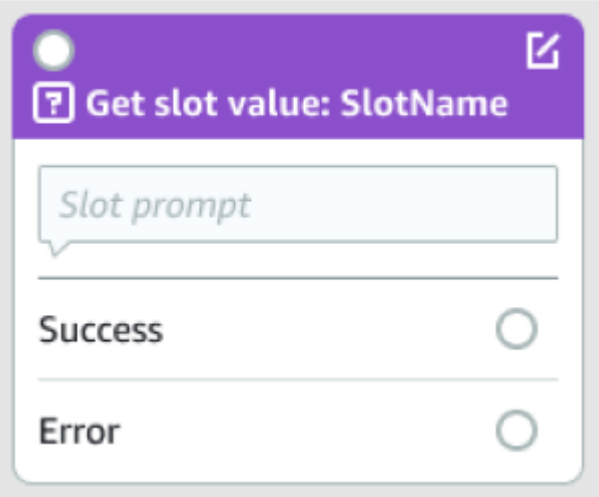
I blocchi sono gli elementi costitutivi della progettazione del flusso di conversazione. Rappresentano diversi stati all'interno dell'intento, che va dall'inizio dell'intento, all'input dell'utente, fino alla chiusura.

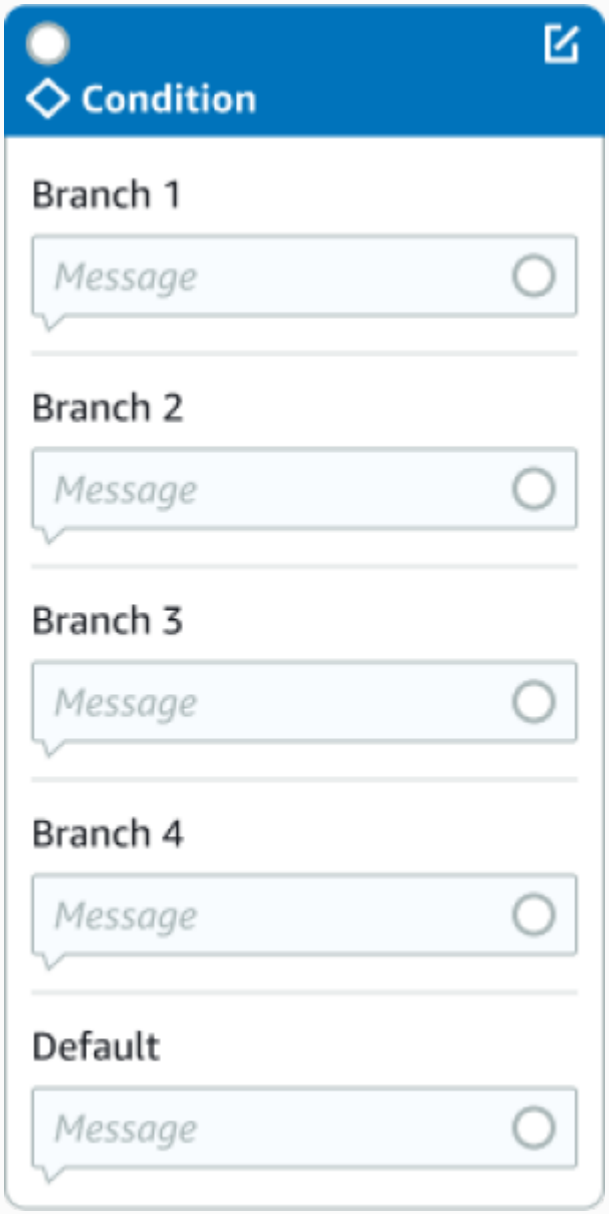
Ogni blocco ha un punto di ingresso e uno o più punti di uscita in base al tipo di blocco. Ogni punto di uscita può essere configurato con un messaggio corrispondente man mano che la conversazione procede attraverso i punti di uscita. Per i blocchi con più punti di uscita, i punti di uscita si riferiscono allo stato corrispondente al nodo. Per un nodo di condizione, i punti di uscita rappresentano le diverse condizioni.

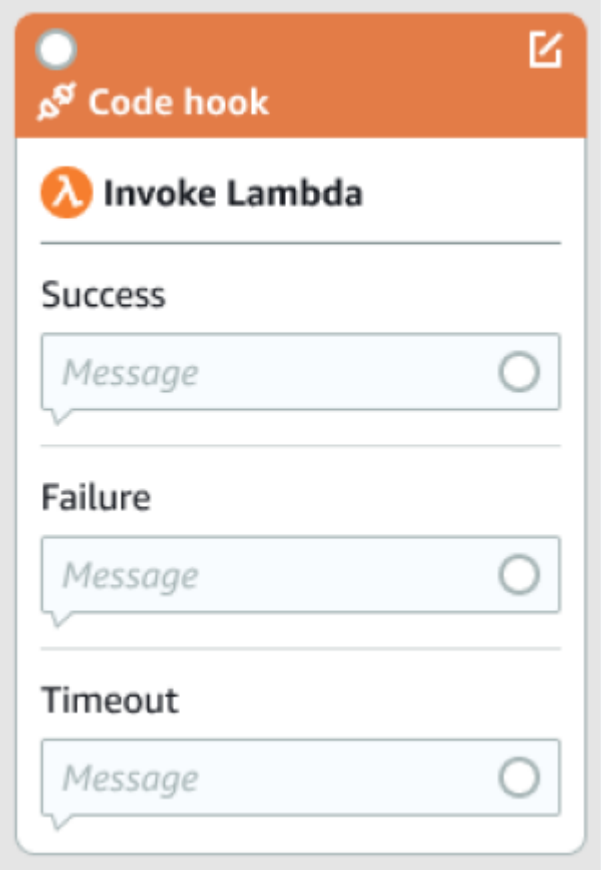
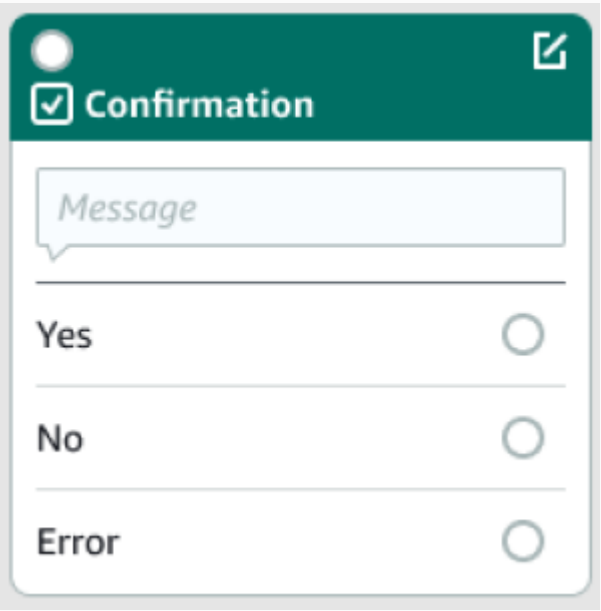
Ogni blocco ha un pannello di configurazione, che si apre facendo clic sulModificaicona nell'angolo in alto a destra del blocco. Il pannello di configurazione contiene campi dettagliati che possono essere configurati per corrispondere a ciascun blocco.

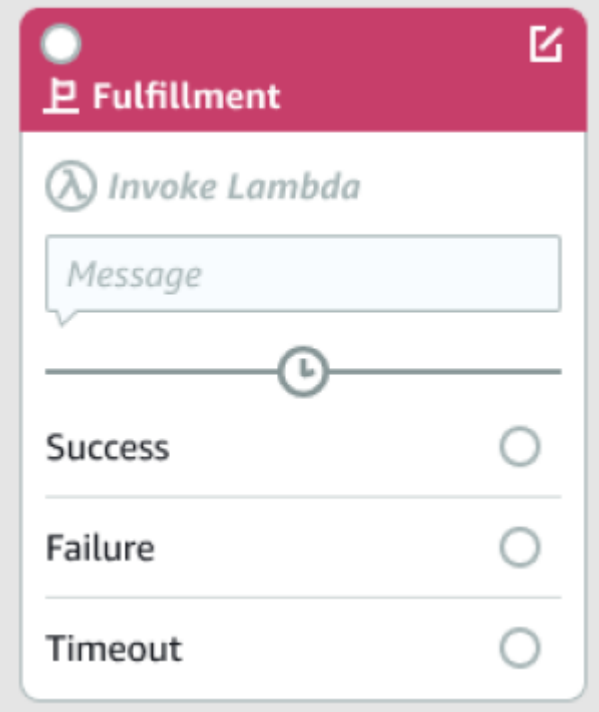
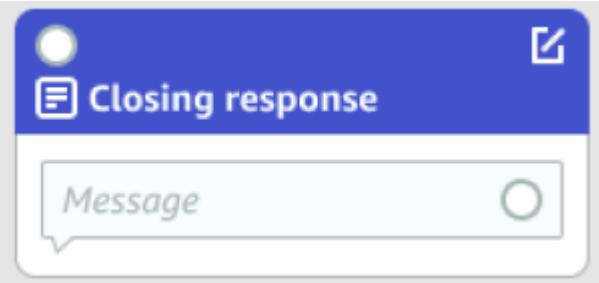
I prompt e i messaggi del bot possono essere configurati direttamente sul nodo trascinando un nuovo blocco, oppure possono essere modificati all'interno del pannello di destra, insieme ad altri attributi del blocco.

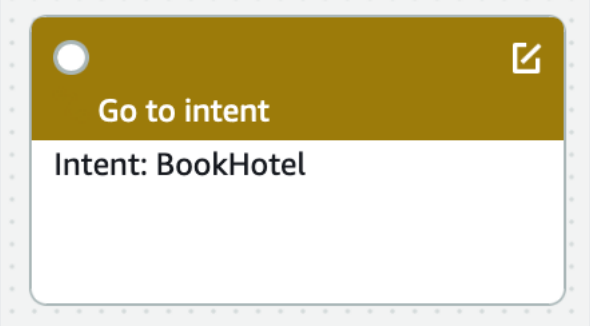
Tipi di blocchi— Ecco i tipi di blocchi che puoi usare con Visual Conversation Builder.

Tipo di blocco	Blocco
Inizio— La radice o il primo blocco del flusso di conversazione. Questo blocco può anche essere configurato in modo che il bot possa inviare una risposta iniziale (messaggio che indica che l'intento è stato riconosciuto). Per ulteriori informazioni, consulta Risposta iniziale.	
Ottieni il valore dello slot— Questo blocco cerca di ottenere valore per un singolo slot. Questo blocco ha un'impostazione per attendere la risposta del cliente alla richiesta di richiesta dello slot. Per ulteriori informazioni, consulta Slot.	

Tipo di blocco	Blocco
Condizione— Questo blocco contiene condizionali. Contiene fino a 4 rami personalizzati (con condizioni) e un ramo predefinito. Per ulteriori informazioni, consulta Aggiungi condizioni alle conversazioni delle filiali.	

Tipo di blocco	Blocco
Gancio del codice di dialogo— Questo blocco gestisce l'invocazione della funzione Dialog Lambda. Questo blocco contiene le risposte dei bot basate sulla funzione di dialogo Lambda che ha avuto esito positivo, negativo o scaduto. Per ulteriori informazioni, consulta Richiama l'hook del codice di dialogo.	
Conferma— Questo blocco interroga il cliente prima dell'adempimento dell'intento. Contiene le risposte dei bot basate sul fatto che il cliente dice sì o no alla richiesta di conferma. Per ulteriori informazioni, consulta Conferma.	

Tipo di blocco	Blocco
Adempimento— Questo blocco gestisce l'adempimento dell'intento, di solito dopo l'elicitazione degli slot. Può essere configurato per richiamare le funzioni Lambda e per rispondere con messaggi, se l'adempimento ha esito positivo o negativo. Per ulteriori informazioni, consulta Compimento.	
Risposta di chiusura— Questo blocco consente al bot di rispondere con un messaggio prima di terminare la conversazione. Per ulteriori informazioni, consulta Risposta di chiusura.	
Termina conversazione— Questo blocco indica la fine del flusso di conversazione.	
Attendi l'input dell'utente— Questo blocco può essere utilizzato per acquisire input dal cliente e passare a un altro intento in base all'enunciato.	

Tipo di blocco	Blocco
Vai all'intento— Questo blocco può essere utilizzato per passare a un nuovo intento o per richiamare direttamente uno spazio specifico di tale intento.	

Tipi di porte

Tutti i blocchi contengono una porta di ingresso, che viene utilizzata per connettere i blocchi principali. La conversazione può fluire verso la porta di ingresso di un determinato blocco solo dalla porta di uscita del blocco padre. Tuttavia, i blocchi possono contenere zero, una o più porte di uscita. I blocchi senza porte di uscita indicano la fine del flusso di conversazione nell'intento corrente (`GoToIntent`,`EndConversation`,`WaitForUserInput`).

Regole di progettazione degli intenti:

- Tutti i flussi di un intento iniziano con il blocco iniziale.
- I messaggi corrispondenti a ciascun punto di uscita sono opzionali.
- È possibile configurare i blocchi per impostare i valori corrispondenti a ciascun punto di uscita nel pannello di configurazione.
- In un singolo flusso all'interno di un intento può esistere solo un singolo blocco di avvio, conferma, adempimento e chiusura. Potrebbero esistere più condizioni, agganciare il codice di dialogo, ottenere i valori degli slot, terminare la conversazione, trasferire e attendere che l'utente inserisca blocchi.
- Un blocco di condizioni non può avere una connessione diretta a un blocco di condizioni. Lo stesso vale per l'hook del codice di dialogo.
- I flussi circolari sono consentiti per tre blocchi, ma non è consentito un connettore in ingresso per Start Intent.
- Uno slot opzionale non dispone di un connettore in ingresso o di una connessione in uscita e viene utilizzato principalmente per acquisire i dati presenti durante la richiesta di intenti. Ogni altro slot che fa parte del percorso di conversazione deve essere uno slot obbligatorio.

Blocchi:

- Il blocco iniziale deve avere un bordo in uscita.
- Ogni blocco get slot value deve avere un bordo in uscita dalla porta di successo, se lo slot è richiesto.
- Ogni blocco condizionale deve avere un bordo in uscita da ogni ramo se il blocco è attivo.
- Un blocco di condizioni non può avere più di un genitore.
- Un blocco di condizioni attivo deve avere un bordo in entrata.
- Ogni blocco di code hook attivo deve avere un vantaggio in uscita da ciascuna porta: successo, errore e timeout.
- Un blocco di code hook attivo deve avere un edge in entrata.
- Un blocco di conferma attivo deve avere un edge in entrata.
- Un blocco logistico attivo deve avere un vantaggio in entrata.
- Un blocco di chiusura attivo deve avere un bordo in entrata.
- Un blocco di condizioni deve avere almeno un ramo non predefinito.
- Un blocco di accesso all'intento deve avere un intento specificato.

Bordi:

- Un blocco di condizioni non può essere collegato a un altro blocco di condizioni.
- Un blocco code hook non può essere collegato a un altro blocco di code hook.
- Un blocco di condizioni può essere collegato solo a zero o a un blocco di codice hook.
- La connessione (code hook -> condition -> code hook) non è valida.
- Un blocco logistico non può avere un blocco code hook da bambino.
- Un blocco di condizioni, che è figlio del blocco logistico, non può avere un blocco code hook secondario.
- Un blocco di chiusura non può avere un blocco code hook come elemento secondario.
- Un blocco di condizioni figlio del blocco di chiusura non può avere un blocco code hook figlio.
- Un blocco di valore dello slot start, di conferma o get slot non può avere più di un blocco di code hook nella sua catena di dipendenze.

> **Note**
>
> Il 17 agosto 2022, Amazon Lex V2 ha rilasciato una modifica al modo in cui le conversazioni vengono gestite con l'utente. Questa modifica consente un maggiore controllo sul percorso che l'utente segue durante la conversazione. Per ulteriori informazioni, consulta Comprendere la gestione del flusso di conversazione. I bot creati prima del 17 agosto 2022 non supportano i messaggi hook con codice di dialogo, l'impostazione di valori, la configurazione dei passaggi successivi e l'aggiunta di condizioni.

Intenti incorporati

Per le 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 menu a sinistra, scegli la lingua e poi scegli Intenti.
4. Scegli Aggiungi intento, quindi scegli Usa intento integrato.
5. In Intento integrato, scegli l'intento da usare.
6. Assegna un nome all'intento, quindi scegli Aggiungi.
7. Usa l'editor degli intenti per configurare l'intento come richiesto per il tuo bot.

Argomenti

- AMAZON.CancelIntent
- AMAZON.FallbackIntent

- AMAZON.HelpIntent
- AMAZON.KendraSearchIntent
- AMAZON.PauseIntent
- AMAZON.QnAIntent
- AMAZON.RepeatIntent
- AMAZON.ResumeIntent
- AMAZON.StartOverIntent
- AMAZON.StopIntent

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 V2 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 V2 richiama l'intento di fallback per gestire la risposta.

Il tipo di `AMAZON.FallbackIntent` intento integrato viene aggiunto automaticamente al bot quando si crea un bot utilizzando la console o quando si aggiunge una localizzazione a un bot utilizzando l'operazione. CreateBotLocale

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.

Amazon Lex V2 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 V2 non è in grado di determinare quale intento richiamare.

L'intento di fallback viene richiamato quando:

- 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.

Non puoi aggiungere quanto segue a un intento di fallback:

- Enunciazioni
- Slot
- Prompt di conferma

Utilizzo di una funzione Lambda con un intento di fallback

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

- Restituisce le informazioni sull'intento all'applicazione client.
- Richiama la funzione Lambda di convalida e adempimento degli alias. 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 `fulfillmentCodeHook` dell'operazione CreateIntent.

Se si utilizza la funzione Lambda con l'intento di fallback, è possibile utilizzare questa funzione per chiamare un altro intent o per eseguire una qualche forma di comunicazione con l'utente, ad esempio raccogliere un numero di callback o aprire una sessione con un rappresentante del servizio clienti.

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 V2 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.

Puoi configurare la tua 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 della 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 V2 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 nelle impostazioni locali in inglese (USA) (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 V2 chiama l'intento ogni volta che non è in grado di determinare l'espressione dell'intento da parte dell'utente. Se non viene ricevuta alcuna risposta da Amazon Kendra, la conversazione continua come configurato nel bot.

> **Note**
>
> Amazon Lex V2 attualmente non supporta l'elicitazione `AMAZON.KendraSearchIntent` durante lo slot. Se Amazon Lex V2 non è in grado di determinare l'enunciato dell'utente per uno slot, chiama il. `AMAZON.FallbackIntent`

Quando usi il `AMAZON.KendraSearchIntent` con lo `AMAZON.FallbackIntent` stesso bot, Amazon Lex V2 utilizza gli intenti come segue:

1. Amazon Lex V2 chiama il`AMAZON.KendraSearchIntent`. L'intento chiama l'operazione Amazon Kendra`Query`.
2. Se Amazon Kendra restituisce una risposta, Amazon Lex V2 mostra il risultato all'utente.
3. Se non viene ricevuta alcuna risposta da Amazon Kendra, Amazon Lex V2 richiede nuovamente una richiesta all'utente. L'operazione successiva dipende dalla risposta dell'utente.
   - Se la risposta dell'utente contiene un enunciato riconosciuto da Amazon Lex V2, 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 V2, Amazon Lex V2 effettua un'altra chiamata all'operazione. `Query`
4. Se non viene fornita alcuna risposta dopo il numero di tentativi configurato, Amazon Lex V2 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 V2 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 V2 utilizza il filtro nella richiesta di query.

- Aggiungi parametri di query aggiuntivi alla richiesta per restringere i risultati della ricerca utilizzando la funzione Lambda. 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. Puoi creare una richiesta di query Amazon Kendra completa inviata da Amazon Lex V2. 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 V2 invia ad Amazon Kendra, puoi specificare una query `kendraQueryRequestPayload` nel campo della tua funzione Lambda. Se la query non è valida, Amazon Lex V2 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 V2 di assumere un ruolo di runtime autorizzato a chiamare 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 V2, 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 V2 o utilizzare un ruolo specifico per chiamare l'operazione Amazon Kendra. `Query` Quando utilizzi l'SDK AWS CLI o un SDK per creare l'intento, devi utilizzare un ruolo specifico per chiamare 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 V2. 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 V2 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/lexv2.amazonaws.com/
AWSServiceRoleForLexBots*"
    },
    {
        "Effect": "Allow",
        "Action": "iam:ListRoles",
        "Resource": "*"
    }
]
}
```

Specifica di un ruolo

Puoi utilizzare la console AWS 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 V2 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. In AWS 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 V2 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 V2 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 della richiesta. Per ulteriori informazioni sugli attributi di sessione, consulta Impostazione degli attributi della 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 in una risposta dalla dichiarazione di intento. `IntentClosingSetting` L'intento deve avere un'`closingResponse`istruzione a meno che una funzione Lambda non produca un messaggio di risposta di chiusura.

Amazon Kendra offre cinque tipi di risposte.

- Le due risposte seguenti richiedono la configurazione di una FAQ per il tuo indice Amazon Kendra. Per maggiori dettagli, consulta Aggiungere domande e risposte direttamente a un indice.
  - `x-amz-lex:kendra-search-response-question_answer-question-<N>`— La domanda di una FAQ che corrisponde alla ricerca.
  - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— La risposta di una FAQ che corrisponde alla ricerca.
- Le tre risposte seguenti richiedono la configurazione di un'origine dati per il tuo indice Amazon Kendra. Per maggiori dettagli, consulta Creazione di una fonte di dati.
  - `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 `closingResponse` 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 V2 crea una query a partire 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 qualsiasi funzionalità di Amazon Kendra. Se non si specifica una query valida, si ottiene un'eccezione `InvalidLambdaResponseException`. Per ulteriori informazioni, consulta Query nella Amazon Kendra Developer Guide.

Dopo aver creato il filtro o la stringa di query, invii la risposta ad Amazon Lex V2 con il `dialogAction` campo della risposta impostato `delegate` su. Amazon Lex V2 invia la query ad Amazon Kendra e quindi restituisce la risposta alla query all'hook del codice di evasione dell'ordine.

Utilizzo dell'hook del codice di adempimento per la risposta

Dopo che Amazon Lex V2 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.

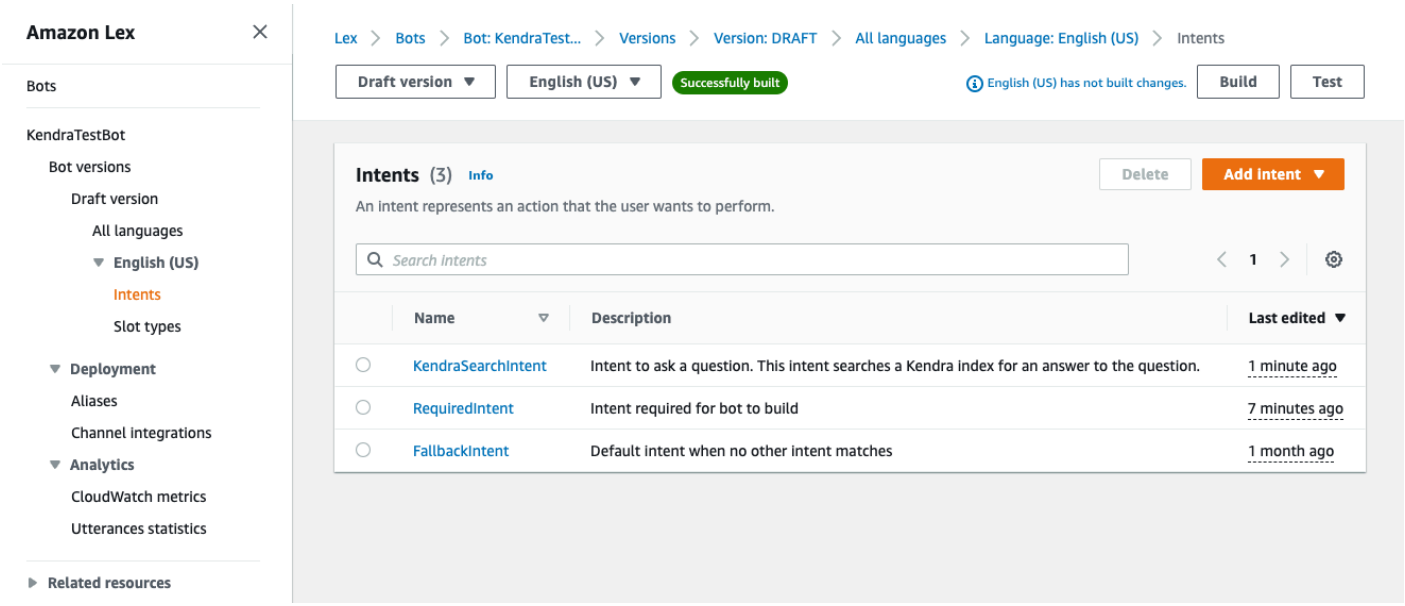
L'hook del codice di adempimento è facoltativo. Se non ne esiste uno o se il code hook non restituisce un messaggio nella risposta, Amazon Lex V2 utilizza l'`closingResponse`istruzione per le risposte.

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

Questo esempio crea un bot Amazon Lex V2 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. Ecco un riepilogo di come creerai il tuo bot per le domande frequenti utilizzando un indice Amazon Kendra:

1. Creare un bot con cui i clienti interagiranno per ottenere risposte dal bot.

2. Creare un intento personalizzato. Poiché `AMAZON.KendraSearchIntent` e `AMAZON.FallbackIntent` sono intenti di backup, il bot richiede almeno un altro intento che deve contenere almeno un'enunciazione. Questo intento consente la c reazione del bot, ma non viene usato altrimenti. Il tuo FAQ bot conterrà quindi almeno tre intenti, come nell'immagine seguente:



3. Aggiungi l'`AMAZON.KendraSearchIntent`intento al tuo bot e configuralo in modo che funzioni con il tuo indice Amazon Kendra.
4. Testa il bot effettuando una query e verificando che i risultati del tuo indice Amazon Kendra siano documenti che rispondono alla query.

Prerequisiti

Prima di poter utilizzare questo esempio, devi creare un indice Amazon Kendra. Per ulteriori informazioni, consulta la sezione Guida introduttiva alla console Amazon Kendra nella Amazon Kendra Developer Guide. Per questo esempio, scegli il set di dati di esempio (documentazione Sample AWS) come fonte di dati.

Per creare un bot per le 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. Scegli Crea bot.

    a. Per il metodo di creazione, scegli Crea un bot vuoto.

b. Nella sezione Configurazione del bot, assegna al bot un nome che ne indichi lo scopo, ad esempio**`KendraTestBot`**, e una descrizione opzionale. Il nome deve essere univoco nel tuo account.

c. Nella sezione Autorizzazioni IAM, scegli Crea un ruolo con autorizzazioni Amazon Lex di base. Questo creerà un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni necessarie ad Amazon Lex V2 per eseguire il bot.

d. Nella sezione Children's Online Privacy Protection Act (COPPA), scegli No.

e. Nelle sezioni Timeout della sessione inattiva e Impostazioni avanzate, lascia le impostazioni predefinite e scegli Avanti.

f. Ora ti trovi nella sezione Aggiungi lingua al bot. Nel menu sotto Interazione vocale, seleziona Nessuno. Questa è solo un'applicazione basata su testo. Lascia le impostazioni predefinite per i campi rimanenti.

g. Seleziona Fatto. Amazon Lex V2 crea il bot e un intento predefinito chiamato NewIntente ti porta alla pagina per configurare questo intento

Per creare correttamente un bot, devi creare almeno un intento separato da e da. `AMAZON.FallbackIntent` `AMAZON.KendraSearchIntent` Questo intento è necessario per creare un bot Amazon Lex V2, ma non viene utilizzato per la risposta alle domande frequenti. Questo intento deve contenere almeno un esempio di enunciato e l'enunciato non deve essere applicabile a nessuna delle domande poste dal cliente.

Per creare l'intento richiesto:

1. Nella sezione Dettagli sull'intento, assegna un nome all'intento, ad esempio. **`RequiredIntent`**
2. Nella sezione Esempi di enunciati, digitate un enunciato nella casella accanto a Aggiungi enunciato, ad esempio. **`Required utterance`** Quindi scegli Aggiungi enunciato.
3. Scegliere Salva intento.

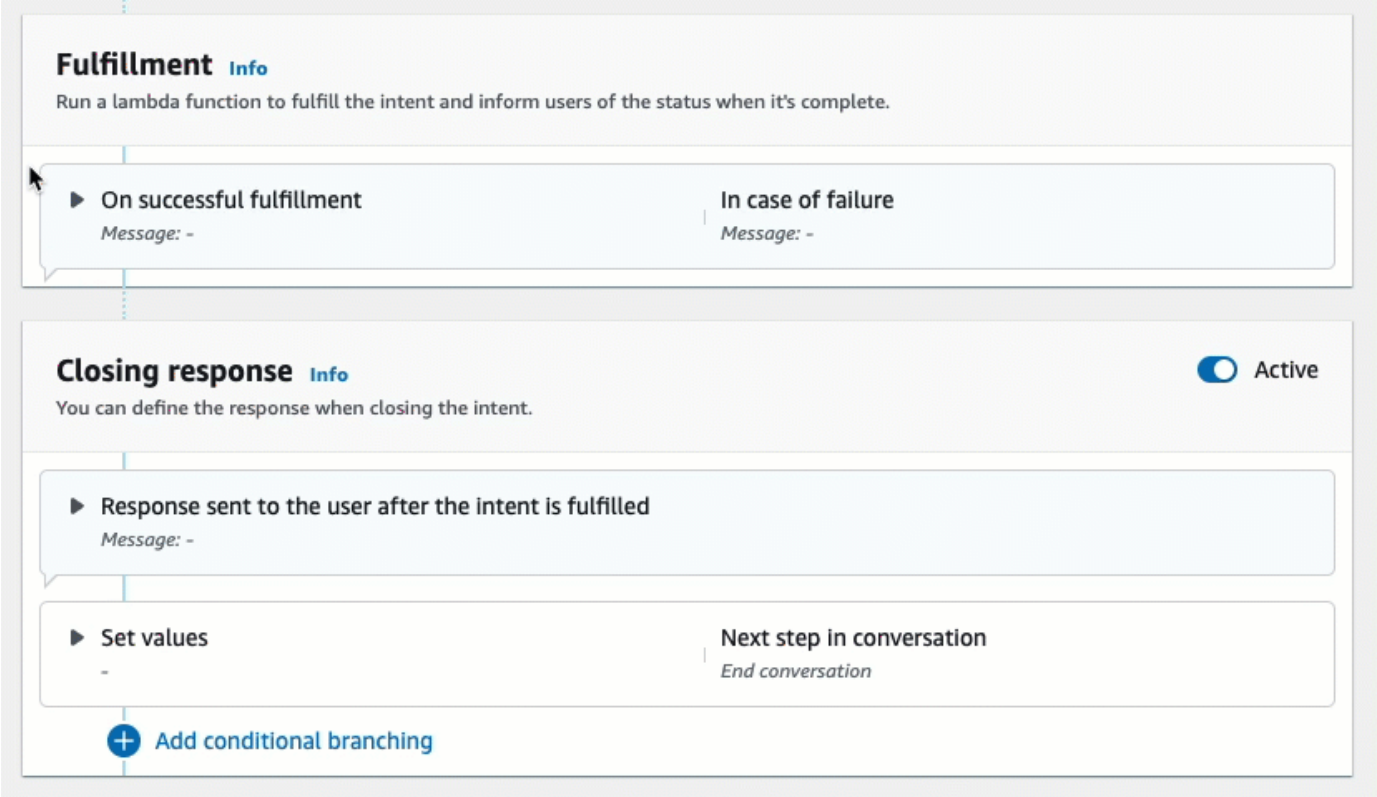
Crea l'intento di cercare un indice Amazon Kendra e il messaggio di risposta che dovrebbe restituire.

Per creare un AMAZON. KendraSearchIntent messaggio di intento e risposta:

1. Seleziona Torna all'elenco degli intenti nel riquadro di navigazione per tornare alla pagina degli intenti del bot. Scegli Aggiungi intento e seleziona Usa intento integrato dal menu a discesa.

2. Nella casella che si apre, seleziona il menu sotto Integrato intento. Entra **`AMAZON.KendraSearchIntent`** nella barra di ricerca e poi selezionalo dall'elenco.
3. Assegna un nome all'intento, ad esempio**`KendraSearchIntent`**.
4. Dal menu a discesa dell'indice di Amazon Kendra, scegli l'indice che desideri venga cercato. L'indice che hai creato nella sezione Prerequisiti dovrebbe essere disponibile.
5. Selezionare Aggiungi.
6. Nell'editor degli intenti, scorri verso il basso fino alla sezione Adempimento, seleziona la freccia destra per espandere la sezione e aggiungi il seguente messaggio nella casella In caso di adempimento riuscito:

```
I found a link to a document that could help you: ((x-amz-lex:kendra-search-
response-document-link-1)).
```



Per ulteriori informazioni su Amazon Kendra Search Response, consulta Using the Search Response.

7. Selezionare Salva intento, quindi Crea per creare il bot. Quando il bot è pronto, il banner nella parte superiore dello schermo diventa verde e mostra un messaggio di successo.

Infine, usa la finestra di test della console per testare le risposte dal bot.

Per testare il tuo FAQ bot:

1. Dopo che il bot è stato creato con successo, scegli Test.
2. Entra **`What is Amazon Kendra?`** nella finestra di test della console. Verifica che il bot risponda con un link.
3. Per ulteriori informazioni sulla configurazione`AMAZON.KendraSearchIntent`, consulta AMAZON.KendraSearchIntente. KendraConfiguration

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 quello

AMAZON.QnAIntent

> **Note**
>
> Prima di poter sfruttare le funzionalità di intelligenza artificiale generativa, è necessario soddisfare i seguenti prerequisiti
>
> 1. Accedi alla console Amazon Bedrock e registrati per accedere al modello Anthropic Claude che intendi utilizzare (per ulteriori informazioni, consulta Model access). Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock.
> 2. Attiva le funzionalità di intelligenza artificiale generativa per le impostazioni locali del tuo bot. Per farlo, segui i passaggi indicati inOttimizza la creazione e le prestazioni dei bot con l'IA generativa.

Risponde alle domande dei clienti utilizzando Amazon Bedrock FM per cercare e riepilogare le risposte alle domande frequenti. Questo intento si attiva quando un enunciato non è classificato in nessuno degli altri intenti presenti nel bot. Nota che questo intento non verrà attivato in caso di enunciazioni perse quando si ottiene un valore di slot. Una volta riconosciuto`AMAZON.QnAIntent`, utilizza il modello Amazon Bedrock specificato per effettuare ricerche nella knowledge base configurata e rispondere alla domanda del cliente.

Se la risposta dalla FM non è soddisfacente o la chiamata alla FM non riesce, Amazon Lex V2 richiama il. `AMAZON.FallbackIntent`

> **Warning**
>
> Non è possibile utilizzare `AMAZON.QnAIntent` and the nella stessa lingua del bot`AMAZON.KendraSearchIntent`.

Sono disponibili le seguenti opzioni del knowledge store. È necessario aver già creato il Knowledge Store e indicizzato i documenti al suo interno.

- OpenSearch Dominio di servizio: contiene documenti indicizzati. Per creare un dominio, segui i passaggi indicati in Creazione e gestione dei domini Amazon OpenSearch Service.
- Indice Amazon Kendra: contiene documenti di domande frequenti indicizzati. Per creare un indice Amazon Kendra, segui la procedura descritta in Creazione di un indice.
- Knowledge base Amazon Bedrock: contiene fonti di dati indicizzate. Per configurare una knowledge base, segui i passaggi riportati in Creazione di una knowledge base.

Se selezioni questo intento, configuri i seguenti campi e quindi seleziona Aggiungi per aggiungere l'intento.

- Modello Bedrock: scegli il fornitore e il modello di base da utilizzare per questo intento. Attualmente sono supportati Anthropic Claude V2 e Anthropic Claude Instant.
- Knowledge Store: scegli la fonte da cui desideri che il modello estragga le informazioni per rispondere alle domande dei clienti. Sono disponibili le seguenti fonti.
  - OpenSearch— Configurare i seguenti campi.
    - Endpoint di dominio: fornisci l'endpoint di dominio che hai creato per il dominio o che ti è stato fornito dopo la creazione del dominio.

    - Nome dell'indice: fornisce l'indice per la ricerca. Per ulteriori informazioni, consulta Indicizzazione dei dati in Amazon OpenSearch Service.
    - Scegli come restituire la risposta al cliente.
        - Risposta esatta: quando questa opzione è abilitata, il valore nel campo Risposta viene utilizzato così com'è per la risposta del bot. Il modello di base Amazon Bedrock configurato viene utilizzato per selezionare l'esatto contenuto della risposta così com'è, senza alcuna sintesi o riepilogo del contenuto. Specificare il nome dei campi di domanda e risposta configurati nel database. OpenSearch
        - Includi campi: restituisce una risposta generata dal modello utilizzando i campi specificati. Specificate il nome di un massimo di cinque campi configurati nel OpenSearch database. Utilizzate un punto e virgola (;) per separare i campi.
- Amazon Kendra: configura i seguenti campi.
    - Indice Amazon Kendra: seleziona l'indice Amazon Kendra in cui desideri che il bot cerchi.
    - Filtro Amazon Kendra: per creare un filtro, seleziona questa casella di controllo. Per ulteriori informazioni sul formato JSON del filtro di ricerca Amazon Kendra, consulta Utilizzo degli attributi del documento per filtrare i risultati della ricerca.
    - Risposta esatta: per consentire al bot di restituire la risposta esatta restituita da Amazon Kendra, seleziona questa casella di controllo. Altrimenti, il modello Amazon Bedrock selezionato genera una risposta basata sui risultati.

      > **Note**
      >
      > Per utilizzare questa funzionalità, devi prima aggiungere domande frequenti al tuo indice seguendo i passaggi riportati in Aggiungere domande frequenti (FAQ) a un indice.
- Knowledge base Amazon Bedrock: se scegli questa opzione, specifica l'ID della knowledge base. Puoi trovare l'ID controllando la pagina dei dettagli della knowledge base nella console o inviando una GetKnowledgeBaserichiesta.

Le risposte di QNAIntent verranno memorizzate negli attributi della richiesta come mostrato di seguito:

- `x-amz-lex:qnA-search-response`— La risposta di QNAIntent alla domanda o all'enunciato.

- `x-amz-lex:qnA-search-response-source`— Indirizza il documento o l'elenco di documenti utilizzati per generare la risposta.

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'`PutSessio`noperazione per ottenere nuovamente il valore del primo slot.

Enunciati comuni:

- ricominciare da capo
- 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

Aggiungere tipi di slot

I tipi di slot definiscono i valori che gli utenti possono fornire per le variabili di intento. Si definiscono i tipi di slot per ogni lingua in modo che i valori siano specifici per quella lingua. Ad esempio, per un tipo di slot che elenca i colori della vernice, è possibile includere il valore red "" in inglese, rouge "" in francese e "rojo" in spagnolo.

Questo argomento descrive come creare tipi di slot personalizzati che forniscano valori per gli slot del vostro intento. È inoltre possibile utilizzare tipi di slot incorporati per valori standard. Ad esempio, è possibile utilizzare il tipo di slot integrato `AMAZON.Country` per un elenco di paesi nel mondo.

Per creare un tipo di slot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli il bot a cui desideri aggiungere la lingua, quindi scegli Struttura della conversazione e quindi Tutte le lingue.
3. Scegli la lingua a cui aggiungere il tipo di slot, quindi scegli Tipi di slot.
4. Scegli Aggiungi tipo di slot, dai un nome al tipo di slot, quindi scegli Aggiungi.
5. Nell'editor dei tipi di slot, aggiungi i dettagli del tipo di slot.

    - Risoluzione dei valori degli slot: determina come vengono risolti i valori degli slot. Se scegli Espandi valori, Amazon Lex V2 utilizza i valori come valori rappresentativi per la formazione.

Se utilizzi Limita ai valori dello slot, i valori consentiti per lo slot sono limitati a quelli forniti da te.

- Valori del tipo di slot: i valori per lo slot. Se si sceglie Limita ai valori dello slot, è possibile aggiungere sinonimi per il valore. Ad esempio, per il valore «calcio» puoi aggiungere il sinonimo «calcio». Se l'utente inserisce «calcio» in una conversazione con il bot, il valore effettivo dello slot è «calcio».
- Usa i valori degli slot come vocabolario personalizzato: abilita questa opzione per migliorare il riconoscimento dei valori degli slot e dei sinonimi nelle conversazioni audio. Non abilitate questa opzione quando i valori degli slot sono termini comuni, come «sì», «no», «uno», «due», «tre», ecc.

6. Scegli Salva il tipo di slot.

Amazon Lex V2 offre i seguenti tipi di slot:

Argomenti

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.AlphaNumeric	Riconosce parole costituite da lettere e numeri.	Tutte le versioni locali tranne il coreano (ko-KR)	

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
Amazon.city	Riconosce le parole che rappresentano una città.	Tutte le località	
Amazon. Conferma	Riconosce le parole che significano «Sì», «No», «Forse» e «Non so» e le converte in un formato standard (Sì/No/Forse/Non so).	Inglese (en-US, en-GB, en-AU, en-IN, en-ZA)	
Amazon.country	Riconosce le parole che rappresentano un paese.	Tutte le località	
Amazon.date	Riconosce le parole che rappresentano una data e le converte in un formato standard.	Tutte le versioni locali	
Amazon.Durata	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	

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
AMAZON. FirstName	Riconosce le parole che rappresentano un nome.	Tutte le impostazioni locali	
AMAZON. 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	
AMAZON.Percentage	Riconosce le parole che rappresentano una percentuale e le converte in un numero e un segno di percentuale (%).	Tutte le impostazioni locali	
AMAZON. PhoneNumber	Riconosce le parole che rappresentano un numero di telefono e le converte in una stringa numerica.	Tutte le impostazioni locali	
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 località	

Tipo di slot	Breve descrizione	Impostazioni locali supportate	
Amazon.time	Riconosce le parole che indicano l'ora e le converte in un formato orario.	Tutte le impostazioni locali	
AMAZON.UK PostalCode	Riconosce le parole che rappresentano un codice postale del Regno Unito e le converte in un modulo standard.	Solo inglese (britannico) (en-GB)	
AMAZON. FreeFormInput	Riconosce le stringhe costituite da qualsiasi parola o carattere.	Tutte le impostazioni locali	

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`**

Il tipo di `AMAZON.AlphaNumeric` slot supporta input che utilizzano stili di ortografia. Puoi utilizzare gli spell-by-word stili spell-by-letter e per aiutare i tuoi clienti a inserire le lettere. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

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 canadesi
- Numeri della patente di guida
- Numeri di identificazione del veicolo

Usa un'espressione regolare standard. Amazon Lex V2 supporta i seguenti caratteri nell'espressione regolare:

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

Amazon Lex V2 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 un tipo di `AMAZON.AlphaNumeric` 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 CreateSlotType. L'espressione regolare viene convalidata quando si salva il tipo di slot. Se l'espressione non è valida, Amazon Lex V2 restituisce un messaggio di errore.

Quando usi un'espressione regolare in un tipo di slot, Amazon Lex V2 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 V2 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 V2 non si converte da variante a nome ufficiale.

Esempi:

- New York
- Reykjavik
- Tokyo
- Versailles

Amazon. Conferma

Questo tipo di slot riconosce le frasi e le parole di input che corrispondono a «Sì», «No», «Forse» e «Non so» per Amazon Lex V2 e le converte in uno dei quattro valori. Può essere utilizzato per acquisire la conferma o la conferma da parte dell'utente. In base al valore finale risolto, puoi creare le condizioni per progettare più percorsi di conversazione.

Per esempio:

se {confirmation} = «Sì», soddisfa l'intento

altrimenti, ottieni un altro slot

Esempi:

- Sì, sì, sì, ok, certo, ce l'ho, sono d'accordo...
- No: No, Negativo, No, Scordatelo, rifiuterò, Assolutamente no...
- Forse: è possibile, forse, a volte, potrei, potrebbe essere giusto...
- Non lo so: non lo so, sconosciuto, non ne ho idea, non ne sono sicuro, chissà...

A partire dal 17 agosto 2023, se esiste un tipo di slot personalizzato denominato «Confirmation», il nome deve essere modificato per evitare conflitti con lo slot di conferma integrato. Nella barra di

navigazione a sinistra della console Lex, vai al tipo di slot (per un tipo di slot personalizzato esistente denominato Confirmation) e aggiorna il nome del tipo di slot. Il nome del nuovo tipo di slot non deve essere «Confirmation», che è una parola chiave riservata per il tipo di slot di conferma integrato.

Amazon.country

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 si riferiscono a una settimana futura, ad esempio «settimana prossima», vengono convertiti nella data dell'ultimo giorno della settimana corrente. 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. `2020-11-29` Le date che corrispondono alla settimana corrente o precedente vengono convertite nel primo giorno della settimana. Ad esempio, se oggi è il 25/11/2020, «settimana scorsa» viene convertito in. `2020-11-16`
- Gli enunciati che si riferiscono a un mese futuro, 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» viene convertito in. `2020-12-31` Per le date che corrispondono al mese corrente o precedente, convertite al primo giorno del mese. Ad esempio, se oggi è il 25/11/2020, «questo mese» corrisponde a. `2020-11-01`
- Gli enunciati che si riferiscono a un anno futuro, ma non a un mese o a un giorno specifici, ad esempio «anno prossimo», vengono convertiti nell'ultimo giorno dell'anno successivo. Ad esempio,

se oggi è il 25/11/2020, «l'anno prossimo» verrà convertito in. `2021-12-31` Per le date che corrispondono all'anno corrente o precedente, convertite al primo giorno dell'anno. Ad esempio, se oggi è il 25/11/2020, «anno scorso» viene convertito in. `2019-01-01`

Amazon.Durata

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` `P`indica 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 (+).

Il tipo di `AMAZON.EmailAddress` slot supporta input che utilizzano stili di ortografia. Puoi utilizzare gli spell-by-word stili spell-by-letter e per aiutare i tuoi clienti a inserire gli indirizzi e-mail. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

AMAZON. FirstName

Nomi di battesimo comunemente usati. Questo tipo di slot riconosce nomi formali, soprannomi informali e nomi composti da più di una parola. Il nome inviato all'intento è il valore inviato dall'utente. Amazon Lex V2 non viene convertito dal soprannome al nome formale.

Per i nomi che suonano allo stesso modo ma sono scritti in modo diverso, Amazon Lex V2 invia alle tue intenzioni un unico modulo comune.

Il tipo di `AMAZON.FirstName` slot supporta input che utilizzano stili di ortografia. Puoi utilizzare gli spell-by-word stili spell-by-letter e per aiutare i clienti a inserire i nomi. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

Esempi:

- Emilia
- John
- Sofia
- Anil Kumar

AMAZZONE. FirstName restituisce anche un elenco di nomi strettamente correlati in base al valore originale. È possibile utilizzare l'elenco di valori risolti per correggere errori di battitura, confermare il nome con l'utente o eseguire una ricerca nel database dei nomi validi nella directory utente.

Ad esempio, l'input «John» può comportare la restituzione di nomi correlati aggiuntivi come «John J» e «John-Paul».

Di seguito viene illustrato il formato di risposta per il tipo di slot `AMAZON.FirstName` integrato:

```
"value": {
    "originalValue": "John",
    "interpretedValue": "John",
    "resolvedValues": [
        "John",
        "John J.",
        "John-Paul"
    ]
}
```

AMAZON. LastName

Cognomi di uso comune. Per i nomi che hanno lo stesso suono e che vengono scritti in modo diverso, Amazon Lex V2 invia all'intento un'unica forma comune.

Il tipo di `AMAZON.LastName` slot supporta input che utilizzano stili di ortografia. Puoi utilizzare gli spell-by-word stili spell-by-letter e per aiutare i clienti a inserire i nomi. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

Esempi:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON. LastName restituisce anche un elenco di nomi strettamente correlati in base al valore originale. È possibile utilizzare l'elenco di valori risolti per correggere errori di battitura, confermare il nome con l'utente o eseguire una ricerca nel database dei nomi validi nella directory utente.

Ad esempio, l'input «Smith» può comportare la restituzione di nomi correlati aggiuntivi come «Smyth» e «Smithe».

Di seguito viene illustrato il formato di risposta per il tipo di slot integrato: `AMAZON.LastName`

```
"value": {
    "originalValue": "Smith",
    "interpretedValue": "Smith",
    "resolvedValues": [
        "Smith",
        "Smyth",
        "Smithe"
    ]
}
```

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
-15	-15
meno 15	-15

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%

Input	Risposta
venticinque percento	25%

AMAZON. PhoneNumber

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

Type	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 senza prefisso internazionale o prefisso	555-1212	`5551212`

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.

Esempi:

- Viale Canberra
- Strada anteriore
- Strada del mercato

Amazon.time

Converte le parole che rappresentano i tempi in valori temporali. `AMAZON.Time`può risolvere orari esatti, valori ambigui e intervalli di tempo. Il valore dello slot può corrispondere ai seguenti intervalli di tempo:

- AM
- PM
- MO (mattina)
- AF (pomeriggio)
- EV (sera)
- NI (notte)

Quando un utente inserisce un orario ambiguo, Amazon Lex V2 utilizza l'`slots`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 nel `interpretedValue` campo è `null` e il `resolvedValues` campo contiene le due possibili risoluzioni dell'ora. Amazon Lex V2 inserisce quanto segue nella funzione Lambda:

```
"slots": {
    "deliveryTime": {
```

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

Quando l'utente risponde con un orario inequivocabile, Amazon Lex V2 invia l'ora alla funzione Lambda nel `interpretedValue` campo dell'attributo `slots` dell'evento Lambda. Ad esempio, se l'utente risponde alla richiesta di un orario di consegna con "10:00», Amazon Lex V2 inserisce quanto segue nella funzione Lambda:

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

Quando l'utente risponde a una richiesta di orario di consegna con «al mattino», Amazon Lex V2 inserisce quanto segue nella funzione Lambda:

```
"slots": {
    "deliveryTime": {
    "value": {
        "originalValue": "morning",
        "interpretedValue": "MO",
        "resolvedValues": [
            "MO"
        ]
    }
}
```

Per ulteriori informazioni sui dati inviati da Amazon Lex V2 a una funzione Lambda, consulta. [Interpretazione del formato dell'evento di input]

AMAZON.UK PostalCode

Converte le parole che rappresentano un codice postale del Regno Unito in un formato standard per i codici postali del Regno Unito. Il tipo di `AMAZON.UKPostalCode` slot convalida e risolve il codice postale in un insieme di formati standardizzati, ma non verifica che il codice postale sia valido. L'applicazione deve convalidare il codice postale.

Il tipo di `AMAZON.UKPostalCode` slot è disponibile solo nella versione locale inglese (UK) (en-GB).

Il tipo di `AMAZON.UKPostalCode` slot supporta input che utilizzano stili di ortografia. Puoi utilizzare gli spell-by-word stili spell-by-letter e per aiutare i tuoi clienti a inserire le lettere. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

Il tipo di slot riconosce solo i formati di codice postale validi elencati di seguito, utilizzati nel Regno Unito. I formati validi sono («A» rappresenta una lettera e «9" rappresenta una cifra):

- AA9A 9AA
- A9A 9AA
- A9 9AA
- A99 9AA
- A9 9AA
- A9 9AA

Per l'immissione di testo, l'utente può inserire qualsiasi combinazione di lettere maiuscole e minuscole. L'utente può utilizzare o omettere lo spazio nel codice postale. Il valore risolto includerà sempre lo spazio nella posizione corretta per il codice postale.

Per l'input vocale, l'utente può pronunciare i singoli caratteri oppure utilizzare pronunce doppie, come «doppia A» o «doppia 9". Possono anche usare pronunce a due cifre, come «novantanove» per «99".

> **Note**
>
> Non tutti i codici postali del Regno Unito sono riconosciuti. Sono supportati solo i formati sopra elencati.

AMAZON. FreeFormInput

`AMAZON.FreeFormInput`può essere utilizzato per acquisire input in formato libero dall'utente finale. Riconosce le stringhe costituite da parole o caratteri. Il valore risolto è l'intero enunciato di input.

Esempio:

Bot: fornisci un feedback sulla tua esperienza di chiamata.

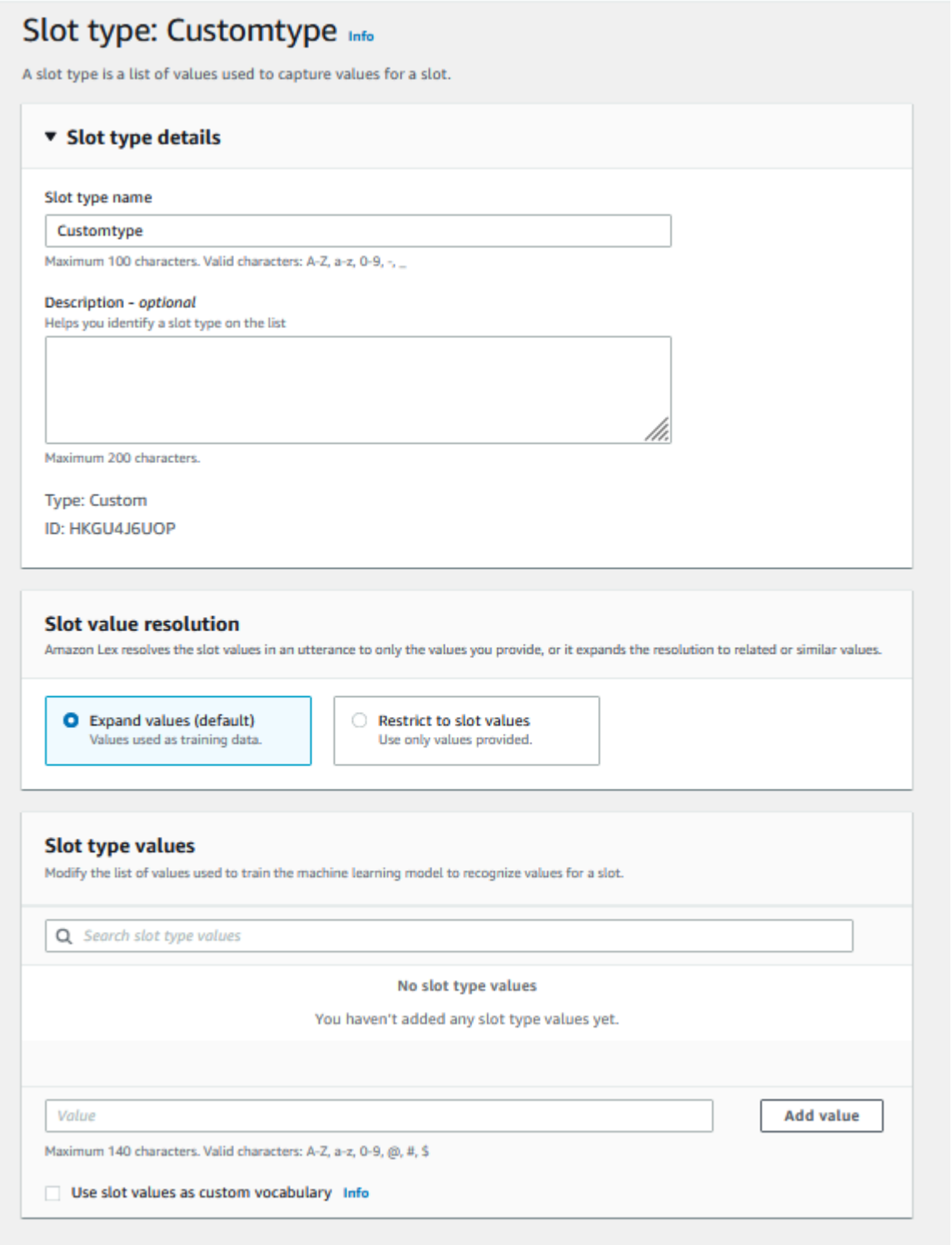
Utente: Ho ricevuto le risposte a tutte le mie domande e sono riuscito a completare la transazione.

Nota:

- `AMAZON.FreeFormInput`può essere utilizzato per acquisire l'input in formato libero così com'è dall'utente finale.
- `AMAZON.FreeFormInput`non può essere utilizzato in esempi di enunciati intenti.
- `AMAZON.FreeFormInput`non può avere enunciati di esempio a slot.
- `AMAZON.FreeFormInput`viene riconosciuto solo quando viene richiesto per.
- `AMAZON.FreeFormInput`non supporta wait and continue.
- `AMAZON.FreeFormInput`al momento non è supportato nel canale Amazon Connect Chat.
- Quando viene attivato uno `AMAZON.FreeFormInput` slot, `FallbackIntent` non viene attivato.
- Quando viene attivato uno `AMAZON.FreeFormInput` slot, non verrà attivato alcun interruttore di intento.

Tipo di slot personalizzato

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 V2 utilizza per addestrare il modello di apprendimento automatico a riconoscere i valori di uno slot. Ad esempio, puoi definire un tipo di slot chiamato `Genres` con valori come «commedia», «avventura», «documentario», ecc. È possibile definire sinonimi per il valore di un tipo di slot. Ad esempio, puoi definire i sinonimi "divertente" e "spiritoso" per il valore "commedia".



È possibile configurare il tipo di slot per espandere i valori dello slot. I valori dello slot verranno utilizzati come dati di addestramento e il modello risolverà lo slot in base al valore fornito dall'utente se è simile ai valori dello slot e ai sinonimi di tali valori. Questo è il comportamento che segue di default. Amazon Lex V2 mantiene un elenco di possibili risoluzioni per uno slot. Ogni voce dell'elenco

fornisce un valore risolto che Amazon Lex V2 ha riconosciuto come possibilità aggiuntive per lo slot. Un valore risolto è lo sforzo migliore per corrispondere al valore dello slot. L'elenco contiene fino a cinque valori.

In alternativa, è possibile configurare il tipo di slot per limitare la risoluzione ai valori dello slot. In questo caso, il modello risolverà un valore di slot inserito dall'utente in un valore di slot esistente solo se è uguale a quel valore di slot o è un sinonimo. Ad esempio, se l'utente immette "divertente" determinerà il valore di slot "commedia".

Quando il valore inserito dall'utente è sinonimo di un valore di tipo di slot, il modello restituisce quel valore del tipo di slot come prima voce nell'elenco di. `resolvedValues` Ad esempio, se l'utente inserisce «funny», il modello compila il `originalValue` campo con il valore «funny» e la prima voce nel campo ResolvedValues con «comedy». Puoi configurare il `valueSelectionStrategy` quando crei o aggiorni un tipo di slot con l'operazione CreateSlotType in modo che il valore di slot venga compilato con il primo valore dell'elenco di risoluzione.

I tipi di slot personalizzati supportano gli input che utilizzano stili ortografici. Puoi utilizzare gli spell-by-word stili spell-by-letter e per aiutare i tuoi clienti a inserire le lettere. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

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

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

Un tipo di slot personalizzato non deve avere lo stesso nome dei tipi di slot incorporati. Ad esempio, un tipo di slot personalizzato non deve essere denominato con le parole chiave riservate di Data, Numero o Conferma. Queste parole chiave sono riservate ai tipi di slot incorporati. Per un elenco di tutti i tipi di slot integrati, vedereTipi di slot integrati.

Tipo di slot grammaticale

Con il tipo di slot grammaticale, puoi creare la tua grammatica in formato XML secondo le specifiche SRGS per raccogliere informazioni in una conversazione. Amazon Lex V2 riconosce le espressioni che corrispondono alle regole specificate nella grammatica. È inoltre possibile fornire regole di interpretazione semantica utilizzando i tag ECMAScript all'interno dei file grammaticali. Amazon Lex restituisce quindi le proprietà impostate nei tag come valori risolti quando si verifica una corrispondenza.

Puoi creare tipi di slot grammaticali solo nelle lingue inglese (Australia), inglese (Regno Unito) e inglese (Stati Uniti).

Un tipo di slot grammaticale è composto da due parti. La prima è la grammatica stessa scritta utilizzando il formato delle specifiche SRGS. La grammatica interpreta l'enunciato dell'utente. Se l'enunciato è accettato dalla grammatica, viene abbinato, altrimenti viene rifiutato. Se un enunciato corrisponde, viene passato allo script, se ce n'è uno.

Il secondo, che fa parte di un tipo di slot grammaticale, è uno script opzionale scritto in ECMAScript che trasforma l'input nei valori risolti restituiti dal tipo di slot. Ad esempio, puoi usare uno script per convertire i numeri vocali in cifre. <tag>Le istruzioni ECMAScript sono racchiuse nell'elemento.

L'esempio seguente è in formato XML secondo la specifica SRGS che mostra una grammatica valida accettata da Amazon Lex V2. Definisce un tipo di slot grammaticale che accetta i numeri delle carte e determina se sono per account normali o premium. Per ulteriori informazioni sulla sintassi accettabile, vedere gli Formato dello script argomenti Definizione grammaticale e i relativi.

```
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar"
         xml:lang="en-US" tag-format="semantics/1.0" root="card_number">

   <rule id="card_number" scope="public">
        <item repeat="0-1">
```

```
            card number
        </item>
        <item>
            seven
            <tag>out.value = "7";</tag>
         </item>
         <item>
            <one-of>
               <item>
                  two four one
                  <tag> out.value = out.value + "241"; out.card_type = "premium"; </
tag>
               </item>
               <item>
                  zero zero one
                  <tag> out.value = out.value + "001"; out.card_type = "regular";</tag>
               </item>
            </one-of>
         </item>
   </rule>
</grammar>
```

La grammatica di cui sopra accetta solo due tipi di numeri di carta: 7241 o 7001. Entrambi possono essere opzionalmente preceduti dal prefisso «numero della carta». Contiene anche tag ECMAScript che possono essere utilizzati per l'interpretazione semantica. Con l'interpretazione semantica, l'espressione «carta numero sette due quattro uno» restituirebbe il seguente oggetto:

```
{
    "value": "7241",
    "card_type": "premium"
}
```

Questo oggetto viene restituito come stringa serializzata in JSON nell'`resolvedValues`oggetto restituito dalle operazioni RecognizeText, RecognizeUtterancee. StartConversation

Aggiungere un tipo di slot grammaticale

Per aggiungere uno slot grammaticale, digitare

1. Carica la definizione XML del tuo tipo di slot su un bucket S3. Prendi nota del nome del bucket e del percorso del file.

> **Note**
>
> La dimensione massima del file è 100 KB.

2. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
3. Dal menu a sinistra, scegli Bot, quindi scegli il bot a cui aggiungere il tipo di slot grammaticale.
4. Scegli Visualizza lingue, quindi scegli la lingua a cui aggiungere il tipo di slot grammaticale.
5. Scegli Visualizza i tipi di slot.
6. Scegli Aggiungi tipo di slot, quindi scegli Aggiungi tipo di slot grammaticale.
7. Assegna un nome al tipo di slot, quindi scegli Aggiungi.
8. Scegli il bucket S3 che contiene il tuo file di definizione e inserisci il percorso del file. Scegli Salva tipo di slot.

Definizione grammaticale

Questo argomento mostra le parti della specifica SRGS supportate da Amazon Lex V2. Tutte le regole sono definite nelle specifiche SRGS. Per ulteriori informazioni, vedere la versione 1.0 delle specifiche grammaticali del riconoscimento vocale, raccomandazione W3C.

Argomenti

- Dichiarazioni di intestazione
- Elementi XML supportati
- Gettoni
- Riferimento alla regola
- Sequenze e incapsulamento
- Ripetizioni
- Linguaggio
- Tag
- Pesi

Questo documento include materiale copiato e derivato dalla versione 1.0 della specifica grammaticale del riconoscimento vocale del W3C (disponibile all'indirizzo https://www.w3.org/TR/speech-grammar/). Di seguito sono riportate le informazioni sulla citazione:

Copyright © 2004 W3C® (MIT, ERCIM, Keio, Tutti i diritti riservati. Si applicano le regole del W3C sulla responsabilità, sui marchi, sull'uso dei documenti e sulle licenze del software.

Il documento sulle specifiche SRGS, una raccomandazione del W3C, è disponibile presso il W3C con la seguente licenza.

Testo della licenza

Licenza

Utilizzando e/o copiando questo documento, o il documento W3C da cui è collegata questa dichiarazione, l'utente (il licenziatario) accetta di aver letto, compreso e di rispettare i seguenti termini e condizioni:

Con la presente si concede il permesso di copiare e distribuire il contenuto di questo documento o del documento W3C da cui è collegata questa dichiarazione, su qualsiasi supporto per qualsiasi scopo e senza commissioni o royalty, a condizione che l'utente includa quanto segue su TUTTE le copie del documento, o parti di esso, che utilizza:

- Un link o un URL al documento W3C originale.
- L'avviso di copyright preesistente dell'autore originale o, se non esiste, un avviso (l'ipertesto è preferito, ma è consentita una rappresentazione testuale) del formato: «Copyright © [$date-of-document] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang). http://www.w3.org/Consortium/Legal/2015/doc-license»
- Se esiste, lo STATO del documento W3C.

Quando lo spazio lo consente, deve essere fornita l'inclusione del testo completo del presente AVVISO. Richiediamo che l'attribuzione della paternità sia fornita in qualsiasi software, documento o altro elemento o prodotto creato in base all'implementazione del contenuto di questo documento o di qualsiasi parte di esso.

Nessun diritto di creare modifiche o derivati dei documenti del W3C è concesso ai sensi di questa licenza, ad eccezione di quanto segue: Per facilitare l'implementazione delle specifiche tecniche stabilite in questo documento, chiunque può preparare e distribuire opere derivate e parti di

questo documento nel software, nei materiali di supporto che accompagnano il software e nella documentazione del software, A CONDIZIONE che tutte queste opere includano l'avviso seguente. TUTTAVIA, la pubblicazione di opere derivate da questo documento per l'uso come specifica tecnica è espressamente vietata.

Inoltre, i «Code Components» (Web IDL nelle sezioni chiaramente contrassegnati come Web IDL), il markup definito dal W3C (HTML, CSS, ecc.) e il codice del linguaggio di programmazione per computer chiaramente contrassegnati come esempi di codice, sono concessi in licenza ai sensi della W3C Software License.

L'avviso è:

«Diritti d'autore © 2015 W3C® (MIT, ERCIM, Keio, Beihang). Questo software o documento include materiale copiato o derivato da [titolo e URI del documento W3C].»

Avvertenze

QUESTO DOCUMENTO È FORNITO «COSÌ COM'È» E I TITOLARI DEI DIRITTI D'AUTORE NON RILASCIANO ALCUNA DICHIARAZIONE O GARANZIA, ESPRESSA O IMPLICITA, INCLUSE, A TITOLO ESEMPLIFICATIVO, GARANZIE DI COMMERCIABILITÀ, IDONEITÀ PER UNO SCOPO PARTICOLARE, NON VIOLAZIONE O TITOLO; CHE IL CONTENUTO DEL DOCUMENTO SIA ADATTO A QUALSIASI SCOPO; NÉ CHE L'IMPLEMENTAZIONE DI TALI CONTENUTI NON VIOLERÀ BREVETTI, DIRITTI D'AUTORE, MARCHI O ALTRI DIRITTI DI TERZI.

I DETENTORI DEI DIRITTI D'AUTORE NON SARANNO RESPONSABILI PER EVENTUALI DANNI DIRETTI, INDIRETTI, SPECIALI O CONSEQUENZIALI DERIVANTI DALL'USO DEL DOCUMENTO O DALL'ESECUZIONE O DALL'IMPLEMENTAZIONE DEI SUOI CONTENUTI.

Il nome e i marchi dei titolari del copyright NON possono essere utilizzati nella pubblicità o nella pubblicità relativa a questo documento o al suo contenuto senza una specifica autorizzazione scritta. Il diritto d'autore contenuto in questo documento rimarrà sempre di proprietà dei detentori dei diritti d'autore.

Dichiarazioni di intestazione

La tabella seguente mostra le dichiarazioni di intestazione supportate dal tipo di slot grammaticale. Per ulteriori informazioni, vedere Dichiarazioni di intestazione grammaticale nella raccomandazione W3C delle specifiche grammaticali di riconoscimento vocale versione 1.

Dichiarazione	Requisito di specificazione	Modulo XML	Supporto Amazon Lex	Specifica
Versione grammaticale	Obbligatorio	4.3: `version` attributo sull'elemento `grammar`	Obbligatorio	SERGI
Namespace XML	Obbligatorio (solo XML)	4.3: `xmlns` attributo sull'elemento `grammar`	Obbligatorio	SERGI
Tipo di documento	Obbligatorio (solo XML)	4.3: TIPO DI DOCUMENTO XML	Consigliato	SERGI
Codifica caratteri	Consigliato	4.4: `encoding` attributo nella dichiarazione XML	Consigliato	SERGI
Linguaggio	Richiesto in modalità vocale Ignorato in modalità DTMF	4.5: `xml:lang` attributo sull'elemento `grammar`	Richiesto in modalità vocale Ignorato in modalità DTMF	SERGI
Mode	Facoltativo	4.6: `mode` attributo sull'elemento `grammar`	Facoltativo	SERGI
Regola principale	Facoltativo	4.7: `root` attributo sull'elemento `grammar`	Campo obbligatorio	SERGI

Dichiarazione	Requisito di specificazione	Modulo XML	Supporto Amazon Lex	Specifica
Formato dei tag	Facoltativo	4.8: `tag-format` attributo sull'elemento `grammar`	String literal ed ECMAScript sono supportati	SRGS, SORELLA
URI di base	Facoltativo	4.9: `xml:base` attributo sull'elemento `grammar`	Facoltativo	SERGI
Lessico della pronuncia	Facoltativo, sono consentiti più di	4.10: elemento `lexicon`	Non supportato	SRGS, PER FAVORE
Metadati	Facoltativo, sono consentiti più di	4.11.1: elemento `meta`	Obbligatorio	SERGI
Metadati XML	Facoltativo, solo XML	4.11.2: elemento `metadata`	Facoltativo	SERGI
Tag	Facoltativo, sono consentiti più di	4.12: elemento `tag`	Tag globali non supportati	SERGI

Esempio

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
                 "http://www.w3.org/TR/speech-grammar/grammar.dtd">

<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xml:base="http://www.example.com/base-file-path"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US"
         version="1.0"
```

```
        mode="voice"
        root="city"
        tag-format="semantics/1.0">
```

Elementi XML supportati

Amazon Lex V2 supporta i seguenti elementi XML per grammatiche personalizzate:

- `<item>`
- `<token>`
- `<tag>`
- `<one-of>`
- `<rule-ref>`

Gettoni

La tabella seguente mostra le specifiche dei token supportate dal tipo di slot grammaticale. Per ulteriori informazioni, consulta Tokens in the Speech Recognition Grammar Specification version 1 Raccomandation W3C.

Tipo di token	Esempio	Supportata?
Token singolo non quotato	ciao	Sì
Token singolo non quotato: non alfabetico	2	Sì
Token con virgolette singole, senza spazi bianchi	"hello"	Sì, inserisci le virgolette doppie quando contiene un solo token
Due gettoni delimitati da spazi bianchi	buon viaggio	Sì
Quattro gettoni delimitati da spazi bianchi	questo è un test	Sì

Tipo di token	Esempio	Supportata?
Token tra virgolette singole, spazio bianco incluso	«San Francisco	No
Singolo token XML nel <token>tag	<token>San Francisco</token>	No (uguale al token tra virgolette singole con spazio bianco)

Note

- Token tra virgolette singole incluso spazio bianco: la specifica richiede che le parole racchiuse tra virgolette doppie vengano trattate come un singolo token. Amazon Lex V2 li tratta come token delimitati da spazi vuoti.
- Token XML singolo in<token>: la specifica richiede che le parole delimitate <token>da rappresentino un token. Amazon Lex V2 li tratta come token delimitati da spazi vuoti.
- Amazon Lex V2 genera un errore di convalida quando uno degli usi viene rilevato nella grammatica.

Esempio

```
<rule id="state" scope="public">
    <one-of>
        <item>FL</item>
        <item>MA</item>
        <item>NY</item>
    </one-of>
</rule>
```

Riferimento alla regola

La tabella seguente riassume le varie forme di riferimento alle regole possibili all'interno dei documenti grammaticali. Per ulteriori informazioni, vedere Riferimento alla regola nella raccomandazione W3C delle specifiche grammaticali di riconoscimento vocale versione 1.

Tipo di riferimento	Modulo XML	Supportato
2.2.1 Riferimento esplicito alle regole locali	`<ruleref uri="#rulename"/>`	Sì
2.2.2 Riferimento esplicito a una regola denominata di una grammatica identificata da un URI	`<ruleref uri="grammarURI#rulename"/>`	No
2.2.2 Riferimento implicito alla regola fondamentale di una grammatica identificata da un URI	`<ruleref uri="grammarURI"/>`	No
2.2.2 Riferimento esplicito a una regola grammaticale denominata identificata da un URI con un tipo di supporto	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	No
2.2.2 Riferimento implicito alla regola fondamentale di una grammatica identificata da un URI con un tipo di supporto	`<ruleref uri="grammarURI" type="media-type"/>`	No

2.2.3 Definizioni di regole speciali	`<ruleref special="NULL"/>`
	`<ruleref special="VOID"/>`
	`<ruleref special="GARBAGE"/>`
		No

Note

1. L'URI grammaticale è un URI esterno. Ad esempio, `http://grammar.example.com/world-cities.grxml`.

2. Il tipo di supporto può essere:
   - `application/srgs+xml`
   - `text/plain`

Esempio

```
<rule id="city" scope="public">
    <one-of>
        <item>Boston</item>
        <item>Philadelphia</item>
        <item>Fargo</item>
    </one-of>
</rule>

<rule id="state" scope="public">
    <one-of>
        <item>FL</item>
        <item>MA</item>
        <item>NY</item>
    </one-of>
</rule>

<!-- "Boston MA" -> city = Boston, state = MA -->
<rule id="city_state" scope="public">
    <ruleref uri="#city"/> <ruleref uri="#state"/>
</rule>
```

Sequenze e incapsulamento

L'esempio seguente mostra le sequenze supportate. Per ulteriori informazioni, vedere Sequenze e incapsulamento nella raccomandazione W3C delle specifiche grammaticali di riconoscimento vocale versione 1.

Esempio

```
<!-- sequence of tokens -->
this is a test

<!--sequence of rule references-->
<ruleref uri="#action"/> <ruleref uri="#object"/>
```

```
<!--sequence of tokens and rule references-->
the <ruleref uri="#object"/> is <ruleref uri="#color"/>

<!-- sequence container -->
<item>fly to <ruleref uri="#city"/> </item>
```

Ripetizioni

La tabella seguente mostra le espansioni ripetute supportate per le regole. Per ulteriori informazioni, vedere Repeats in the Speech Recognition Grammar Specification versione 1 Raccomandazione W3C.

Modulo XML Esempio	Comportamento	Supportata?
repeat = «n» repeat="6"	L'espressione contenuta viene ripetuta esattamente «n» volte. «n» deve essere «0" o un numero intero positivo.	Sì
repeat = «m-n» repeat="4-6"	L'espansione contenuta viene ripetuta tra «m» e «n» volte (incluse). «m» e «n» devono essere entrambi «0" o un numero intero positivo e «m» deve essere minore o uguale a «n».	Sì
repeat = «m-» repeat = «3-»	L'espansione contenuta viene ripetuta «m» volte o più (inclusa). «m» deve essere «0" o un numero intero positivo. Ad esempio, «3-» dichiara che l'espansione contenuta può verificarsi tre, quattro, cinque o più volte.	Sì

Modulo XML Esempio	Comportamento	Supportata?
repeat="0-1"	L'espansione contenuta è opzionale.	Sì
<item repeat="2-4" repeat-prob="0.8">		No

Linguaggio

La seguente discussione si applica agli identificatori linguistici applicati alle grammatiche. Per ulteriori informazioni, vedere la versione 1 della specifica grammaticale di riconoscimento vocale Language in the Speech Recognition Raccomandazione W3C.

Per impostazione predefinita, una grammatica è un documento in una sola lingua con un identificatore di lingua fornito nella dichiarazione della lingua nell'intestazione grammaticale. Tutti i token all'interno di quella grammatica, salvo diversa indicazione, verranno gestiti in base al linguaggio della grammatica. Le dichiarazioni linguistiche a livello grammaticale non sono supportate.

Nel seguente esempio:

1. La dichiarazione dell'intestazione grammaticale per la lingua «en-US» è supportata da Amazon Lex V2.
2. Gli allegati linguistici a livello di articolo (evidenziati in ***rosso***) non sono supportati. Amazon Lex V2 genera un errore di convalida se un allegato in lingua è diverso dalla dichiarazione di intestazione.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
                  "http://www.w3.org/TR/speech-grammar/grammar.dtd">

<!-- the default grammar language is US English -->
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
```

```
        xml:lang="en-US" version="1.0">

  <!--
     single language attachment to tokens
     "yes" inherits US English language
     "oui" is Canadian French language
  -->
  <rule id="yes">
    <one-of>
      <item>yes</item>
      <item xml:lang="fr-CA">oui</item>
    </one-of>
  </rule>

  <!-- Single language attachment to an expansion -->
  <rule id="people1">
    <one-of xml:lang="fr-CA">
      <item>Michel Tremblay</item>
      <item>André Roy</item>
    </one-of>
  </rule>
</grammar>
```

Tag

La seguente discussione si applica ai tag definiti per le grammatiche. Per ulteriori informazioni, vedere Tags in the Speech Recognition Grammar Specification versione 1 Raccomandazione W3C.

In base alle specifiche SRGS, i tag possono essere definiti nei seguenti modi:

1. Come parte di una dichiarazione di intestazione come descritta inDichiarazioni di intestazione.
2. Come parte di una <rule>definizione.

Sono supportati i seguenti formati di tag:

- `semantics/1.0`(SISR, ECMAScript)
- `semantics/1.0-literals`(stringhe letterali SISR)

I seguenti formati di tag non sono supportati:

- `swi-semantics/1.0`(proprietà di Nuance)

Esempio

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xml:base="http://www.example.com/base-file-path"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US"
         version="1.0"
         mode="voice"
         root="city"
         tag-format="semantics/1.0-literals">
    <rule id="no">
        <one-of>
            <item>no</item>
            <item>nope</item>
            <item>no way</item>
        </one-of>
        <tag>no</tag>
    </rule>
</grammar>
```

Pesi

È possibile aggiungere l'attributo weight a un elemento. Il peso è un valore positivo in virgola mobile che rappresenta il grado in cui la frase nell'articolo viene aumentata durante il riconoscimento vocale. Per ulteriori informazioni, vedere Weights in the Speech Recognition Grammar Specification version 1 Raccomandazione W3C.

I pesi devono essere maggiori di 0 e inferiori o uguali a 10 e possono avere una sola cifra decimale. Se il peso è maggiore di 0 e inferiore a 1, la frase viene potenziata negativamente. Se il peso è maggiore di 1 e inferiore o uguale a 10, la frase viene potenziata positivamente. Un peso di 1 equivale a non dare alcun peso e non c'è alcun potenziamento per la frase.

Assegnare pesi adeguati agli articoli per migliorare le prestazioni di riconoscimento vocale è un compito difficile. Ecco alcuni suggerimenti che puoi seguire per l'assegnazione dei pesi:

- Inizia con una grammatica senza assegnare pesi agli elementi.
- Determina quali schemi del discorso vengono spesso identificati erroneamente.
- Applica valori diversi per i pesi fino a quando non noterai un miglioramento delle prestazioni di riconoscimento vocale e non si verificheranno regressioni.

Esempio 1

Ad esempio, se hai una grammatica per gli aeroporti e osservi che New York viene spesso erroneamente identificata come Newark, puoi dare un impulso positivo a New York assegnandole un peso di 5.

```
<rule> id="airport">
    <one-of>
        <item>
            Boston
            <tag>out="Boston"</tag>
        </item>
        <item weight="5">
            New York
            <tag>out="New York"</tag>
        </item>
        <item>
            Newark
            <tag>out="Newark"</tag>
        </item>
    </one-of>
</rule>
```

Esempio 2

Ad esempio, hai una grammatica per il codice di prenotazione della compagnia aerea che inizia con un alfabeto inglese seguito da tre cifre. Il codice di prenotazione probabilmente inizia con B o D, ma osservi che B viene spesso erroneamente identificato come P e D come T. Puoi potenziare positivamente B e D.

```
<rule> id="alphabet">
    <one-of>
        <item>A<tag>out.letters+='A';</tag></item>
        <item weight="3.5">B<tag>out.letters+='B';</tag></item>
        <item>C<tag>out.letters+='C';</tag></item>
        <item weight="2.9">D<tag>out.letters+='D';</tag></item>
        <item>E<tag>out.letters+='E';</tag></item>
        <item>F<tag>out.letters+='F';</tag></item>
        <item>G<tag>out.letters+='G';</tag></item>
        <item>H<tag>out.letters+='H';</tag></item>
        <item>I<tag>out.letters+='I';</tag></item>
```

```
        <item>J<tag>out.letters+='J';</tag></item>
        <item>K<tag>out.letters+='K';</tag></item>
        <item>L<tag>out.letters+='L';</tag></item>
        <item>M<tag>out.letters+='M';</tag></item>
        <item>N<tag>out.letters+='N';</tag></item>
        <item>O<tag>out.letters+='O';</tag></item>
        <item>P<tag>out.letters+='P';</tag></item>
        <item>Q<tag>out.letters+='Q';</tag></item>
        <item>R<tag>out.letters+='R';</tag></item>
        <item>S<tag>out.letters+='S';</tag></item>
        <item>T<tag>out.letters+='T';</tag></item>
        <item>U<tag>out.letters+='U';</tag></item>
        <item>V<tag>out.letters+='V';</tag></item>
        <item>W<tag>out.letters+='W';</tag></item>
        <item>X<tag>out.letters+='X';</tag></item>
        <item>Y<tag>out.letters+='Y';</tag></item>
        <item>Z<tag>out.letters+='Z';</tag></item>
    </one-of>
</rule>
```

Formato dello script

Amazon Lex V2 supporta le seguenti funzionalità ECMAScript per la definizione delle grammatiche.

Amazon Lex V2 supporta le seguenti funzionalità ECMAScript quando si specificano i tag nella grammatica. `tag-format`deve essere inviato `semantics/1.0` quando i tag ECMAScript vengono utilizzati nella grammatica. Per ulteriori informazioni, vedere la specifica del linguaggio ECMA-262 ECMAScript 2021.

```
<grammar version="1.0"
xmlns="http://www.w3.org/2001/06/grammar"
xml:lang="en-US"
tag-format="semantics/1.0"
root="card_number">
```

Argomenti

- Dichiarazione variabile
- Espressioni
- Se dichiarazione
- Dichiarazione Switch

- Dichiarazioni di funzione
- Dichiarazione di iterazione
- Dichiarazione di blocco
- Commenti
- Dichiarazioni non supportate

Questo documento contiene materiale tratto dallo standard ECMAScript (disponibile all'indirizzo https://www.ecma-international.org/ publications-and-standards /standards/ecma-262/). Il documento con le specifiche del linguaggio ECMAScript è disponibile presso Ecma International con la seguente licenza.

Testo della licenza

© 2020 Ecma International

Questo documento può essere copiato, pubblicato e distribuito ad altri e alcune opere derivate possono essere preparate, copiate, pubblicate e distribuite, in tutto o in parte, a condizione che l'avviso di copyright di cui sopra e la presente licenza e disclaimer sul copyright siano inclusi su tutte queste copie e opere derivate. Le uniche opere derivate consentite dalla presente Licenza di Copyright e dal Disclaimer sono:

(i) opere che incorporano tutto o parte di questo documento allo scopo di fornire commenti o spiegazioni (come una versione annotata del documento),

(ii) opere che incorporano tutto o parte di questo documento allo scopo di incorporare funzionalità che forniscono l'accessibilità,

(iii) traduzioni di questo documento in lingue diverse dall'inglese e in diversi formati e

(iv) funziona facendo uso di queste specifiche in prodotti conformi agli standard implementando (ad esempio copiando e incollando in tutto o in parte) le funzionalità ivi contenute.

Tuttavia, il contenuto di questo documento stesso non può essere modificato in alcun modo, inclusa la rimozione dell'avviso di copyright o dei riferimenti a Ecma International, ad eccezione di quanto richiesto per tradurlo in lingue diverse dall'inglese o in un formato diverso.

La versione ufficiale di un documento di Ecma International è la versione in lingua inglese sul sito web di Ecma International. In caso di discrepanze tra una versione tradotta e la versione ufficiale, prevarrà la versione ufficiale.

Le autorizzazioni limitate concesse sopra sono perpetue e non saranno revocate da Ecma International o dai suoi successori o assegnatari. Questo documento e le informazioni in esso contenute sono forniti «COSÌ COME SONO» ed ECMA INTERNATIONAL DECLINA TUTTE LE GARANZIE, ESPLICITE O IMPLICITE, INCLUSA MA NON LIMITATA A QUALSIASI GARANZIA CHE L'USO DELLE INFORMAZIONI QUI CONTENUTE NON VIOLERÀ ALCUN DIRITTO DI PROPRIETÀ O GARANZIE IMPLICITE DI COMMERCIABILITÀ O IDONEITÀ PER UNO SCOPO PARTICOLARE.»

Dichiarazione variabile

Un'istruzione variabile definisce una o più variabili.

```
var x = 10;
var x = 10, var y = <expression>;
```

Espressioni

Tipo di espressione	Sintassi	Esempio	Supportata?
Espressione regolare letterale	Stringa letterale contenente caratteri speciali regex validi	`"^\d\.$"`	No
Funzione	`function functionName(parameters) { functionBody}`	`var x = function calc() { return 10; }`	No
Delete	`delete expression`	`delete obj.property;`	No
Void	`void expression`	`void (2 == '2');`	No
Tipo di	`typeof expression`	`typeof 42;`	No

Tipo di espressione	Sintassi	Esempio	Supportata?
Indice dei membri	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	Sì
Membro dot	`expression . identifier`	`out.value`	sì
Argomenti	`expression (arguments)`	`new Date('1994-10-11')`	Sì
Incremento dei post	`expression++`	`var x=10; x++;`	Sì
Dopo il decremento	`expression--`	`var x=10; x--;`	Sì
Pre-incremento	`++expression`	`var x=10; ++x;`	Sì
Pre-decremento	`--expression`	`var x=10; --x;`	Sì
Unario più/Unario meno	`+expression / -expression`	`+x / -x;`	Sì
Ma no	`~ expression`	`const a = 5; consol e.log( ~a );`	Sì
Logico no	`! expression`	`!(a > 0 || b > 0)`	Sì
Moltiplicativo	`expression ('*' | '/' | '%') expression`	`(x + y) * (a / b)`	Sì

Tipo di espressione	Sintassi	Esempio	Supportata?
Additivo	`expression ('+' | '-') expression`	`(a + b) - (a - (a + b))`	Sì
Spostamento dei bit	`expression ('<<' | '>>' | '>>>') expression`	`(a >> b) >>> c`	Sì
Parente	`expression ('<' | '>' | '<=' | '>=') expression`	`if (a > b) { ... }`	Sì
In	`expression in expression`	`fruits[0] in otherFruits;`	Sì
Parità	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	Sì
Bit e/xor/or	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	Sì
Logico e/o	`expression ('&&' | '||') expression`	`if (a && (b ||c)) { ...}`	Sì
Ternaria	`expression ? expression : expression`	`a > b ? obj.prop : 0`	Sì

Tipo di espressione	Sintassi	Esempio	Supportata?
Assegnazione	`expression = expression`	`out.value = "string";`	Sì
Operatore di assegnazione	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	Sì
Operatore di assegnazione bit per bit	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	Sì
Identificatore	`identifierSequence` dove IdentifierSequence è una sequenza di caratteri validi	`fruits=[10, 20, 30];`	Sì
Letterale nullo	`null`	`x = null;`	Sì
Letterale booleano	`true | false`	`x = true;`	Sì
Stringa letterale	`'string' / "string"`	`a = 'hello', b = "world";`	Sì
Letterale decimale	`integer [.] digits [exponent]`	`111.11 e+12`	Sì
Letterale esadecimale	`0 (x | X)[0-9a-fA-F]`	`0x123ABC`	Sì

Tipo di espressione	Sintassi	Esempio	Supportata?
Letterale ottale	`0 [0-7]`	`"051"`	Sì
Array letterale	`[ expressio n, ... ]`	`v = [a, b, c];`	Sì
Oggetto letterale	`{property: value, ...}`	`out = {value: 1, flag: false};`	Sì
Tra parentesi	`( expressions )`	`x + (x + y)`	Sì

Se dichiarazione

```
if (expressions) {
    statements;
} else {
    statements;
}
```

Nota: nell'esempio precedente, `expressions` `statements` deve essere uno di quelli supportati da questo documento.

Dichiarazione Switch

```
switch (expression) {
    case (expression):
     statements
     .
     .
     .
   default:
      statements
}
```

Nota: nell'esempio precedente, `expressions` `statements` deve essere uno di quelli supportati da questo documento.

Dichiarazioni di funzione

```
function functionIdentifier([parameterList, ...]) {
     <function body>
}
```

Dichiarazione di iterazione

Le istruzioni di iterazione possono essere una delle seguenti:

```
// Do..While statement
do {
   statements
} while (expressions)


// While Loop
while (expressions) {
   statements
}

// For Loop
for ([initialization]; [condition]; [final-expression])
   statement

// For..In
for (variable in object) {
  statement
}
```

Dichiarazione di blocco

```
{
   statements
}

// Example
{
    x = 10;
   if (x > 10) {
     console.log("greater than 10");
   }
```

```
}
```

Nota: nell'esempio precedente, `statements` fornito nel blocco deve essere uno di quelli supportati da questo documento.

Commenti

```
// Single Line Comments
"// <comment>"

// Multineline comments
/**
<comment>
**/
```

Dichiarazioni non supportate

Amazon Lex V2 non supporta le seguenti funzionalità ECMAScript.

Argomenti

- Dichiarazione vuota
- Continua la dichiarazione
- Dichiarazione di interruzione
- Dichiarazione di restituzione
- Dichiarazione Throw
- Prova la dichiarazione
- Dichiarazione del debugger
- Dichiarazione etichettata
- Dichiarazione di classe

Dichiarazione vuota

L'istruzione vuota viene utilizzata per non fornire alcuna dichiarazione. Di seguito è riportata la sintassi per un'istruzione vuota:

```
;
```

Continua la dichiarazione

L'istruzione continua senza etichetta è supportata daDichiarazione di iterazione. L'istruzione continua con etichetta non è supportata.

```
// continue with label
// this allows the program to jump to a
// labelled statement (see labelled statement below)
continue <label>;
```

Dichiarazione di interruzione

La dichiarazione di interruzione senza etichetta è supportata daDichiarazione di iterazione. La dichiarazione di interruzione con etichetta non è supportata.

```
// break with label
// this allows the program to break out of a
// labelled statement (see labelled statement below)
break <label>;
```

Dichiarazione di restituzione

```
return expression;
```

Dichiarazione Throw

L'istruzione throw viene utilizzata per generare un'eccezione definita dall'utente.

```
throw expression;
```

Prova la dichiarazione

```
try {
  statements
}
catch (expression) {
  statements
}
finally {
  statements
}
```

Dichiarazione del debugger

L'istruzione debugger viene utilizzata per richiamare la funzionalità di debug fornita dall'ambiente.

```
debugger;
```

Dichiarazione etichettata

L'istruzione etichettata può essere utilizzata con `continue` le dichiarazioni `break` or.

```
label:
   statements


// Example
let str = '';

loop1:
for (let i = 0; i < 5; i++) {
  if (i === 1) {
    continue loop1;
  }
  str = str + i;
}

console.log(str);
```

Dichiarazione di classe

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
```

Grammatiche del settore

Le grammatiche di settore sono un insieme di file XML da utilizzare con il tipo di slot grammaticale. Puoi utilizzarli per offrire rapidamente un'esperienza coerente all'utente finale durante la migrazione dei flussi di lavoro di risposta vocale interattiva ad Amazon Lex V2. Puoi scegliere tra una gamma di grammatiche predefinite in tre domini: servizi finanziari, assicurazioni e telecomunicazioni.

Esiste anche un set generico di grammatiche che puoi usare come punto di partenza per le tue grammatiche.

Le grammatiche contengono le regole per raccogliere le informazioni e i tag ECMAScript per l'interpretazione semantica.

Grammatiche per i servizi finanziari (download)

Per i servizi finanziari sono supportate le seguenti grammatiche: numeri di conto e di routing, numeri di carta di credito e prestito, punteggio di credito, date di apertura e chiusura del conto e numero di previdenza sociale.

Numero conto

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My account number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My account number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My account number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
```

```
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">account number is</item>
                <item repeat="0-1">Account Number</item>
                <item repeat="0-1">Here is my Account Number </item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My account Id is</item>
                <item repeat="0-1">This is the account Id</item>
                <item repeat="0-1">account Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
```

```
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numero di routing

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My routing number is 1 2 3 4 5 6 7 8 9
                 Output: 123456789

             Scenario 2:
                 Input: routing number 1 2 3 4 5 6 7 8 9
                 Output: 123456789

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
```

```
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
       <one-of>
          <item repeat="0-1">My routing number</item>
          <item repeat="0-1">Routing number of</item>
          <item repeat="0-1">The routing number is</item>
       </one-of>
    </rule>

    <rule id="hesitation">
        <one-of>
           <item>Hmm</item>
           <item>Mmm</item>
           <item>My</item>
        </one-of>
      </rule>

    <rule id="singleDigit">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.digit=""</tag>
        <item repeat="16">
            <one-of>
                <item>0<tag>out.digit+=0;</tag></item>
                <item>zero<tag>out.digit+=0;</tag></item>
                <item>1<tag>out.digit+=1;</tag></item>
                <item>one<tag>out.digit+=1;</tag></item>
                <item>2<tag>out.digit+=2;</tag></item>
                <item>two<tag>out.digit+=2;</tag></item>
                <item>3<tag>out.digit+=3;</tag></item>
                <item>three<tag>out.digit+=3;</tag></item>
                <item>4<tag>out.digit+=4;</tag></item>
                <item>four<tag>out.digit+=4;</tag></item>
                <item>5<tag>out.digit+=5;</tag></item>
                <item>five<tag>out.digit+=5;</tag></item>
                <item>6<tag>out.digit+=6;</tag></item>
                <item>six<tag>out.digit+=5;</tag></item>
                <item>7<tag>out.digit+=7;</tag></item>
                <item>seven<tag>out.digit+=7;</tag></item>
                <item>8<tag>out.digit+=8;</tag></item>
                <item>eight<tag>out.digit+=8;</tag></item>
                <item>9<tag>out.digit+=9;</tag></item>
                <item>nine<tag>out.digit+=9;</tag></item>
            </one-of>
        </item>
    </rule>
```

```
</grammar>
```

Numero di carta di credito

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My credit card number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

             Scenario 2:
                 Input: card number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My credit card number is</item>
              <item repeat="0-1">card number</item>
           </one-of>
        </rule>

        <rule id="hesitation">
```

```
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="16">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

ID del prestito

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                            http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My loan Id is A B C 1 2 3 4
                Output: ABC1234
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my loan number is</item>
                <item repeat="0-1">The loan number</item>
                <item repeat="0-1">The loan is </item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">loan number</item>
                <item repeat="0-1">loan number of</item>
                <item repeat="0-1">loan Id is</item>
                <item repeat="0-1">My loan Id is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
```

```
            <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
```

```
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Punteggio di credito

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
```

```
                Input: The number is fifteen
                Output: 15

             Scenario 2:
                Input: My credit score is fifteen
                Output: 15
         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
        </rule>

        <rule id="text">
           <one-of>
              <item repeat="0-1">Credit score is</item>
              <item repeat="0-1">Last digits are</item>
              <item repeat="0-1">The number is</item>
              <item repeat="0-1">That's</item>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">My credit score is</item>
           </one-of>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
```

```
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
            <item>forty<tag>out=40;</tag></item>
```

```
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Data di apertura del conto

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I opened account on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: I need account number opened on July Two Thousand and Eleven
                 Output: 07/11
```

```
         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I opened account on </item>
              <item repeat="0-1">I need account number opened on </item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
```

```
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
```

```
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<!--<tag>out=2000;</tag>--></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data di pagamento automatica

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">
```

```
         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to schedule auto pay for twenty five Dollar
                 Output: $25

             Scenario 2:
                 Input: Setup automatic payments for twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to schedule auto pay for</item>
              <item repeat="0-1">Setup automatic payments for twenty five dollars</
item>
              <item repeat="0-1">Auto pay amount of</item>
              <item repeat="0-1">Set it up for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="digits">
```

```
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.num = 0;</tag>
        <one-of>
            <item>0<tag>out.num+=0;</tag></item>
            <item>1<tag>out.num+=1;</tag></item>
            <item>2<tag>out.num+=2;</tag></item>
            <item>3<tag>out.num+=3;</tag></item>
            <item>4<tag>out.num+=4;</tag></item>
            <item>5<tag>out.num+=5;</tag></item>
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
            <item>eighteen<tag>out.teen+=18;</tag></item>
            <item>nineteen<tag>out.teen+=19;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>
```

```
        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>
```

```
        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Data di scadenza della carta di credito

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out="";</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
```

```
        Output: 05 2026

-->

<rule id="text">
   <item repeat="0-1"><ruleref uri="#hesitation"/></item>
   <one-of>
      <item repeat="0-1">My card expiration date is </item>
      <item repeat="0-1">Expiration date is </item>
   </one-of>
</rule>

<rule id="hesitation">
  <one-of>
     <item>Hmm</item>
     <item>Mmm</item>
     <item>My</item>
  </one-of>
</rule>

<rule id="months">
   <item repeat="0-1"><ruleref uri="#text"/></item>
   <one-of>
       <item>january<tag>out="01";</tag></item>
       <item>february<tag>out="02";</tag></item>
       <item>march<tag>out="03";</tag></item>
       <item>april<tag>out="04";</tag></item>
       <item>may<tag>out="05";</tag></item>
       <item>june<tag>out="06";</tag></item>
       <item>july<tag>out="07";</tag></item>
       <item>august<tag>out="08";</tag></item>
       <item>september<tag>out="09";</tag></item>
       <item>october<tag>out="10";</tag></item>
       <item>november<tag>out="11";</tag></item>
       <item>december<tag>out="12";</tag></item>
       <item>jan<tag>out="01";</tag></item>
       <item>feb<tag>out="02";</tag></item>
       <item>aug<tag>out="08";</tag></item>
       <item>sept<tag>out="09";</tag></item>
       <item>oct<tag>out="10";</tag></item>
       <item>nov<tag>out="11";</tag></item>
       <item>dec<tag>out="12";</tag></item>
       <item>1<tag>out="01";</tag></item>
       <item>2<tag>out="02";</tag></item>
```

```
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="year">
      <tag>out.yr="20"</tag>
      <one-of>
          <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
          <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
```

```
      </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
            <item>forty<tag>out=40;</tag></item>
            <item>fifty<tag>out=50;</tag></item>
            <item>sixty<tag>out=60;</tag></item>
            <item>seventy<tag>out=70;</tag></item>
            <item>eighty<tag>out=80;</tag></item>
            <item>ninety<tag>out=90;</tag></item>
            <item>20<tag>out=20;</tag></item>
            <item>30<tag>out=30;</tag></item>
            <item>40<tag>out=40;</tag></item>
            <item>50<tag>out=50;</tag></item>
```

```
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data della dichiarazione

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: Show me statements from July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: Show me statements from July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: Show me statements from July Thirty Two Thousand and Eleven
                 Output: 07/30/11
         -->

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item>
```

```
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                  <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                  <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                  <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
             </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to see bank statements from </item>
              <item repeat="0-1">Show me statements from</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
         </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
        <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
             <item>january<tag>out.mon+="01";</tag></item>
```

```
            <item>february<tag>out.mon+="02";</tag></item>
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
```

```
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

Data della transazione

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
             Scenario 1:
                 Input: My last incorrect transaction date is july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My last incorrect transaction date is july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My last incorrect transaction date is</item>
              <item repeat="0-1">It is</item>
           </one-of>
        </rule>
        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
```

```
            <item>january<tag>out.mon+="01";</tag></item>
            <item>february<tag>out.mon+="02";</tag></item>
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
```

```
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
```

```
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Importo del trasferimento

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to transfer twenty five Dollar
                 Output: $25

             Scenario 2:
                 Input: transfer twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
```

```
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
       <one-of>
          <item repeat="0-1">I want to transfer</item>
          <item repeat="0-1">transfer</item>
          <item repeat="0-1">make a transfer for</item>
       </one-of>
    </rule>

    <rule id="hesitation">
        <one-of>
           <item>Hmm</item>
           <item>Mmm</item>
           <item>My</item>
        </one-of>
      </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.num = 0;</tag>
        <one-of>
            <item>0<tag>out.num+=0;</tag></item>
            <item>1<tag>out.num+=1;</tag></item>
            <item>2<tag>out.num+=2;</tag></item>
            <item>3<tag>out.num+=3;</tag></item>
            <item>4<tag>out.num+=4;</tag></item>
            <item>5<tag>out.num+=5;</tag></item>
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
```

```
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>
```

```
        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Numero di previdenza sociale

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
```

```
            <tag>out=""</tag>
            <ruleref uri="#digits"/><tag>out += rules.digits.numbers;</tag>
        </rule>

        <rule id="digits">
            <tag>out.numbers=""</tag>
            <item repeat="1-12">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                    <item>zero<tag>out.numbers+=0;</tag></item>
                    <item>one<tag>out.numbers+=1;</tag></item>
                    <item>two<tag>out.numbers+=2;</tag></item>
                    <item>three<tag>out.numbers+=3;</tag></item>
                    <item>four<tag>out.numbers+=4;</tag></item>
                    <item>five<tag>out.numbers+=5;</tag></item>
                    <item>six<tag>out.numbers+=6;</tag></item>
                    <item>seven<tag>out.numbers+=7;</tag></item>
                    <item>eight<tag>out.numbers+=8;</tag></item>
                    <item>nine<tag>out.numbers+=9;</tag></item>
                    <item>dash</item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Grammatiche assicurative (download)

Per il settore assicurativo sono supportate le seguenti grammatiche: numeri di sinistro e polizza, numeri di patente di guida e di targa, date di scadenza, date di inizio e date di rinnovo, importi dei reclami e delle polizze.

ID del reclamo

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My claim number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: Claim number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My claim number is</item>
                <item repeat="0-1">Claim number</item>
                <item repeat="0-1">This is for claim</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
```

```
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

ID Policy

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">
```

```
        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My policy number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: This is the policy number 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My policy number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy number is</item>
                <item repeat="0-1">This is the policy number</item>
                <item repeat="0-1">Policy number</item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My policy Id is</item>
                <item repeat="0-1">This is the policy Id</item>
                <item repeat="0-1">Policy Id</item>
            </one-of>
        </rule>
```

```
        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
```

```
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numero della patente di guida

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My drivers license number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: driver license number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My drivers license number is</item>
                <item repeat="0-1">My drivers license id is</item>
                <item repeat="0-1">Driver license number</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
```

```
        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numero di targa

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">
```

```
        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: my license plate is A B C D 1 2
                Output: ABCD12

            Scenario 2:
                Input: license plate number A B C 1 2 3 4
                Output: ABC1234

            Scenario 3:
                Input: my plates say A F G K 9 8 7 6 Thanks
                Output: AFGK9876
        -->

        <rule id="main" scope="public">
            <tag>out.licenseNum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.licenseNum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my license plate is</item>
                <item repeat="0-1">license plate number</item>
                <item repeat="0-1">my plates say</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
```

```
            <item>I think</item>
          </one-of>
        </rule>

      <rule id="alphabets">
          <item repeat="0-1"><ruleref uri="#text"/></item>
          <tag>out.letters=""</tag>
          <tag>out.firstOccurence=""</tag>
          <item repeat="3-4">
              <one-of>
                  <item>A<tag>out.letters+='A';</tag></item>
                  <item>B<tag>out.letters+='B';</tag></item>
                  <item>C<tag>out.letters+='C';</tag></item>
                  <item>D<tag>out.letters+='D';</tag></item>
                  <item>E<tag>out.letters+='E';</tag></item>
                  <item>F<tag>out.letters+='F';</tag></item>
                  <item>G<tag>out.letters+='G';</tag></item>
                  <item>H<tag>out.letters+='H';</tag></item>
                  <item>I<tag>out.letters+='I';</tag></item>
                  <item>J<tag>out.letters+='J';</tag></item>
                  <item>K<tag>out.letters+='K';</tag></item>
                  <item>L<tag>out.letters+='L';</tag></item>
                  <item>M<tag>out.letters+='M';</tag></item>
                  <item>N<tag>out.letters+='N';</tag></item>
                  <item>O<tag>out.letters+='O';</tag></item>
                  <item>P<tag>out.letters+='P';</tag></item>
                  <item>Q<tag>out.letters+='Q';</tag></item>
                  <item>R<tag>out.letters+='R';</tag></item>
                  <item>S<tag>out.letters+='S';</tag></item>
                  <item>T<tag>out.letters+='T';</tag></item>
                  <item>U<tag>out.letters+='U';</tag></item>
                  <item>V<tag>out.letters+='V';</tag></item>
                  <item>W<tag>out.letters+='W';</tag></item>
                  <item>X<tag>out.letters+='X';</tag></item>
                  <item>Y<tag>out.letters+='Y';</tag></item>
                  <item>Z<tag>out.letters+='Z';</tag></item>
              </one-of>
          </item>
    <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
       </rule>

       <rule id="digits">
           <item repeat="0-1"><ruleref uri="#text"/></item>
```

```
            <tag>out.numbers=""</tag>
            <item repeat="2-4">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Data di scadenza della carta di credito

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:
```

```
            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
               <item>april<tag>out="04";</tag></item>
               <item>may<tag>out="05";</tag></item>
               <item>june<tag>out="06";</tag></item>
               <item>july<tag>out="07";</tag></item>
               <item>august<tag>out="08";</tag></item>
               <item>september<tag>out="09";</tag></item>
```

```
            <item>october<tag>out="10";</tag></item>
            <item>november<tag>out="11";</tag></item>
            <item>december<tag>out="12";</tag></item>
            <item>jan<tag>out="01";</tag></item>
            <item>feb<tag>out="02";</tag></item>
            <item>aug<tag>out="08";</tag></item>
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

     <rule id="digits">
         <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
             <item>0<tag>out=0;</tag></item>
             <item>1<tag>out=1;</tag></item>
             <item>2<tag>out=2;</tag></item>
             <item>3<tag>out=3;</tag></item>
             <item>4<tag>out=4;</tag></item>
             <item>5<tag>out=5;</tag></item>
             <item>6<tag>out=6;</tag></item>
             <item>7<tag>out=7;</tag></item>
             <item>8<tag>out=8;</tag></item>
             <item>9<tag>out=9;</tag></item>
             <item>one<tag>out=1;</tag></item>
             <item>two<tag>out=2;</tag></item>
             <item>three<tag>out=3;</tag></item>
             <item>four<tag>out=4;</tag></item>
             <item>five<tag>out=5;</tag></item>
             <item>six<tag>out=6;</tag></item>
```

```
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
```

```
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data di scadenza della polizza, giorno/mese/anno

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My policy expired on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
```

```
                Input: My policy will expire on July Sixteen Two Thousand and Eleven
                Output: 07/16/11

            Scenario 3:
                Input: My policy expired on July Thirty Two Thousand and Eleven
                Output: 07/30/11
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item>
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                  <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                  <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                  <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
             </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My policy expired on</item>
              <item repeat="0-1">My policy will expire on</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
```

```
            <item>Hmm</item>
            <item>Mmm</item>
            <item>My</item>
          </one-of>
        </rule>

      <rule id="months">
         <tag>out.mon=""</tag>
  <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
           <item>january<tag>out.mon+="01";</tag></item>
           <item>february<tag>out.mon+="02";</tag></item>
           <item>march<tag>out.mon+="03";</tag></item>
           <item>april<tag>out.mon+="04";</tag></item>
           <item>may<tag>out.mon+="05";</tag></item>
           <item>june<tag>out.mon+="06";</tag></item>
           <item>july<tag>out.mon+="07";</tag></item>
           <item>august<tag>out.mon+="08";</tag></item>
           <item>september<tag>out.mon+="09";</tag></item>
           <item>october<tag>out.mon+="10";</tag></item>
           <item>november<tag>out.mon+="11";</tag></item>
           <item>december<tag>out.mon+="12";</tag></item>
           <item>jan<tag>out.mon+="01";</tag></item>
           <item>feb<tag>out.mon+="02";</tag></item>
           <item>aug<tag>out.mon+="08";</tag></item>
           <item>sept<tag>out.mon+="09";</tag></item>
           <item>oct<tag>out.mon+="10";</tag></item>
           <item>nov<tag>out.mon+="11";</tag></item>
           <item>dec<tag>out.mon+="12";</tag></item>
         </one-of>
      </rule>

       <rule id="digits">
           <one-of>
               <item>zero<tag>out=0;</tag></item>
               <item>one<tag>out=1;</tag></item>
               <item>two<tag>out=2;</tag></item>
               <item>three<tag>out=3;</tag></item>
               <item>four<tag>out=4;</tag></item>
               <item>five<tag>out=5;</tag></item>
               <item>six<tag>out=6;</tag></item>
               <item>seven<tag>out=7;</tag></item>
               <item>eight<tag>out=8;</tag></item>
               <item>nine<tag>out=9;</tag></item>
```

```
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

Data di rinnovo della polizza, mese/anno

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
```

```
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I renewed my policy on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: My policy will renew on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy will renew on</item>
```

```
            <item repeat="0-1">My policy was renewed on</item>
            <item repeat="0-1">Renew policy on</item>
            <item repeat="0-1">I renewed my policy on</item>
        </one-of>
    </rule>

    <rule id="hesitation">
        <one-of>
           <item>Hmm</item>
           <item>Mmm</item>
           <item>My</item>
        </one-of>
      </rule>

    <rule id="months">
       <item repeat="0-1"><ruleref uri="#text"/></item>
       <tag>out.mon=""</tag>
       <one-of>
           <item>january<tag>out.mon+="01";</tag></item>
           <item>february<tag>out.mon+="02";</tag></item>
           <item>march<tag>out.mon+="03";</tag></item>
           <item>april<tag>out.mon+="04";</tag></item>
           <item>may<tag>out.mon+="05";</tag></item>
           <item>june<tag>out.mon+="06";</tag></item>
           <item>july<tag>out.mon+="07";</tag></item>
           <item>august<tag>out.mon+="08";</tag></item>
           <item>september<tag>out.mon+="09";</tag></item>
           <item>october<tag>out.mon+="10";</tag></item>
           <item>november<tag>out.mon+="11";</tag></item>
           <item>december<tag>out.mon+="12";</tag></item>
           <item>jan<tag>out.mon+="01";</tag></item>
           <item>feb<tag>out.mon+="02";</tag></item>
           <item>aug<tag>out.mon+="08";</tag></item>
           <item>sept<tag>out.mon+="09";</tag></item>
           <item>oct<tag>out.mon+="10";</tag></item>
           <item>nov<tag>out.mon+="11";</tag></item>
           <item>dec<tag>out.mon+="12";</tag></item>
       </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
```

```
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<!--<tag>out=2000;</tag>--></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
```

```
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data di inizio della politica

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I bought my policy on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My policy started on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
```

```
                    <item><ruleref uri="#above_twenty"/><tag>out+=
rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">I bought my policy on</item>
                <item repeat="0-1">I bought policy on</item>
                <item repeat="0-1">My policy started on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
               <item>oct<tag>out.mon+="10";</tag></item>
```

```
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
```

```
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Importo del reclamo

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
```

```
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to make a claim of one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: Requesting claim of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to place a claim for</item>
              <item repeat="0-1">I want to make a claim of</item>
              <item repeat="0-1">I assess damage of</item>
              <item repeat="0-1">Requesting claim of</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
```

```
  <rule id="thanks">
      <one-of>
         <item>Thanks</item>
         <item>I think</item>
      </one-of>
    </rule>

<rule id="digits">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.num = 0;</tag>
    <one-of>
        <item>0<tag>out.num+=0;</tag></item>
        <item>1<tag>out.num+=1;</tag></item>
        <item>2<tag>out.num+=2;</tag></item>
        <item>3<tag>out.num+=3;</tag></item>
        <item>4<tag>out.num+=4;</tag></item>
        <item>5<tag>out.num+=5;</tag></item>
        <item>6<tag>out.num+=6;</tag></item>
        <item>7<tag>out.num+=7;</tag></item>
        <item>8<tag>out.num+=8;</tag></item>
        <item>9<tag>out.num+=9;</tag></item>
        <item>one<tag>out.num+=1;</tag></item>
        <item>two<tag>out.num+=2;</tag></item>
        <item>three<tag>out.num+=3;</tag></item>
        <item>four<tag>out.num+=4;</tag></item>
        <item>five<tag>out.num+=5;</tag></item>
        <item>six<tag>out.num+=6;</tag></item>
        <item>seven<tag>out.num+=7;</tag></item>
        <item>eight<tag>out.num+=8;</tag></item>
        <item>nine<tag>out.num+=9;</tag></item>
    </one-of>
    <item repeat="0-1"><ruleref uri="#currency"/></item>
</rule>

<rule id="teens">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.teen = 0;</tag>
    <one-of>
        <item>ten<tag>out.teen+=10;</tag></item>
        <item>eleven<tag>out.teen+=11;</tag></item>
        <item>twelve<tag>out.teen+=12;</tag></item>
        <item>thirteen<tag>out.teen+=13;</tag></item>
        <item>fourteen<tag>out.teen+=14;</tag></item>
```

```
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
```

```
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Importo del premio

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Premium amounts
             Scenario 1:
                 Input: The premium for one hundre ten dollars
                 Output: $110
```

```
             Scenario 2:
                 Input: RPremium amount of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">A premium of</item>
              <item repeat="0-1">Premium amount of</item>
              <item repeat="0-1">The premium for</item>
              <item repeat="0-1">Insurance premium for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

          <rule id="thanks">
              <one-of>
                 <item>Thanks</item>
                 <item>I think</item>
              </one-of>
            </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
```

```
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
                <item>9<tag>out.num+=9;</tag></item>
                <item>one<tag>out.num+=1;</tag></item>
                <item>two<tag>out.num+=2;</tag></item>
                <item>three<tag>out.num+=3;</tag></item>
                <item>four<tag>out.num+=4;</tag></item>
                <item>five<tag>out.num+=5;</tag></item>
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
```

```
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
```

```
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Quantità della polizza

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: I want policy for ten
                 Output: 10

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
```

```
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <one-of>
              <item repeat="0-1">I want policy for</item>
              <item repeat="0-1">I want to order policy for</item>
              <item repeat="0-1">The number is</item>
           </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
```

```
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
            <item>forty<tag>out=40;</tag></item>
            <item>fifty<tag>out=50;</tag></item>
            <item>sixty<tag>out=60;</tag></item>
            <item>seventy<tag>out=70;</tag></item>
            <item>eighty<tag>out=80;</tag></item>
            <item>ninety<tag>out=90;</tag></item>
            <item>20<tag>out=20;</tag></item>
            <item>30<tag>out=30;</tag></item>
            <item>40<tag>out=40;</tag></item>
```

```
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Grammatiche per le telecomunicazioni (download)

Per le telecomunicazioni sono supportate le seguenti grammatiche: numero di telefono, numero di serie, numero SIM, codice postale statunitense, data di scadenza della carta di credito, date di inizio del piano, date di rinnovo e scadenza, data di inizio del servizio, quantità dell'attrezzatura e importo della fattura.

Numero di telefono

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support 10-12 digits number and here are couple of examples of
 valid inputs:

             Scenario 1:
                 Input: Mmm My phone number is two zero one two five two six seven
 eight five
                 Output: 2012526785

             Scenario 2:
```

```
                Input: My phone number is two zero one two five two six seven eight
 five
                Output: 2012526785

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My phone number is</item>
              <item repeat="0-1">Phone number is</item>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">Yes, it's</item>
              <item repeat="0-1">Yes, it is</item>
              <item repeat="0-1">Yes it is</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="10-12">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
```

```
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numero di serie

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My serial number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

             Scenario 2:
                 Input: Device Serial number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

         -->
```

```
        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My serial number is</item>
                <item repeat="0-1">Device Serial number</item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">The IMEI number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="15">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
```

```
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numero SIM

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My SIM number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My SIM number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: My SIM number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
```

```
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My SIM number is</item>
                <item repeat="0-1">SIM number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
```

```
                <item>E<tag>out.letters+='E';</tag></item>
                <item>F<tag>out.letters+='F';</tag></item>
                <item>G<tag>out.letters+='G';</tag></item>
                <item>H<tag>out.letters+='H';</tag></item>
                <item>I<tag>out.letters+='I';</tag></item>
                <item>J<tag>out.letters+='J';</tag></item>
                <item>K<tag>out.letters+='K';</tag></item>
                <item>L<tag>out.letters+='L';</tag></item>
                <item>M<tag>out.letters+='M';</tag></item>
                <item>N<tag>out.letters+='N';</tag></item>
                <item>O<tag>out.letters+='O';</tag></item>
                <item>P<tag>out.letters+='P';</tag></item>
                <item>Q<tag>out.letters+='Q';</tag></item>
                <item>R<tag>out.letters+='R';</tag></item>
                <item>S<tag>out.letters+='S';</tag></item>
                <item>T<tag>out.letters+='T';</tag></item>
                <item>U<tag>out.letters+='U';</tag></item>
                <item>V<tag>out.letters+='V';</tag></item>
                <item>W<tag>out.letters+='W';</tag></item>
                <item>X<tag>out.letters+='X';</tag></item>
                <item>Y<tag>out.letters+='Y';</tag></item>
                <item>Z<tag>out.letters+='Z';</tag></item>
            </one-of>
        </item>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.numbers=""</tag>
        <item repeat="1-10">
            <one-of>
                <item>0<tag>out.numbers+=0;</tag></item>
                <item>1<tag>out.numbers+=1;</tag></item>
                <item>2<tag>out.numbers+=2;</tag></item>
                <item>3<tag>out.numbers+=3;</tag></item>
                <item>4<tag>out.numbers+=4;</tag></item>
                <item>5<tag>out.numbers+=5;</tag></item>
                <item>6<tag>out.numbers+=6;</tag></item>
                <item>7<tag>out.numbers+=7;</tag></item>
                <item>8<tag>out.numbers+=8;</tag></item>
                <item>9<tag>out.numbers+=9;</tag></item>
            </one-of>
        </item>
    </rule>
```

```
</grammar>
```

Codice postale degli Stati Uniti

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support 5 digits code and here are couple of examples of valid
 inputs:

             Scenario 1:
                 Input: Mmmm My zipcode is umm One Oh Nine Eight Seven
                 Output: 10987

             Scenario 2:
                 Input: My zipcode is One Oh Nine Eight Seven
                 Output: 10987

          -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My zipcode is</item>
              <item repeat="0-1">Zipcode is</item>
              <item repeat="0-1">It is</item>
           </one-of>
        </rule>
```

```
        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="5">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>Oh<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Data di scadenza della carta di credito

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
```

```
        </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
               <item>april<tag>out="04";</tag></item>
               <item>may<tag>out="05";</tag></item>
               <item>june<tag>out="06";</tag></item>
               <item>july<tag>out="07";</tag></item>
               <item>august<tag>out="08";</tag></item>
               <item>september<tag>out="09";</tag></item>
               <item>october<tag>out="10";</tag></item>
               <item>november<tag>out="11";</tag></item>
               <item>december<tag>out="12";</tag></item>
               <item>jan<tag>out="01";</tag></item>
               <item>feb<tag>out="02";</tag></item>
               <item>aug<tag>out="08";</tag></item>
               <item>sept<tag>out="09";</tag></item>
               <item>oct<tag>out="10";</tag></item>
               <item>nov<tag>out="11";</tag></item>
               <item>dec<tag>out="12";</tag></item>
               <item>1<tag>out="01";</tag></item>
               <item>2<tag>out="02";</tag></item>
               <item>3<tag>out="03";</tag></item>
               <item>4<tag>out="04";</tag></item>
               <item>5<tag>out="05";</tag></item>
               <item>6<tag>out="06";</tag></item>
               <item>7<tag>out="07";</tag></item>
               <item>8<tag>out="08";</tag></item>
               <item>9<tag>out="09";</tag></item>
               <item>ten<tag>out="10";</tag></item>
               <item>eleven<tag>out="11";</tag></item>
               <item>twelve<tag>out="12";</tag></item>
           </one-of>
       </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
```

```
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
```

```
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data di scadenza del piano, giorno/mese/anno

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
```

```
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan expires on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: My plan will expire on July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: My plan will expire on July Thirty Two Thousand and Eleven
                 Output: 07/30/11
         -->

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item>
                 <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                   <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                   <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                 </one-of>
                 <one-of>
                     <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                     <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                     <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                     <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
```

```
        </one-of>
    </item>
</rule>

<rule id="text">
  <item repeat="0-1"><ruleref uri="#hesitation"/></item>
  <one-of>
       <item repeat="0-1">My plan expires on</item>
       <item repeat="0-1">My plan expired on</item>
       <item repeat="0-1">My plan will expire on</item>
   </one-of>
</rule>

<rule id="hesitation">
   <one-of>
      <item>Hmm</item>
      <item>Mmm</item>
      <item>My</item>
   </one-of>
 </rule>

<rule id="months">
   <tag>out.mon=""</tag>
<item repeat="0-1"><ruleref uri="#text"/></item>

   <one-of>
     <item>january<tag>out.mon+="01";</tag></item>
     <item>february<tag>out.mon+="02";</tag></item>
     <item>march<tag>out.mon+="03";</tag></item>
     <item>april<tag>out.mon+="04";</tag></item>
     <item>may<tag>out.mon+="05";</tag></item>
     <item>june<tag>out.mon+="06";</tag></item>
     <item>july<tag>out.mon+="07";</tag></item>
     <item>august<tag>out.mon+="08";</tag></item>
     <item>september<tag>out.mon+="09";</tag></item>
     <item>october<tag>out.mon+="10";</tag></item>
     <item>november<tag>out.mon+="11";</tag></item>
     <item>december<tag>out.mon+="12";</tag></item>
     <item>jan<tag>out.mon+="01";</tag></item>
     <item>feb<tag>out.mon+="02";</tag></item>
     <item>aug<tag>out.mon+="08";</tag></item>
     <item>sept<tag>out.mon+="09";</tag></item>
     <item>oct<tag>out.mon+="10";</tag></item>
     <item>nov<tag>out.mon+="11";</tag></item>
```

```
             <item>dec<tag>out.mon+="12";</tag></item>
           </one-of>
        </rule>

         <rule id="digits">
             <one-of>
                 <item>zero<tag>out=0;</tag></item>
                 <item>one<tag>out=1;</tag></item>
                 <item>two<tag>out=2;</tag></item>
                 <item>three<tag>out=3;</tag></item>
                 <item>four<tag>out=4;</tag></item>
                 <item>five<tag>out=5;</tag></item>
                 <item>six<tag>out=6;</tag></item>
                 <item>seven<tag>out=7;</tag></item>
                 <item>eight<tag>out=8;</tag></item>
                 <item>nine<tag>out=9;</tag></item>
             </one-of>
         </rule>

         <rule id="teens">
             <one-of>
                 <item>ten<tag>out=10;</tag></item>
                 <item>eleven<tag>out=11;</tag></item>
                 <item>twelve<tag>out=12;</tag></item>
                 <item>thirteen<tag>out=13;</tag></item>
                 <item>fourteen<tag>out=14;</tag></item>
                 <item>fifteen<tag>out=15;</tag></item>
                 <item>sixteen<tag>out=16;</tag></item>
                 <item>seventeen<tag>out=17;</tag></item>
                 <item>eighteen<tag>out=18;</tag></item>
                 <item>nineteen<tag>out=19;</tag></item>
             </one-of>
         </rule>

         <rule id="thousands">
             <item>two thousand</item>
             <item repeat="0-1">and</item>
             <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
             <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
             <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
         </rule>
```

```
        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

Data di rinnovo del piano, mese/anno

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will renew on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: Renew plan on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
```

```
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My plan will renew on</item>
                <item repeat="0-1">My plan was renewed on</item>
                <item repeat="0-1">Renew plan on</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
```

```
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="thousands">
        <item>two thousand</item>
        <item repeat="0-1">and</item>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data di inizio del piano, mese/giorno

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will start on july twenty three
                 Output: 07/23
```

```
            Scenario 2:
                Input: My plan will start on july fifteen
                Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan started on</item>
                <item repeat="0-1">My plan will start on</item>
                <item repeat="0-1">I paid it on</item>
                <item repeat="0-1">I paid bill for</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
```

```
            <item>february<tag>out.mon+="02";</tag></item>
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
```

```
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
        </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data di inizio del servizio, mese/giorno

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan starts on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: I want to activate on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
```

```
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan starts on</item>
                <item repeat="0-1">I want to start my plan on</item>
                <item repeat="0-1">Activation date of</item>
                <item repeat="0-1">Start activation on</item>
                <item repeat="0-1">I want to activate on</item>
                <item repeat="0-1">Activate plan starting</item>
                <item repeat="0-1">Starting</item>
                <item repeat="0-1">Start on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
```

```
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
```

```
                <item>tenth<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Quantità dell'attrezzatura

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
```

```
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: It is ten
                 Output: 10

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">The number is</item>
              <item repeat="0-1">Order</item>
              <item repeat="0-1">I want to order</item>
              <item repeat="0-1">Total equipment</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
```

```
            <item>Mmm</item>
            <item>My</item>
        </one-of>
      </rule>

    <rule id="thanks">
        <one-of>
            <item>Thanks</item>
            <item>I think</item>
        </one-of>
      </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
```

```
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
```

```
</grammar>
```

Importo della fattura

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

                 Input: I want to make a payment of one hundred ten dollars
                 Output: $110

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to make a payment for</item>
              <item repeat="0-1">I want to make a payment of</item>
              <item repeat="0-1">Pay a total of</item>
              <item repeat="0-1">Paying</item>
              <item repeat="0-1">Pay bill for </item>
           </one-of>
```

```
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

          <rule id="thanks">
              <one-of>
                 <item>Thanks</item>
                 <item>I think</item>
              </one-of>
            </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
                <item>9<tag>out.num+=9;</tag></item>
                <item>one<tag>out.num+=1;</tag></item>
                <item>two<tag>out.num+=2;</tag></item>
                <item>three<tag>out.num+=3;</tag></item>
                <item>four<tag>out.num+=4;</tag></item>
                <item>five<tag>out.num+=5;</tag></item>
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
```

```
        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
```

```
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Grammatiche generiche (download)

Forniamo le seguenti grammatiche generiche: alfanumerico, valuta, data (mm/gg/aa), numeri, saluto, esitazione e agente.

Carattere alfanumerico

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
```

```
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

            Scenario 1:
                Input: A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
```

```
            <one-of>
                <item>A<tag>out.letters+='A';</tag></item>
                <item>B<tag>out.letters+='B';</tag></item>
                <item>C<tag>out.letters+='C';</tag></item>
                <item>D<tag>out.letters+='D';</tag></item>
                <item>E<tag>out.letters+='E';</tag></item>
                <item>F<tag>out.letters+='F';</tag></item>
                <item>G<tag>out.letters+='G';</tag></item>
                <item>H<tag>out.letters+='H';</tag></item>
                <item>I<tag>out.letters+='I';</tag></item>
                <item>J<tag>out.letters+='J';</tag></item>
                <item>K<tag>out.letters+='K';</tag></item>
                <item>L<tag>out.letters+='L';</tag></item>
                <item>M<tag>out.letters+='M';</tag></item>
                <item>N<tag>out.letters+='N';</tag></item>
                <item>O<tag>out.letters+='O';</tag></item>
                <item>P<tag>out.letters+='P';</tag></item>
                <item>Q<tag>out.letters+='Q';</tag></item>
                <item>R<tag>out.letters+='R';</tag></item>
                <item>S<tag>out.letters+='S';</tag></item>
                <item>T<tag>out.letters+='T';</tag></item>
                <item>U<tag>out.letters+='U';</tag></item>
                <item>V<tag>out.letters+='V';</tag></item>
                <item>W<tag>out.letters+='W';</tag></item>
                <item>X<tag>out.letters+='X';</tag></item>
                <item>Y<tag>out.letters+='Y';</tag></item>
                <item>Z<tag>out.letters+='Z';</tag></item>
            </one-of>
        </item>
    </rule>

    <rule id="digits">
        <tag>out.numbers=""</tag>
        <item repeat="1-10">
            <one-of>
                <item>0<tag>out.numbers+=0;</tag></item>
                <item>1<tag>out.numbers+=1;</tag></item>
                <item>2<tag>out.numbers+=2;</tag></item>
                <item>3<tag>out.numbers+=3;</tag></item>
                <item>4<tag>out.numbers+=4;</tag></item>
                <item>5<tag>out.numbers+=5;</tag></item>
                <item>6<tag>out.numbers+=6;</tag></item>
                <item>7<tag>out.numbers+=7;</tag></item>
                <item>8<tag>out.numbers+=8;</tag></item>
```

```
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Valuta

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
                <item>9<tag>out.num+=9;</tag></item>
                <item>one<tag>out.num+=1;</tag></item>
                <item>two<tag>out.num+=2;</tag></item>
```

```
                <item>three<tag>out.num+=3;</tag></item>
                <item>four<tag>out.num+=4;</tag></item>
                <item>five<tag>out.num+=5;</tag></item>
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>
```

```
        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
        </rule>
</grammar>
```

Data, gg/mm

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                              http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                    <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                    <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                </one-of>
                <item><ruleref uri="#months"/><tag>out = out + rules.months;</tag></
item>
            </item>
        </rule>

        <rule id="months">
           <one-of>
             <item>january<tag>out="january";</tag></item>
             <item>february<tag>out="february";</tag></item>
             <item>march<tag>out="march";</tag></item>
             <item>april<tag>out="april";</tag></item>
             <item>may<tag>out="may";</tag></item>
             <item>june<tag>out="june";</tag></item>
             <item>july<tag>out="july";</tag></item>
             <item>august<tag>out="august";</tag></item>
             <item>september<tag>out="september";</tag></item>
             <item>october<tag>out="october";</tag></item>
             <item>november<tag>out="november";</tag></item>
             <item>december<tag>out="december";</tag></item>
             <item>jan<tag>out="january";</tag></item>
             <item>feb<tag>out="february";</tag></item>
             <item>aug<tag>out="august";</tag></item>
             <item>sept<tag>out="september";</tag></item>
             <item>oct<tag>out="october";</tag></item>
             <item>nov<tag>out="november";</tag></item>
             <item>dec<tag>out="december";</tag></item>
           </one-of>
```

```
        </rule>

        <rule id="digits">
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>first<tag>out=1;</tag></item>
                <item>second<tag>out=2;</tag></item>
                <item>third<tag>out=3;</tag></item>
                <item>fourth<tag>out=4;</tag></item>
                <item>fifth<tag>out=5;</tag></item>
                <item>sixth<tag>out=6;</tag></item>
                <item>seventh<tag>out=7;</tag></item>
                <item>eighth<tag>out=8;</tag></item>
                <item>ninth<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
```

```
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Data, mm/aa

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
```

```
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
               <item>december<tag>out.mon+="december";</tag></item>
               <item>jan<tag>out.mon+="january";</tag></item>
               <item>feb<tag>out.mon+="february";</tag></item>
               <item>aug<tag>out.mon+="august";</tag></item>
               <item>sept<tag>out.mon+="september";</tag></item>
               <item>oct<tag>out.mon+="october";</tag></item>
               <item>nov<tag>out.mon+="november";</tag></item>
               <item>dec<tag>out.mon+="december";</tag></item>
           </one-of>
       </rule>

        <rule id="digits">
            <one-of>
                <item>zero<tag>out=0;</tag></item>
                <item>one<tag>out=1;</tag></item>
```

```
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

       <!--  <rule id="singleDigit">
            <item><ruleref uri="#digits"/><tag>out += rules.digits;</tag></item>
        </rule> -->

        <rule id="thousands">
            <!-- <item>
                <ruleref uri="#digits"/>
                <tag>out = (1000 * rules.digits);</tag>
                thousand
            </item> -->
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
```

```
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Data, gg/mm/aaaa

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out="";</tag>
             <item repeat="1-10">
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                   <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                   <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                 </one-of>
```

```
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
               <item>december<tag>out.mon+="december";</tag></item>
               <item>jan<tag>out.mon+="january";</tag></item>
               <item>feb<tag>out.mon+="february";</tag></item>
               <item>aug<tag>out.mon+="august";</tag></item>
               <item>sept<tag>out.mon+="september";</tag></item>
               <item>oct<tag>out.mon+="october";</tag></item>
               <item>nov<tag>out.mon+="november";</tag></item>
               <item>dec<tag>out.mon+="december";</tag></item>
           </one-of>
       </rule>

        <rule id="digits">
            <one-of>
                <item>zero<tag>out=0;</tag></item>
                <item>one<tag>out=1;</tag></item>
```

```
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
```

```
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

 </grammar>
```

Numeri, cifre

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="singleDigit">
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
```

```
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=6;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Numeri, ordinali

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
```

```
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Agente

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
```

```
          <one-of>
             <item>Can I talk to the agent<tag>out="You will be trasnfered to the
 agent in a while"</tag></item>
             <item>talk to an agent<tag>out="You will be trasnfered to the agent in a
 while"</tag></item>
          </one-of>
        </rule>
</grammar>
```

Saluto

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>hey<tag>out="Greeting"</tag></item>
              <item>hi<tag>out="Greeting"</tag></item>
              <item>Hi<tag>out="Greeting"</tag></item>
              <item>Hey<tag>out="Greeting"</tag></item>
              <item>Hello<tag>out="Greeting"</tag></item>
              <item>hello<tag>out="Greeting"</tag></item>
           </one-of>
         </rule>
</grammar>
```

Esitazione

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                            http://www.w3.org/TR/speech-grammar/grammar.xsd"
        xml:lang="en-US" version="1.0"
        root="main"
        mode="voice"
        tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <ruleref uri="#text"/><tag>out = rules.text</tag>
        </rule>

        <rule id="text">
          <one-of>
             <item>Hmm<tag>out="Waiting for your input"</tag></item>
             <item>Mmm<tag>out="Waiting for your input"</tag></item>
             <item>Can you please wait<tag>out="Waiting for your input"</tag></item>
          </one-of>
        </rule>
</grammar>
```

Tipo di slot composito

Uno slot composito è una combinazione di due o più slot che acquisiscono più informazioni in un singolo input dell'utente. Ad esempio, puoi configurare il bot per ottenere la posizione richiedendo «città e stato o codice postale». Al contrario, quando la conversazione è configurata per utilizzare tipi di slot separati, si ottiene un'esperienza conversazionale rigida («Qual è la città?» seguita da «Cos'è il codice postale?»). Con uno slot composito, puoi acquisire tutte le informazioni attraverso un unico slot. Uno slot composito è una combinazione di slot chiamati subslot, come città, stato e codice postale.

Puoi utilizzare una combinazione di tipi di slot Amazon Lex disponibili (integrati) e slot personalizzati (slot personalizzati). È possibile progettare espressioni logiche per acquisire informazioni all'interno dei sottoslot richiesti. Ad esempio: città e stato o codice postale.

Il tipo di slot composito è disponibile solo in en-US.

Creazione di un tipo di slot composito

Per utilizzare i subslot all'interno di uno slot composito, è necessario innanzitutto configurare il tipo di slot composito. A tale scopo, utilizza i passaggi della console per aggiungere un tipo di slot o

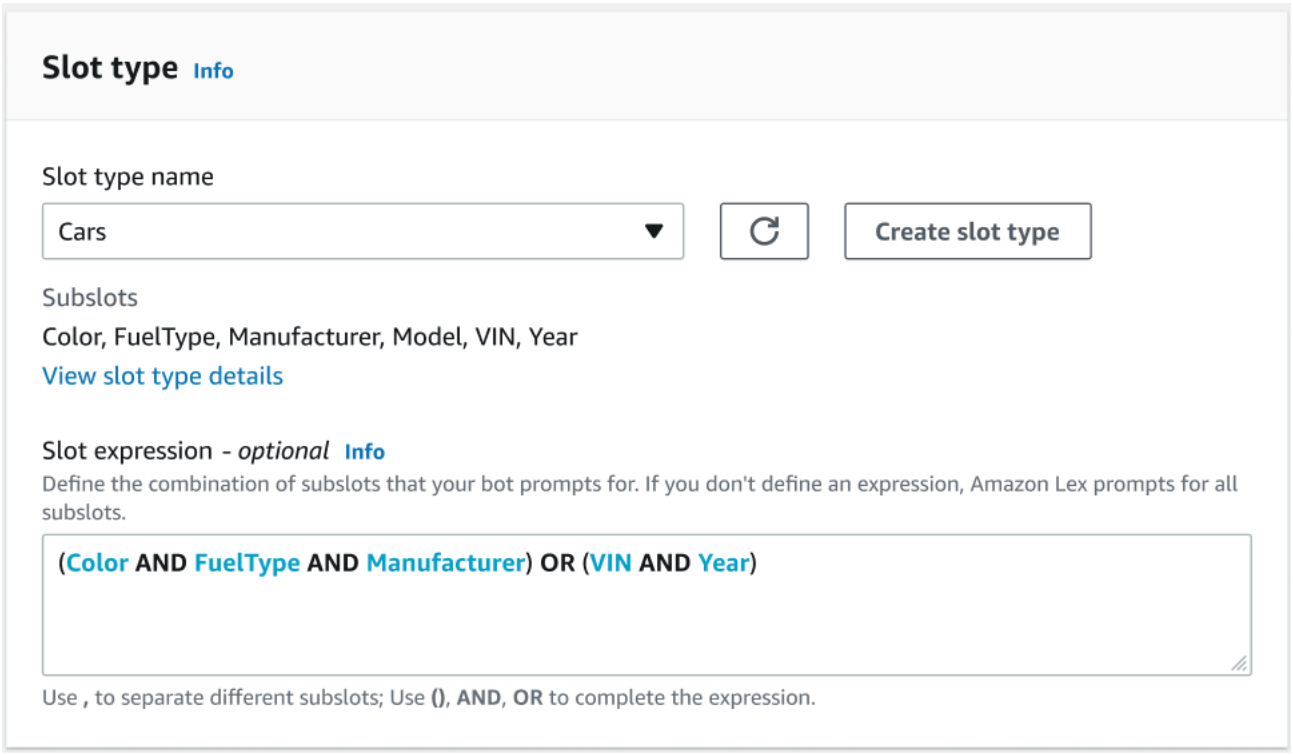
l'operazione API. Dopo aver scelto il nome e la descrizione del tipo di slot composito, è necessario fornire informazioni per i sottoslot. Per ulteriori informazioni sull'aggiunta di un tipo di slot, vedere Aggiungere tipi di slot

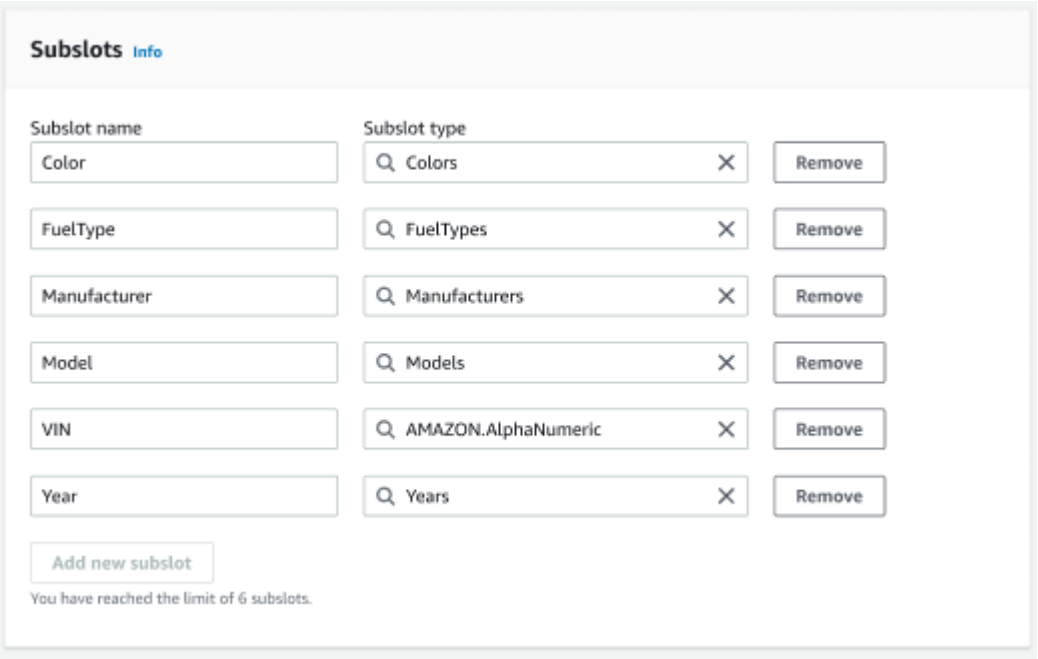
Sottoslot

Un tipo di slot composito richiede la configurazione degli slot sottostanti, chiamati subslot. Se desideri ottenere più informazioni da un cliente in un'unica richiesta, configura una combinazione di subslot. Ad esempio: città, stato e codice postale. È possibile aggiungere fino a 6 sottoslot per uno slot composito.

È possibile utilizzare slot di tipi di slot singoli per aggiungere sottoslot al tipo di slot composito. Tuttavia, non è possibile utilizzare un tipo di slot composito come tipo di slot per un sottoslot.

Le immagini seguenti illustrano uno slot composito «Auto», che è una combinazione di sottoslot: Colore, Produttore, ModelloFuelType, VIN e Anno.





Generatore di espressioni

Per favorire la realizzazione di uno slot composito, puoi opzionalmente utilizzare il generatore di espressioni. Con il generatore di espressioni, è possibile progettare un'espressione di slot logica per acquisire i valori di subslot richiesti nell'ordine desiderato. Come parte dell'espressione booleana, è possibile utilizzare operatori come AND e OR. In base all'espressione progettata, quando i sottoslot richiesti sono soddisfatti, lo slot composito viene considerato soddisfatto.

Utilizzo di un tipo di slot composito

Per alcuni scopi, potresti voler catturare slot diversi come parte di un singolo slot. Ad esempio, un bot per la pianificazione della manutenzione dell'auto potrebbe avere un intento con la seguente espressione:

```
My car is a {car}
```

L'intento prevede che lo slot composito {car} contenga un elenco degli slot, comprensivo dei dettagli dell'auto. Ad esempio, «Toyota Camry bianca 2021».

Lo slot composito è diverso da uno slot multivalore. Lo slot composito è composto da più slot, ognuno con il proprio valore. Invece, uno slot multivalore è uno slot singolo che può contenere un elenco di valori. Per ulteriori informazioni sugli slot multivalore, vedere Utilizzo di più valori in uno slot

Per uno slot composito, Amazon Lex restituisce un valore per ogni sottoslot in risposta all'operazione `RecognizeText` or`RecognizeUtterance`. Di seguito sono riportate le informazioni sullo slot

restituite per l'enunciato: «Voglio programmare un servizio per la mia «Toyota Camry bianca 2021» dal CarService bot.

```
"slots": {
    "CarType": {
        "value": {
            "originalValue": "White Toyota Camry 2021",
            "interpretedValue": "White Toyota Camry 2021",
            "resolvedValues": [
                "white Toyota Camry 2021"
            ]
        },
        "subSlots": {
            "Color": {
                "value": {
                    "originalValue": "White",
                    "interpretedValue": "White",
                    "resolvedValues": [
                        "white"
                    ]
                },
                "shape": "Scalar"
            },
            "Manufacturer": {
                "value": {
                    "originalValue": "Toyota",
                    "interpretedValue": "Toyota",
                    "resolvedValues": [
                        "Toyota"
                    ]
                },
                "shape": "Scalar"
            },
            "Model": {
                "value": {
                    "originalValue": "Camry",
                    "interpretedValue": "Camry",
                    "resolvedValues": [
                       "Camry"
                    ]
                },
                "shape": "Scalar"
            },
```

```
            "Year": {
                "value": {
                    "originalValue": "2021",
                    "interpretedValue": "2021",
                    "resolvedValues": [
                        "2021"
                    ]
                },
                "shape": "Scalar"
            }
        }
    },
    ...
}
```

È possibile creare uno slot composito nel primo o nell'n-esimo turno di una conversazione. In base ai valori di input forniti, lo slot composito può richiedere i restanti sottoslot richiesti.

Gli slot compositi restituiscono sempre un valore per ogni sottoslot. Quando l'enunciato non contiene un valore riconoscibile per un determinato sottoslot, non viene restituita alcuna risposta per quel particolare sottoslot.

Gli slot compositi funzionano con input sia testuali che vocali.

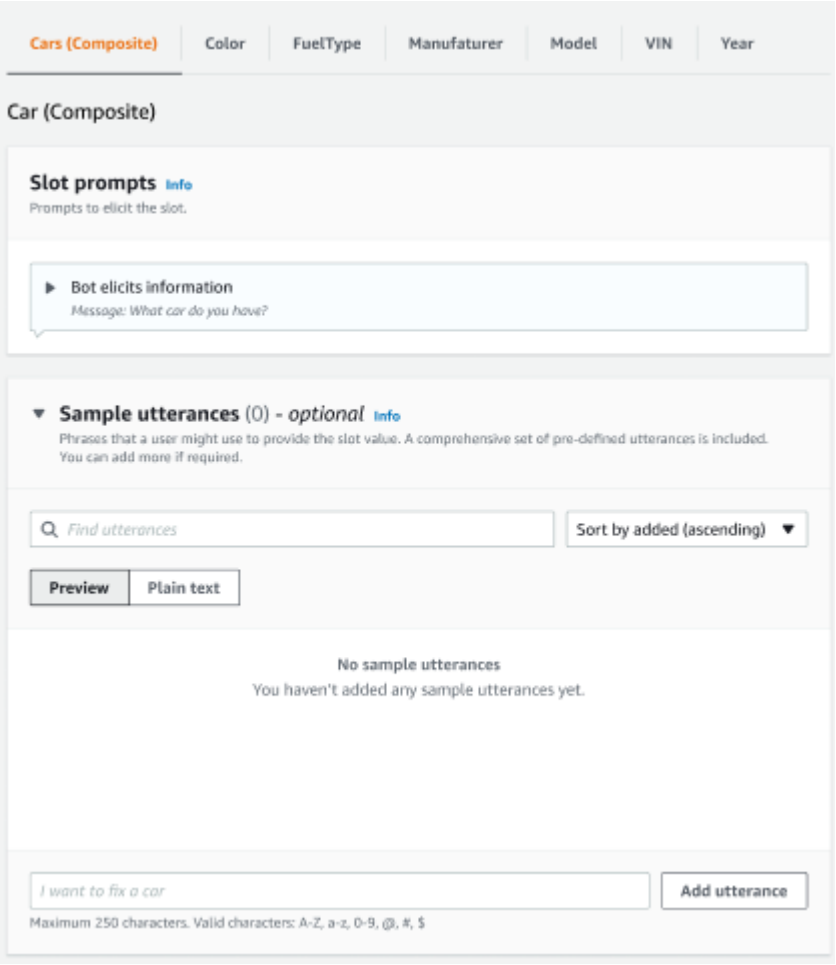
Quando si aggiunge uno slot a un intento, uno slot composito è disponibile solo come tipo di slot personalizzato.

È possibile utilizzare gli slot compositi nei prompt. Ad esempio, è possibile impostare la richiesta di conferma per un intento.

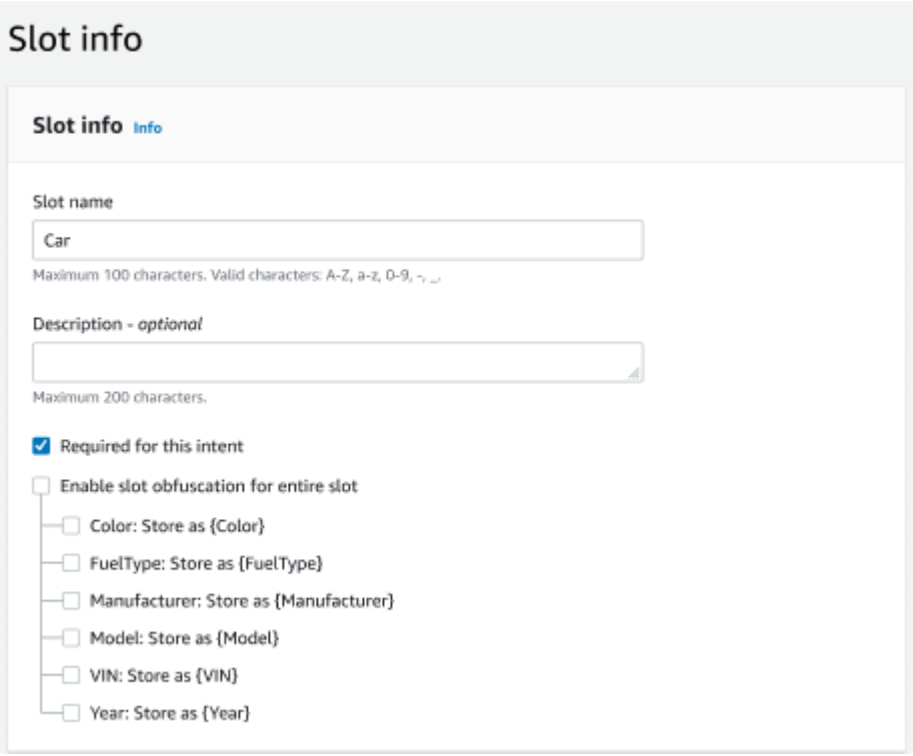
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

Quando Amazon Lex invia la richiesta all'utente, invia «Vuoi che pianifichi l'assistenza per la tua Toyota Camry bianca del 2021?»

Ogni sottoslot è configurato come slot. È possibile aggiungere i prompt degli slot per richiamare il subslot e gli enunciati di esempio. Puoi abilitare wait and continue per un subslot e per i valori predefiniti. Per ulteriori informazioni, consulta Utilizzo dei valori di slot predefiniti.



È possibile utilizzare l'offuscamento degli slot per mascherare l'intero slot composito nei registri delle conversazioni. Tieni presente che l'offuscamento degli slot viene applicato a livello di slot composito e, quando abilitato, i valori dei sottoslot appartenenti a uno slot composito vengono offuscati. Quando offuscate i valori degli slot, il valore di ciascuno dei valori degli slot viene sostituito con il nome dello slot. Per ulteriori informazioni, consulta Oscurare i valori degli slot nei registri delle conversazioni.



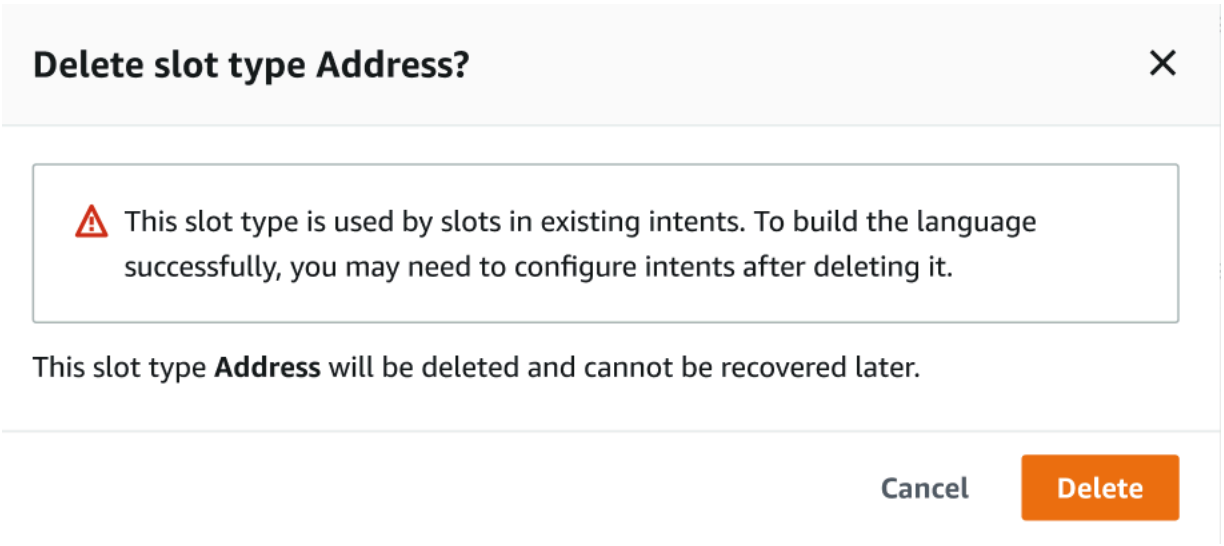
Modifica di un tipo di slot composito

È possibile modificare un sottoslot dall'interno della configurazione composita dello slot per modificare il nome e il tipo di slot secondario. Tuttavia, quando uno slot composito viene utilizzato da un intento, è necessario modificare gli intenti prima di modificare il sottoslot.

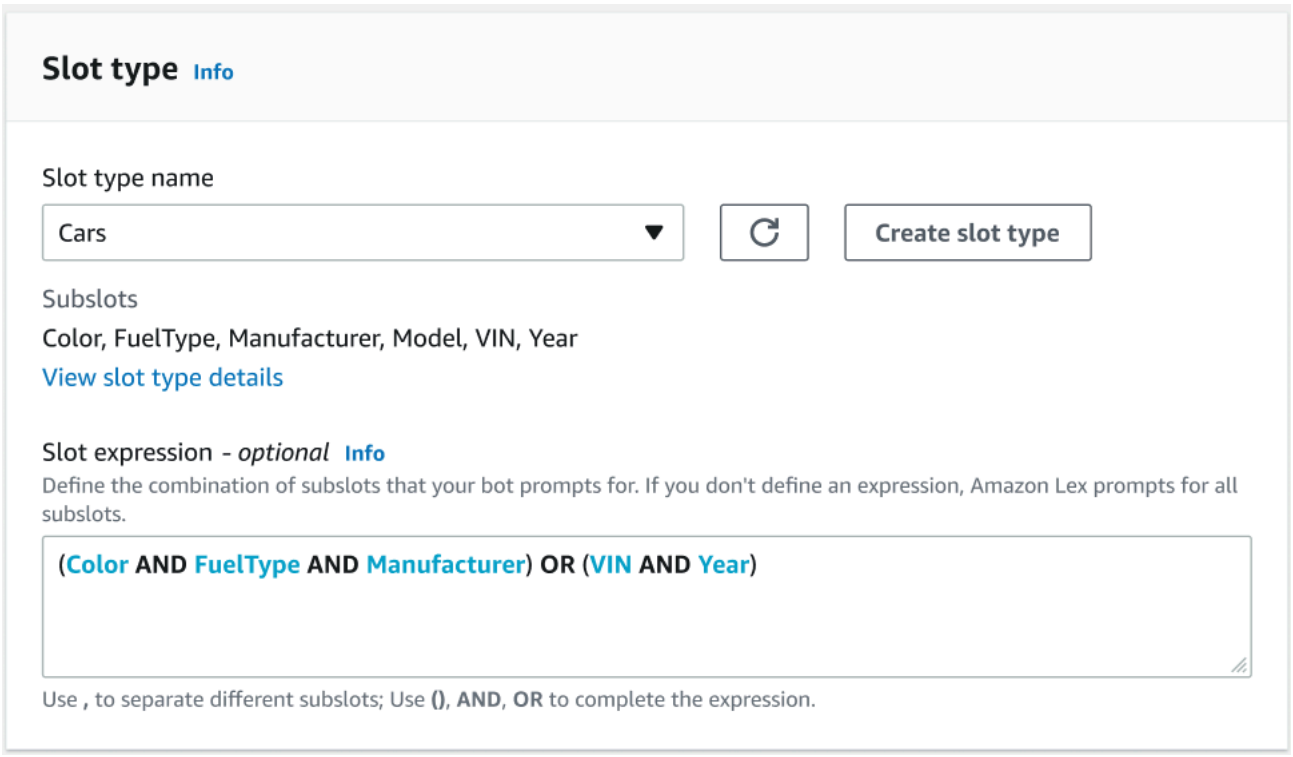
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

Eliminazione di un tipo di slot composito

È possibile eliminare un sottoslot dall'interno della configurazione dello slot composito. Tieni presente che quando un subslot è in uso nell'ambito di un intento, i sottoslot vengono comunque rimossi da tale intento.



L'espressione slot nel generatore di espressioni fornisce un avviso per informare sui sottoslot eliminati.



Testare un bot utilizzando la console

La console Amazon Lex V2 contiene una finestra di test che puoi utilizzare per testare l'interazione con il tuo bot. Utilizzi la finestra di test per avere una conversazione di prova con il tuo bot e per vedere le risposte che la tua applicazione riceve dal bot.

Esistono due tipi di test che puoi eseguire con il tuo bot. Il primo, il test rapido, ti consente di testare il tuo bot con le frasi esatte che hai usato per creare il bot. Ad esempio, se hai aggiunto l'espressione «Voglio raccogliere fiori» al tuo intento, puoi testare il bot usando quella frase esatta.

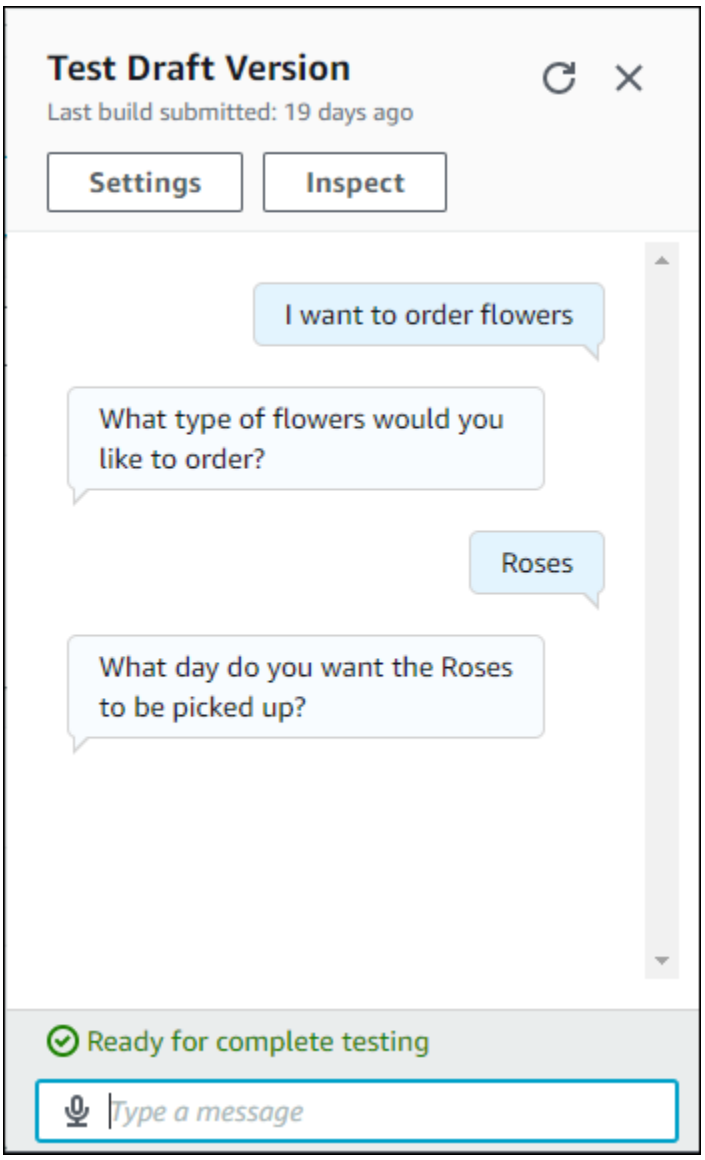
Il secondo tipo, test completo, ti consente di testare il tuo bot utilizzando frasi relative agli enunciati che hai configurato. Ad esempio, puoi usare la frase «Posso ordinare fiori» per iniziare una conversazione con il tuo bot.

Esegui il test di un bot utilizzando un alias e una lingua specifici. Se stai testando la versione di sviluppo del bot, usi l'`TestBotAlias`alias per il test.

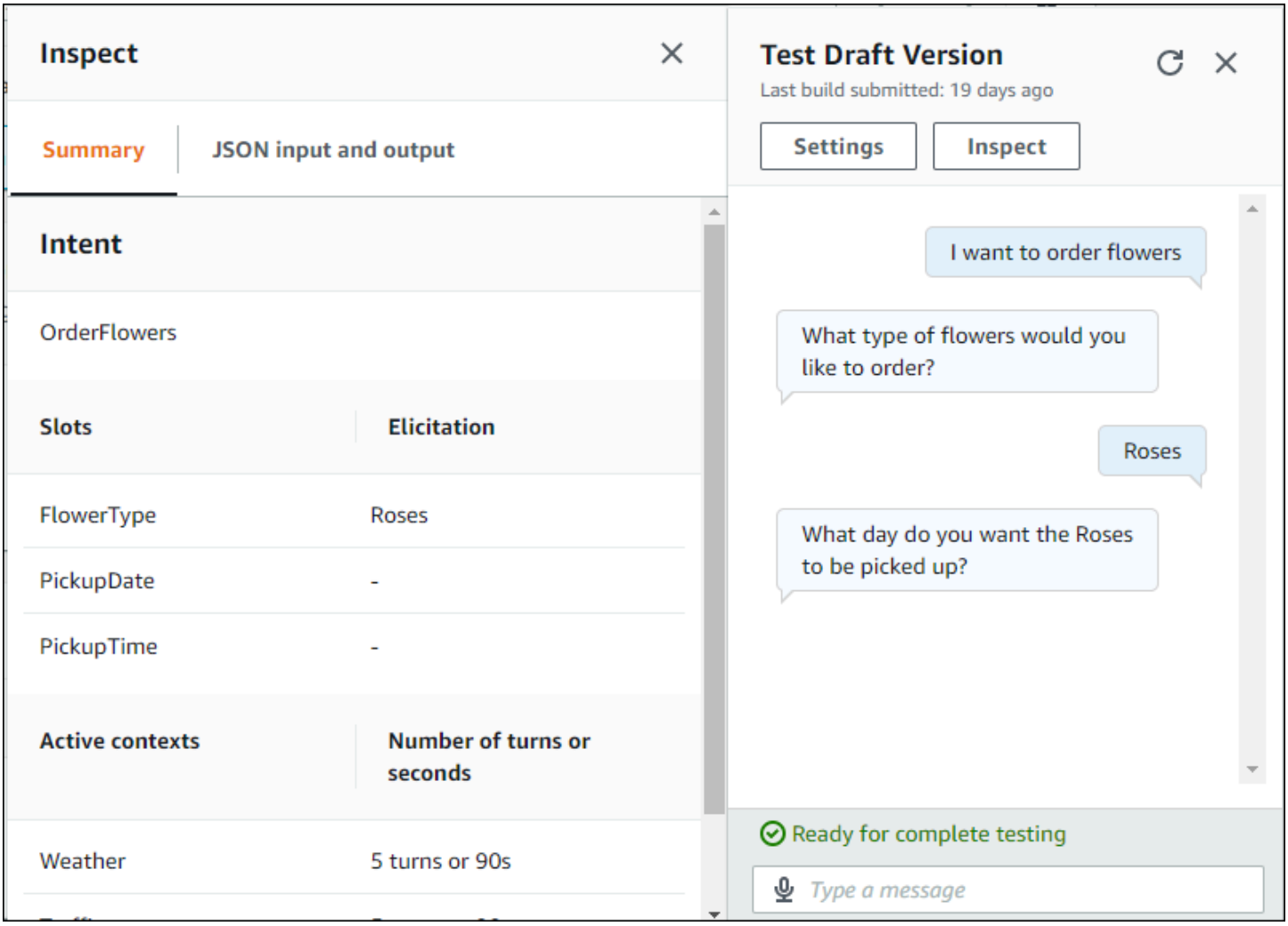
Per aprire la finestra di test

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot da testare dall'elenco dei bot.
3. Dal menu a sinistra, scegli Alias.
4. Dall'elenco degli alias, scegli l'alias da testare.
5. In Lingue, scegli il pulsante di opzione della lingua da testare, quindi scegli Test.

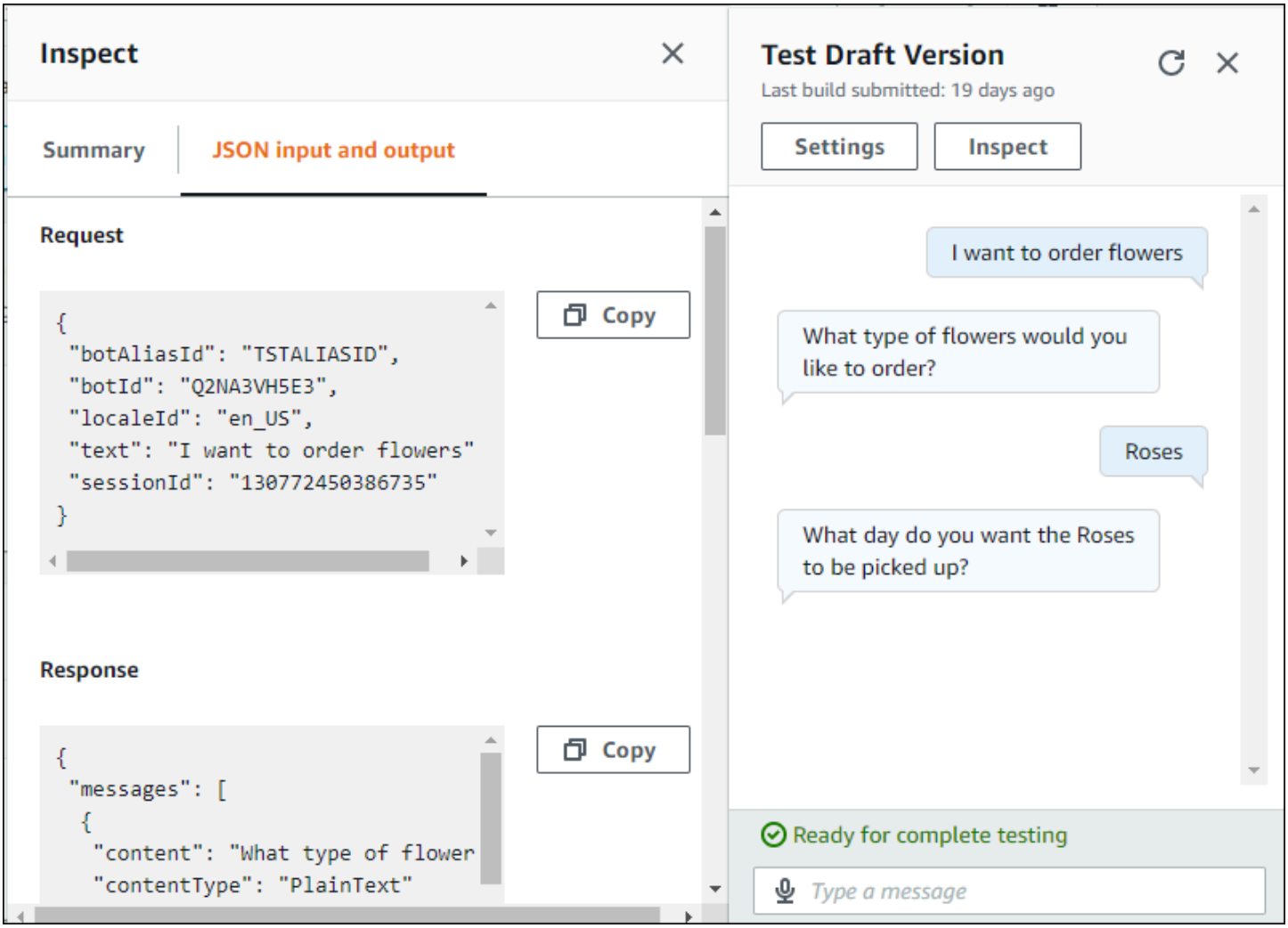
Dopo aver scelto Test, la finestra di test si apre nella console. Puoi utilizzare la finestra di test per interagire con il tuo bot, come mostrato nel seguente grafico.



Oltre alla conversazione, puoi anche scegliere Ispeziona nella finestra di test per vedere le risposte restituite dal bot. La prima visualizzazione mostra un riepilogo delle informazioni restituite dal bot alla finestra di test.



Puoi anche utilizzare la finestra di ispezione del test per vedere le strutture JSON inviate tra il bot e la finestra di test. Puoi visualizzare sia la richiesta dalla finestra di test che la risposta di Amazon Lex V2.


Inspect
Summary
JSON input and output
Request
{
"botAliasId": "TSTALIASID",
"botId": "Q2NA3VH5E3",
"localeId": "en_US",
"text": "I want to order flowers"
"sessionId": "130772450386735"
}
Copy
Response
{
"messages": [
{
"content": "What type of flower
"contentType": "PlainText"
Copy
Test Draft Version
Last build submitted: 19 days ago
Settings
Inspect
I want to order flowers
What type of flowers would you like to order?
Roses
What day do you want the Roses to be picked up?
Ready for complete testing
Type a message

Ottimizza la creazione e le prestazioni dei bot con l'IA generativa

> **Note**
>
> Queste funzionalità utilizzano l'intelligenza artificiale generativa. Quando utilizzi il servizio, ricorda che potrebbe fornire risposte imprecise o inappropriate. Per ulteriori informazioni, consulta la policy di AWS Responsible AI.
> Realizzato da Amazon Bedrock: AWS implementa il rilevamento automatico degli abusi. Poiché le funzionalità di intelligenza artificiale generativa di Amazon Lex V2 sono basate su Amazon Bedrock, gli utenti ereditano i controlli implementati in Amazon Bedrock per rafforzare la sicurezza e l'uso responsabile dell'IA.

Sfrutta le funzionalità di intelligenza artificiale generativa di Amazon Bedrock per automatizzare e velocizzare il processo di creazione di bot Amazon Lex V2. Puoi eseguire i seguenti processi con l'aiuto di Amazon Bedrock.

- Crea nuovi bot e popolali con intenti e tipi di slot pertinenti in modo efficiente utilizzando la descrizione in linguaggio naturale.
- Genera automaticamente esempi di enunciati per le intenzioni del tuo bot.
- Migliora le prestazioni di risoluzione degli slot dei tuoi bot.
- Crea l'intento di contribuire a rispondere alle domande dei tuoi clienti.

Puoi attivare funzionalità di intelligenza artificiale generativa per Amazon Lex V2 tramite la console o l'API.

> **Note**
>
> Prima di poter sfruttare le funzionalità di intelligenza artificiale generativa, devi soddisfare i seguenti prerequisiti
>
> 1. Accedi alla console Amazon Bedrock e registrati per accedere al modello Anthropic Claude che intendi utilizzare (per ulteriori informazioni, consulta Model access). Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock.

2. Attiva le funzionalità di intelligenza artificiale generativa per le impostazioni locali del tuo bot. Per farlo, segui i passaggi indicati inOttimizza la creazione e le prestazioni dei bot con l'IA generativa.

Using the console

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Seleziona il bot e le impostazioni locali nel bot per cui desideri attivare le funzionalità di intelligenza artificiale generativa.
3. Nella sezione Configurazioni AI generative, seleziona Configura.
4. Attiva il pulsante Abilitato per ogni funzionalità che desideri attivare. Selezionate il modello e la versione che desiderate utilizzare per quella feature. L'attivazione di una funzionalità può comportare costi aggiuntivi. Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock. Per ulteriori informazioni su una funzionalità, seleziona l'argomento corrispondente dall'elenco seguente. Seleziona Salva dopo aver attivato le funzionalità che desideri attivare. Viene visualizzato un banner verde di successo per confermare che le funzionalità sono attivate.

Using the API

1. Per abilitare le funzionalità di intelligenza artificiale generativa per un nuovo bot, utilizza l'CreateBotoperazione per creare un nuovo bot.
2. Invia una CreateBotLocalerichiesta, modificando l'`generativeAISettings`oggetto se necessario. Se stai abilitando le funzionalità di un bot esistente, invia invece una UpdateBotLocalerichiesta.

    a. Per abilitare l'utilizzo del bot builder descrittivo, modifica l'`descriptiveBotBuilder`oggetto. Specificate il modello di base da utilizzare nel `modelArn` campo e impostate il `enabled` valore su. `True`

    b. Per consentire il miglioramento della risoluzione degli slot, modificate l'`slotResolutionImprovement`oggetto. Specificate il modello di base da utilizzare nel `modelArn` campo e impostate il `enabled` valore su`True`.

c. Per abilitare la generazione di enunciati di esempio, modificate l'`sampleUtteranceGeneration`oggetto. Specificate il modello di base da utilizzare nel `modelArn` campo e impostate il `enabled` valore su. `True`

Argomenti

Utilizzo del bot builder descrittivo

> **Note**
>
> Prima di poter sfruttare le funzionalità di intelligenza artificiale generativa, è necessario soddisfare i seguenti prerequisiti
>
> 1. Accedi alla console Amazon Bedrock e registrati per accedere al modello Anthropic Claude che intendi utilizzare (per ulteriori informazioni, consulta Model access). Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock.
> 2. Attiva le funzionalità di intelligenza artificiale generativa per le impostazioni locali del tuo bot. Per farlo, segui i passaggi indicati inOttimizza la creazione e le prestazioni dei bot con l'IA generativa.

Il generatore di bot descrittivo ti consente di sfruttare l'accesso di Amazon Bedrock a modelli linguistici di grandi dimensioni per migliorare l'efficienza del processo di creazione dei bot. Fornisci un prompt utilizzando un linguaggio naturale che include lo scopo del bot e le azioni che deve eseguire. Amazon Lex V2 sfrutta le funzionalità di Amazon Bedrock per generare intenti e tipi di slot pertinenti per il tuo bot in base alla tua descrizione. Una volta scelti gli intenti e i tipi di slot che desideri conservare, puoi utilizzare il bot per modificarlo in base al tuo caso d'uso specifico. Il bot builder descrittivo ti fa risparmiare tempo evitando di dover creare manualmente intenti e tipi di slot per il bot.

Il bot builder descrittivo è disponibile nelle versioni locali inglesi (vedi le versioni locali che iniziano con nella tabella in). en_ Lingue e impostazioni locali supportate da Amazon Lex V2

Prima di creare il bot, procedi come segue.

1. Verifica che il tuo ruolo disponga delle autorizzazioni corrette esaminando i passaggi riportati inAutorizzazioni necessarie per creare un bot con descrizione in linguaggio naturale.
2. Decidi la descrizione da utilizzare. Puoi fare riferimento a Esempi di descrizioni dei bot per esempi di descrizioni dei bot.

Crea un bot utilizzando il linguaggio naturale per descrivere ciò che il bot dovrebbe essere in grado di fare. Amazon Lex V2 richiama i modelli Amazon Bedrock per generare intenti e tipi di slot adatti al caso d'uso del bot. Puoi creare il bot con la console o l'API.

Console

Crea un bot utilizzando il generatore di bot descrittivo

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Nella pagina Bot, seleziona Crea bot.
3. Per il metodo di creazione, scegli Descriptive Bot Builder - GenAI.
4. Assegna al bot un nome e una descrizione opzionale, configura le autorizzazioni IAM e scegli se il bot è soggetto al COPPA o meno. Quindi seleziona Avanti.
5. Seleziona una lingua in cui creare il bot, una voce per il bot e una soglia di confidenza per la classificazione degli intenti (per ulteriori informazioni, consultaUtilizzo dei punteggi di confidenza delle intenzioni.
6. In Descriptive Bot Builder - GenAI, fornisci una descrizione del bot che desideri creare. La tua descrizione deve essere dettagliata e precisa per aiutare a generare intenti appropriati e sufficienti per il tuo bot. Includi un elenco di azioni per migliorare il processo di creazione degli intenti.
7. Seleziona un fornitore di modelli e un modello in Seleziona modello.
8. Per creare il bot in un'altra lingua, scegli Aggiungi un'altra lingua. Quando hai finito di aggiungere le lingue, seleziona Fine. Amazon Lex V2 crea il tuo bot e il generatore descrittivo di bot genera intenti e slot per esso. Una volta generate le impostazioni locali, il banner passa dal blu al verde. Seleziona Review per vedere gli intenti generati e i tipi di slot.

> **Note**
>
> Il bot builder descrittivo è attualmente disponibile solo nelle versioni locali inglesi. Tuttavia, è possibile copiare un bot in una versione locale diversa dall'inglese dopo averlo creato.

Controlla gli intenti e i tipi di slot generati e aggiungili al tuo bot

1. Se ci sono abbastanza intenti e tipi di slot adatti al caso d'uso del bot, puoi esaminare gli intenti generati.
   a. Rivedi gli intenti generati.
      i. Scegli una casella di controllo accanto a un intento per rimuoverlo dall'elenco degli intenti da aggiungere al bot.
      ii. Scegli il nome dell'intento per visualizzare gli enunciati di esempio e gli slot generati per l'intento.
      iii. Per impostazione predefinita, sono selezionati tutti gli enunciati e gli slot. Scegliete una casella di controllo per rimuovere quell'elemento dall'intento. Seleziona Aggiungi alla selezione per mantenere gli elementi selezionati nell'intento.
   b. Esamina i tipi di slot generati.
      i. Scegli una casella di controllo accanto a un tipo di slot per rimuoverlo dall'elenco degli intenti da aggiungere al bot.
      ii. Puoi aggiungere valori a un tipo di slot dopo averlo aggiunto al bot
2. Quando sei soddisfatto delle tue intenzioni e dei tipi di slot, seleziona Aggiungi intenti e tipi di slot nella parte superiore della pagina per aggiungere gli intenti e i tipi di slot al tuo bot.
3. Al termine dell'aggiunta delle risorse, viene visualizzato un banner verde di successo. Vai a Intents e Slot types per modificare quelli generati e aggiungere altri valori.
4. Se i tipi di slot Generated Intents e Generated sono per lo più inapplicabili al bot che desideri creare, procedi nel seguente modo.
   a. Seleziona Nuova generazione nella sezione dei dettagli del generatore di bot descrittivi.
   b. Riscrivi il prompt e seleziona Rigenera per generare nuovi intenti e tipi di slot. I risultati sono diversi se si utilizza un modello diverso.

> **Important**
>
> Non vi è alcuna garanzia che vengano generati gli stessi intenti e gli stessi slot. Ti viene addebitato un importo ogni volta che rigeneri gli intenti e i tipi di slot.

API

Crea il bot usando una descrizione in linguaggio naturale

Quando utilizzi il generatore descrittivo di bot tramite l'API, crea una definizione di bot in un file.zip in un bucket Amazon S3. Scarica questo file e importa la definizione del bot in Amazon Lex V2 per creare il tuo bot.

1. Invia una CreateBotrichiesta per creare un nuovo bot. Quindi invia una CreateBotLocalerichiesta per creare una versione locale per il bot.
2. Invia una StartBotResourceGenerationrichiesta, specificando l'ID, la versione e le impostazioni locali del bot. Puoi usare `DRAFT` per la versione bot. Fornisci la tua richiesta nel `generationInputPrompt` campo. La tua descrizione deve essere dettagliata e precisa per aiutare a generare intenti appropriati e sufficienti per il tuo bot. Includi un elenco di azioni per migliorare il processo di creazione degli intenti.
3. Prendi nota di quanto indicato `generationId` nella risposta.
4. Invia una DescribeBotResourceGenerationrichiesta utilizzando il `generationId` codice che hai ricevuto nella `StartBotResourceGeneration` risposta. Includi l'ID del bot, la versione e le impostazioni locali.
5. Se `generationStatus` nella `DescribeBotResourceGeneration` risposta è`Complete`, anche il `generatedBotLocaleUrl` campo verrà compilato. Usa questo URI Amazon S3 per scaricare la definizione del bot seguendo la procedura descritta in Download di un oggetto.

Controlla la definizione del bot generata e importala

1. Utilizza l'URI di Amazon S3 contenuto `generationStatus` nella `DescribeBotResourceGeneration` risposta per scaricare la definizione del bot seguendo i passaggi riportati in Download di un oggetto.

2. Puoi modificare direttamente il contenuto generato per il caso d'uso specifico del tuo bot modificando il file. Puoi anche inviare un'altra `StartBotResourceGeneration` richiesta per rigenerare intent e slot.

   > **Important**
   >
   > Non vi è alcuna garanzia che vengano generati gli stessi intenti e gli stessi slot. Ti viene addebitato un importo ogni volta che rigeneri gli intenti e i tipi di slot.

3. Per importare la definizione del bot, segui i passaggi riportati in. Importing (Importazione)
4. Dopo l'importazione, è possibile modificare gli intenti e gli slot generati utilizzando le operazioni UpdateIntentUpdateSlot, e UpdateSlotType.

Per elencare i metadati relativi a tutti gli elementi generati per un bot locale, utilizzate l'operazione. ListBotResourceGenerations Utilizza uno qualsiasi dei `generationId` valori restituiti in una `DescribeBotResourcGeneration` richiesta per recuperare l'URI di Amazon S3 per una definizione di bot generata.

Argomenti

- Esempi di descrizioni dei bot
- Autorizzazioni necessarie per creare un bot con descrizione in linguaggio naturale

Esempi di descrizioni dei bot

Industry	Richiesta di esempio
Servizi finanziari	Siamo un servizio di carte finanziarie che aiuta gli utenti a svolgere alcune attività quando ricevono una nuova carta, ad esempio attivare la carta, inviare via e-mail o posta un PIN, verificare una nuova carta (utilizzando un codice postale). Li aiutiamo anche con le attività associate alla loro carta esistente, come richiedere informazioni sui vantaggi della carta di credito, segnalare uno smarrimento della

Industry	Richiesta di esempio
	carta, richiedere una nuova carta, reimpostare il PIN di una carta o pagare una fattura.
Servizi di ristorazione	Voglio un bot che aiuti i clienti a ordinare il cibo (utilizzando l'ID dell'articolo, la quantità, le dimensioni), a controllare lo stato dell'ordi ne e ad annullare un ordine. Usa Order ID per indicizzare gli ordini.
Linea aerea	Siamo un dominio di una compagnia aerea che aiuta gli utenti a prenotare biglietti aerei, controllare i dettagli di una prenotazione, ottenere la ricevuta di un volo prenotato, verificare lo stato del volo, riprogrammare i voli prenotati, ottenere i dettagli del volo e cancellar e i voli prenotati. Puoi anche generare intenti aggiuntivi se aiutano a supportare le funzioni nella descrizione del dominio.
Assicurazione	Obiettivo: siamo una compagnia assicurativa che vende polizze assicurative per auto, casa e rendite vitalizia. Voglio un bot in grado di controllare lo stato del reclamo, presentare un reclamo, effettuare i pagamenti delle polizze e annullare una polizza. Utilizziamo policy_id e gli ultimi 4 di SSN per l'identificazione e la convalida dell'account. Mi aspetto che il bot abbia almeno i seguenti intenti e slot: authentic ation - policy_id, last4SSNTipo di policy: car, home, annuitypolicy status: verifica saldo, verifica data di scadenza, copertura degli assegni Effettua un pagamento: pagamento una tantum, rate, importo

Industry	Richiesta di esempio
Gestione del veicolo	Stiamo costruendo un bot Towed Cars Lookup che aiuta i conducenti di una città la cui auto è stata rimorchiata a trovare dove si trova l'auto. Questo bot dovrebbe chiedere l'indirizzo o il luogo da cui è stata rimorchiata l'automob ile e dettagli sul veicolo come targa e marca, modello e anno dell'auto. Il bot dovrebbe rispondere indicando l'ubicazione del parcheggi o trainato e gli orari di apertura.
Viaggia	Sono un agente di viaggi e voglio un bot che aiuti i miei clienti a prenotare un viaggio a Disney. Disney ha diversi parchi in tutto il mondo tra cui scegliere e dispone anche di hotel, ristoranti e intrattenimenti speciali che possono essere prenotati. Gli utenti del bot dovrebbero essere in grado di modificare o cancellare la propria prenotazione. Le prenotazi oni devono includere almeno il parco, le date e l'hotel. L'inclusione della ristorazione o dell'intr attenimento è facoltativa e può essere aggiunta o modificata in un secondo momento.

Autorizzazioni necessarie per creare un bot con descrizione in linguaggio naturale

- Per accedere a questa funzionalità sulla console Amazon Lex V2, assicurati che il ruolo della console sia `bedrock:ListFoundationModels` autorizzato.
- Il ruolo IAM associato al bot deve essere `bedrock:InvokeModel` autorizzato. Quando abiliti la funzionalità con la console Amazon Lex, la policy verrà aggiunta automaticamente al ruolo bot a condizione che il bot utilizzi un ruolo collegato al servizio generato da Amazon Lex.

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

```
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
      }
  ]
}
```

Generazione di enunciati

> **Note**
>
> Prima di poter sfruttare le funzionalità di intelligenza artificiale generativa, devi soddisfare i seguenti prerequisiti
>
> 1. Accedi alla console Amazon Bedrock e registrati per accedere al modello Anthropic Claude che intendi utilizzare (per ulteriori informazioni, consulta Model access). Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock.
> 2. Attiva le funzionalità di intelligenza artificiale generativa per le impostazioni locali del tuo bot. Per farlo, segui i passaggi indicati inOttimizza la creazione e le prestazioni dei bot con l'IA generativa.

Utilizzate la generazione di enunciati per automatizzare la creazione di enunciati di esempio per il vostro intento. Invece di inserire manualmente enunciati di esempio, Amazon Lex V2 genera enunciati di esempio in base al nome dell'intento, alla descrizione e agli enunciati di esempio esistenti, in modo da ridurre il tempo e lo sforzo necessari per scoprire e scrivere i propri enunciati di esempio. Dopo che Amazon Lex V2 ha generato gli enunciati, puoi modificarli ed eliminarli. Usa questo strumento per accelerare la creazione di enunciati di esempio per il processo di riconoscimento degli intenti.

Per consentire la generazione di enunciati, segui i passaggi indicati Ottimizza la creazione e le prestazioni dei bot con l'IA generativa per attivare le funzionalità di intelligenza artificiale generativa.

Puoi generare enunciati con la console o l'API.

Console

1. Vai alla sezione Sample utterances di qualsiasi intento nel tuo bot (nel Visual Conversation Builder, si trova nel blocco Start).
2. Seleziona il pulsante Genera enunciati per generare 5 enunciati di esempio. Se l'intento contiene più di 25 enunciati di esempio, il pulsante Genera enunciati viene disabilitato.
3. Gli enunciati generati vengono visualizzati con un banner verde che differenzia gli enunciati generati dagli enunciati esistenti.
4. Passa il mouse su un enunciato per visualizzare le opzioni per modificare, eliminare e ordinare gli enunciati generati.

API

1. Inviate una GenerateBotElementrichiesta, inserendo l'ID dell'intento e del bot, la versione e la lingua per cui desiderate generare enunciati di esempio.
2. La risposta restituisce un elenco di SampleUtteranceoggetti, ognuno dei quali contiene un enunciato generato.
3. Per aggiungere gli enunciati all'intento, inviate una UpdateIntentrichiesta e aggiungete gli enunciati al campo. `sampleUtterances`

Argomenti

- Autorizzazioni per la generazione di enunciati

Autorizzazioni per la generazione di enunciati

Per accedere a questa funzionalità sulla console Amazon Lex V2, assicurati che la console disponga del ruolo `bedrock:ListFoundationModels` e delle `bedrock:InvokeModel` autorizzazioni.

Utilizzo della risoluzione assistita degli slot

> **Note**
>
> Prima di poter sfruttare le funzionalità di intelligenza artificiale generativa, è necessario soddisfare i seguenti prerequisiti

1. Accedi alla console Amazon Bedrock e registrati per accedere al modello Anthropic Claude che intendi utilizzare (per ulteriori informazioni, consulta Model access). Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock.
2. Attiva le funzionalità di intelligenza artificiale generativa per le impostazioni locali del tuo bot. Per farlo, segui i passaggi indicati inOttimizza la creazione e le prestazioni dei bot con l'IA generativa.

Puoi migliorare la precisione di alcuni slot incorporati nel flusso di conversazione del tuo bot utilizzando la risoluzione assistita degli slot. La risoluzione assistita degli slot utilizza Amazon Bedrock Large Language Models (LLM) per migliorare il riconoscimento di alcuni slot integrati, il che si traduce in una migliore interpretazione delle risposte dei clienti durante l'elezione degli slot. Per gli enunciati che non possono essere risolti normalmente, Amazon Lex tenterà di risolverli una seconda volta utilizzando Amazon Bedrock.

La risoluzione assistita degli slot consente di utilizzare la potenza dei modelli Amazon Bedrock Foundation per migliorare la precisione dei seguenti slot integrati:

- `AMAZON.Alphanumeric`senza supporto per regex
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

È possibile abilitare la risoluzione assistita degli slot per qualsiasi scopo che utilizzi gli slot integrati sopra elencati. La risoluzione assistita degli slot non si applica agli slot personalizzati o agli slot integrati di Amazon non elencati sopra.

Puoi raccogliere dati sui miglioramenti della precisione dopo aver abilitato la risoluzione assistita degli slot nel tuo bot Amazon Lex utilizzando i log e le metriche delle conversazioni.

- Registri delle conversazioni: le interpretazioni avranno lo `interpretationSource` stesso valore`Bedrock`, se per risolvere lo slot fosse stato utilizzato Amazon Bedrock.

- CloudWatch metriche: le metriche verranno pubblicate secondo le dimensioni elencate nella metrica. CloudWatch Per ulteriori informazioni, consulta Monitoraggio di Amazon Lex con Amazon CloudWatch.

Per utilizzare il generatore di bot descrittivo, assicurati che il tuo ruolo IAM disponga delle autorizzazioni appropriate seguendo i passaggi riportati di seguito. Autorizzazioni per la risoluzione assistita degli slot

Argomenti

- Esempi di risoluzione assistita degli slot
- Abilita la risoluzione assistita degli slot nella schermata di configurazione dell'IA generativa
- Abilita la risoluzione assistita degli slot nelle impostazioni degli slot
- Autorizzazioni per la risoluzione assistita degli slot

Esempi di risoluzione assistita degli slot

Di seguito sono riportati alcuni esempi in cui la risoluzione assistita degli slot è in grado di risolvere in modo intelligente le espressioni dell'utente in un valore.

Amazon.number

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
Viaggio	Numero Amazon	numberOfNightsRimasto	Quante notti hai soggiornato durante il viaggio?	Un'intera settimana, 7 notti.	7
Servizi bancari	Amazon. Numero	numberOfPeopleOnTheAccount	Quante persone hanno l'account?	Io e mia moglie.	2
Viaggiare	Numero Amazon	numberOfStops	Quante fermate?	Una volta in Giappone.	2

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
				Una volta a Los Angeles.	

AMAZON. AlphaNumeric

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
Autonoleggio	Amazon.alfanumerico	ID transazione	Qual è l'ID della tua transazione?	Credo fosse Alpha Whisky Echo Echo Eight Three Four Nine Romeo Juliet.	AWE8349RJ
Viaggio	Amazon.alfanumerico	Codice di conferma	Qual è il numero di conferma della tua prenotazione?	Il numero di conferma è BLT2UE.	BLT2UE

Amazon.date

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto	Data corrente
Autonoleggio	Amazon.date	Data di scadenza	Quando scadrà il contratto di noleggio?	Il contratto di locazione scade il 1° del mese prossimo.	01/12/2023	2023-11-09

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto	Data corrente
Viaggio	Amazon.date	Data di restituzione	Quando tornerai?	Più tardi oggi verso le 7.	2023-11-09	2023-11-09

AMAZZONE. PhoneNumber

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
Assicurazione	AMAZON. PhoneNumber	Titolare della polizza	Qual è il numero di telefono del titolare della polizza?	Il numero di telefono del titolare della polizza è 123-456-7890.	1234567890
Vendita al dettaglio	AMAZON. PhoneNumber	Ricerca telefonica	Qual è il tuo numero di telefono per poter trovare il tuo account?	Penso che sia inferiore al 413-570-9617, fammi ricontrollare.	4135709617

Amazon.paese

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
Viaggio	Amazon.paese	Paese nativo	Qual è il tuo paese di origine?	Sono indiano.	India
Servizi bancari	Amazon.paese	Itinerario nazionale	In quali paesi viaggerai con	Viaggerò a Nuova Delhi.	India

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
			la tua carta di debito?		

Amazon.city

Vertical	Tipo di slot	Intento	Domanda	Risposta	Valore risolto
Assicurazione	Amazon.city	policyHolderCity	In quale città risiede il titolare della polizza?	Vivo a Springfield.	Springfield
Viaggio	Amazon.city	Città di destinazione	In quale città stai viaggiando?	Sto andando a Tokyo.	Tokyo

Amazon. Conferma

Vertical	Tipo di slot	slotName	SlotPrompt	enunciato	Valore risolto
Assicurazione	Amazon. Conferma	Polizza scaduta	La polizza assicurativa è scaduta?	Sì, purtroppo è scaduta.	Sì
Servizi bancari	Amazon. Conferma	Ha investimenti	Hai degli investimenti?	Non ho ancora investito in niente.	No

Abilita la risoluzione assistita degli slot nella schermata di configurazione dell'IA generativa

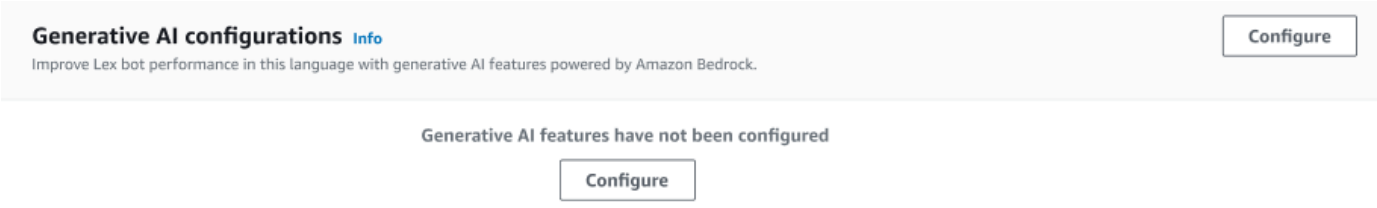
Puoi abilitare la risoluzione assistita degli slot integrati per gli slot integrati supportati accedendo alla schermata Generative AI.

Se lo slot è uno slot integrato supportato, avrai la possibilità di attivare la risoluzione assistita dello slot a livello di slot.

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Nel riquadro di navigazione sotto Bots, seleziona il bot che desideri utilizzare per la risoluzione assistita degli slot.
3. Seleziona la lingua inglese (Stati Uniti) per il bot che desideri abilitare.
4. Vai alla sezione Configurazione dell'IA generativa sullo schermo.
5. Seleziona Vai ad Amazon Bedrock per registrarti e abilitare la funzionalità, se la funzionalità non è stata abilitata.

   > **Note**
   >
   > Se non hai accesso ai modelli Amazon Bedrock Foundation, dovresti vedere Vai ad Amazon Bedrock. Fai clic su Vai ad Amazon Bedrock per accedere alla pagina Amazon Bedrock dove puoi registrarti per accedere ai modelli Foundation. La risoluzione assistita degli slot attualmente supporta Claude V2 e Claude Instant V1. Suggeriamo di utilizzare Claude V2 per ottenere i migliori risultati.

6. Se hai già accesso ai modelli Bedrock Foundation, dovresti vedere un pulsante Configura. Fai clic su questo pulsante per accedere alla pagina di configurazione dell'IA generativa e attivare le funzionalità di intelligenza artificiale generativa in Lex.



7. Nell'angolo in alto a destra della casella, sposta il cursore verso destra per scegliere l'impostazione Abilitato.

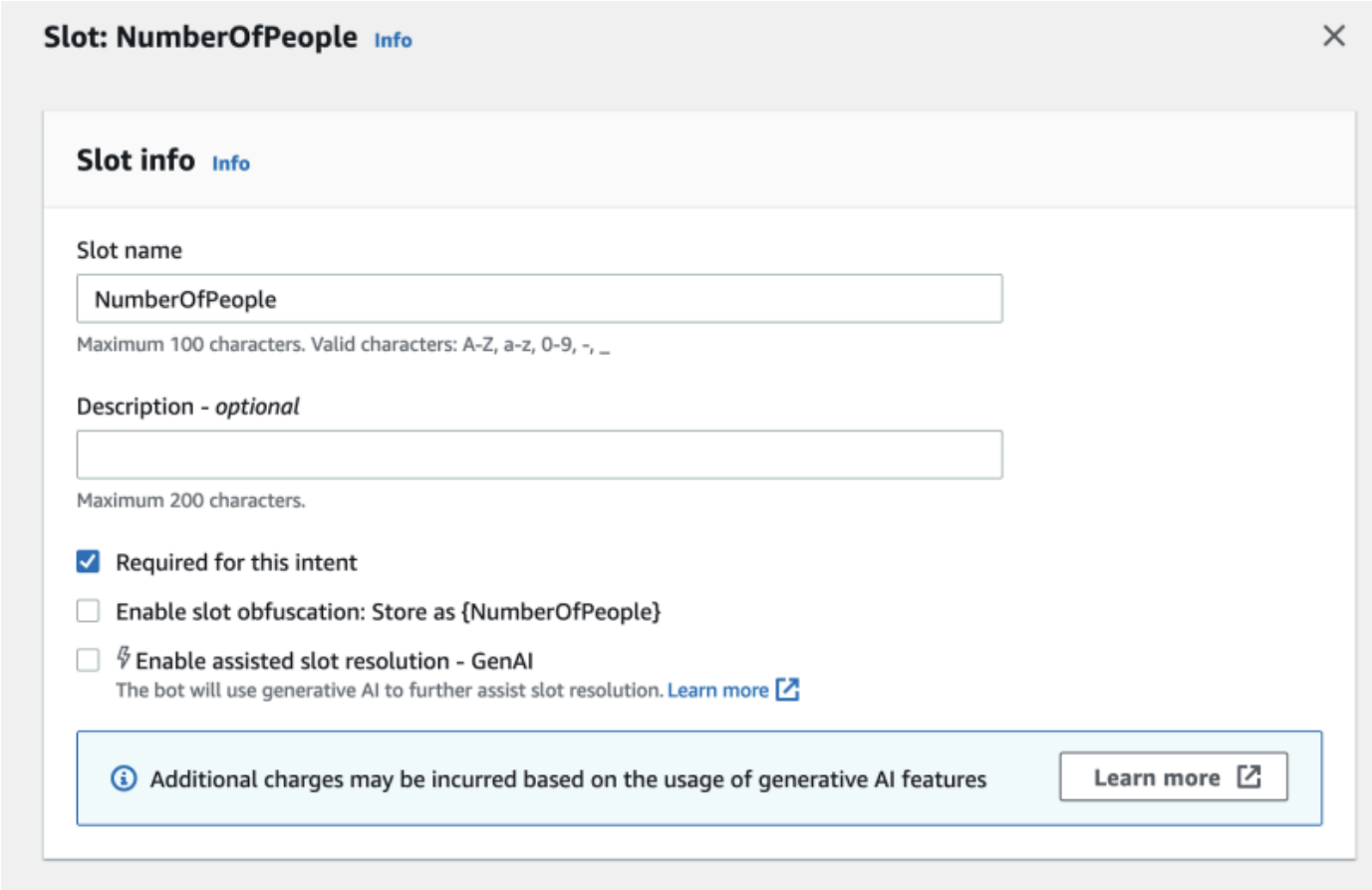
8. Scegliete il pulsante Abilita per attivare la risoluzione assistita degli slot per gli slot selezionati.
9. È possibile disattivare la risoluzione assistita degli slot selezionando gli slot dall'elenco e selezionando il pulsante Disabilita.

Abilita la risoluzione assistita degli slot nelle impostazioni degli slot

È possibile abilitare la risoluzione assistita degli slot integrati supportati accedendo al livello di slot per ogni intent dotato di slot. Gli slot devono essere uno degli slot integrati supportati elencati sopra per avere la possibilità di attivare la risoluzione assistita degli slot. Se lo slot non dispone della possibilità di attivare la risoluzione assistita degli slot, l'opzione sarà disattivata.

> **Note**
>
> È necessario prima attivare la funzione di risoluzione assistita degli slot sul pannello Generative AI per attivare la funzione per i singoli slot.

1. Accedi alla Console di gestione AWS e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Nel riquadro di navigazione sotto Bots, seleziona il bot che desideri utilizzare per la risoluzione assistita degli slot.
3. In Tutte le lingue, seleziona Inglese (Stati Uniti) per espandere l'elenco.
4. Nel pannello laterale sinistro, scegli Intenti per visualizzare un elenco di intenti nel bot selezionato.
5. Nella schermata Intenti, scegli l'intento che contiene gli slot che desideri modificare.
6. Selezionate il nome dell'intento per visualizzare gli slot relativi a quell'intento.
7. Seleziona il pulsante Opzioni avanzate nella sezione Slot.
8. Seleziona la casella di controllo Abilita la risoluzione assistita degli slot per abilitare la funzione.



9. Scegli il pulsante Update Slot nell'angolo in basso a destra dello schermo. Ciò attiverà la risoluzione assistita degli slot per gli slot che hai scelto.

Puoi abilitare la risoluzione assistita degli slot integrati per gli slot integrati supportati effettuando chiamate API.

- Segui i passaggi indicati Ottimizza la creazione e le prestazioni dei bot con l'IA generativa per abilitare la risoluzione assistita degli slot per le impostazioni locali del bot.
- Inviate una UpdateSlotrichiesta, specificando lo slot per il quale desiderate abilitare la risoluzione assistita degli slot. Nel `slotResolutionSetting` campo, imposta il `slotResolutionStrategy` valore come. `EnhancedFallback` Per creare un nuovo slot con la risoluzione assistita degli slot abilitata, invia invece una CreateSlotrichiesta.

Autorizzazioni per la risoluzione assistita degli slot

- Per accedere a questa funzionalità sulla console Amazon Lex V2, assicurati che il ruolo della console sia `bedrock:ListFoundationModels` autorizzato.

- Il ruolo IAM associato al bot deve essere `bedrock:InvokeModel` autorizzato. Quando abiliti la funzionalità con la console Amazon Lex, la policy verrà aggiunta automaticamente al ruolo bot a condizione che il bot utilizzi un ruolo collegato al servizio generato da Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:Region::foundation-model/modelId"
          ]
      }
  ]
}
```

Amazon.qnaIntent

> **Note**
>
> Prima di poter sfruttare le funzionalità di intelligenza artificiale generativa, devi soddisfare i seguenti prerequisiti
>
> 1. Accedi alla console Amazon Bedrock e registrati per accedere al modello Anthropic Claude che intendi utilizzare (per ulteriori informazioni, consulta Model access). Per informazioni sui prezzi per l'utilizzo di Amazon Bedrock, consulta i prezzi di Amazon Bedrock.
> 2. Attiva le funzionalità di intelligenza artificiale generativa per le impostazioni locali del tuo bot. Per farlo, segui i passaggi indicati inOttimizza la creazione e le prestazioni dei bot con l'IA generativa.

Puoi sfruttare i FM di Amazon Bedrock per rispondere alle domande dei clienti in una conversazione con i bot. Amazon Lex V2 offre una funzionalità integrata `AMAZON.QnAIntent` che puoi aggiungere al tuo bot. Questo intento sfrutta le funzionalità di intelligenza artificiale generativa di Amazon

Bedrock riconoscendo le domande dei clienti e cercando una risposta nei seguenti knowledge store (ad esempio,). **`Can you provide me details on the baggage limits for my international flight?`** Questa funzionalità riduce la necessità di configurare domande e risposte utilizzando un dialogo orientato alle attività all'interno degli intenti di Amazon Lex V2. Questo intento riconosce anche le domande successive (ad esempio**`What about domestic flight?`**) in base alla cronologia delle conversazioni e fornisce la risposta di conseguenza.

Assicurati che il tuo ruolo IAM disponga delle autorizzazioni appropriate per accedere a `AMAZON.QnAIntent` seguendo i passaggi riportati di seguito. Autorizzazioni per AMAZON.QnAIntent

Per usufruire di, `AMAZON.QnAIntent` è necessario aver configurato uno dei seguenti knowledge store.

- Database Amazon OpenSearch Service: per ulteriori informazioni, consulta Creazione e gestione di domini Amazon OpenSearch Service.
- Indice Amazon Kendra: per ulteriori informazioni, consulta Creazione di un indice.
- Knowledge base Amazon Bedrock: per ulteriori informazioni, consulta Creazione di una knowledge base.

Puoi configurarlo `AMAZON.QnAIntent` in due modi:

Per eseguire la configurazione utilizzando configurazioni di intelligenza artificiale generativa

1. Nella console Amazon Lex V2, seleziona Bot dal riquadro di navigazione a sinistra e scegli il bot per cui desideri aggiungere l'intento dalla sezione Bot.
2. Dal riquadro di navigazione a sinistra, seleziona la lingua per la quale desideri aggiungere l'intento.
3. Nella sezione Configurazioni AI generative, seleziona Configura.
4. Nella sezione Configurazioni QnA, seleziona Crea intento QnA.

Da configurare aggiungendo un intento integrato al bot

1. Nella console Amazon Lex V2, seleziona Bot dal riquadro di navigazione a sinistra e scegli il bot per cui desideri aggiungere l'intento dalla sezione Bot.
2. Dal riquadro di navigazione a sinistra, seleziona Intents nella lingua per cui desideri aggiungere l'intento.

3. Seleziona Aggiungi intento e scegli Usa intento integrato dal menu a discesa.
4. Per maggiori dettagli sulle configurazioni per, consulta. `AMAZON.QnAIntent` AMAZON.QnAIntent

> **Note**
>
> `AMAZON.QnAIntent`Si attiva quando un enunciato non è classificato in nessuno degli altri intenti presenti nel bot. Questo intento si attiva quando un'enunciazione non è classificata in nessuno degli altri intenti presenti nel bot. Nota che questo intento non verrà attivato in caso di enunciazioni perse quando si ottiene un valore di slot. Una volta riconosciuto, `AMAZON.QnAIntent` utilizza il modello Amazon Bedrock specificato per effettuare ricerche nella knowledge base configurata e rispondere alla domanda del cliente.

Argomenti

- Autorizzazioni per AMAZON.QnAIntent

Autorizzazioni per AMAZON.QnAIntent

Per accedere a questa funzionalità sulla console Amazon Lex V2, assicurati che il ruolo della console disponga delle `bedrock:ListFoundationModels` autorizzazioni.

Il ruolo IAM associato al bot dovrebbe avere le seguenti autorizzazioni richieste per. `AMAZON.QnAIntent` Il ruolo bot deve avere le autorizzazioni per le chiamate. `bedrock:InvokeModel` È inoltre necessario allegare un'istruzione per ogni archivio di dati specificato nei bot `AMAZON.QnAIntent` (vedere le `Permissions to access knowledge base in Amazon Bedrock` istruzioni `Permissions to access Amazon Kendra index`Permissions to access `OpenSearch Service index`, e nella politica riportata di seguito). Quando abiliti la funzionalità con la console Amazon Lex, le policy verranno automaticamente aggiunte al ruolo bot a condizione che il bot utilizzi un ruolo collegato al servizio generato da Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "Permissions to invoke Amazon Bedrock foundation models",
```

```
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:region::foundation-model/model-id"
            ]
        },
        {
          "Sid": "Permissions to access Amazon Kendra index",
          "Effect": "Allow",
          "Action": [
              "kendra:Query",
              "kendra:Retrieve"
          ],
          "Resource": [
              "arn:aws:kendra:region:account-id:index/kendra-index"
          ]
       },
       {
          "Sid": "Permissions to access OpenSearch Service index",
          "Effect": "Allow",
          "Action": [
              "es:ESHttpGet",
              "es:ESHttpPost"
          ],
          "Resource": [
              "arn:aws:es:region:account-id:domain/domain-name/index-name/_search"
          ]
       },
       {
          "Sid": "Permissions to access knowledge base in Amazon Bedrock",
          "Effect": "Allow",
          "Action": [
              "bedrock:Retrieve"
          ],
          "Resource": [
              "arn:aws:bedrock:region:account-id:knowledge-base/knowledge-base"
          ]
       }
    ]
}
```

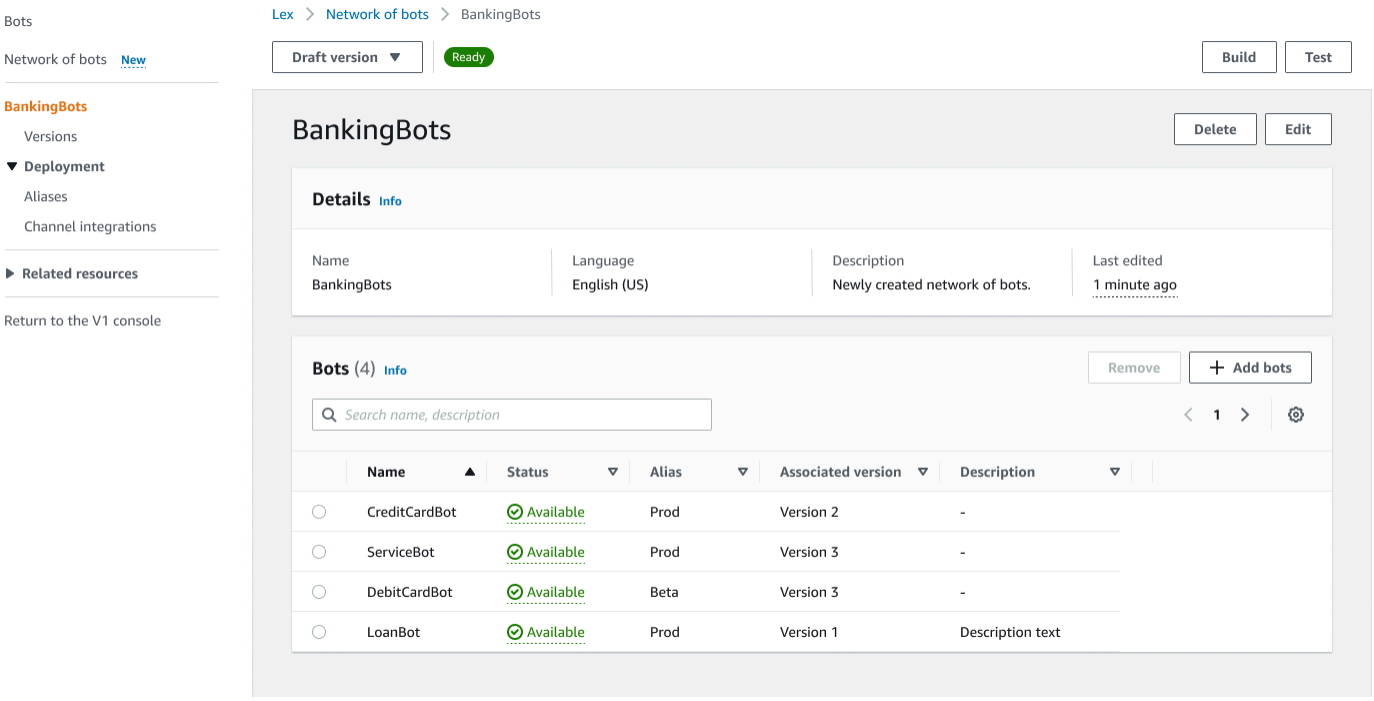
Creare una rete di bot

Network of Bots consente alle aziende di offrire un'esperienza utente unificata su più bot. Con Network of Bots, le aziende possono aggiungere più bot a un'unica rete per consentire una gestione flessibile e indipendente del ciclo di vita dei bot. La rete espone un'unica interfaccia unificata all'utente finale e indirizza la richiesta al bot appropriato in base all'input dell'utente.

I team possono collaborare per creare una rete di bot in grado di soddisfare diverse esigenze aziendali gestendo e aggiungendo bot alla rete man mano che i bot migliorati vengono implementati in produzione. Gli sviluppatori possono semplificare e velocizzare l'implementazione e i miglioramenti integrando più bot in un'unica rete.

La rete di bot è attualmente disponibile solo nella lingua en-USA.

> **Note**
>
> Attualmente, una rete di bot è limitata a un account. Non puoi aggiungere bot da altri account.



Crea una rete di bot

Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home. Scegli Rete di bot dal menu laterale. Devi aver creato almeno un bot per creare una rete di bot.

Passaggio 1: configurare la rete di impostazioni dei bot

1. Nella sezione Dettagli, inserisci il nome della tua rete e fornisci una descrizione facoltativa.
2. Nella sezione Autorizzazioni IAM, scegli un ruolo AWS Identity and Access Management (IAM) che fornisca ad Amazon Lex V2 l'autorizzazione per accedere ad altri AWS servizi, come Amazon. CloudWatch Puoi fare in modo che Amazon Lex V2 crei il ruolo oppure puoi scegliere un ruolo esistente con CloudWatch autorizzazioni. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Lex V2.
3. Nella sezione Children's Online Privacy Protection Act (COPPA), scegli la risposta appropriata. DataPrivacyPer ulteriori informazioni, consulta.
4. Nella sezione Timeout della sessione di inattività, scegli la durata in cui Amazon Lex V2 mantiene aperta una sessione con un utente. Amazon Lex V2 mantiene le variabili di sessione per tutta la durata della sessione in modo che il bot possa riprendere una conversazione con le stesse variabili. Per ulteriori informazioni, vedere Impostazione del timeout della sessione.
5. Nella sezione Aggiungi impostazioni della lingua, scegli una voce per far interagire il tuo bot con gli utenti. Puoi digitare una frase in Voice sample e selezionare Riproduci per ascoltare la voce.
6. Nella sezione Impostazioni avanzate, aggiungi facoltativamente dei tag che aiutano a identificare il bot. I tag possono essere usati per controllare l'accesso e monitorare le risorse. Per ulteriori informazioni, consulta Risorse per l'etichettatura.
7. Scegli Avanti per creare la rete di bot e passare all'aggiunta di bot.

Passaggio 2: aggiungi bot

1. Nella sezione Bot, seleziona + Aggiungi bot.
2. Apparirà una modalità Aggiungi bot. Scegli un bot da aggiungere dal menu a discesa Bot e l'alias del bot che desideri utilizzare dal menu a discesa Alias.

   L'alias deve fare riferimento a una versione numerata del bot e non alla versione bozza. Puoi aggiungere fino a 5 bot. Un bot può essere aggiunto a un massimo di 25 reti diverse.

3. Seleziona + Aggiungi bot per aggiungere altri bot alla tua rete. Per rimuovere un bot, seleziona Rimuovi accanto al bot che desideri rimuovere. Quando hai finito di aggiungere i bot, scegli Salva per chiudere la modalità.
4. Seleziona Salva per completare la creazione della rete.

Gestisci la tua rete di bot

Dopo aver creato la tua rete di bot, verrai indirizzato a una pagina in cui puoi gestire e costruire la tua rete. Oppure puoi raggiungere questa pagina selezionando Rete di bot nel menu laterale e scegliendo il nome della rete da gestire.

1. Per modificare le informazioni relative alla rete, seleziona Modifica sopra la sezione Dettagli. Per eliminare la rete, seleziona Elimina sopra la sezione Dettagli.
2. Nella sezione Bot, puoi aggiungere altri bot selezionando + Aggiungi bot. Puoi anche aggiungere bot se accedi alla pagina Bot nel menu laterale della console Amazon Lex V2. Attiva il pulsante di scelta accanto al bot che desideri aggiungere e seleziona Aggiungi a una rete di bot dal menu a discesa Azioni.

   Dal menu a discesa Rete di bot nella modalità che si apre, scegli la rete a cui desideri aggiungere il bot. Quindi scegli l'alias del bot che desideri utilizzare dal menu a discesa Alias Bot. Seleziona Aggiungi per aggiungere il bot alla rete che hai scelto.
3. Puoi rimuovere i bot dalla tua rete premendo il pulsante di opzione accanto a un bot e scegliendo Rimuovi.
4. Al termine della configurazione della rete, seleziona Crea in alto a destra per creare la tua rete. La costruzione potrebbe richiedere alcuni minuti. Se la build ha esito positivo, nella parte superiore della pagina viene visualizzato un banner verde di successo.
5. Una volta creata la rete, puoi selezionare Test in alto a destra per visualizzare una finestra di chat nell'angolo in basso a destra. Puoi utilizzare questa finestra di chat per conversare con i bot della tua rete e assicurarti che i flussi di conversazione e le transizioni siano configurati correttamente.

> **Note**
>
> Se aggiungi, rimuovi o aggiorni bot all'interno della tua rete, devi ricostruire la rete.

Versioni

Puoi creare diverse versioni della tua rete di bot. Per gestire le versioni, scegli la tua rete dal menu laterale della console Amazon Lex V2 e seleziona Versioni.

1. Seleziona Crea versione per creare una nuova versione della tua rete di bot. Puoi aggiungere una descrizione opzionale. Scegli Crea per creare la versione.
2. Quando attivi il pulsante di scelta accanto a una versione della tua rete di bot, puoi selezionare Associa alias alla versione per indirizzare un alias a questa versione.
3. Per gestire una versione della rete, seleziona il nome della versione nella sezione Versioni. Nella pagina seguente, puoi modificare i dettagli della versione e gestire i bot all'interno della versione e del relativo alias associato.

Alias

Puoi usare gli alias per implementare le tue reti. Per gestire gli alias, scegli la tua rete dal menu laterale della console Amazon Lex V2 e seleziona Alias.

1. Seleziona Crea alias per creare un nuovo alias.
2. Assegna un nome all'alias e una descrizione facoltativa nella sezione Dettagli alias. Puoi scegliere una versione per associare l'alias alla sezione Associa a una versione e aggiungere tag nella sezione Tag. Scegli Crea per creare l'alias.
3. Per gestire un alias per la tua rete, seleziona il nome dell'alias nella sezione Alias. Nella pagina seguente, puoi modificare i dettagli dell'alias e gestirne i tag, le integrazioni dei canali e la politica basata sulle risorse. Puoi anche visualizzare la cronologia della sua associazione con le versioni della rete.

Integrazioni di canali

Per integrare la tua rete di bot con una piattaforma di messaggistica, scegli la tua rete di bot dal menu laterale della console Amazon Lex V2. Quindi seleziona Integrazioni dei canali.

1. Seleziona Aggiungi canale per integrare la tua rete con un nuovo canale.
2. Nella sezione Piattaforma, scegli la piattaforma su cui vuoi distribuire il tuo bot in Seleziona piattaforma. Verrà creato un ruolo IAM. Scegli una chiave dal menu a discesa sotto la chiave KMS per proteggere le tue informazioni.

3. Nel canale di configurazione dell'integrazione, inserisci il nome e una descrizione opzionale. Scegli un alias dal menu a discesa.
4. Ottieni il SID del tuo account e il token di autenticazione dalla piattaforma e compila i campi SID dell'account e Token di autenticazione. Per ulteriori informazioni, consulta Integrazione dei bot.
5. Seleziona Crea per completare l'integrazione dei canali.

> **Note**
>
> La rete di bot non è attualmente disponibile nella chat o nella voce di Amazon Connect.

Implementazione di bot

Dopo aver creato e testato il bot, è pronto per l'implementazione per interagire con i tuoi clienti. In questa sezione, impara a creare versioni del tuo bot dopo aver effettuato un aggiornamento. Usa gli alias per indicare diverse versioni del bot quando sono pronte per la distribuzione. Scopri come integrare i bot con piattaforme di messaggistica, applicazioni mobili e siti Web.

Argomenti

Controllo delle versioni e alias

Amazon Lex V2 supporta la creazione di versioni e alias di bot e reti di bot in modo da poter controllare l'implementazione utilizzata dalle applicazioni client. Una versione funge da istantanea numerata del tuo lavoro. Puoi indicare un alias alla versione del bot che desideri rendere disponibile ai tuoi clienti. Tra una creazione e l'altra, puoi continuare ad aggiornare la `Draft` versione del tuo bot senza influire sull'esperienza utente.

Versioni

Amazon Lex V2 supporta la creazione di versioni di bot in modo da poter controllare l'implementazione utilizzata dalle applicazioni client. Una versione è un'istantanea numerata del tuo lavoro che puoi creare per utilizzarla in diverse parti del tuo flusso di lavoro, come sviluppo, distribuzione beta e produzione.

La versione bozza

Quando crei un bot Amazon Lex V2, esiste solo una versione, la `Draft` versione.

`Draft`è la copia funzionante del tuo bot. Puoi aggiornare solo la `Draft` versione e fino a quando non crei la tua prima versione, `Draft` è l'unica versione del bot che hai.

La `Draft` versione del tuo bot è associata a`TestBotAlias`. `TestBotAlias`Deve essere usato solo per test manuali. Amazon Lex V2 limita il numero di richieste di runtime che puoi effettuare all'`TestBotAlias`alias del bot.

Creazione di una versione

Quando crei una versione di un bot Amazon Lex V2, crei uno snapshot numerato del bot in modo da poter utilizzare il bot così com'era al momento della creazione della versione. Una volta creata una versione numerica, questa rimarrà invariata mentre continuerai a lavorare sulla versione bozza dell'applicazione.

Quando create una versione, potete scegliere le versioni locali da includere nella versione. Non è necessario scegliere tutte le versioni locali in un bot. Inoltre, quando si crea una versione, è possibile scegliere una versione locale da una versione precedente. Ad esempio, se disponi di tre versioni di un bot, puoi scegliere una locale dalla `Draft` versione e una dalla versione due quando crei la versione quattro.

Se eliminate una versione locale dalla `Draft` versione, questa non viene eliminata da una versione numerata.

Se una versione bot non viene utilizzata per sei mesi, Amazon Lex V2 contrassegnerà la versione come inattiva. Quando una versione è inattiva, non è possibile utilizzare le operazioni di runtime con il bot. Per rendere attivo il bot, ricostruisci tutte le lingue associate alla versione.

Aggiornamento di un bot Amazon Lex V2

Puoi aggiornare solo la `Draft` versione di un bot Amazon Lex V2. Le versioni non possono essere modificate. Puoi creare una nuova versione in qualsiasi momento dopo aver aggiornato una risorsa nella console o con l'CreateBotVersionoperazione.

Eliminazione di un bot o di una versione di Amazon Lex V2

Amazon Lex V2 supporta l'eliminazione di un bot o di una versione utilizzando la console o una delle operazioni API:

- DeleteBot
- DeleteBotVersion

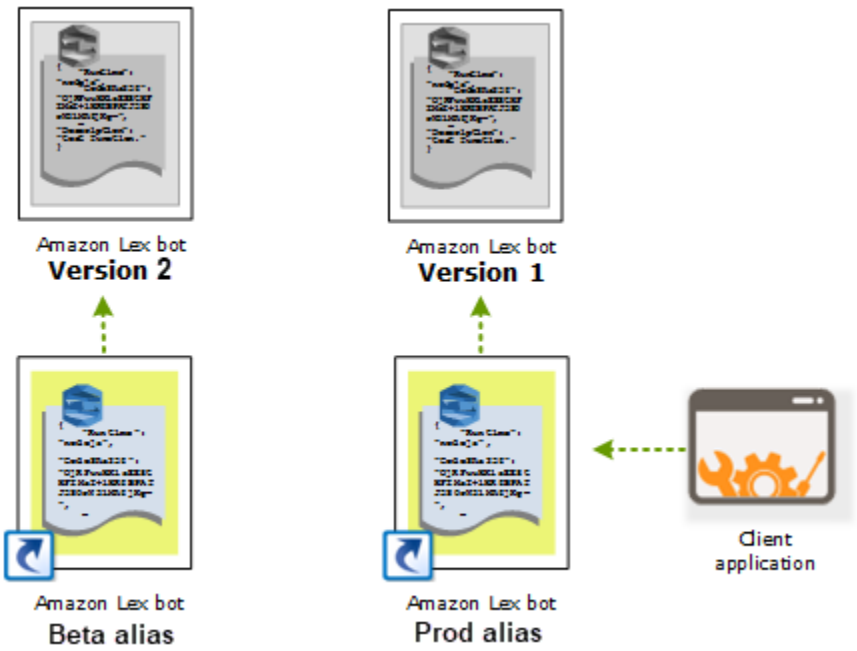
Alias

I bot Amazon Lex V2 supportano 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 sei pronto per aggiornare il bot, crei la versione 2 e modifichi 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.

Un alias è un puntatore a una versione specifica di un bot Amazon Lex V2. 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.

Quando crei un bot, Amazon Lex V2 crea un alias chiamato `TestBotAlias` che puoi usare per testare il bot. L'`TestBotAlias`alias è sempre associato alla `Draft` versione del bot. È necessario utilizzare l'`TestBotAlias`alias solo per i test, Amazon Lex V2 limita il numero di richieste di runtime che è possibile effettuare all'alias.

L'esempio seguente mostra due versioni di un bot Amazon Lex V2, 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 UpdateBotAlias 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.

Quando configuri le tue applicazioni client per chiamare le API Amazon Lex Runtime V2 per consentire ai clienti di interagire con il tuo bot, usi l'alias che indica la versione che desideri che i tuoi clienti utilizzino.

> **Note**
>
> Sebbene tu possa testare la `Draft` versione di un bot nella console, quando integri un bot con la tua applicazione client, ti consigliamo di creare prima una versione e creare un alias che punti a quella versione. Utilizzare l'alias nell'applicazione client per i motivi illustrati in questa sezione. Quando aggiorni un alias, Amazon Lex V2 utilizzerà la versione corrente per tutte le sessioni in corso. Le nuove sessioni utilizzano la nuova versione.

Utilizzo di un'applicazione Java per interagire con un bot Amazon Lex V2

La AWS SDK for Javaversione 2.0 fornisce un'interfaccia che puoi utilizzare dalle tue applicazioni Java per interagire con i tuoi bot. Utilizza SDK for Java per creare applicazioni client per gli utenti.

La seguente applicazione interagisce con il OrderFlowers bot che hai creato inEsercizio 1: Creare un bot da un esempio. Utilizza il file`LexRuntimeV2Client` dall'SDK for Java per chiamare l'RecognizeTextoperazione per condurre una conversazione con il bot.

L'output della conversazione ha il seguente aspetto:

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

Per le strutture JSON inviate tra l'applicazione client e il bot Amazon Lex V2, consultaEsercizio 2: Rivedi il flusso della conversazione.

Per eseguire l'applicazione, è necessario fornire le informazioni riportate di seguito:

- botId: l'identificativo assegnato al bot al momento della creazione. Puoi vedere l'ID del bot nella console Amazon Lex V2 nella pagina Impostazioni del bot.
- botAliasId — L'identificativo assegnato all'alias del bot al momento della creazione. Puoi vedere l'ID alias del bot nella console Amazon Lex V2 nella pagina Alias. Se non riesci a vedere l'alias ID nell'elenco, scegli l'icona a forma di ingranaggio in alto a destra e attiva Alias ID.
- localeID: l'identificatore della lingua che hai usato per il tuo bot. Per un elenco delle impostazioni locali, consultaLingue e impostazioni locali supportate da Amazon Lex V2.
- accessKey e secretKey: le chiavi di autenticazione per il tuo account. Se non si dispone di un set di chiavi, è necessario crearli utilizzando laAWS Identity and Access Management console.
- sessionId: un identificatore per la sessione con il bot Amazon Lex V2. In questo caso, il codice utilizza un UUID casuale.
- regione se il tuo bot nella regione Stati Uniti orientali (Virginia settentrionale), settentrionale), settentrionale), settentrionale), settentrionale), se il tuo bot nella regione Stati Uniti orientali (Virginia settentrionale), settentrionale), settentrionale), settentrionale), settentrionale

Le applicazioni utilizzano una funzione chiamata`getRecognizeTextRequest` per creare richieste individuali al bot. La funzione crea una richiesta con i parametri richiesti da inviare ad Amazon Lex V2.

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

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2Client;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextRequest;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextResponse;

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


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

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

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

        LexRuntimeV2Client lexV2Client = LexRuntimeV2Client
                .builder()
                .credentialsProvider(awsCredentialsProvider)
                .region(region)
```

```
                .build();

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

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

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

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

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

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

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

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

```
            System.out.println("Bot : " + message.content());
        });

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

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

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

Resilienza globale

> **Note**
>
> Questa funzionalità è disponibile solo per le istanze Amazon Connect e Amazon Lex V2 create nelle regioni Stati Uniti orientali (Virginia settentrionale) e Stati Uniti occidentali (Oregon).
>
> Per accedere a questa funzionalità, contatta il Solutions Architect o il Technical Account Manager Amazon Connect.

Global Resiliency ti consente di replicare un bot in una regione secondaria. La regione secondaria può essere resa attiva con la replica automatica del bot dell'utente in entrambe le regioni. Avrai una regione di backup in caso di interruzione regionale. Una volta che Global Resiliency è attiva, i nuovi bot creati vengono replicati in una seconda regione. AWS

Dopo aver attivato questa funzionalità, puoi automatizzare la replica dei bot di Amazon Lex V2 e delle relative risorse, versioni e alias in una regione associata quasi in tempo reale. AWS Con questa funzionalità, puoi monitorare il numero di versione del bot originale e di replica per assicurarti che la replica del bot rimanga sincronizzata con il bot originale. Quando abiliti la replica, puoi attivare la AWS regione predeterminata in cui desideri che il bot venga replicato (le regioni si basano su coppie predeterminate). Qualsiasi aggiornamento al bot di origine nella regione di origine viene automaticamente aggiornato al bot replicato nella seconda regione.

> **Note**
>
> Quando Global Resiliency è attivata, solo i bot, le versioni e gli alias creati dopo l'attivazione della funzionalità verranno replicati nell'area replicata. I bot, le versioni e gli alias creati in precedenza non saranno presenti nell'area replicata. La seconda regione identificata è di sola lettura e suddivisa in coppie predeterminate. Gli aggiornamenti dei bot sono limitati alla regione in cui il bot è stato inizialmente creato.

Informazioni aggiuntive sull'utilizzo di Global Resiliency:

- Attualmente Global Resiliency funziona solo con coppie di regioni predeterminate.

us-east-1	us-west-2
eu-west-2	eu-central-1

- Puoi creare una replica di qualsiasi bot Amazon Lex V2. È necessario creare una nuova versione e un nuovo alias per il bot dopo aver abilitato Global Resiliency.
- Gli alias abilitati in Global Resiliency possono essere associati solo a versioni abilitate per Global Resiliency.

Restrizioni:

- Global Resiliency non replica i bot creati con slot che utilizzano LLM come CFAQ e Utterance Generation.
- Global Resiliency non replica una rete di bot, ma qualsiasi bot che fa parte della rete di bot può comunque essere replicato individualmente.

Argomenti

Autorizzazioni per replicare i bot e gestire le repliche dei bot

Se a un ruolo IAM è associata la AmazonLexFullAccesspolicy, può creare e gestire repliche di bot.

Se preferisci creare un ruolo con autorizzazioni minime per Global Resiliency, utilizza la seguente policy, che contiene le seguenti istruzioni.

- Autorizzazioni per accedere al ruolo collegato al servizio Amazon Lex V2 per la replica dei bot.
- Autorizzazioni per consentire ad Amazon Lex V2 di creare un ruolo collegato al servizio per la replica dei bot per tuo conto.
- Autorizzazioni per chiamare le API di replica dei bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetReplicationSLR",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "CreateReplicationSLR",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:CreateServiceLinkedRole",
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowBotReplicaActions",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBotReplica",
                "lex:DescribeBotReplica",
                "lex:ListBotReplica",
                "lex:ListBotVersionReplicas",
                "lex:ListBotAliasReplicas",
                "lex:DeleteBotReplica"
            ],
            "Resource": [
                "arn:aws:lex:*:*:bot/*",
                "arn:aws:lex:*:*:bot-alias/*"
            ]
        }
    ]
}
```

È possibile limitare ulteriormente le autorizzazioni modificandole come segue.

- *Sostituisci** con ID di bot o alias di bot specifici per limitare le autorizzazioni a bot o alias di bot specifici.
- Utilizza un sottoinsieme di `lex BotReplica` azioni per limitare il ruolo a azioni specifiche.

Per vedere un esempio, consulta Consenti agli utenti di creare e visualizzare le repliche dei bot, ma non di eliminarle.

Implementazione della resilienza globale

Pannello informativo sulla resilienza globale

È possibile accedere alle seguenti informazioni nel pannello Global Resiliency:

- Dettagli sulla fonte: informazioni sulla regione di origine del bot, sul tipo di replica, sulla data di attivazione della replica e sull'ultima versione creata. Utilizza queste informazioni per tenere traccia delle iterazioni del bot.
- Dettagli sulla replica: dopo aver creato la replica del bot, è possibile tenere traccia della regione replicata, del tipo di replica, della data di sincronizzazione dell'ultima versione e dell'ultima versione replicata. Utilizzate queste informazioni per tracciare la sincronizzazione della replica del bot.
- Regione di origine: la regione in cui è abilitata la resilienza globale. È possibile apportare modifiche nella regione di origine per replicare il bot in entrambe le regioni.
- Tipo di replica: indica se il bot è di sola lettura o in grado di leggere e scrivere in base alla regione.
- Regione di replica: l'area secondaria utilizzata per replicare il bot di origine per Global Resiliency. Global Resiliency attualmente funziona solo con coppie regionali IAD/PDX e LDN/FRA.
- Data di attivazione della replica: la data e l'ora in cui è stata abilitata la replica del bot.
- Ultima versione creata: l'ultima versione del bot associata alla replica nell'area di origine.

Abilitare la resilienza globale

> **Note**
>
> Questa funzionalità è disponibile solo per le istanze Amazon Connect e Amazon Lex V2 create nelle regioni Stati Uniti orientali (Virginia settentrionale) e Stati Uniti occidentali (Oregon).
> Per accedere a questa funzionalità, contatta il Solutions Architect o il Technical Account Manager Amazon Connect.

Prima di attivare Global Resiliency nella console Amazon Lex V2, devi assicurarti che l'utente che abilita la replica dei bot sia autorizzato a creare Service Linked Roles (SLR). Global Resiliency utilizzerà queste credenziali FAS per creare una SLR nell'account abilitato quando viene richiamato. CreateReplica Per ulteriori informazioni sulla configurazione della reflex per Global Resiliency in Amazon Lex V2, consulta la policy gestita di AWS:. AmazonLexFullAccess

Attiva Global Resiliency e configura la replica dei bot per una seconda regione:

1. Accedi alla Console di gestione AWS e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot che desideri replicare dalla navigazione Bots nel pannello di navigazione a sinistra.
3. Scegli Deployment > Global Resiliency.
4. Seleziona il pulsante Crea replica nell'angolo in alto a destra della finestra per creare una bozza del bot.

   > **Note**
   >
   > Assicurati di non avere bot nella regione secondaria con lo stesso nome del bot che desideri replicare. (Il bot deve avere un nome univoco).

5. Vai a Global Resiliency, fai clic su Crea replica: questa azione crea una bozza del tuo bot. (non è necessario tornare alla scheda Global Resiliency se non per rivedere lo stato o vedere i dettagli delle future build).

   > **Note**
   >
   > Puoi anche creare un bot Alias per la replica in Global Resiliency andando su Alias e selezionando Crea nuovo alias per il bot abilitato a Global Resiliency. Verranno replicati solo gli alias creati dopo l'attivazione della replica.

6. Vai a Alias - Crea un nuovo alias per il bot abilitato alla resilienza globale. Verranno replicati solo gli alias creati dopo l'attivazione della replica.
7. Vai a Versione: crea una nuova versione per il bot abilitato alla resilienza globale. Verranno replicate solo le versioni create dopo l'attivazione della replica.

> **Note**
>
> I clienti hanno ancora il pieno controllo della gestione delle politiche e dei tag basati sulle risorse per i bot replicati. Le funzioni Lambda e CloudWatch i Logs Groups dovranno essere distribuiti in entrambe le regioni con gli stessi identificatori. Gli utenti non dovranno associare nuovamente la funzione lambda nell'area di replica.

Disabilitazione della resilienza globale

Puoi disabilitare Global Resiliency in qualsiasi momento selezionando il pulsante Disable Global Resiliency. Questa azione impedisce che il bot di origine e tutti gli alias e le versioni ad esso associati vengano replicati in altre regioni.

Utilizzo di API con Global Resiliency

Puoi effettuare chiamate API in Global Resiliency utilizzando le seguenti API. Ulteriori informazioni sulle API Global Resiliency e Amazon Lex V2 sono disponibili nella Amazon Lex V2 API Guide.

- CreateBotReplica

  Abilita Global Resiliency e crea un bot replicato. Richiede `replicaRegion`.

  Per ulteriori informazioni, CreateBotReplicaconsulta la Guida alle API Lex.
- DeleteBotReplica

  Disabilita Global Resiliency ed elimina il bot replicato. Richiede `replicaRegion` e `botId`.

  Per ulteriori informazioni, DeleteBotReplicaconsulta la Guida alle API Lex.
- ListBotReplicas

  Elenca i bot replicati nella zona secondaria. Richiede `botId`.

  Per ulteriori informazioni, ListBotReplicasconsulta la Guida alle API Lex.
- DescribeBotReplica

  Riepilogo delle informazioni per il bot replicato. Richiede `replicaRegion` e `botId`.

  Per ulteriori informazioni, DescribeBotReplicaconsulta la Guida alle API Lex.

Integrazione di un bot Amazon Lex V2 con una piattaforma di messaggistica

Questa sezione spiega come integrare i bot di Amazon Lex V2 con le piattaforme di messaggistica Facebook, Slack e Twilio. Se non disponi già di un bot Amazon Lex V2, creane uno. In questo argomento, supponiamo che tu stia utilizzando il bot in Esercizio 1: Creare un bot da un esempio cui hai creato. Tuttavia, puoi usare qualsiasi bot.

> **Note**
>
> Quando archivia le configurazioni di Facebook, Slack o Twilio, Amazon Lex V2 utilizza un AWS KMS key per crittografare le informazioni. La prima volta che crei un canale verso una di queste piattaforme di messaggistica, Amazon Lex V2 crea una chiave gestita dal cliente predefinita (`aws/lex`) nel tuo AWS account oppure puoi selezionare la tua chiave gestita dal cliente. Amazon Lex V2 supporta solo chiavi simmetriche. 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 V2, include informazioni specifiche della piattaforma come attributo di richiesta della funzione Lambda. Usa questo attributo per personalizzare il comportamento del bot. Per ulteriori informazioni, consulta Impostazione degli attributi della richiesta.

Attributo di richiesta comune

Attributo	Descrizione
x-amz-lexAttributo di richiesta comune ----sep----:channels:platform	Uno dei seguenti valori: • `Facebook` • `Slack` • `Twilio`

Integrazione di un bot Amazon Lex V2 con Facebook Messenger

Puoi ospitare il tuo bot Amazon Lex V2 in Facebook Messenger. Quando lo fai, gli utenti di Facebook possono interagire con il tuo bot per soddisfare gli intenti.

Prima di iniziare, devi creare un account sviluppatore Facebook all'indirizzo https://developers.facebook.com.

È necessario completare la seguente procedura:

Argomenti

- Fase 1: Creazione di un'applicazione Facebook
- Passaggio 2: integra Facebook Messenger con il bot Amazon Lex V2

- Fase 3: Integrazione completa con Facebook
- Fase 4: Test dell'integrazione

Fase 1: Creazione di un'applicazione Facebook

Sul portale per gli sviluppatori di Facebook, crea un'applicazione e una pagina Facebook.

Per creare un'applicazione Facebook

1. Apri https://developers.facebook.com/apps
2. Scegli Create App (Crea app).
3. Nella pagina Crea un'app, scegli Business, quindi scegli Avanti.
4. Per i campi Nome app aggiuntiva, email di contatto dell'app e Account aziendale, effettua le scelte appropriate per la tua app. Scegli Crea app per continuare.
5. Da Aggiungi prodotti alla tua app, scegli Configura dal riquadro Messenger.
6. Nella sezione Token di accesso, scegli Aggiungi o rimuovi pagine.
7. Scegli una pagina da utilizzare con l'app, quindi scegli Avanti.
8. Per Cosa può fare l'app, lascia le impostazioni predefinite e scegli Fine.
9. Nella pagina di conferma scegli OK.
10. Nella sezione Token di accesso, scegli Genera token, quindi copia il token. Inserisci questo token nella console Amazon Lex V2.
11. Dal menu a sinistra, scegli Impostazioni, quindi scegli Basic.
12. Per App Secret, scegli Mostra, quindi copia il segreto. Inserisci questo token nella console Amazon Lex V2.

Approfondimenti

Passaggio 2: integra Facebook Messenger con il bot Amazon Lex V2

Passaggio 2: integra Facebook Messenger con il bot Amazon Lex V2

In questo passaggio colleghi il tuo bot Amazon Lex V2 a Facebook.

1. Accedere aAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.

2. Dall'elenco dei bot, scegli il bot Amazon Lex V2 che hai creato.
3. Nel menu a sinistra, scegli Integrazioni dei canali, quindi scegli Aggiungi canale.
4. In Crea canale, procedi come segue:

    a. Per Platform, scegli Facebook.
    b. Per le politiche di identità, scegli laAWS KMS chiave per proteggere le informazioni sul canale. La chiave predefinita è Amazon Lex V2.
    c. Per la configurazione dell'integrazione, assegnare al canale un nome e una descrizione facoltativa. Scegli l'alias che punta alla versione del bot che si desidera utilizzare e scegliere la lingua supportata dal canale.
    d. Per Configurazione aggiuntiva, inserisci quanto segue:

        - Alias: una stringa che identifica l'app che chiama Amazon Lex V2. Puoi usare qualsiasi stringa. Registra questa stringa, la inserisci nella console degli sviluppatori di Facebook.
        - Token di accesso alla pagina: il token di accesso alla pagina che hai copiato dalla console degli sviluppatori di Facebook.
        - Chiave segreta dell'app: la chiave segreta che hai copiato dalla console degli sviluppatori di Facebook.

    e. Scegli Create (Crea)
    f. Amazon Lex V2 mostra l'elenco dei canali del tuo bot. Dall'elenco, scegliere il canale appena creato.
    g. Dall'URL di richiamata, registra l'URL di richiamata. Inserisci questo URL nella console degli sviluppatori di Facebook.

Approfondimenti

Fase 3: Integrazione completa con Facebook

Fase 3: Integrazione completa con Facebook

In questo passaggio, utilizza la console per sviluppatori di Facebook per completare l'integrazione con Amazon Lex V2.

Per completare l'integrazione con Facebook Messenger

1. Apri https://developers.facebook.com/apps
2. Dall'elenco delle app, scegli l'app che stai integrando con Facebook Messenger.

3. Nel menu a sinistra, scegli Messenger, quindi scegli Impostazioni.
4. Nella sezione Webhook:
    a. Scegli Aggiungi URL di richiamata.
    b. In Modifica URL di richiamata, inserisci quanto segue:
        - URL di richiamata: inserisci l'URL di callback che hai registrato dalla console Amazon Lex V2.
        - Token di verifica: inserisci l'alias che hai inserito nella console Amazon Lex V2.
    c. Seleziona Verify and Save (Verifica e salva).
    d. Scegli Aggiungi abbonamenti in Webhook accanto alla tua pagina.
    e. Nella finestra che si apre, scegli`messages` e fai clic su Salva.

Approfondimenti

Fase 4: Test dell'integrazione

Fase 4: Test dell'integrazione

Ora puoi avviare una conversazione da Facebook Messenger con il tuo bot Amazon Lex V2.

Per testare l'integrazione tra Facebook Messenger e un bot Amazon Lex V2

1. Apri la pagina Facebook che hai associato al tuo bot nel passaggio 1.
2. Nella finestra di Messenger, usa le espressioni di prova fornite inEsercizio 1: Creare un bot da un esempio.

Integrazione di un bot Amazon Lex V2 con Slack

Questo argomento fornisce istruzioni per l'integrazione di un bot Amazon Lex V2 con l'applicazione di messaggistica Slack. Completa la seguente procedura:

Argomenti

- Passaggio 4: Integrazione completa con Slack
- Passaggio 5: Test dell'integrazione

Passaggio 1: registrazione a Slack e creazione di un team Slack

Registra un account Slack e crea un team Slack. Per istruzioni, consulta Utilizzo di Slack. Nella sezione successiva crei un'applicazione Slack, che qualsiasi team Slack può installare.

Approfondimenti

Passaggio 2: creazione di un'applicazione Slack

Passaggio 2: creazione di un'applicazione Slack

In questa sezione effettuerai le operazioni seguenti:

1. Crea un'applicazione Slack nella console dell'API Slack.
2. Configura l'applicazione per aggiungere messaggi interattivi al tuo bot.

Alla fine di questa sezione, ottieni le credenziali dell'applicazione (ID cliente, segreto del cliente e token di verifica). Nel passaggio successivo, utilizzi queste informazioni per integrare il bot nella console Amazon Lex V2.

Per creare un'applicazione Slack

1. Accedi alla console delle API di Slack all'indirizzo https://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, scegli Interattività 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

Approfondimenti

Fase 3: Integrare l'applicazione Slack con il bot Amazon Lex V2

Fase 3: Integrare l'applicazione Slack con il bot Amazon Lex V2

In questa sezione, integra l'applicazione Slack che hai creato con il bot Amazon Lex V2 che hai creato utilizzando le integrazioni di canale.

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli il bot Amazon Lex V2 che hai creato.
3. Nel menu a sinistra, scegli Integrazioni dei canali, quindi scegli Aggiungi canale.
4. In Crea canale, procedi come segue:
   a. Per Platform, scegli Slack.
   b. Per le politiche di identità, scegli laAWS KMS chiave per proteggere le informazioni sul canale. La chiave predefinita è fornita da Amazon Lex V2.
   c. Per la configurazione dell'integrazione, assegnare al canale un nome e una descrizione facoltativa. Scegliere l'alias che punta alla versione del canale. Il canale supporta l'alias che punta alla versione del canale.

> **Note**
>
> Se il bot è disponibile in più lingue, devi creare un canale diverso e un'applicazione diversa per ogni lingua.

d. Per Configurazione aggiuntiva, inserisci quanto segue:

- ID cliente: inserisci l'ID client da Slack.
- Segreto del cliente: inserisci il segreto del cliente da Slack.
- Token di verifica: inserisci il token di verifica di Slack.
- URL della pagina di successo: l'URL della pagina che Slack deve aprire quando l'utente viene autenticato. In genere questo spazio rimane vuoto.

5. Scegli Crea per creare il canale.
6. Amazon Lex V2 mostra l'elenco dei canali del tuo bot. Dall'elenco, scegliere il canale appena creato.
7. Dall'URL di callback, registra l'endpoint e l'endpoint OAuth.

Approfondimenti

Passaggio 4: Integrazione completa con Slack

Passaggio 4: Integrazione completa con Slack

In questa sezione, usa la console dell'API Slack per completare l'integrazione con l'applicazione Slack.

1. Accedi alla console dell'API Slack all'indirizzo https://api.slack.com. Seleziona l'app che hai creato in Passaggio 2: creazione di 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 Reindirizza gli URL, aggiungi l'endpoint OAuth fornito da Amazon Lex nel passaggio precedente. Seleziona Add (Aggiungi), 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 della richiesta all'endpoint fornito da Amazon Lex nel passaggio precedente. Inserisci l'endpoint che hai salvato nel passaggio 3, quindi scegli Salva modifiche.
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'endpoint fornito da Amazon Lex nel passaggio precedente.
   - Nella sezione Iscriviti agli eventi dei bot, seleziona Aggiungi evento utente bot e aggiungi l'evento**`message.im`** bot 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:
   - Dal menu a sinistra, scegli App Home.
   - Nella sezione Mostra schede, scegli Consenti agli utenti di inviare comandi e messaggi Slash dalla scheda messaggi.
6. Seleziona Manage Distribution (Gestisci la distribuzione) in Settings (Impostazioni). Seleziona Add to Slack (Aggiungi a Slack) per installare l'applicazione. Se sei autenticato in più aree di lavoro, scegli innanzitutto l'area di lavoro corretta nell'angolo in alto a destra dall'elenco a discesa. Quindi, seleziona Consenti per autorizzare il bot a rispondere ai messaggi.

> **Note**
>
> Se apporti modifiche alle impostazioni dell'applicazione Slack in un secondo momento, devi ripetere questo passaggio secondario.

Approfondimenti

Passaggio 5: Test dell'integrazione

Passaggio 5: Test dell'integrazione

Ora usa una finestra del browser per testare l'integrazione di Slack con il tuo bot Amazon Lex V2.

Per testare la tua applicazione Slack

1. Avvia Slack. Dal menu a sinistra, nella sezione Messaggi diretti, scegli il tuo bot. Se non lo trovi, seleziona il simbolo "più" (+) accanto all'opzione Direct Messages (Messaggi diretti) per cercarlo.
2. Partecipa a una chat con la tua applicazione Slack. Il bot risponde ai messaggi.

   Se hai creato il bot utilizzandoEsercizio 1: Creare un bot da un esempio, puoi usare le conversazioni di esempio di quell'esercizio.

Integrazione di un bot Amazon Lex V2 con Twilio SMS

Questo argomento fornisce istruzioni per l'integrazione di un bot Amazon Lex V2 con il servizio di messaggistica semplice (SMS) Twilio. Completa la seguente procedura:

Argomenti

- Fase 1: creazione di un account SMS Twilio
- Fase 2: Integrazione dell'endpoint del servizio di messaggistica Twilio con il bot Amazon Lex V2
- Fase 3: integrazione completa con Twilio
- Fase 4: esecuzione del test dell'integrazione

Fase 1: creazione di un account SMS Twilio

Effettua la registrazione di un account Twilio e registra le seguenti informazioni relative all'account:

- SID ACCOUNT
- TOKEN AUTORIZ

Per le istruzioni di registrazione, consulta https://www.twilio.com/console.

Approfondimenti

Fase 2: Integrazione dell'endpoint del servizio di messaggistica Twilio con il bot Amazon Lex V2

Fase 2: Integrazione dell'endpoint del servizio di messaggistica Twilio con il bot Amazon Lex V2

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 il bot Amazon Lex V2 che hai creato.
3. Nel menu a sinistra, scegli Integrazioni dei canali, quindi scegli Aggiungi canale.
4. In Crea canale, procedi come segue:
   a. Per Platform, scegli Twilio.
   b. Per le politiche di identità, scegli laAWS KMS chiave per proteggere le informazioni sul canale. La chiave predefinita è fornita da Amazon Lex V2.
   c. Per la configurazione dell'integrazione, assegnare al canale un nome e una descrizione facoltativa. Scegli l'alias che punta alla versione del bot che si desidera utilizzare.
   d. Per una configurazione aggiuntiva, inserisci il SID dell'account e il token di autenticazione dalla dashboard di Twilio.
5. Seleziona Create (Crea).
6. Dall'elenco dei canali, scegliere il canale appena creato.
7. Copia l'URL del callback.

Approfondimenti

Fase 3: integrazione completa con Twilio

Fase 3: integrazione completa con Twilio

Usa la console Twilio per completare l'integrazione del tuo bot Amazon Lex V2 con Twilio SMS.

1. Apri la console Twilio all'indirizzo https://www.twilio.com/console.
2. Dal menu a sinistra, scegli Tutti i prodotti e servizi, quindi scegli Numero di telefono.
3. Se hai un numero di telefono, sceglilo. Se non disponi di un numero di telefono, scegli Acquista un numero per ottenerne uno.
4. Nella sezione Messaggi, in A MESSAGE COMES IN, inserisci l'URL di callback dalla console Amazon Lex V2.
5. Seleziona Salva.

Approfondimenti

Fase 4: esecuzione del test dell'integrazione

Fase 4: esecuzione del test dell'integrazione

Utilizza il tuo cellulare per testare l'integrazione tra Twilio SMS e il tuo bot. Utilizzando il tuo cellulare, invia messaggi al numero Twilio.

Se hai creato il bot utilizzandoEsercizio 1: Creare un bot da un esempio, puoi usare le conversazioni di esempio di quell'esercizio.

Integrazione di un bot Amazon Lex V2 con un contact center

Puoi integrare i bot di Amazon Lex V2 con i tuoi contact center per abilitare casi d'uso self-service utilizzando l'API di streaming Amazon Lex V2. Usa questi bot come agenti di risposta vocale interattiva (IVR) per la telefonia o come chatbot testuale integrato nel tuo contact center. Per ulteriori informazioni sulle API di streaming, consulta. Streaming su un bot Amazon Lex V2

Con le API di streaming, puoi abilitare le seguenti funzionalità:

- Interruzioni («barge-in»): i chiamanti possono interrompere il bot e rispondere a una domanda prima che il prompt sia completo. Per ulteriori informazioni, consulta Consentire al bot di essere interrotto dall'utente.
- Attendi e continua: i chiamanti possono indicare al bot di attendere se hanno bisogno di tempo per recuperare informazioni aggiuntive durante una chiamata, come il numero della carta di credito o l'ID di prenotazione. Per ulteriori informazioni, consulta Abilitare il bot ad attendere che l'utente fornisca ulteriori informazioni.
- Supporto DTMF: i chiamanti possono fornire informazioni vocali o DTMF in modo intercambiabile.
- Supporto SSML: puoi configurare i prompt dei bot di Amazon Lex V2 utilizzando tag SSML per un maggiore controllo sulla generazione vocale dal testo. Per ulteriori informazioni, consulta Generazione vocale da documenti SSML nella guida per sviluppatori di Amazon Polly.
- Timeout configurabili: puoi configurare il tempo di attesa che i clienti finiscano di parlare prima che Amazon Lex V2 raccolga i loro input vocali, ad esempio rispondendo a una domanda sì o no o fornendo una data o un numero di carta di credito. Per ulteriori informazioni, consulta Configurazione dei timeout per l'acquisizione dell'input dell'utente.
- Aggiornamenti sull'avanzamento dell'evasione degli ordini: puoi configurare il bot in modo che risponda con più messaggi in base allo stato di evasione durante l'esecuzione della logica

aziendale per l'adempimento degli intenti. Puoi impostare il bot in modo che risponda con messaggi all'inizio e al termine dell'adempimento e fornisca aggiornamenti periodici per le funzioni Lambda a lunga esecuzione. Per ulteriori informazioni, consulta Configurazione degli aggiornamenti sullo stato di avanzamento dell'evasione.

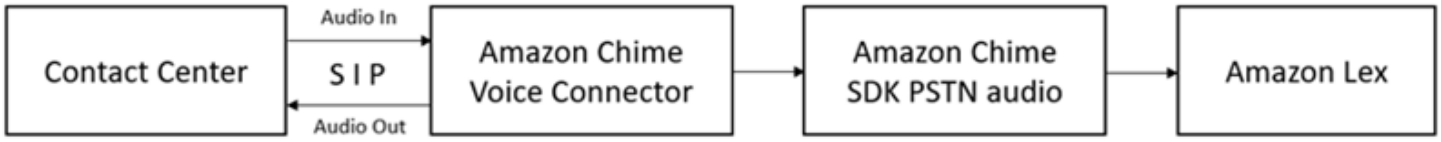
SDK Amazon Chime

Usa Amazon Chime SDK per aggiungere funzionalità audio, video, di condivisione dello schermo e di messaggistica in tempo reale alle tue applicazioni web o mobili. L'SDK Amazon Chime fornisce un servizio audio PSTN (Public Switched Telephone Network) in modo da poter creare applicazioni di telefonia personalizzate con unaAWS Lambda funzione.

L'audio PSTN di Amazon Chime è integrato con Amazon Lex V2. Puoi utilizzare questa integrazione per accedere ai bot di Amazon Lex V2 come sistemi di risposta vocale interattiva (IVR) nei contact center per le interazioni audio. Utilizzalo per integrare Amazon Lex V2 utilizzando i servizi audio PSTN nei seguenti scenari.

Integrazioni con i contact center: puoi utilizzare il servizio audio PSTN Amazon Chime Voice Connector e Amazon Chime SDK per accedere ai bot Amazon Lex V2. Utilizzali in qualsiasi applicazione di contact center che utilizza il protocollo di avvio della sessione (SIP) per le comunicazioni vocali. Questa integrazione aggiunge esperienze di conversazione vocale in linguaggio naturale al tuo contact center locale o basato sul cloud esistente con supporto SIP. Per un elenco delle piattaforme di contact center supportate, consulta le risorse relative ai connettori vocali Amazon Chime.

Il diagramma seguente mostra l'integrazione tra un contact center che utilizza SIP e Amazon Lex V2.



Supporto telefonico diretto: puoi creare soluzioni IVR personalizzate per accedere direttamente ai bot di Amazon Lex V2 utilizzando un numero di telefono fornito in Amazon Chime SDK.

Per ulteriori informazioni, consulta gli argomenti seguenti nella guida di Amazon Chenenenenenenenenenenenenenenenenenenenenenenen

- Integrazione SIP tramite un connettore vocale Amazon Chime
- Utilizzo del servizio audio PSTN Amazon Chime SDK

- Integrazione dell'audio PSTN Amazon Chime con Amazon Lex V2

Quando Amazon Chime SDK invia una richiesta ad Amazon Lex V2, include informazioni specifiche della piattaforma nella funzione Lambda e nei registri delle conversazioni. Utilizza queste informazioni per determinare l'applicazione del contact center che invia traffico al tuo bot.

Atenencomuni	Value (Valore)
x-amz-lex:canali:piattaforma	`Amazon Chime SDK PSTN Audio`

Amazon Connect

Amazon Connect è un contact center cloud omnicanale. Puoi configurare un contact center in pochi passaggi, aggiungere agenti dislocati ovunque e iniziare a interagire con i tuoi clienti. Per ulteriori informazioni, consulta la sezione Introduzione ad Amazon Connect nella guida per amministratori di Amazon Connect.

Puoi creare esperienze personalizzate per i tuoi clienti utilizzando le comunicazioni omnichannel. Ad esempio, puoi offrire chat e contatti vocali in base alle preferenze del cliente e ai tempi di attesa stimati. Nel frattempo, gli agenti possono gestire tutti i clienti da un'unica interfaccia. Ad esempio, possono chattare con i clienti e creare attività o rispondervi man mano che queste vengono instradate a ognuno.

Puoi utilizzare Amazon Connect per le interazioni audio con i tuoi clienti o Amazon Connect Chat per le interazioni di solo testo.

Per ulteriori informazioni, consulta i seguenti argomenti nella guida per amministratori di Amazon Connect.

- Cos'è Amazon Connect
- Aggiungere un bot Amazon Lex V2
- Amazon Connect riceve il blocco dei contatti di input dei clienti

Quando un contact center invia una richiesta ad Amazon Lex V2, include informazioni specifiche della piattaforma come attributo di richiesta nella funzione Lambda e nei log delle conversazioni. Utilizza queste informazioni per determinare quale applicazione del contact center invia traffico al tuo bot.

Attributo di richiesta comune

Attributo	Valore
x-amz-lexAttributo di richiesta comune ----sep----:channels:platform	Uno dei seguenti valori: • `Connect` • `Connect Chat`

Genesys Cloud

Genesys Cloud è una suite di servizi cloud per la comunicazione aziendale, la collaborazione e la gestione dei contact center. Genesys Cloud si basa su AWS e utilizza un ambiente cloud distribuito che fornisce un accesso sicuro alle organizzazioni di tutto il mondo del lavoro.

Per ulteriori informazioni, consulta le seguenti pagine sul sito web di Genesys Cloud.

- Informazioni sul contact center Genesys Cloud
- Informazioni sull'integrazione con Amazon Lex V2

Quando un contact center invia una richiesta ad Amazon Lex V2, include informazioni specifiche della piattaforma come attributo di richiesta nella funzione Lambda e nei log delle conversazioni. Utilizza queste informazioni per determinare quale applicazione del contact center invia traffico al tuo bot.

Attributo di richiesta comune

Attributo	Valore
x-amz-lexAttributo di richiesta comune ----sep----:channels:platform	• `Genesys Cloud`

Ulteriori informazioni

- Potenzia il tuo contact center con Amazon Lex e Genesys Cloud

Gestire le conversazioni

Dopo aver creato un bot, integri l'applicazione client con le operazioni di runtime di Amazon Lex V2 per tenere conversazioni con il bot.

Quando un utente avvia una conversazione con il tuo bot, Amazon Lex V2 crea una sessione. Una sessione incapsula le informazioni scambiate tra l'applicazione e il bot. Per ulteriori informazioni, consulta Gestione delle sessioni con l'API Amazon Lex V2.

Una conversazione tipica prevede un flusso avanti e indietro tra l'utente e un bot. Ad esempio:

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

Quando si utilizza l'RecognizeUtteranceoperazione RecognizeTextor, è necessario gestire la conversazione nell'applicazione client. Quando utilizzi l'StartConversationoperazione, Amazon Lex V2 gestisce la conversazione per te.

Per gestire la conversazione, devi inviare le espressioni degli utenti al bot fino a quando la conversazione non raggiunge una conclusione logica. La conversazione corrente viene acquisita nello stato della sessione. Lo stato della sessione viene aggiornato dopo ogni enunciato dell'utente. Lo stato della sessione contiene lo stato corrente della conversazione e viene restituito dal bot in una risposta. a ciascuna espressione dell'utente.

Una conversazione può avere uno dei seguenti stati:

- ElicitIntent— Indica che il bot non ha ancora determinato l'intento dell'utente.
- ElicitSlot— Indica che il bot ha rilevato l'intento dell'utente e sta raccogliendo le informazioni necessarie per soddisfare l'intento.
- ConfirmIntent— Indica che il bot è in attesa che l'utente confermi che le informazioni raccolte sono corrette.

- Chiuso: indica che l'intento dell'utente è completo e che la conversazione con il bot ha raggiunto una conclusione logica.

Un utente può specificare un nuovo intento dopo il completamento del primo intento. Per ulteriori informazioni, consulta Gestire il contesto della conversazione.

Un intento può avere uno dei seguenti stati:

- InProgress— Indica che il bot sta raccogliendo le informazioni necessarie per completare l'intento. Questo è in concomitanza con lo stato della `ElicitSlot` conversazione.
- In attesa: indica che l'utente ha richiesto al bot di attendere quando il bot ha richiesto informazioni per uno slot specifico.
- Soddisfatto: indica che la logica aziendale in una funzione Lambda associata all'intento è stata eseguita correttamente.
- ReadyForFulfillment— Indica che il bot ha raccolto tutte le informazioni necessarie per soddisfare l'intento e che l'applicazione client può eseguire la logica aziendale di adempimento.
- Fallito: indica che un intento è fallito.

Consulta i seguenti argomenti per scoprire come utilizzare le API Amazon Lex V2 per gestire il contesto delle conversazioni e le sessioni tra il bot e gli utenti.

Argomenti

- Gestire il contesto della conversazione
- Gestione delle sessioni con l'API Amazon Lex V2

Gestire il contesto della conversazione

Il contesto della conversazione è costituito da informazioni che l'utente, l'applicazione client o una funzione Lambda forniscono a un bot Amazon Lex per soddisfare un intento. Il contesto della conversazione include i dati relativi agli slot forniti dall'utente, gli attributi di 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 della sessione
- Impostazione degli attributi della richiesta
- Impostazione del timeout della sessione
- Condivisione di informazioni tra intenti
- Impostazione di attributi complessi

Impostazione del contesto dell'intento

Puoi fare in modo che Amazon Lex attivi gli intenti 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 vengono configurati quando si crea l'intento utilizzando la console o utilizzando l'operazione. CreateIntent Puoi utilizzare il contesto solo nella lingua inglese (USA) (en-US).

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 dall'RecognizeUtteranceoperazione RecognizeTextor e viene 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 intento può essere riconosciuto durante una conversazione solo quando tutti i suoi contesti di input sono attivi. Un intento privo di contesti di input è sempre idoneo al riconoscimento.

Amazon Lex gestisce automaticamente il ciclo di vita dei contesti che vengono attivati soddisfacendo gli intenti con contesti di output. È inoltre possibile impostare contesti attivi in una chiamata all'operazione `RecognizeText` or`RecognizeUtterance`.

Puoi anche 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 Abilitazione della logica personalizzata con AWS Lambda funzioni.

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 utilizzarlo 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 attiva i contesti di output di un intento quando l'intento è soddisfatto. È possibile utilizzare il contesto di output per controllare gli intenti idonei a seguire l'intento corrente.

Ogni contesto ha un elenco di parametri che vengono mantenuti nella sessione. I parametri sono i valori degli slot per l'intento soddisfatto. 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'CreateIntentoperazione. 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 nella risposta or. RecognizeTextRecognizeUtterance

Quando definisci un contesto di output, ne definisci anche la durata di vita, il periodo di tempo o il numero di turni in cui il contesto viene 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 in base alle esigenze. Ad esempio, l'applicazione può utilizzare il contesto di output per:

- Cambia 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

Si imposta un contesto di input per limitare i punti della conversazione in cui l'intento viene riconosciuto. Gli intenti senza un contesto di input sono sempre idonei per essere riconosciuti.

Si impostano i contesti di input a cui risponde un intento utilizzando la console o l'operazione. `CreateIntent` Un intento può avere più di un contesto di input.

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'PutSessionoperazione RecognizeTextRecognizeUtterance,, or.

È possibile configurare gli slot con l'intento di prendere 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. Specificate il nome del contesto e il nome dello slot nel modulo `#context-name.parameter-name` quando definite 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 un'origine per un valore di slot da riempire per nuovi intenti quando l'input dell'utente non fornisce alcuno slot. Questa fonte può essere costituita da attributi precedenti di dialogo, richiesta o sessione o da un valore fisso impostato in fase di compilazione.

È possibile utilizzare quanto segue come origine per i valori predefiniti.

- 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 al precedente

Quando si utilizza l'CreateIntentoperazione 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, supponiamo di avere un intento con uno slot con la seguente definizione:

```
"slots": [
    {
        "botId": "string",
        "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 contesto book-car-fulfilled "" è attivo, il valore del parametro «startDate» viene utilizzato come valore predefinito.
2. Se il contesto book-car-fulfilled "" non è attivo o se il parametro «startDate» non è impostato, il valore dell'attributo reservationStartDate "" session 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 attivato anche se è richiesto.

Impostazione degli attributi della 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 trasmette 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 trasmette le nuove informazioni all'applicazione client.

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 pizza, l'applicazione client trasmette la posizione dell'utente come attributo di sessione nella prima chiamata all'RecognizeTextoperazione or. RecognizeUtterance Ad esempio, `"Location": "111 Maple Street"`. La funzione Lambda utilizza queste informazioni per trovare la pizzeria più vicina a cui effettuare l'ordine.
- Personalizza i prompt: configura i prompt e le schede di risposta per fare riferimento agli attributi della sessione. Ad esempio, «Ehi [FirstName], quali condimenti vorresti?» Se passi il nome dell'utente come attributo di sessione (`{"FirstName": "Vivian"}`), Amazon Lex sostituisce il nome al posto del segnaposto. Quindi invia un messaggio personalizzato all'utente: «Ehi Vivian, quali condimenti desideri?»

Gli attributi della sessione persistono per tutta la durata della sessione. Amazon Lex li archivia in un archivio dati crittografato fino al termine della sessione. Il client può creare attributi di sessione in una richiesta chiamando l'RecognizeUtteranceoperazione RecognizeTextor con il `sessionAttributes` campo 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'`RecognizeUtterance`operazione `RecognizeText` or senza specificare il `sessionAttributes` campo, Amazon Lex chiama la funzione Lambda con gli attributi della sessione memorizzati (`{"x": 1, "y": 2}`. Se la funzione Lambda non restituisce gli attributi di sessione, Amazon Lex restituisce gli attributi della sessione memorizzati all'applicazione client.

Se l'applicazione client o una funzione Lambda superano 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, devi creare una string-to-string mappa degli attributi. Di seguito viene spiegato come mappare gli attributi di sessione:

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

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

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

Per l'`RecognizeUtterance`operazione, si codifica la mappa in base64 e quindi la si invia come parte dell'intestazione. `x-amz-lex-session-state`

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 della 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 RecognizeUtterance o il campo `requestAttributes` in una richiesta RecognizeText. Poiché, a differenza degli attributi di sessione, gli attributi di richiesta non vengono mantenuti tra richieste diverse, non vengono restituiti nelle risposte `RecognizeUtterance` o `RecognizeText`.

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

Impostazione degli 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 `RecognizeUtterance` o nel campo `requestAttributes` di una richiesta `RecognizeText`.

Per inviare gli attributi della richiesta ad Amazon Lex, devi 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 della sessione

Amazon Lex conserva le informazioni contestuali (dati sugli slot e attributi della sessione) 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 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 sugli slot ma non completa l'acquisto della scarpa, Amazon Lex ricorda tutti i dati degli slot e gli attributi della sessione per l'intera sessione. Se l'utente torna alla sessione prima della scadenza, può fornire i dati relativi agli slot rimanenti 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 (AWS CLI) o l'API, puoi impostare il timeout quando crei un bot con l'CreateBotoperazione impostando il campo InSecondsIdleSessionTTL.

Condivisione di informazioni tra intenti

Amazon Lex supporta la condivisione di informazioni tra intenti. Per condividere gli intenti, usa i contesti di output o gli attributi della sessione.

Per utilizzare i contesti di output, si definisce un contesto di output quando si crea o si aggiorna un intento. Quando l'intento è soddisfatto, le risposte di Amazon Lex V2 contengono il contesto e i valori degli slot dell'intento come parametri di contesto. È possibile utilizzare questi parametri come valori predefiniti negli intenti successivi o nel codice dell'applicazione o nelle funzioni Lambda.

Per utilizzare gli attributi di sessione, impostate gli attributi nel codice Lambda o dell'applicazione. 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 soddisfa l'ordine imposta l'attributo di `orderNumber` 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.

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 serie 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 RecognizeText e RecognizeUtterance.

Ad esempio, se si utilizzaJavaScript, 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 attributi con l'`RecognizeUtterance`operazione, è necessario codificare gli attributi in base64 prima di aggiungerli all'intestazione della richiesta, come mostrato nel codice seguente: JavaScript

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

Puoi inviare i dati binari alle operazioni `RecognizeText` e `RecognizeUtterance` 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"
}
```

Gestione delle sessioni con l'API Amazon Lex V2

Quando un utente avvia una conversazione con il tuo bot, Amazon Lex V2 crea una sessione. Le informazioni scambiate tra l'applicazione e Amazon Lex V2 costituiscono lo stato della sessione della conversazione. Quando si effettua una richiesta, la sessione viene identificata da un identificatore

specificato dall'utente. Per ulteriori informazioni sull'identificatore di sessione, vedere il `sessionId` campo nell'RecognizeUtteranceoperazione RecognizeTextor.

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.

Esistono tre modi per aggiornare lo stato della sessione.

- Passa le informazioni sulla sessione in linea come parte di una chiamata all'`RecognizeUtterance`operazione `RecognizeText` or.
- Usa una funzione Lambda con l'`RecognizeUtterance`operazione `RecognizeText` or che viene chiamata dopo ogni turno della conversazione. Per ulteriori informazioni, consulta Abilitazione della logica personalizzata con AWS Lambda funzioni. L'altro consiste nell'utilizzare l'API di runtime di Amazon Lex V2 nell'applicazione per apportare modifiche allo stato della sessione.
- Usa le operazioni che ti consentono di gestire le informazioni sulla sessione per una conversazione con il tuo bot. Le operazioni sono l'PutSessionoperazione, l'GetSessionoperazione e l'DeleteSessionoperazione. 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 corrente della sessione, incluso lo stato della finestra di dialogo con l'utente, gli eventuali attributi di sessione impostati e i valori degli slot per l'intento corrente e qualsiasi altro intento che Amazon Lex V2 ha identificato come possibili intenti che corrispondono all'enunciato dell'utente.

L'operazione `PutSession` consente di manipolare direttamente lo stato attuale della sessione. Puoi impostare la sessione, incluso il tipo di azione di dialogo che il bot eseguirà successivamente e i messaggi che Amazon Lex V2 invia all'utente. in modo da avere il controllo sul flusso della conversazione con il bot. Imposta il `type` campo di azione della finestra `Delegate` di dialogo in modo che Amazon Lex V2 determini l'azione successiva per il 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.

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

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 funzionano con le funzioni di adempimento Lambda. Ad esempio, se la funzione Lambda ritorna `Failed` come stato di adempimento, puoi utilizzare l'`PutSession`operazione per impostare il tipo di azione di dialogo `close` e `fulfillmentState` `ReadyForFulfillment` 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.

Avvio di 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 della tua intenzione di iniziare la conversazione con l'utente.

Cambio di intenti

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.

Ripresa di un intento precedente

Per ripristinare un intento precedente, utilizzate l'`GetSession`operazione per ottenere lo stato dell'intento, eseguire l'interazione necessaria e quindi utilizzare l'`PutSession`operazione per impostare l'intento allo stato di dialogo precedente.

- Richiama l'operazione `GetSession`. Memorizza lo stato dell'intento.
- Esegui un'altra interazione, ad esempio soddisfacendo un intento diverso.
- Utilizzando le informazioni salvate per l'intento precedente, richiamate l'`PutSession`operazione. 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.

Convalida dei valori degli 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, puoi impostare i `messageFormat` campi `message` and se desideri modificare il messaggio utilizzato da Amazon Lex per ottenere il valore dello slot.

Abilitazione della logica personalizzata con AWS Lambda funzioni

Con AWS Lambdale funzioni, puoi controllare meglio il comportamento del tuo bot Amazon Lex V2 attraverso funzioni personalizzate da te definite.

Amazon Lex V2 utilizza una funzione Lambda per alias bot per lingua anziché una funzione Lambda per ogni intento.

Per integrare una funzione Lambda con il tuo bot Amazon Lex V2, esegui i seguenti passaggi:

1. Determina da quali campi dell'evento di input desideri trarre informazioni da utilizzare nella tua funzione Lambda.
2. Determina quali campi della risposta desideri manipolare e restituire dalla tua funzione Lambda.
3. Crea una funzione AWS Lambda utilizzando il linguaggio di programmazione che preferisci e scrivi lo script.
4. Assicurati che la funzione restituisca una struttura corrispondente al formato di risposta.
5. Distribuire la funzione Lambda.
6. Associa la funzione Lambda a un alias bot di Amazon Lex V2 alle operazioni della console o dell'API.
7. Seleziona le fasi della conversazione in cui desideri richiamare la funzione Lambda con le operazioni della console o dell'API.
8. Crea il tuo bot Amazon Lex V2 e verifica che la funzione Lambda funzioni come previsto. Esegui il debug della tua funzione con l'aiuto di Amazon CloudWatch.

Argomenti

- Interpretazione del formato dell'evento di input
- Preparazione del formato di risposta
- Strutture comuni nell'evento e nella risposta Lambda
- Creazione e associazione di una funzione Lambda a un alias bot
- Debugging della funzione Lambda

Interpretazione del formato dell'evento di input

Il primo passaggio per integrare una funzione Lambda nel tuo bot Amazon Lex V2 consiste nel comprendere i campi dell'evento Amazon Lex V2 e determinare le informazioni da questi campi che desideri utilizzare durante la scrittura dello script. Il seguente oggetto JSON mostra il formato generale di un evento Amazon Lex V2 passato a una funzione Lambda:

> **Note**
>
> Il formato di input può cambiare senza una modifica corrispondente a. `messageVersion` Il codice non dovrebbe generare un errore se sono presenti nuovi campi.

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook | FulfillmentCodeHook",
    "inputMode": "DTMF | Speech | Text",
    "responseContentType": "audio/mpeg | audio/ogg | audio/pcm | text/plain;
 charset=utf-8",
    "sessionId": string,
    "inputTranscript": string,
    "invocationLabel": string,
    "bot": {
        "id": string,
        "name": string,
        "localeId": string,
        "version": string,
        "aliasId": string,
        "aliasName": string
    },
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "intent": {
                // see Intento for details about the structure
            },
            "nluConfidence": number,
            "sentimentResponse": {
                "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
                "sentimentScore": {
                    "mixed": number,
```

```
                    "negative": number,
                    "neutral": number,
                    "positive": number
                }
            }
        },
        ...
    ],
    "proposedNextState": {
        "dialogAction": {
            "slotToElicit": string,
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
        },
        "intent": {
            // see Intento for details about the structure
        },
        "prompt": {
            "attempt": string
        }
    },
    "requestAttributes": {
        string: string,
        ...
    },
    "sessionState": {
        // see Stato della sessione for details about the structure
    },
    "transcriptions": [
        {
            "transcription": string,
            "transcriptionConfidence": number,
            "resolvedContext": {
                "intent": string
            },
            "resolvedSlots": {
                slot name: {
                    // see Slot for details about the structure
                },
                ...
            }
        },
        ...
    ]
```

```
}
```

Ogni campo dell'evento di input è descritto di seguito:

messageVersion

La versione del messaggio che identifica il formato dei dati degli eventi che entrano nella funzione Lambda e il formato previsto della risposta da una funzione Lambda.

> **Note**
>
> Puoi configurare questo valore quando definisci un intento. Nell'attuale implementazione, Amazon Lex V2 supporta solo la versione 1.0 dei messaggi. Pertanto, la console presuppone il valore predefinito di 1.0 e non mostra la versione del messaggio.

invocationSource

Il code hook che ha chiamato la funzione Lambda. I valori possibili sono i seguenti:

`DialogCodeHook`— Amazon Lex V2 ha chiamato la funzione Lambda dopo l'input dell'utente.

`FulfillmentCodeHook`— Amazon Lex V2 ha richiamato la funzione Lambda dopo aver riempito tutti gli slot richiesti e l'intento è pronto per l'adempimento.

Modalità di input

La modalità di espressione dell'utente. I valori possibili sono i seguenti:

`DTMF`— L'utente inserisce l'enunciato utilizzando una tastiera tattile (Dual Tone Multi-Frequency).

`Speech`— L'utente ha pronunciato l'enunciato.

`Text`— L'utente ha digitato l'enunciato.

responseContentType

La modalità di risposta del bot all'utente. `text/plain; charset=utf-8`indica che l'ultima espressione è stata scritta, mentre un valore che inizia con `audio` indica che l'ultima espressione è stata pronunciata.

sessionId

L'identificatore alfanumerico di sessione utilizzato per la conversazione.

inputTranscript

Una trascrizione dell'input dell'utente.

- Per l'immissione di testo, si tratta del testo digitato dall'utente. Per l'input DTMF, questa è la chiave immessa dall'utente.
- Per l'input vocale, questo è il testo in cui Amazon Lex V2 converte l'enunciato dell'utente per richiamare un intento o riempire uno slot.

Etichetta di chiamata

Un valore che indica la risposta che ha richiamato la funzione Lambda. È possibile impostare etichette di chiamata per la risposta iniziale, gli slot e la risposta di conferma.

bot

Informazioni sul bot che ha elaborato la richiesta, costituite dai seguenti campi:

- id — L'identificatore assegnato al bot al momento della sua creazione. Puoi visualizzare l'ID del bot nella console Amazon Lex V2 nella pagina delle impostazioni del bot.
- nome: il nome che hai dato al bot quando lo hai creato.
- localeID: l'identificatore del locale che hai usato per il bot. Per un elenco delle impostazioni locali, consulta. Lingue e impostazioni locali supportate da Amazon Lex V2
- version — La versione del bot che ha elaborato la richiesta.
- aliasID: l'identificatore assegnato all'alias del bot al momento della creazione. Puoi visualizzare l'ID alias del bot nella console Amazon Lex V2 nella pagina Alias. Se non riesci a visualizzare l'ID dell'alias nell'elenco, scegli l'icona a forma di ingranaggio in alto a destra e attiva l'ID alias.
- AliasName: il nome che hai assegnato all'alias del bot.

interpretazioni

Un elenco di informazioni sugli intenti che Amazon Lex V2 considera possibili corrispondenze all'enunciato dell'utente. Ogni elemento è una struttura che fornisce informazioni sulla corrispondenza dell'enunciato a un intento, con il seguente formato:

```
{
    "intent": {
        // see Intento for details about the structure
    },
    "interpretationSource": "Bedrock | Lex",
    "nluConfidence": number,
    "sentimentResponse": {
        "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
        "sentimentScore": {
            "mixed": number,
            "negative": number,
            "neutral": number,
            "positive": number
        }
    }
}
```

I campi all'interno della struttura sono i seguenti:

- intento: una struttura contenente informazioni sull'intento. IntentoPer ulteriori dettagli sulla struttura, vedere.
- NLUConfidence: un punteggio che indica quanto Amazon Lex V2 sia sicuro che l'intento corrisponda all'intento dell'utente.
- SentimentResponse — Un'analisi del sentimento della risposta, contenente i seguenti campi:
  - sentiment: indica se il sentimento dell'enunciato è,,, o. `POSITIVE` `NEGATIVE` `NEUTRAL` `MIXED`
  - SentimentScore: una struttura che associa ogni sentimento a un numero che indica quanto Amazon Lex V2 sia sicuro che l'enunciato trasmetta quel sentimento.
- InterpretationSource: indica se uno slot viene risolto da Amazon Lex o Amazon Bedrock.

proposedNextState

Se la funzione Lambda imposta `dialogAction` of `sessionState` to`Delegate`, questo campo viene visualizzato e mostra la proposta di Amazon Lex V2 per il passaggio successivo della conversazione. Altrimenti, lo stato successivo dipende dalle impostazioni restituite nella risposta della funzione Lambda. Questa struttura è presente solo se entrambe le seguenti affermazioni sono vere:

1. Il `invocationSource` valore è `DialogCodeHook`
2. Il previsto `type` di `dialogAction` è`ElicitSlot`.

Puoi utilizzare queste informazioni per aggiungere `runtimeHints` qualcosa al momento giusto della conversazione. Miglioramento del riconoscimento dei valori degli slot con suggerimenti di runtimePer ulteriori informazioni, consulta. `proposedNextState`è una struttura contenente i seguenti campi:

La struttura di `proposedNextState` è la seguente:

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Intento for details about the structure
    },
    "prompt": {
        "attempt": string
    }
}
```

- DialogAction: contiene informazioni sulla fase successiva proposta da Amazon Lex V2. I campi della struttura sono i seguenti:
  - slotToElicit— Lo slot da ottenere successivamente, come proposto da Amazon Lex V2. Questo campo viene visualizzato solo se è`type`. `ElicitSlot`
  - type — La fase successiva della conversazione proposta da Amazon Lex V2. I valori possibili sono i seguenti:

    `Delegate`— Amazon Lex V2 determina l'azione successiva.

    `ElicitIntent`— L'azione successiva consiste nel suscitare un intento da parte dell'utente.

    `ElicitSlot`— L'azione successiva consiste nell'ottenere un valore di slot dall'utente.

    `Close`— Termina il processo di adempimento degli intenti e indica che non ci sarà alcuna risposta da parte dell'utente.

    `ConfirmIntent`— L'azione successiva consiste nel chiedere all'utente se gli slot sono corretti e l'intento è pronto per essere soddisfatto.
- intento: l'intento che il bot ha determinato che l'utente sta cercando di soddisfare. Vedi Intento per i dettagli sulla struttura.

- prompt: una struttura contenente il campo`attempt`, che corrisponde a un valore che specifica quante volte Amazon Lex V2 ha richiesto all'utente lo slot successivo. I valori possibili sono `Initial` per il primo tentativo e`Retry1`,, `Retry2` `Retry3`Retry4, e `Retry5` per i tentativi successivi.

requestAttributes

Una struttura contenente gli 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, consulta Impostazione degli attributi della richiesta.

SessionState

Lo stato attuale della conversazione tra l'utente e il tuo bot Amazon Lex V2. Stato della sessionePer ulteriori dettagli sulla struttura, consulta.

trascrizioni

Un elenco di trascrizioni che Amazon Lex V2 considera possibili corrispondenze all'enunciato dell'utente. Per ulteriori informazioni, consulta Utilizzo dei punteggi di confidenza nella trascrizione vocale. Ogni elemento è un oggetto con il seguente formato, contenente informazioni su una possibile trascrizione:

```
{
    "transcription": string,
    "transcriptionConfidence": number,
    "resolvedContext": {
        "intent": string
    },
    "resolvedSlots": {
        slot name: {
            // see Slot for details about the structure
        },
        ...
    }
}
```

I campi sono descritti di seguito:

- trascrizione: una trascrizione che Amazon Lex V2 considera una possibile corrispondenza con l'enunciato audio dell'utente.
- TranscriptionConfidence: un punteggio che indica quanto Amazon Lex V2 sia sicuro che l'intento corrisponda all'intento dell'utente.
- ResolvedContext: una struttura contenente il campo`intent`, che corrisponde all'intento a cui si riferisce l'enunciato.
- resolvedSlots — Una struttura le cui chiavi sono i nomi di ogni slot che viene risolto dall'enunciato. Ogni nome di slot corrisponde a una struttura contenente informazioni su quello slot. Vedi Slot per i dettagli sulla struttura.

Preparazione del formato di risposta

Il secondo passaggio per integrare una funzione Lambda nel tuo bot Amazon Lex V2 consiste nel comprendere i campi nella risposta della funzione Lambda e determinare quali parametri desideri manipolare. Il seguente oggetto JSON mostra il formato generale di una risposta Lambda restituita ad Amazon Lex V2:

```
{
    "sessionState": {
        // see Stato della sessione for details about the structure
    },
    "messages": [
        {
            "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
            "content": string,
            "imageResponseCard": {
                "title": string,
                "subtitle": string,
                "imageUrl": string,
                "buttons": [
                    {
                        "text": string,
                        "value": string
                    },
                    ...
                ]
            }
        },
        ...
```

```
    ],
    "requestAttributes": {
        string: string,
        ...
    }
}
```

Ogni campo della risposta è descritto di seguito:

Stato della sessione

Lo stato della conversazione tra l'utente e il bot Amazon Lex V2 che desideri restituire. Stato della sessionePer ulteriori dettagli sulla struttura, consulta. Questo campo è sempre obbligatorio.

messaggi

Un elenco di messaggi che Amazon Lex V2 restituisce al cliente per la fase successiva della conversazione. Se il messaggio `contentType` che fornisci è `PlainTextCustomPayload`, o`SSML`, scrivi il messaggio che desideri restituire al cliente nel `content` campo. Se `contentType` lo fornisci`ImageResponseCard`, inserisci i dettagli della carta nel `imageResponseCard` campo. Se non fornisci messaggi, Amazon Lex V2 utilizza il messaggio appropriato definito al momento della creazione del bot.

Il `messages` campo è obbligatorio se `dialogAction.type` è `ElicitIntent` o`ConfirmIntent`.

Ogni elemento dell'elenco è una struttura nel formato seguente, contenente informazioni su un messaggio da restituire all'utente. Ecco un esempio:

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
                "value": string
            },
            ...
        ]
```

```
    }
}
```

Di seguito viene fornita una descrizione per ogni campo:

- contentType: il tipo di messaggio da utilizzare.

  `CustomPayload`— Una stringa di risposta che è possibile personalizzare per includere dati o metadati per l'applicazione.

  `ImageResponseCard`— Un'immagine con pulsanti selezionabili dal cliente. Vedi ImageResponseCardper ulteriori informazioni.

  `PlainText`— Una stringa di testo semplice.

  `SSML`— Una stringa che include Speech Synthesis Markup Language per personalizzare la risposta audio.
- content — Il messaggio da inviare all'utente. Utilizza questo campo se il tipo di messaggio è `PlainTextCustomPayload`, o`SSML`.
- imageResponseCard— Contiene la definizione della scheda di risposta da mostrare all'utente. Utilizza questo campo se il tipo di messaggio è`ImageResponseCard`. Esegue il mapping su una struttura contenente i seguenti campi:
  - title: il titolo della scheda di risposta.
  - sottotitolo: la richiesta all'utente di scegliere un pulsante.
  - ImageURL — Un collegamento a un'immagine per la scheda.
  - buttons — Un elenco di strutture contenenti informazioni su un pulsante. Ogni struttura contiene un `text` campo con il testo da visualizzare e un `value` campo con il valore da inviare ad Amazon Lex V2 se il cliente seleziona quel pulsante. Puoi includere fino a tre pulsanti.

requestAttributes

Una struttura contenente gli attributi specifici della richiesta per la risposta al cliente. Per ulteriori informazioni, consulta Impostazione degli attributi della richiesta. Questo campo è facoltativo.

Campi obbligatori nella risposta

Come minimo, la risposta Lambda deve includere `sessionState` un oggetto. All'interno di ciò, fornisci un `dialogAction` oggetto e specifica il `type` campo. A seconda `type` di `dialogAction`

quello fornito, potrebbero esserci altri campi obbligatori per la risposta Lambda. Questi requisiti sono descritti di seguito, insieme a esempi di funzionamento minimi:

Delegato

Delegate consente ad Amazon Lex V2 di determinare il passaggio successivo. Non sono richiesti altri campi.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntentrichiede al cliente di esprimere un'intenzione. È necessario includere almeno un messaggio nel `messages` campo per sollecitare l'individuazione di un intento.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlotrichiede al cliente di fornire un valore di slot. È necessario includere il nome dello slot nel `slotToElicit` campo dell'oggetto. `dialogAction` È inoltre necessario includere il `name` di `intent` nell'`sessionState`oggetto.

```
{`
    "sessionState": {
        "dialogAction": {
```

```
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
    }
}
```

ConfirmIntent

ConfirmIntentconferma i valori degli slot del cliente e se l'intento è pronto per essere soddisfatto. È necessario includere il `name` di `intent` nell'`sessionState`oggetto e il codice `slots` da confermare. È inoltre necessario includere almeno un messaggio nel `messages` campo per chiedere all'utente la conferma dei valori dello slot. Il messaggio dovrebbe richiedere una risposta «sì» o «no». Se l'utente risponde «sì», Amazon Lex V2 imposta l'`confirmationState`intento su. `Confirmed` Se l'utente risponde «no», Amazon Lex V2 imposta l'`confirmationState`intento su. `Denied`

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
        "intent": {
            "name": "BookFlight",
            "slots": {
                "DepartureDate": {
                    "value": {
                        "originalValue": "tomorrow",
                        "interpretedValue": "2023-05-09",
                        "resolvedValues": [
                            "2023-05-09"
                        ]
                    }
                },
                "DestinationCity": {
                    "value": {
                        "originalValue": "sf",
                        "interpretedValue": "sf",
                        "resolvedValues": [
                            "sf"
                        ]
                    }
```

```
                },
                "OriginCity": {
                    "value": {
                        "originalValue": "nyc",
                        "interpretedValue": "nyc",
                        "resolvedValues": [
                            "nyc"
                        ]
                    }
                }
            }
        }
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "Okay, you want to fly from {OriginCity} to \
            {DestinationCity} on {DepartureDate}. Is that correct?"
        }
    ]
}
```

Chiudi

Close termina il processo di adempimento dell'intento e indica che non sono previste ulteriori risposte da parte dell'utente. È necessario includere la `name` e `state` del `intent` nell'oggetto. `sessionState` Gli stati di intento compatibili sono `FailedFulfilled`, `eInProgress`.

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Strutture comuni nell'evento e nella risposta Lambda

All'interno della risposta Lambda, ci sono diverse strutture che si ripetono. I dettagli su queste strutture comuni sono forniti in questa sezione.

Intento

```
"intent": {
    "confirmationState": "Confirmed | Denied | None",
    "name": string,
    "slots": {
        // see Slot for details about the structure
    },
    "state": "Failed | Fulfilled | FulfillmentInProgress | InProgress |
 ReadyForFulfillment | Waiting",
    "kendraResponse": {
        // Only present when intent is KendraSearchIntent. For details, see
// https://docs.aws.amazon.com/kendra/latest/dg/API_Query.html#API_Query_ResponseSyntax
       }
}
```

Il `intent` campo è mappato su un oggetto con i seguenti campi:

Stato di conferma

Indica se l'utente ha confermato gli slot per l'intento e l'intento è pronto per l'adempimento. I valori possibili sono i seguenti:

`Confirmed`— L'utente conferma che i valori degli slot sono corretti.

`Denied`— L'utente indica che i valori dello slot non sono corretti.

`None`— L'utente non ha ancora raggiunto la fase di conferma.

nome

Il nome dell'intento.

slots

Informazioni sugli slot necessari per soddisfare l'intento. Vedi Slot per i dettagli sulla struttura.

stato

Indica lo stato di adempimento dell'intento. I valori possibili sono i seguenti:

`Failed`— Il bot non è riuscito a soddisfare l'intento.

`Fulfilled`— Il bot ha completato l'adempimento dell'intento.

`FulfillmentInProgress`— Il bot è nel bel mezzo del raggiungimento dell'intento.

`InProgress`— Il bot sta cercando di ottenere i valori dello slot necessari per soddisfare l'intento.

`ReadyForFulfillment`— Il bot ha ottenuto tutti i valori dello slot per l'intento ed è pronto a soddisfare l'intento.

`Waiting`— Il bot è in attesa di una risposta dall'utente (limitata alle conversazioni in streaming).

kendraResponse

Contiene informazioni sui risultati della query di ricerca di Kendra. Questo campo viene visualizzato solo se l'intento è un. `KendraSearchIntent` Per ulteriori informazioni, consulta la sintassi della risposta nella chiamata all'API Query per Kendra.

Slot

Il `slots` campo esiste all'interno di una `intent` struttura ed è mappato su una struttura le cui chiavi sono i nomi degli slot relativi a tale scopo. Se lo slot non è uno slot multivalore (vedi Utilizzo di più valori in uno slot per maggiori dettagli), viene mappato su una struttura con il seguente formato. Nota che lo è. `shape` `Scalar`

```
{
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    }
}
```

Se lo slot è uno slot multivalore, l'oggetto a cui è mappato contiene un altro campo chiamato`values`, che è mappato su un elenco di strutture, ciascuna contenente informazioni su uno slot che costituisce

lo slot multivalore. Il formato di ogni oggetto nell'elenco corrisponde a quello dell'oggetto a cui è mappato uno slot normale. Nota che lo `shape` è`List`, ma lo `values` è `Scalar` lo slot `shape` dei componenti sottostanti.

```
{
    slot name: {
    "shape": "List",
    "value": {
        "originalValue": string,
        "interpretedValue": string,
        "resolvedValues": [
            string,
            ...
        ]
    },
    "values": [
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        ...
    ]
}
```

I campi dell'oggetto slot sono descritti di seguito:

shape

La forma dello slot. Questo valore è `List` se ci sono più valori nello slot (vedi Utilizzo di più valori in uno slot per maggiori dettagli) e non lo `Scalar` è.

value

Un oggetto contenente informazioni sul valore fornito dall'utente per uno slot e sull'interpretazione di Amazon Lex, nel seguente formato:

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
        ...
    ]
}
```

I campi sono descritti di seguito:

- OriginalValue: la parte della risposta dell'utente all'elicitazione dello slot che Amazon Lex determina è rilevante per il valore dello slot.
- InterpretedValue: il valore che Amazon Lex determina per lo slot, in base all'input dell'utente.
- ResolvedValues: un elenco di valori che Amazon Lex determina come possibili risoluzioni per l'input dell'utente.

values

Un elenco di oggetti contenente informazioni sugli slot che compongono lo slot multivalore. Il formato di ogni oggetto corrisponde a quello di uno slot normale, con i `value` campi `shape` e descritti sopra. `values`appare solo se lo slot è composto da più valori (vedi Utilizzo di più valori in uno slot per maggiori dettagli). Il seguente oggetto JSON mostra due slot componenti:

```
"values": [
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
```

```
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    ...
]
```

Stato della sessione

Il `sessionState` campo è mappato su un oggetto contenente informazioni sullo stato della conversazione con l'utente. I campi effettivi che appaiono nell'oggetto dipendono dal tipo di azione di dialogo. Vedi Campi obbligatori nella risposta i campi obbligatori in una risposta Lambda. Il formato dell'`sessionState`oggetto è il seguente:

```
"sessionState": {
    "activeContexts": [
        {
            "name": string,
            "contextAttributes": {
                string: string
            },
            "timeToLive": {
                "timeToLiveInSeconds": number,
                "turnsToLive": number
            }
        },
        ...
    ],
    "sessionAttributes": {
        string: string,
```

```
        ...
    },
    "runtimeHints": {
        "slotHints": {
            intent name: {
                slot name: {
                    "runtimeHintValues": [
                        {
                            "phrase": string
                        },
                        ...
                    ]
                },
                ...
            },
            ...
        }
    },
    "dialogAction": {
        "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Intento for details about the structure
    },
    "originatingRequestId": string
}
```

I campi sono descritti di seguito:

Contesti attivi

Un elenco di oggetti contenenti informazioni su un contesto utilizzato da un utente in una sessione. Utilizza i contesti per facilitare e controllare il riconoscimento degli intenti. Per ulteriori informazioni sui contesti, vedere. Impostazione del contesto dell'intento Ogni oggetto è formattato come segue:

```
{
    "name": string,
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
```

```
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

I campi sono descritti di seguito:

- name: il nome del contesto.
- contextAttributes — Un oggetto contenente i nomi degli attributi per il contesto e i valori a cui sono mappati.
- timeToLive— Un oggetto che specifica per quanto tempo il contesto rimane attivo. Questo oggetto può contenere uno o entrambi i seguenti campi:
  - timeToLiveInSeconds— Il numero di secondi in cui il contesto rimane attivo.
  - turnsToLive— Il numero di turni in cui il contesto rimane attivo.

sessionAttributes

Una mappa di coppie chiave/valore che rappresentano informazioni di contesto specifiche della sessione. Per ulteriori informazioni, consulta Impostazione degli attributi della sessione. L'oggetto è formattato come segue:

```
{
    string: string,
    ...
}
```

RuntimeHints

Fornisce suggerimenti sulle frasi che un cliente probabilmente utilizzerà per uno slot per migliorare il riconoscimento audio. I valori forniti nei suggerimenti aumentano il riconoscimento audio di tali valori rispetto a parole dal suono simile. Il formato dell'`runtimeHints`oggetto è il seguente:

```
{
    "slotHints": {
        intent name: {
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
```

```
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

Il `slotHints` campo è mappato su un oggetto i cui campi sono i nomi degli intenti nel bot. Ogni nome di intento è mappato a un oggetto i cui campi sono i nomi degli slot per quell'intento. Ogni nome di slot è mappato a una struttura con un singolo campo`runtimeHintValues`, che è un elenco di oggetti. Ogni oggetto contiene un `phrase` campo che corrisponde a un suggerimento.

dialogAction

Determina l'azione successiva da intraprendere per Amazon Lex V2. Il formato dell'oggetto è il seguente:

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

I campi sono descritti di seguito:

- slotElicitationStyle— Determina il modo in cui Amazon Lex V2 interpreta l'input audio dell'utente, se l'`type`o è. `dialogAction` `ElicitSlot` Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia. I valori possibili sono i seguenti:

  `Default`— Amazon Lex V2 interpreta l'ingresso audio nel modo predefinito per soddisfare uno slot.

  `SpellByLetter`— Amazon Lex V2 ascolta l'ortografia del valore dello slot da parte dell'utente.

  `SpellByWord`— Amazon Lex V2 ascolta l'ortografia del valore dello slot da parte dell'utente utilizzando parole associate a ciascuna lettera (ad esempio, «a as in apple»).
- slotToElicit— Definisce lo slot da richiedere all'utente se l'of è. `type` `dialogAction` `ElicitSlot`
- tipo — Definisce l'azione che il bot deve eseguire. I valori possibili sono i seguenti:

`Delegate`— Consente ad Amazon Lex V2 di determinare il passaggio successivo.

`ElicitIntent`— Richiede al cliente di esprimere un'intenzione.

`ConfirmIntent`— Conferma i valori degli slot assegnati al cliente e se l'intento è pronto per l'adempimento.

`ElicitSlot`— Richiede al cliente di fornire un valore di slot per un intento.

`Close`— Termina il processo di adempimento degli intenti.

intento

Vedi Intento la struttura del `intent` campo.

originatingRequestId

Un identificatore univoco per la richiesta. Questo campo è facoltativo per la risposta Lambda.

Creazione e associazione di una funzione Lambda a un alias bot

Creazione della funzione Lambda

Per creare una funzione Lambda per il tuo bot Amazon Lex V2, accedi AWS Lambda dal tuo AWS Management Console e crea una nuova funzione. Puoi fare riferimento alla guida per AWS Lambda sviluppatori per maggiori dettagli in merito. AWS Lambda

1. Accedi a AWS Management Console e apri la console AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegli Funzioni nella barra laterale sinistra.
3. Seleziona Crea funzione.
4. Puoi selezionare Author partendo da zero per iniziare con un codice minimo, Usa un blueprint per selezionare da un elenco il codice di esempio per i casi d'uso più comuni o Container image per selezionare un'immagine del contenitore da distribuire per la tua funzione. Se selezioni Author da zero, continua con i seguenti passaggi:
   a. Assegna alla funzione un nome di funzione significativo per descrivere cosa fa.
   b. Scegli una lingua dal menu a discesa sotto Runtime in cui scrivere la tua funzione.

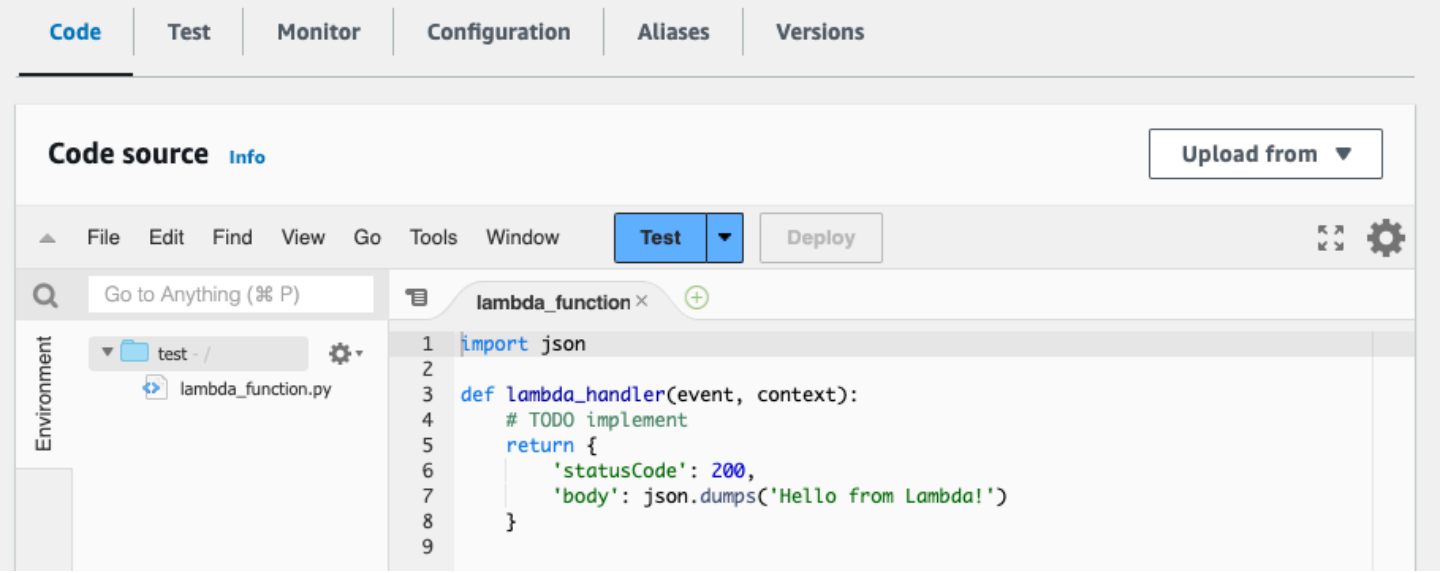
c. Seleziona un set di istruzioni Architecture per la tua funzione.

d. Per impostazione predefinita, Lambda crea un ruolo con autorizzazioni di base. Per utilizzare un ruolo esistente o creare un ruolo utilizzando modelli di AWS policy, espandi il menu Cambia il ruolo di esecuzione predefinito e seleziona un'opzione.

e. Espandi il menu Impostazioni avanzate per configurare altre opzioni.

5. Seleziona Crea funzione.

L'immagine seguente mostra ciò che vedi quando crei una nuova funzione da zero:



La funzione di gestione Lambda varia a seconda della lingua utilizzata. Accetta almeno un oggetto `event` JSON come argomento. Puoi visualizzare i campi forniti da Amazon Lex V2 all'indirizzoInterpretazione del formato dell'evento di input. `event` Modifica la funzione di gestione per restituire infine un oggetto `response` JSON che corrisponda al formato descritto in. Preparazione del formato di risposta

Una volta terminata la scrittura della funzione, selezionate Deploy per consentire l'utilizzo della funzione.

Ricorda che puoi associare ogni alias del bot al massimo a una funzione Lambda. Tuttavia, puoi definire tutte le funzioni necessarie per il tuo bot all'interno del codice Lambda e richiamare queste funzioni nella funzione di gestione Lambda. Ad esempio, mentre tutti gli intenti nello stesso alias bot devono chiamare la stessa funzione Lambda, puoi creare una funzione router che attivi una funzione separata per ogni intento. Di seguito è riportato un esempio di funzione router che potete utilizzare o modificare per la vostra 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
```

Aggiungere e richiamare una funzione Lambda

Per chiamare la funzione Lambda nel tuo bot Amazon Lex V2, devi prima collegare la funzione a un alias bot e quindi impostare i punti della conversazione in cui il bot richiama la funzione. Puoi eseguire questi passaggi con le operazioni della console o dell'API.

È possibile utilizzare le funzioni Lambda nei seguenti punti di una conversazione con un utente:

- Nella risposta iniziale dopo il riconoscimento dell'intento. Ad esempio, dopo che l'utente ha dichiarato di voler ordinare una pizza.
- Dopo aver ottenuto il valore di uno slot dall'utente. Ad esempio, dopo che l'utente ha detto al bot la dimensione della pizza che desidera ordinare.
- Tra un tentativo e l'altro per ottenere uno slot. Ad esempio, se il cliente non utilizza una dimensione di pizza riconosciuta.
- Quando si conferma un intento. Ad esempio, quando si conferma un ordine di pizza.

- Per soddisfare un intento. Ad esempio, per ordinare una pizza.
- Dopo il raggiungimento dell'intento e prima che il bot chiuda la conversazione. Ad esempio, per passare all'intenzione di ordinare un drink.

Argomenti

Utilizzo della console

Associa una funzione Lambda a un alias bot

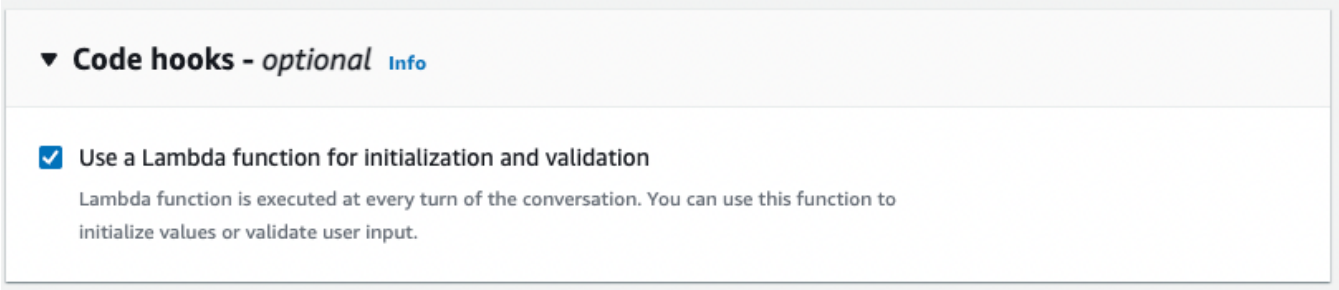
1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli Bot dal pannello laterale sinistro e dall'elenco dei bot, scegli il nome del bot a cui vuoi collegare una funzione Lambda.
3. Dal pannello laterale sinistro, seleziona Alias nel menu Deployment.
4. Dall'elenco degli alias, scegli il nome dell'alias a cui desideri associare una funzione Lambda.
5. Nel pannello Lingue, seleziona la lingua in cui desideri utilizzare una funzione Lambda. Seleziona Gestisci le lingue nell'alias per aggiungere una lingua se non è presente nel pannello.
6. Nel menu a discesa Source, scegli il nome della funzione Lambda che desideri allegare.
7. Nel menu a discesa versione o alias della funzione Lambda, scegli la versione o l'alias della funzione Lambda che desideri utilizzare. Quindi, seleziona Salva. La stessa funzione Lambda viene utilizzata a tutti gli effetti in un linguaggio supportato dal bot.

Imposta l'intento di richiamare la funzione Lambda

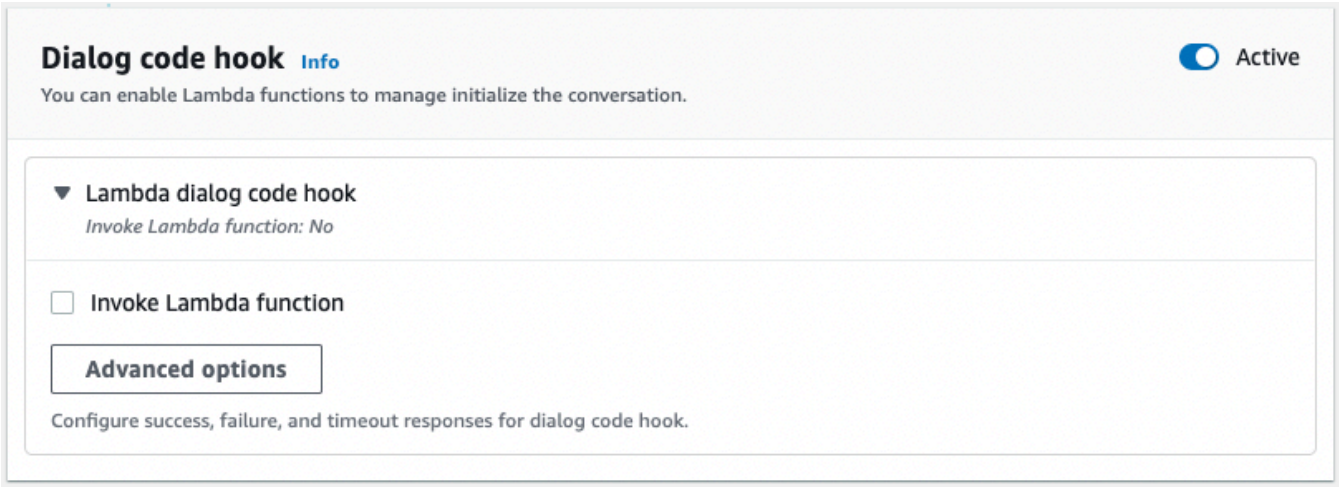
1. Dopo aver selezionato un bot, seleziona Intents nel menu a sinistra sotto la lingua del bot per cui desideri richiamare la funzione Lambda.
2. Scegliete l'intento in cui desiderate richiamare la funzione Lambda per aprire l'editor degli intenti.
3. Esistono due opzioni per impostare il code hook Lambda:

    1. Per richiamare la funzione Lambda dopo ogni fase della conversazione, scorri fino alla sezione Code hook nella parte inferiore dell'editor degli intenti e seleziona la casella di

controllo Usa una funzione Lambda per l'inizializzazione e la convalida, come nell'immagine seguente:



2. In alternativa, utilizzate la sezione Dialog code hook nelle fasi di conversazione in cui richiamare la funzione Lambda. La sezione Dialog code hook viene visualizzata come segue:



Esistono due modi per controllare il modo in cui Amazon Lex V2 chiama il code hook per una risposta:

- Attiva il pulsante Attivo per contrassegnarlo come attivo o inattivo. Quando un code hook è attivo, Amazon Lex V2 lo richiamerà. Quando il code hook è inattivo, Amazon Lex V2 non lo esegue.
- Espandi la sezione Lambda dialog code hook e seleziona la casella di controllo Invoke Lambda function per contrassegnarla come abilitata o disabilitata. Puoi abilitare o disabilitare un code hook solo quando è contrassegnato come attivo. Quando è contrassegnato come abilitato, il code hook viene eseguito normalmente. Quando è disabilitato, l'hook di codice non viene chiamato e Amazon Lex V2 agisce come se fosse stato restituito correttamente. Per configurare le risposte dopo l'esito positivo, negativo o il timeout dell'hook di dialogo, seleziona Opzioni avanzate

L'hook di codice Lambda può essere richiamato nelle seguenti fasi di conversazione:

- Per richiamare la funzione come risposta iniziale, scorri fino alla sezione Risposta iniziale, espandi la freccia accanto a Risposta per confermare la richiesta dell'utente e seleziona Opzioni avanzate. Trova la sezione Dialog code hook nella parte inferiore del menu che si apre.
- Per richiamare la funzione dopo l'attivazione dello slot, scorri fino alla sezione Slot, espandi la freccia accanto al relativo prompt for slot e seleziona Opzioni avanzate. Trova la sezione Dialog code hook nella parte inferiore del menu che si apre, appena sopra i valori predefiniti.

  Puoi anche richiamare la funzione dopo ogni elicitazione. A tale scopo, espandi Bot elicits information nella sezione Slot prompt, seleziona Altre opzioni di prompt e seleziona la casella di controllo accanto a Invoke Lambda code hook dopo ogni elicitazione.
- Per richiamare la funzione di conferma dell'intento, scorri fino alla sezione Conferma, espandi la freccia accanto a Richieste per confermare l'intento e seleziona Opzioni avanzate. Trova la sezione Dialog code hook nella parte inferiore del menu che si apre.
- Per richiamare la funzione per l'adempimento degli intenti, scorri fino alla sezione Adempimento. Attiva il pulsante Attivo per attivare il code hook. Espandi la freccia accanto a In caso di evasione avvenuta con successo e seleziona Opzioni avanzate. Seleziona la casella di controllo accanto a Usa una funzione Lambda per l'adempimento nella sezione Fulfillment Lambda code hook per impostare l'hook di codice su abilitato.

4. Dopo aver impostato le fasi della conversazione in cui richiamare la funzione Lambda, crea nuovamente il bot per testare la funzione.

Utilizzo delle operazioni API

Associa una funzione Lambda a un alias bot

Se stai creando un nuovo alias bot, usa l'CreateBotAliasoperazione per allegare una funzione Lambda. Per collegare una funzione Lambda a un alias bot esistente, usa l'operazione. UpdateBotAlias Modifica il `botAliasLocaleSettings` campo in modo che contenga le impostazioni corrette:

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
                "lambdaCodeHook": {
```

```
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. Il `botAliasLocaleSettings` campo è mappato su un oggetto le cui chiavi sono le impostazioni locali in cui si desidera collegare la funzione Lambda. Consultate Lingue e impostazioni locali supportate per un elenco delle impostazioni locali supportate e dei codici che costituiscono chiavi valide.
2. Per trovare la funzione `lambdaARN` for a Lambda, apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/home, seleziona Funzioni nella barra laterale sinistra e seleziona la funzione da associare all'alias bot. Sul lato destro della panoramica delle funzioni, trova la `lambdaARN` sezione Funzione ARN. Dovrebbe contenere una regione, un ID account e il nome della funzione.
3. Per consentire ad Amazon Lex V2 di richiamare la funzione Lambda per l'alias, imposta il campo su. `enabled` `true`

Imposta l'intento di richiamare la funzione Lambda

Per impostare la chiamata della funzione Lambda durante un intento, usa l'CreateIntentoperazione se stai creando un nuovo intento o l'UpdateIntentoperazione se stai invocando la funzione in un intento esistente. I campi che controllano l'invocazione della funzione Lambda nelle operazioni intent sono`dialogCodeHook`,, e. `initialResponseSetting` `intentConfirmationSetting` `fulfillmentCodeHook`

Se richiamate la funzione durante l'elicitazione di uno slot, utilizzate l'CreateSlotoperazione se state creando un nuovo slot o l'UpdateSlotoperazione per richiamare la funzione in uno slot esistente. Il campo che controlla l'invocazione della funzione Lambda nelle operazioni dello slot è l'`slotCaptureSetting`oggetto. `valueElicitationSetting`

1. Per impostare l'hook del codice di dialogo Lambda in modo che venga eseguito dopo ogni turno della conversazione, imposta il `enabled` campo del seguente DialogCodeHookSettingsoggetto nel `dialogCodeHook` campo su: `true`

```
"dialogCodeHook": {
    "enabled": boolean
}
```

2. In alternativa, puoi impostare l'hook di dialogo Lambda in modo che venga eseguito solo in punti specifici delle conversazioni modificando il `elicitationCodeHook` campo `codeHook` e/o all'interno delle strutture che corrispondono alle fasi della conversazione in cui desideri richiamare la funzione. Per utilizzare l'hook di codici di dialogo Lambda per l'adempimento degli intenti, usa il `fulfillmentCodeHook` campo nell'operazione or. CreateIntentUpdateIntent Le strutture e gli usi di questi tre tipi di code hook sono i seguenti:

CodeHook

Il `codeHook` campo definisce le impostazioni per l'esecuzione del code hook in una determinata fase della conversazione. È un DialogCodeHookInvocationSettingoggetto con la seguente struttura:

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- Cambia il `active` campo in modo `true` che Amazon Lex V2 richiami il code hook in quel momento della conversazione.
- Cambia il `enableCodeHookInvocation` campo in `true` Amazon Lex V2 per consentire al code hook di funzionare normalmente. Se lo contrassegni`false`, Amazon Lex V2 si comporta come se il code hook fosse stato restituito correttamente.
- `invocationLabel`Indica la fase di dialogo da cui viene richiamato l'hook di codice.
- Utilizzate il `postCodeHookSpecification` campo per specificare le azioni e i messaggi che si verificano dopo il successo, l'errore o il timeout del code hook.

elicitationCodeHook

Il `elicitationCodeHook` campo definisce le impostazioni per l'esecuzione del code hook nel caso in cui sia necessario richiamare nuovamente uno slot o più slot. Questo scenario può verificarsi se l'elicitazione degli slot fallisce o la conferma dell'intento viene negata. Il `elicitationCodeHook` campo è un ElicitationCodeHookInvocationSettingoggetto con la seguente struttura:

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- Cambia il `enableCodeHookInvocation` campo in `true` Amazon Lex V2 per consentire al code hook di funzionare normalmente. Se lo contrassegni`false`, Amazon Lex V2 si comporta come se il code hook fosse stato restituito correttamente.
- `invocationLabel`Indica la fase di dialogo da cui viene richiamato l'hook di codice.

fulfillmentCodeHook

Il `fulfillmentCodeHook` campo definisce le impostazioni per l'esecuzione del code hook per soddisfare l'intento. È mappato al seguente FulfillmentCodeHookSettingsoggetto:

```
"fulfillmentCodeHook": {
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- Cambia il `active` campo in modo `true` che Amazon Lex V2 richiami il code hook in quel momento della conversazione.
- Cambia il `enabled` campo in `true` Amazon Lex V2 per consentire al code hook di funzionare normalmente. Se lo contrassegni`false`, Amazon Lex V2 si comporta come se il code hook fosse stato restituito correttamente.
- Utilizza il `fulfillmentUpdatesSpecification` campo per specificare i messaggi che appaiono per aggiornare l'utente durante l'adempimento dell'intento e la tempistica ad essi associata.

- Utilizzate il `postFulfillmentStatusSpecification` campo per specificare i messaggi e le azioni che si verificano dopo il successo, l'errore o il timeout del codehook.

Puoi richiamare il code hook Lambda nei seguenti punti di una conversazione impostando `active` i campi `enableCodeHookInvocation` eenabled/su: `true`

Durante la risposta iniziale

Per richiamare la funzione Lambda nella risposta iniziale dopo il riconoscimento dell'intento, utilizzate `codeHook` la struttura nel campo `initialResponse` dell'CreateIntentoperazione or. UpdateIntent Il `initialResponse` campo è mappato al seguente oggetto: InitialResponseSetting

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

Dopo l'elicitazione dello slot o durante la rielicitazione dello slot

Per richiamare la funzione Lambda dopo aver ottenuto un valore di slot, utilizzate `slotCaptureSetting` il campo all'interno del campo `valueElicitation` dell'CreateSlotoperazione or. UpdateSlot Il `slotCaptureSetting` campo è mappato al seguente oggetto: SlotCaptureSetting

```
"slotCaptureSetting": {
    "captureConditional": ConditionalSpecification object,
    "captureNextStep": DialogState object,
    "captureResponse": ResponseSpecification object,
    "codeHook": {
        "active": true,
        "enableCodeHookInvocation": true,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
```

```
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
    "failureResponse": ResponseSpecification object
}
```

- Per richiamare la funzione Lambda dopo che l'elicitazione dello slot ha avuto esito positivo, utilizzate il campo. `codeHook`
- Per richiamare la funzione Lambda dopo che l'elicitazione dello slot fallisce e Amazon Lex V2 tenta di riprovare l'elicitazione dello slot, usa il campo. `elicitationCodeHook`

Dopo la conferma o il rifiuto dell'intenzione

Per richiamare la funzione Lambda quando si conferma un intento, utilizzare `intentConfirmationSetting` il campo dell'operazione or. CreateIntentUpdateIntent Il `intentConfirmation` campo è mappato al seguente oggetto: IntentConfirmationSetting

```
"intentConfirmationSetting": {
    "active": boolean,
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "confirmationConditional": ConditionalSpecification object,
    "confirmationNextStep": DialogState object,
    "confirmationResponse": ResponseSpecification object,
    "declinationConditional": ConditionalSpecification object,
    "declinationNextStep": FulfillmentUpdatesSpecification object,
    "declinationResponse": PostFulfillmentStatusSpecification object,
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
    "failureResponse": ResponseSpecification object,
```

```
    "promptSpecification": PromptSpecification object
}
```

- Per richiamare la funzione Lambda dopo che l'utente ha confermato l'intento e i relativi slot, usa il campo. `codeHook`
- Per richiamare la funzione Lambda dopo che l'utente ha negato la conferma dell'intento e Amazon Lex V2 ha tentato di ripetere l'elicitazione dello slot, usa il campo. `elicitationCodeHook`

Durante l'adempimento dell'intento

Per richiamare la funzione Lambda per soddisfare un intento, usa `fulfillmentCodeHook` il campo nell'CreateIntentoperazione or. UpdateIntent Il `fulfillmentCodeHook` campo è mappato al seguente oggetto: FulfillmentCodeHookSettings

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. Una volta impostate le fasi della conversazione in cui richiamare la funzione Lambda, utilizzate `BuildBotLocale` l'operazione per ricostruire il bot per testare la funzione.

Debugging della funzione Lambda

Amazon CloudWatch Logs è uno strumento per tracciare le chiamate e i parametri delle API che puoi utilizzare per eseguire il debug delle funzioni Lambda. Quando esegui il test del bot nella console o con chiamate API, CloudWatch registra ogni fase della conversazione. Se usi una funzione di stampa nel tuo codice Lambda, la CloudWatch visualizza anche.

Per visualizzare CloudWatch i log della funzione Lambda

1. Accedi AWS Management Console e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel menu Registri nella barra laterale sinistra, seleziona Gruppi di log.
3. Seleziona il gruppo di log delle funzioni Lambda, che dovrebbe avere il formato. `/aws/lambda/`*`function-name`*

4. L'elenco dei flussi di log contiene un registro per ogni sessione con un bot. Scegli un flusso di log per visualizzarlo.
5. Nell'elenco degli eventi di registro, seleziona la freccia destra accanto al timestamp per espandere i dettagli dell'evento. Tutto ciò che stampi dal codice Lambda verrà visualizzato come evento di registro. Usa queste informazioni per eseguire il debug del codice.
6. Dopo aver eseguito il debug del codice, ricordati di implementare la funzione Lambda e, se stai usando la console, di ricaricare la finestra Test prima di testare nuovamente il comportamento del bot.

Personalizzazione delle interazioni con i bot

Scopri le seguenti funzionalità che puoi utilizzare per personalizzare le interazioni dei bot con i tuoi utenti espandendo e regolando il loro comportamento predefinito:

Argomenti

Analisi del sentimento delle espressioni degli utenti

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 il sentimento degli utenti. La risposta di Amazon Comprehend indica se il sentimento complessivo 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 l'API Amazon Lex. Abilita l'analisi del sentiment su un alias per il bot. Sulla console Amazon Lex:

1. Scegli un alias.
2. In Dettagli, scegli Modifica.
3. Scegli Attiva o disattiva l'analisi del sentiment per l'analisi del sentiment.
4. Quindi scegliere Confirm (Conferma) per salvare le modifiche.

Se si utilizza l'API, chiamare l'operazione CreateBotAlias con il campo `detectSentiment` impostato su `true`.

Quando l'analisi del sentiment è abilitata, la risposta RecognizeUtterancedelle operazioni RecognizeTextand restituisce un campo richiamato `sentimentResponse` nella `interpretations`

struttura con altri metadati. Il campo `sentimentResponse` ha due campi, `sentiment` e `sentimentScore`, che contengono il risultato dell'analisi dell'emozione. Se si utilizza una funzione Lambda, il `sentimentResponse` campo viene incluso nei dati dell'evento inviati alla funzione.

Di seguito è riportato un esempio del campo `sentimentResponse` restituito come parte della risposta `RecognizeText` o `RecognizeUtterance`.

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

Amazon Lex chiama Amazon Comprehend per tuo conto per determinare il sentimento contenuto in ogni enunciato elaborato dal bot. Abilitando l'analisi del sentiment, accetti i termini e i contratti di servizio per Amazon Comprehend. Per ulteriori informazioni sui prezzi di Amazon Comprehend, consulta Amazon Comprehend Pricing.

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

Utilizzo dei punteggi di confidenza

Amazon Lex V2 utilizza due passaggi per determinare cosa dice un utente. Il primo, il riconoscimento vocale automatico (ASR), crea una trascrizione dell'enunciato audio dell'utente. La seconda, la comprensione del linguaggio naturale (NLU), determina il significato dell'enunciato dell'utente per riconoscere l'intento dell'utente o il valore degli slot.

Per impostazione predefinita, Amazon Lex V2 restituisce il risultato più probabile di ASR e NLU. A volte può essere difficile per Amazon Lex V2 determinare il risultato più probabile. In tal caso, restituisce diversi risultati possibili insieme a un punteggio di affidabilità che indica la probabilità che il risultato sia corretto. Un punteggio di affidabilità è una valutazione fornita da Amazon Lex V2 che mostra la fiducia relativa di Amazon nei risultati. I punteggi di confidenza vanno da 0,0 a 1,0.

Puoi utilizzare la tua conoscenza del dominio con il punteggio di confidenza per determinare la corretta interpretazione del risultato ASR o NLU.

Il punteggio di confidenza ASR, o trascrizione, è una valutazione della sicurezza di Amazon Lex V2 nella correttezza di una determinata trascrizione. L'NLU, o intent, confidence score, è una valutazione della sicurezza di Amazon Lex V2 nella correttezza dell'intento specificato nella trascrizione superiore. Usa il punteggio di confidenza più adatto alla tua applicazione.

Argomenti

Utilizzo dei punteggi di confidenza delle intenzioni

Quando un utente pronuncia un'enunciazione, Amazon Lex V2 utilizza la comprensione del linguaggio naturale (NLU) per comprendere la richiesta dell'utente e restituire l'intento corretto. Per impostazione predefinita, Amazon Lex V2 restituisce l'intento più probabile definito dal bot.

In alcuni casi può essere difficile per Amazon Lex V2 determinare l'intento più probabile. Ad esempio, l'utente potrebbe pronunciare un'espressione ambigua o potrebbero esserci due intenti simili. Per determinare l'intento corretto, puoi combinare la tua conoscenza del dominio con i punteggi di confidenza NLU in un elenco di interpretazioni. Un punteggio di affidabilità è una valutazione fornita da Amazon Lex V2 che dimostra quanto sia sicuro che un intento sia l'intento corretto.

Per determinare la differenza tra due intenti all'interno di un'interpretazione, 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, esiste un'ambiguità tra i due intenti che potresti essere in grado di discriminare utilizzando la conoscenza del dominio nella tua applicazione.

Puoi anche utilizzare i punteggi di confidenza per creare applicazioni di test che determinano se le modifiche alle espressioni di un intento fanno la differenza nel comportamento del bot. Ad esempio, puoi ottenere i punteggi di confidenza per le intenzioni di un bot utilizzando una serie di enunciati, quindi aggiornare gli intenti con nuove espressioni. Puoi quindi controllare i punteggi di fiducia per vedere se c'è stato un miglioramento.

I punteggi di affidabilità restituiti da Amazon Lex V2 sono valori comparativi. Non dovresti fare affidamento su di loro come punteggio assoluto. I valori possono cambiare in base ai miglioramenti apportati ad Amazon Lex V2.

Amazon Lex V2 restituisce l'intento più probabile e fino a 4 intenti alternativi con i relativi punteggi nella `interpretations` struttura di ciascuna risposta. Il seguente codice JSON mostra la `interpretations` struttura della risposta dell'RecognizeTextoperazione:

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

AMAZON. FallbackIntent

Amazon Lex V2 torna a `AMAZON.FallbackIntent` essere l'intento principale in due situazioni:

1. Se i punteggi di confidenza di tutti gli intenti possibili sono inferiori alla soglia di confidenza. È possibile utilizzare la soglia predefinita o impostare una soglia personalizzata. Se hai `AMAZON.KendraSearchIntent` configurato, Amazon Lex V2 lo restituisce anche in questa situazione.
2. Se la fiducia nell'interpretazione per `AMAZON.FallbackIntent` è superiore alla fiducia interpretativa di tutti gli altri intenti.

Tieni presente che Amazon Lex V2 non mostra un punteggio di affidabilità per`AMAZON.FallbackIntent`.

Impostazione e modifica della soglia di confidenza

La soglia di confidenza deve essere un numero compreso tra 0,00 e 1,00. Puoi impostare la soglia per ogni lingua nel tuo bot nei seguenti modi:

Utilizzo della console Amazon Lex V2

- Per impostare la soglia quando aggiungi una lingua al tuo bot con Aggiungi lingua, puoi inserire il valore desiderato nel pannello Soglia del punteggio di confidenza.
- Per aggiornare la soglia, puoi selezionare Modifica nel pannello dei dettagli della lingua in una lingua per il tuo bot. Quindi inserisci il valore desiderato nel pannello Soglia del punteggio di fiducia.

Utilizzo delle operazioni API

- Per impostare la soglia, imposta il `nluIntentConfidenceThreshold` parametro dell'CreateBotLocaleoperazione.
- Per aggiornare la soglia di confidenza, impostate il `nluIntentConfidenceThreshold` parametro dell'UpdateBotLocaleoperazione.

Gestione delle sessioni

Per modificare l'intento utilizzato da Amazon Lex V2 in una conversazione con l'utente, puoi utilizzare la risposta della funzione Lambda del codice di dialogo oppure puoi utilizzare le API di gestione delle sessioni nella tua applicazione personalizzata.

Utilizzo di una funzione Lambda

Quando usi una funzione Lambda, Amazon Lex V2 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 V2 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 affidabilità assegnato da Amazon Lex V2 all'intento.

Per utilizzare un intento alternativo, è necessario specificarlo nell'azione `ConfirmIntent` o nella `ElicitSlot` finestra di dialogo della funzione Lambda.

Per ulteriori informazioni, consulta Abilitazione della logica personalizzata con AWS Lambda funzioni.

Utilizzo dell'API di gestione delle sessioni

Per utilizzare un intento diverso da quello corrente, utilizzate l'PutSessionoperazione. Ad esempio, se decidi che la prima alternativa è preferibile all'intento scelto da Amazon Lex V2, puoi utilizzare l'`PutSession`operazione per modificare gli intenti in modo che l'intento successivo con cui l'utente interagisce sia quello selezionato.

Per ulteriori informazioni, consulta Gestione delle sessioni con l'API Amazon Lex V2.

Utilizzo dei punteggi di confidenza nella trascrizione vocale

Quando un utente pronuncia una voce, Amazon Lex V2 utilizza il riconoscimento vocale automatico (ASR) per trascrivere la richiesta dell'utente prima che venga interpretata. Per impostazione predefinita, Amazon Lex V2 utilizza la trascrizione più probabile dell'audio per l'interpretazione.

In alcuni casi potrebbe esserci più di una possibile trascrizione dell'audio. Ad esempio, un utente potrebbe emettere un enunciato con un suono ambiguo, ad esempio «Mi chiamo John», che potrebbe essere inteso come «Mi chiamo Juan». In questo caso, puoi utilizzare tecniche di disambiguazione o combinare le tue conoscenze di dominio con il punteggio di confidenza della trascrizione per determinare quale trascrizione in un elenco di trascrizioni è quella corretta.

Amazon Lex V2 include la trascrizione principale e fino a due trascrizioni alternative per l'input dell'utente nella richiesta alla funzione code hook Lambda. Ogni trascrizione contiene un punteggio di affidabilità che indica che si tratta della trascrizione corretta. Ogni trascrizione include anche tutti i valori degli slot desunti dall'input dell'utente.

È possibile confrontare i punteggi di confidenza di due trascrizioni per determinare se vi è ambiguità tra di esse. Ad esempio, se una trascrizione ha un punteggio di confidenza di 0,95 e l'altra ha un punteggio di confidenza di 0,65, la prima trascrizione è probabilmente corretta e l'ambiguità tra di esse è bassa. Se le due trascrizioni hanno punteggi di confidenza di 0,75 e 0,72, l'ambiguità tra loro è elevata. Potresti essere in grado di distinguerle utilizzando le tue conoscenze di dominio.

Ad esempio, se i valori degli slot dedotti in due trascrizioni con un punteggio di confidenza di 0,75 e 0,72 sono «John» e «Juan», puoi interrogare gli utenti del tuo database per verificare l'esistenza di questi nomi ed eliminare una delle trascrizioni. Se «John» non è un utente nel tuo database e «Juan» lo è, puoi usare l'hook del codice di dialogo per modificare il valore dello slot dedotto per il nome in «Juan».

I punteggi di fiducia restituiti da Amazon Lex V2 sono valori comparativi. Non fare affidamento su di essi come punteggio assoluto. I valori possono cambiare in base ai miglioramenti apportati ad Amazon Lex V2.

I punteggi di confidenza nella trascrizione audio sono disponibili solo nelle lingue inglese (GB) (en_GB) e inglese (US) (en_US). I punteggi di confidenza sono supportati solo per l'ingresso audio a 8 kHz. I punteggi di confidenza nella trascrizione non vengono forniti per l'ingresso audio dalla finestra di test sulla console Amazon Lex V2 perché utilizza un ingresso audio a 16 kHz.

> **Note**
>
> Prima di poter utilizzare i punteggi di confidenza della trascrizione audio con un bot esistente, devi prima ricostruire il bot. Le versioni esistenti di un bot non supportano i punteggi di confidenza nella trascrizione. È necessario creare una nuova versione del bot per utilizzarle.

Puoi utilizzare i punteggi di confidenza per più modelli di progettazione delle conversazioni:

- Se il punteggio di confidenza più elevato scende al di sotto di una soglia a causa di un ambiente rumoroso o di una scarsa qualità del segnale, puoi chiedere all'utente con la stessa domanda di acquisire un audio di qualità migliore.
- Se più trascrizioni hanno punteggi di confidenza simili per i valori degli slot, ad esempio «John» e «Juan», puoi confrontare i valori con un database preesistente per eliminare gli input oppure puoi richiedere all'utente di selezionare uno dei due valori. Ad esempio, «pronuncia 1 per John o 2 per Juan».
- Se la tua logica aziendale richiede il cambio di intento in base a parole chiave specifiche in una trascrizione alternativa con un punteggio di affidabilità vicino alla trascrizione superiore, puoi modificare l'intento utilizzando la funzione Lambda del codice di dialogo o utilizzando le operazioni di gestione delle sessioni. Per ulteriori informazioni, consulta Gestione della sessione.

Amazon Lex V2 invia la seguente struttura JSON con un massimo di tre trascrizioni per l'input dell'utente alla funzione code hook Lambda:

```
"transcriptions": [
    {
        "transcription": "string",
        "rawTranscription": "string",
```

```
            "transcriptionConfidence": "number",
            },
            "resolvedContext": {
                "intent": "string"
            },
            "resolvedSlots": {
                "string": {
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "resolvedValues": [
                            "string"
                        ]
                    },
                    "values": [
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "resolvedValues": [
                                    "string"
                                ]
                            }
                        },
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "resolvedValues": [
                                    "string"
                                ]
                            }
                        }
                    ]
                }
            }
        }
    ]
```

La struttura JSON contiene il testo di trascrizione, l'intento che è stato risolto per l'enunciato e i valori per tutti gli slot rilevati nell'enunciato. Per quanto riguarda l'immissione di testo da parte dell'utente, le trascrizioni contengono una sola trascrizione con un punteggio di confidenza di 1,0.

Il contenuto delle trascrizioni dipende dalla svolta della conversazione e dall'intento riconosciuto.

Per il primo turno, ossia l'evocazione degli intenti, Amazon Lex V2 determina le prime tre trascrizioni. Per la trascrizione principale, restituisce l'intento e tutti i valori degli slot dedotti nella trascrizione.

Nei turni successivi, ossia l'elicitazione degli slot, i risultati dipendono dall'intento dedotto per ciascuna delle trascrizioni, come segue.

- Se l'intento dedotto per la trascrizione principale è lo stesso del turno precedente e tutte le altre trascrizioni hanno lo stesso intento, allora
  - Tutte le trascrizioni contengono valori di slot dedotti.

- Se l'intento dedotto per la trascrizione principale è diverso dal turno precedente e tutte le altre trascrizioni hanno l'intento precedente, allora
  - La trascrizione superiore contiene i valori degli slot dedotti per il nuovo intento.
  - Le altre trascrizioni hanno l'intento precedente e i valori degli slot dedotti per l'intento precedente.

- Se l'intento dedotto per la trascrizione principale è diverso dal turno precedente, una trascrizione è uguale all'intento precedente e una trascrizione ha un intento diverso, allora
  - La trascrizione principale contiene il nuovo intento dedotto e tutti i valori degli slot dedotti nell'enunciato.
  - La trascrizione che ha l'intento dedotto precedente contiene i valori degli slot dedotti per quell'intento.
  - La trascrizione con l'intento diverso non ha un nome di intento dedotto né valori di slot dedotti.

- Se l'intento dedotto per la trascrizione principale è diverso da quello del turno precedente e tutte le altre trascrizioni hanno intenti diversi, allora
  - La trascrizione principale contiene il nuovo intento dedotto e tutti i valori degli slot dedotti nell'enunciato.
  - Le altre trascrizioni non contengono intenti dedotti né valori di slot dedotti.

- Se l'intento dedotto per le prime due trascrizioni è lo stesso e diverso dal turno precedente, e la terza trascrizione ha un intento diverso, allora

- Le prime due trascrizioni contengono il nuovo intento dedotto e tutti i valori degli slot dedotti nell'enunciato.
- La terza trascrizione non ha un nome di intento e non ha valori di slot risolti.

Gestione della sessione

Per modificare l'intento utilizzato da Amazon Lex V2 in una conversazione con l'utente, usa la risposta della funzione Lambda del tuo hook di codice di dialogo. Oppure puoi utilizzare le API di gestione delle sessioni nella tua applicazione personalizzata.

Utilizzo di una funzione Lambda

Quando usi una funzione Lambda, Amazon Lex V2 la chiama con una struttura JSON che contiene l'input della funzione. La struttura JSON contiene un campo chiamato `transcriptions` che contiene le possibili trascrizioni che Amazon Lex V2 ha determinato per l'enunciato. Il `transcriptions` campo contiene da una a tre trascrizioni possibili, ognuna con un punteggio di affidabilità.

Per utilizzare l'intento di una trascrizione alternativa, lo specifichi nell'azione di `ElicitSlot` dialogo `ConfirmIntent` o nella funzione Lambda. Per utilizzare un valore di slot da una trascrizione alternativa, imposta il valore nel `intent` campo nella risposta della funzione Lambda. Per ulteriori informazioni, consulta Abilitazione della logica personalizzata con AWS Lambda funzioni.

Codice di esempio

Il seguente esempio di codice è una funzione Python Lambda che utilizza trascrizioni audio per migliorare l'esperienza di conversazione per l'utente.

Per utilizzare il codice di esempio, è necessario disporre di:

- Un bot con una sola lingua, inglese (GB) (en_GB) o inglese (US) (en_US).
- Un unico intento,. OrderBirthStone Assicurati che la funzione Usa una funzione Lambda per l'inizializzazione e la convalida sia selezionata nella sezione Code hook della definizione dell'intento.
- L'intento deve avere due slot, "BirthMonth" e «Name», entrambi di tipo. `AMAZON.AlphaNumeric`
- Un alias con la funzione Lambda definita. Per ulteriori informazioni, consulta Creazione e associazione di una funzione Lambda a un alias bot.

```
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)


# --- Helpers that build all of the responses ---

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit, message):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'ElicitSlot',
                'slotToElicit': slot_to_elicit
            },
            'intent': {
                'name': intent_request['sessionState']['intent']['name'],
                'slots': slots,
                'state': 'InProgress'
            },
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'e3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'sessionId': intent_request['sessionId'],
        'messages': [message],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': '3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
```

```
        'messages': [message],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def delegate(intent_request, session_attributes):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'Delegate'
            },
            'intent': intent_request['sessionState']['intent'],
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'abc'
        },
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']

    return {}


def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']


""" --- Functions that control the behavior of the bot --- """


def order_birth_stone(intent_request):
    """
    Performs dialog management and fulfillment for ordering a birth stone.
    Beyond fulfillment, the implementation for this intent demonstrates the following:
    1) Use of N best transcriptions to re prompt user when confidence for top
 transcript is below a threshold
```

```
    2) Overrides resolved slot for birth month from a known fixed list if the top
 transcript
    is not accurate.
    """

    transcriptions = intent_request['transcriptions']

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Disambiguate if there are multiple transcriptions and the top transcription
        # confidence is below a threshold (0.8 here)
        if len(transcriptions) > 1 and transcriptions[0]['transcriptionConfidence'] <
 0.8:
            if transcriptions[0]['resolvedSlots'] is not {} and 'Name' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['Name'] is not None:
                return prompt_for_name(intent_request)
            elif transcriptions[0]['resolvedSlots'] is not {} and 'BirthMonth' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['BirthMonth'] is not None:
                return validate_month(intent_request)

    return continue_conversation(intent_request)


def prompt_for_name(intent_request):
    """
    If the confidence for the name is not high enough, re prompt the user with the
 recognized names
    so it can be confirmed.
    """
    resolved_names = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'Name' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['Name'] is not None:
            resolved_names.append(transcription['resolvedSlots']['Name']['value']
['originalValue'])
    if len(resolved_names) > 1:
        session_attributes = get_session_attributes(intent_request)
        slots = get_slots(intent_request)
        return elicit_slot(session_attributes, intent_request, slots, 'Name',
                           {'contentType': 'PlainText',
                            'content': 'Sorry, did you say your name is {} ?'.format("
 or ".join(resolved_names))})
```

```
    else:
        return continue_conversation(intent_request)


def validate_month(intent_request):
    """
    Validate month from an expected list, if not valid looks for other transcriptions
 and to see if the month
    recognized there has an expected value. If there is, replace with that and if not
 continue conversation.
    """

    expected_months = ['january', 'february', 'march']
    resolved_months = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'BirthMonth' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['BirthMonth'] is not None:
            resolved_months.append(transcription['resolvedSlots']['BirthMonth']
['value']['originalValue'])

    for resolved_month in resolved_months:
        if resolved_month in expected_months:
            intent_request['sessionState']['intent']['slots']['BirthMonth']
['resolvedValues'] = [resolved_month]
            break

    return continue_conversation(intent_request)


def continue_conversation(event):
    session_attributes = get_session_attributes(event)

    if event["invocationSource"] == "DialogCodeHook":
        return delegate(event, session_attributes)


# --- Intents ---


def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """
```

```
    logger.debug('dispatch sessionId={},
 intentName={}'.format(intent_request['sessionId'],

 intent_request['sessionState']['intent']['name']))

    intent_name = intent_request['sessionState']['intent']['name']

    # Dispatch to your bot's intent handlers
    if intent_name == 'OrderBirthStone':
        return order_birth_stone(intent_request)

    raise Exception('Intent with name ' + intent_name + ' not supported')


# --- Main handler ---


def lambda_handler(event, context):
    """
    Route the incoming request based on intent.
    The JSON body of the request is provided in the event slot.

    """
    # By default, treat the user request as coming from the America/New_York time
 zone.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()
    logger.debug('event={}'.format(event))

    return dispatch(event)
```

Utilizzo dell'API di gestione delle sessioni

Per utilizzare un intento diverso da quello corrente, usa l'PutSessionoperazione. Ad esempio, se decidi che la prima alternativa è preferibile all'intento scelto da Amazon Lex V2, puoi utilizzare l'`PutSession`operazione per modificare gli intenti. In questo modo l'intento successivo con cui l'utente interagisce sarà quello selezionato.

È inoltre possibile utilizzare l'`PutSession`operazione per modificare il valore dello slot nella `intent` struttura in modo da utilizzare un valore proveniente da una trascrizione alternativa.

Per ulteriori informazioni, consulta Gestione delle sessioni con l'API Amazon Lex V2.

Personalizzazione delle trascrizioni vocali

Il comportamento predefinito del bot a volte può comportare trascrizioni vocali imprecise. Le seguenti funzionalità sono disponibili per aiutare il tuo bot a riconoscere parole o nomi meno comuni o facilmente confusi.

Argomenti

Migliorare il riconoscimento vocale con un vocabolario personalizzato

Puoi fornire ad Amazon Lex V2 ulteriori informazioni su come elaborare le conversazioni audio con un bot creando un vocabolario personalizzato in una lingua specifica. Un vocabolario personalizzato è un elenco di frasi specifiche che desideri che Amazon Lex V2 riconosca nell'ingresso audio. In genere si tratta di nomi propri o parole specifiche del dominio che Amazon Lex V2 non riconosce.

Ad esempio, supponiamo di avere un bot di supporto tecnico. Puoi aggiungere «backup» a un vocabolario personalizzato per aiutare il bot a trascrivere correttamente l'audio come «backup», anche quando l'audio suona come «fai le valigie». Un vocabolario personalizzato può anche aiutare a riconoscere parole rare nell'audio, come «solvibilità» per i servizi finanziari o sostantivi propri come «Cognito» o «Monitron».

Nozioni di base sul vocabolario personalizzato

- Un vocabolario personalizzato funziona sulla trascrizione dell'input audio in un bot. È necessario fornire enunciati di esempio per riconoscere un intento o un valore di slot.
- Un vocabolario personalizzato è unico per una lingua specifica. È necessario configurare vocabolari personalizzati in modo indipendente per ogni lingua. I vocabolari personalizzati sono supportati solo per le lingue inglese (Regno Unito) e inglese (Stati Uniti).
- Sono disponibili vocabolari personalizzati con integrazioni di contact center supportate da Amazon Lex V2. La finestra di test nella console Amazon Lex V2 supporta vocabolari personalizzati per tutti i bot Amazon Lex V2 creati a partire dal 31 luglio 2022. Se riscontri problemi con i vocabolari personalizzati nella finestra di test, ricostruisci il bot e riprova.

Amazon Lex V2 utilizza vocabolari personalizzati per suscitare sia gli intenti che gli slot. Lo stesso file di vocabolario personalizzato viene utilizzato per intenti e slot. Puoi disattivare selettivamente la funzionalità del vocabolario personalizzato per uno slot quando aggiungi un tipo di slot.

Suscitare un intento: puoi creare un vocabolario personalizzato per suscitare un intento. Queste frasi vengono utilizzate per la trascrizione quando il bot determina l'intento dell'utente. Ad esempio, se hai configurato la frase «backup» nel tuo vocabolario personalizzato, Amazon Lex V2 trascrive l'input dell'utente in «puoi fare il backup delle mie foto?» —anche quando l'audio suona come «puoi impacchettare le mie foto». È possibile specificare il grado di incremento per ogni frase configurando un peso di 0, 1, 2 o 3. Puoi anche specificare una rappresentazione alternativa per la frase nell'output finale dal discorso al testo aggiungendo un `displayAs` campo.

Le frasi del vocabolario personalizzate utilizzate per migliorare la trascrizione durante l'evocazione dell'intento non influiscono sulle trascrizioni durante l'attivazione degli slot. Per ulteriori informazioni sulla creazione di un vocabolario personalizzato per suscitare intenti, vedere. Creazione di un vocabolario personalizzato per suscitare intenti e slot

Ottenere slot personalizzati: puoi utilizzare un vocabolario personalizzato per migliorare il riconoscimento degli slot per le conversazioni audio. Per migliorare la capacità del tuo bot Amazon Lex V2 di riconoscere i valori degli slot, crea uno slot personalizzato e aggiungi i valori degli slot allo slot personalizzato, quindi scegli Usa i valori degli slot come vocabolario personalizzato. Esempi di valori di slot includono nomi di prodotti, cataloghi o nomi propri. Non dovresti usare parole o frasi comuni come «sì» e «no» nei vocabolari personalizzati.

Dopo aver aggiunto i valori degli slot, questi valori vengono utilizzati per migliorare il riconoscimento degli slot quando il bot aspetta un input per lo slot personalizzato. Questi valori non vengono utilizzati per la trascrizione quando si suscita un intento. Per ulteriori informazioni, consulta Aggiungere tipi di slot.

Procedure consigliate per creare un vocabolario personalizzato

Suscitare un intento

- I vocabolari personalizzati funzionano in modo ottimale se utilizzati finalizzandoli a parole o locuzioni specifiche. Aggiungi parole a un vocabolario personalizzato solo se non sono immediatamente riconosciute da Amazon Lex V2.
- Decidi quanto peso dare a una parola in base alla frequenza con cui la parola non viene riconosciuta nella trascrizione e alla rarità della parola nell'input. Le parole difficili da pronunciare richiedono un peso maggiore.

- Utilizza un set di test rappresentativo per determinare se un peso è appropriato. Puoi raccogliere un set di test audio attivando la registrazione audio nei registri delle conversazioni.
- Evita di usare parole brevi come «on», «it», «to», «yes», «no» in un vocabolario personalizzato.

Ottenere uno slot personalizzato

- Aggiungi i valori al tipo di slot personalizzato che prevedi venga riconosciuto. Aggiungi tutti i possibili valori di slot per il tipo di slot personalizzato, indipendentemente da quanto sia comune o raro il valore dello slot.
- Attivate l'opzione solo quando il tipo di slot personalizzato contiene un elenco di valori o entità del catalogo, come nomi di prodotti o fondi comuni di investimento.
- Disabilita l'opzione se il tipo di slot viene utilizzato per acquisire frasi generiche come «sì», «no», «non lo so», «forse» o parole generiche come «uno», «due», «tre».
- Limita il numero di valori e sinonimi degli slot a 500 o meno per ottenere prestazioni ottimali.

Inserisci acronimi o altre parole le cui lettere devono essere pronunciate singolarmente come lettere singole separate da un punto e uno spazio. Non utilizzare singole lettere a meno che non facciano parte di una frase, come «J. P. Morgan» o «A. W. S.» È possibile utilizzare lettere maiuscole o minuscole per definire un acronimo.

Creazione di un vocabolario personalizzato per suscitare intenti e slot

Puoi usare la console Amazon Lex V2 per creare e gestire un vocabolario personalizzato oppure puoi utilizzare le operazioni dell'API Amazon Lex V2. Esistono 2 modi per creare un vocabolario personalizzato tramite la console:

Console

Importa vocabolario personalizzato nella console:

1. Apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home
2. Dall'elenco dei bot, scegli il bot a cui desideri aggiungere il vocabolario personalizzato.
3. Nella pagina dei dettagli del bot, dalla sezione Aggiungi lingue, scegli Visualizza lingue.
4. Dall'elenco delle lingue, scegli la lingua a cui desideri aggiungere il vocabolario personalizzato.

Crea un nuovo vocabolario personalizzato direttamente dalla console:

1. Fai clic su Crea nella sezione Vocabolario personalizzato della pagina dei dettagli della lingua. Si aprirà una finestra di modifica in cui non è presente alcun vocabolario personalizzato.
2. Aggiungi gli input per frase e peso come richiesto. DisplayAs Puoi inoltre apportare modifiche in linea agli elementi aggiunti aggiornando i relativi campi o eliminandoli dall'elenco.
3. Fai clic su Salva. Nota: il nuovo vocabolario personalizzato viene salvato nel tuo bot solo dopo aver fatto clic su Salva.
4. Puoi continuare ad apportare modifiche in linea in questa pagina e fare clic su Salva quando hai finito.
5. Questa pagina consente anche di importare, esportare ed eliminare un file di vocabolario personalizzato dal menu a discesa in alto a destra.

API

Usa l'**`ListCustomVocabularyItems`**API per visualizzare le voci del vocabolario personalizzate:

1. Usa l'`ListCustomVocabularyItems`operazione per visualizzare le voci del vocabolario personalizzate. Il corpo della richiesta sarà simile al seguente:

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. Tieni presente che `maxResults` e `nextToken` sono campi opzionali per il corpo della richiesta.
3. La risposta dell'`ListCustomVocabularyItems`operazione è la seguente:

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
```

```
   ]
}
```

Usa l'**`BatchCreateCustomVocabularyItem`**API per creare nuove voci di vocabolario personalizzate:

1. Se la lingua del tuo bot non ha ancora creato un vocabolario personalizzato, segui i passaggi per utilizzarlo per StartImportcreare un vocabolario personalizzato.
2. Dopo aver creato il vocabolario personalizzato, usa l'`BatchCreateCustomVocabularyItem`operazione per creare nuove voci di vocabolario personalizzate. Il corpo della richiesta sarà simile al seguente:

```
{
   "customVocabularyItemList": [
        {
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Tieni presente che `weight` e `displayAs` sono campi opzionali per il corpo della richiesta.
4. La risposta del testamento `BatchCreateCustomVocabularyItem` sarà simile a questa:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
```

```
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Poiché si tratta di un'operazione batch, la richiesta non avrà esito negativo se uno degli elementi non viene creato. L'elenco degli errori conterrà informazioni sul motivo per cui l'operazione non è riuscita per quella voce specifica. L'elenco delle risorse conterrà tutte le voci che sono state create con successo.
6. InfattiBatchCreateCustomVocabularyItem, puoi aspettarti di vedere questi tipi di errori:

   - RESOURCE_DOES_NOT_EXIST: Il vocabolario personalizzato non esiste. Segui i passaggi per creare un vocabolario personalizzato prima di avviare questa operazione.
   - DUPLICATE_INPUT: L'elenco degli input contiene frasi duplicate.
   - RESOURCE_ALREADY_EXISTS: La frase indicata per la voce esiste già nel tuo vocabolario personalizzato.
   - INTERNAL_SERVER_FAILURE: Si è verificato un errore nel backend durante l'elaborazione della richiesta. Ciò può indicare un'interruzione del servizio o un altro problema.

Usa l'**BatchDeleteCustomVocabularyItem**API per eliminare le voci di vocabolario personalizzate esistenti:

1. Se la lingua del tuo bot non ha ancora creato un vocabolario personalizzato, segui i passaggi in Usa il per creare un vocabolario personalizzato StartImportper crearne uno.
2. Dopo aver creato il vocabolario personalizzato, usa l'BatchDeleteCustomVocabularyItemoperazione per eliminare le voci del vocabolario personalizzato esistenti. Il corpo della richiesta sarà simile al seguente:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. La risposta del testamento BatchDeleteCustomVocabularyItem sarà simile a questa:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

4. Poiché si tratta di un'operazione batch, la richiesta non avrà esito negativo se uno degli elementi non viene eliminato. L'elenco degli errori conterrà informazioni sul motivo per cui l'operazione non è riuscita per quella voce specifica. L'elenco delle risorse conterrà tutte le voci che sono state eliminate con successo.
5. Infatti`BatchDeleteCustomVocabularyItem`, puoi aspettarti di vedere questi tipi di errori:
   - `RESOURCE_DOES_NOT_EXIST`: la voce del vocabolario personalizzata che stai cercando di eliminare non esiste.
   - `INTERNAL_SERVER_FAILURE`: Si è verificato un errore nel backend durante l'elaborazione della richiesta. Ciò può indicare un'interruzione del servizio o un altro problema.

Usa l'**`BatchUpdateCustomVocabularyItem`**API per aggiornare le voci di vocabolario personalizzate esistenti:

1. Se la lingua del tuo bot non ha ancora creato un vocabolario personalizzato, segui i passaggi in Usa il per creare un vocabolario personalizzato StartImportper creare un vocabolario personalizzato.

2. Dopo aver creato il vocabolario personalizzato, usa l'`BatchUpdateCustomVocabularyItem`operazione per aggiornare le voci del vocabolario personalizzato esistenti. Il corpo della richiesta sarà simile al seguente:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Tieni presente che `weight` e `displayAs` sono campi opzionali per il corpo della richiesta.
4. La risposta del testamento `BatchUpdateCustomVocabularyItem` sarà simile a questa:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Poiché si tratta di un'operazione batch, la richiesta non avrà esito negativo se uno degli elementi non viene eliminato. L'elenco degli errori conterrà informazioni sul motivo per cui l'operazione

non è riuscita per quella voce specifica. L'elenco delle risorse conterrà tutte le voci che sono state aggiornate con successo.

6. InfattiBatchUpdateCustomVocabularyItem, puoi aspettarti di vedere questi tipi di errori:

   - RESOURCE_DOES_NOT_EXIST: la voce del vocabolario personalizzata che stai cercando di aggiornare non esiste.
   - DUPLICATE_INPUT: l'elenco degli input contiene ItemID duplicati.
   - RESOURCE_ALREADY_EXISTS: La frase indicata per la voce esiste già nel tuo vocabolario personalizzato.
   - INTERNAL_SERVER_FAILURE: Si è verificato un errore nel backend durante l'elaborazione della richiesta. Ciò può indicare un'interruzione del servizio o un altro problema.

Creazione di un file di vocabolario personalizzato

Un file di vocabolario personalizzato è un elenco di valori separato da tabulazioni che contiene la frase da riconoscere, un peso per dare una spinta e un displayAs campo che sostituirà la frase nella trascrizione del discorso. È più probabile che le frasi con un valore di boost più elevato vengano utilizzate quando appaiono nell'ingresso audio.

Il file di vocabolario personalizzato deve avere un nome **CustomVocabulary.tsv** e deve essere compresso in un file zip prima di poter essere importato. Il file zip deve avere una dimensione inferiore a 300 MB. Il numero massimo di frasi in un vocabolario personalizzato è 500.

- frase 1—4 parole che devono essere riconosciute. Separa le parole nella frase con spazi. Non puoi inserire frasi duplicate nel file. Il campo della frase è obbligatorio.
- peso: il grado in cui viene potenziato il riconoscimento della frase. Il valore è un numero intero 0, 1, 2 o 3. Se non specifichi un peso, il valore predefinito è 1. Decidi il peso in base alla frequenza con cui la parola non viene riconosciuta nella trascrizione e alla rarità della parola nell'input. Il peso 0 indica che non verrà applicato alcun potenziamento e la voce verrà utilizzata solo per eseguire sostituzioni utilizzando il displayAs campo.
- DisplayAs: definisce come vuoi che la tua frase appaia nell'output della trascrizione. Questo è un campo opzionale nel vocabolario personalizzato.

Il file di vocabolario personalizzato deve contenere una riga di intestazione con le intestazioni «phrase», «weight» e «displayAs». Le intestazioni possono essere in qualsiasi ordine, ma devono seguire la nomenclatura precedente.

L'esempio seguente è un file di vocabolario personalizzato. Il carattere di tabulazione richiesto per separare la frase, il peso e il display As è rappresentato dal testo «[TAB]». Se usi questo esempio, sostituisci il testo con un carattere di tabulazione.

```
phrase[TAB]weight[TAB]displayAs
Newcastle[TAB]2
Hobart[TAB]2[TAB]Hobart, Australia
U. Dub[TAB]1[TAB]University of Washington, Seattle
W. S. U.[TAB]3
Issaquah
Kennewick
```

Miglioramento del riconoscimento dei valori degli slot con suggerimenti di runtime

Con i suggerimenti di runtime puoi assegnare ad Amazon Lex V2 una serie di valori degli slot in base al contesto per ottenere un migliore riconoscimento nelle conversazioni audio e una migliore risoluzione degli slot. Puoi utilizzare i suggerimenti di runtime per fornire un elenco di frasi in fase di esecuzione che diventano candidate per la risoluzione del valore di uno slot.

Ad esempio, se un utente che interagisce con un bot per la prenotazione di voli si reca spesso a San Francisco, Giacarta, Seul e Mosca, puoi configurare suggerimenti di runtime con un elenco di queste quattro città quando cerchi la destinazione per migliorare il riconoscimento delle città più frequentate.

I suggerimenti di runtime sono disponibili solo nelle lingue inglese (Stati Uniti) e inglese (Regno Unito). Possono essere utilizzati con i seguenti tipi di slot:

- Tipi di slot personalizzati
- Amazon.city
- Amazon.paese
- AMAZON. FirstName
- AMAZON. LastName
- Amazon.stato
- AMAZZONE. StreetName

Nozioni di base sui suggerimenti di runtime

- I suggerimenti di runtime vengono utilizzati solo quando si ottiene un valore di slot da un utente.
- Quando si utilizzano i suggerimenti di runtime, i valori dei suggerimenti sono preferiti rispetto a valori simili. Ad esempio, per un robot che ordina cibo, puoi impostare un elenco di voci di menu come suggerimenti in fase di esecuzione e, allo stesso tempo, inserire i prodotti alimentari in uno slot personalizzato per preferire «filetto» a «amico» dal suono simile.
- Se l'input dell'utente è diverso dai valori forniti nei suggerimenti di runtime, per lo slot verrà utilizzato l'input originale dell'utente.
- Per i tipi di slot personalizzati, i valori forniti come suggerimenti di runtime verranno utilizzati per la risoluzione dello slot anche se non fanno parte dello slot personalizzato durante la creazione del bot.
- I suggerimenti di runtime sono supportati solo per l'ingresso audio a 8 kHz. Sono disponibili con integrazioni di contact center supportate da Amazon Lex V2. Non vengono forniti suggerimenti di runtime per l'ingresso audio dalla finestra di test sulla console Amazon Lex V2 perché utilizza un ingresso audio a 16 kHz.

> **Note**
>
> Prima di poter utilizzare i suggerimenti di runtime con un bot esistente, devi prima ricostruire il bot. Le versioni esistenti di un bot non supportano i suggerimenti di runtime. È necessario creare una nuova versione del bot per utilizzarle.

Puoi inviare suggerimenti di runtime ad Amazon Lex V2 utilizzando l'operazione PutSession, RecognizeTextRecognizeUtterance, o StartConversation. Puoi anche aggiungere suggerimenti di runtime utilizzando una funzione Lambda.

Puoi inviare suggerimenti di runtime all'inizio di una conversazione per configurare i suggerimenti per ogni slot utilizzato nel bot oppure puoi inviare suggerimenti come parte dello stato della sessione durante una conversazione. L'`runtimeHints`attributo associa uno slot ai suggerimenti per quello slot.

Una volta inviato un suggerimento di runtime ad Amazon Lex V2, esso persiste per ogni turno della conversazione fino al termine della sessione. Se invii una `runtimeHints` struttura nulla, vengono utilizzati i suggerimenti esistenti. È possibile modificare i suggerimenti nei seguenti modi:

- Invio di una nuova `runtimeHints` struttura al bot. I contenuti della nuova struttura sostituiscono quelli esistenti.

- Invio di una `runtimeHints` struttura vuota al bot. Questo cancella i suggerimenti di runtime per il bot.

Aggiungere valori di slot nel contesto

Aggiungete un contesto al bot fornendo i valori degli slot previsti come suggerimenti di runtime quando l'applicazione dispone di informazioni sulla prossima frase probabile dell'utente. Aggiungi un hook di dialogo Lambda al tuo bot (vedi Abilitazione della logica personalizzata con AWS Lambda funzioni per maggiori informazioni) e usa il proposedNextStatecampo in Interpretazione del formato dell'evento di input per determinare i suggerimenti di runtime da includere per migliorare la conversazione con l'utente.

Ad esempio, in un'app bancaria puoi generare un elenco di nickname di account per un utente specifico e quindi utilizzare l'elenco per richiedere l'account a cui l'utente desidera accedere.

Invia suggerimenti di runtime all'inizio della conversazione quando disponi di un contesto che aiuti il bot a interpretare l'input dell'utente. Ad esempio, se disponi del numero di telefono dell'utente, puoi utilizzare queste informazioni per cercarlo in modo da poter utilizzare l'`StartConversation`operazione `PutSession` o per passare suggerimenti su nome e cognome al bot se stai chiedendo al nome dell'utente di convalidare le sue credenziali.

Durante una conversazione, potresti raccogliere informazioni da un valore di slot che possono aiutarti con un altro valore di slot. Ad esempio, in un'app per la cura dell'auto, quando disponi del numero di account dell'utente, puoi cercare le auto di proprietà del cliente e passarle come suggerimenti a un altro slot.

Inserisci gli acronimi, o altre parole le cui lettere devono essere pronunciate singolarmente, come lettere singole separate da un punto e uno spazio. Non utilizzare singole lettere a meno che non facciano parte di una frase, come «J. P. Morgan» o «A.W.S». È possibile utilizzare lettere maiuscole o minuscole per definire un acronimo.

Aggiungere suggerimenti a uno slot

Per aggiungere suggerimenti di runtime a uno slot, si utilizza la `runtimeHints` struttura che fa parte della `sessionState` struttura. Di seguito è riportato un esempio di `runtimeHints` struttura. Fornisce suggerimenti per due slot, "" e FirstName "LastName" per l'intento MakeAppointment "».

```
{
```

```
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "Stiles"
                            },
                            {
                                "phrase": "Major"
                            }
                        ]
                    }
                }
            }
        }
    }
}
```

Puoi anche usare una funzione Lambda per aggiungere suggerimenti di runtime durante una conversazione. Per aggiungere suggerimenti di runtime, aggiungi la `runtimeHints` struttura allo stato della sessione della risposta che la tua funzione Lambda invia ad Amazon Lex V2. Per ulteriori informazioni, consulta Preparazione del formato di risposta.

È necessario specificare un `intentName` e valido `slotName` nella richiesta, altrimenti Amazon Lex V2 restituisce un errore di runtime.

Acquisizione dei valori degli slot con stili di ortografia

Amazon Lex V2 offre slot integrati per acquisire informazioni specifiche dell'utente come nome, cognome, indirizzo e-mail o identificatori alfanumerici. Ad esempio, puoi utilizzare lo `AMAZON.LastName` slot per acquisire cognomi come «Jackson» o «Garcia». Tuttavia, Amazon Lex V2 può confondersi con cognomi difficili da pronunciare o non comuni in un locale, come «Xiulan». Per acquisire tali nomi, puoi chiedere all'utente di inserire i dati ortografia per lettera o ortografia per stile di parola.

Amazon Lex V2 offre tre stili di elicitazione degli slot da utilizzare. Quando si imposta uno stile di elicitazione degli slot, cambia il modo in cui Amazon Lex V2 interpreta l'input dell'utente.

Ortografia per lettera: con questo stile, puoi indicare al bot di ascoltare l'ortografia anziché l'intera frase. Ad esempio, per registrare un cognome come «Xiulan», puoi dire all'utente di pronunciare il cognome una lettera alla volta. Il bot acquisirà l'ortografia e risolverà le lettere in una parola. Ad esempio, se l'utente dice «x i u l a n», il bot acquisisce il cognome come «xiulan».

Ortografia per parola: nelle conversazioni vocali, specialmente al telefono, alcune lettere, come «t», «b», «p», hanno un suono simile. Quando l'acquisizione di valori alfanumerici o l'ortografia dei nomi restituisce un valore errato, è possibile richiedere all'utente di fornire una parola identificativa insieme alla lettera. Ad esempio, se la risposta vocale a una richiesta di un ID di prenotazione è «abp123", il bot potrebbe invece riconoscere la frase «ab b 123". Se questo è un valore errato, puoi chiedere all'utente di fornire l'input come «a come in alpha b come in boy p come in peter one two three». Il bot risolverà l'input in «abp123».

Quando si utilizza ortografia per parola, è possibile utilizzare i seguenti formati:

- «come in» (come in apple)
- «per» (a per mela)
- «mi piace» (una mela simile)

Predefinito: questo è lo stile naturale di acquisizione delle slot utilizzando la pronuncia delle parole. Ad esempio, può acquisire nomi come «John Stiles» in modo naturale. Se non viene specificato uno stile di elicitazione degli slot, il bot utilizza lo stile predefinito. Per i tipi di slot `AMAZON.AlphaNumeric` e `AMAZON.UKPostal` code, lo stile predefinito supporta l'immissione ortografica per lettera.

Se il nome «Xiulan» viene pronunciato usando una combinazione di lettere e parole, ad esempio «x as in x-ray i u l as in lion a n», lo stile di elicitazione degli slot deve essere impostato su style. spell-by-word Lo spell-by-letter stile non lo riconoscerà.

È necessario creare un'interfaccia vocale che acquisisca i valori degli slot con uno stile di conversazione naturale per un'esperienza migliore. Per gli input che non vengono acquisiti correttamente utilizzando lo stile naturale, è possibile ripetere la richiesta all'utente e impostare lo stile di elicitazione dello slot su or. spell-by-letter spell-by-word

È possibile utilizzare spell-by-word e spell-by-letter stili per i seguenti tipi di slot nelle lingue inglese (Stati Uniti), inglese (Regno Unito) e inglese (Australia):

- AMAZON. AlphaNumeric
- AMAZZONE. EmailAddress
- AMAZON. FirstName
- AMAZON. LastName
- AMAZON.UK PostalCode
- Tipi di slot personalizzati

Abilitare l'ortografia

Si attiva spell-by-letter e spell-by-word in fase di esecuzione quando si ottengono slot dall'utente. È possibile impostare lo stile di ortografia con l'PutSessionoperazione, RecognizeTextRecognizeUtterance, o StartConversation. È inoltre possibile abilitare spell-by-letter e spell-by-word utilizzare una funzione Lambda.

Puoi impostare lo stile di ortografia utilizzando il `dialogAction` campo del `sessionState` campo nella richiesta di una delle suddette operazioni API o durante la configurazione della risposta Lambda (Preparazione del formato di rispostavedi per ulteriori informazioni). È possibile impostare lo stile solo quando il tipo di azione del dialogo è `ElicitSlot` e quando lo slot da suscitare è uno dei tipi di slot supportati.

Il seguente codice JSON mostra il `dialogAction` campo impostato per utilizzare lo spell-by-word stile:

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
```

```
    "type": "ElicitSlot"
}
```

Il campo `slotElicitationStyle` può essere impostato su `SpellByLetter`, `SpellByWord` o `Default`. Se non si specifica un valore, il valore viene impostato `Default` su.

> **Note**
>
> Non puoi abilitare spell-by-letter o spell-by-word evocare stili tramite la console.

Codice di esempio

La modifica dello stile ortografico viene in genere eseguita se il primo tentativo di risolvere un valore dello slot non ha funzionato. Il seguente esempio di codice è una funzione Python Lambda che utilizza spell-by-word lo stile nel secondo tentativo di risolvere uno slot.

Per utilizzare il codice di esempio, è necessario disporre di:

- Un bot con una sola lingua, inglese (GB) (en_GB).
- Un intento, "CheckAccount" con un esempio di espressione: «Vorrei controllare il mio account». Assicurati che l'opzione Usa una funzione Lambda per l'inizializzazione e la convalida sia selezionata nella sezione Code hook della definizione dell'intento.
- L'intento deve avere uno slot, "PostalCode«, del tipo integrato. `AMAZON.UKPostalCode`
- Un alias con la funzione Lambda definita. Per ulteriori informazioni, consulta Creazione e associazione di una funzione Lambda a un alias bot.

```
import json
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

# --- Helpers that build all of the responses ---

def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']
```

```
def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']
    return {}

def get_slot(intent_request, slotName):
    slots = get_slots(intent_request)
    if slots is not None and slotName in slots and slots[slotName] is not None:
        logger.debug('resolvedValue={}'.format(slots[slotName]['value']
['resolvedValues']))
        return slots[slotName]['value']['resolvedValues']
    else:
        return None

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit,
 slot_elicitation_style, message):
    return {'sessionState': {'dialogAction': {'type': 'ElicitSlot',
                                              'slotToElicit': slot_to_elicit,
                                              'slotElicitationStyle':
 slot_elicitation_style
                                              },
                             'intent': {'name': intent_request['sessionState']
['intent']['name'],
                                        'slots': slots,
                                        'state': 'InProgress'
                                        },
                             'sessionAttributes': session_attributes,
                             'originatingRequestId': 'REQUESTID'
                             },
            'sessionId': intent_request['sessionId'],
            'messages': [ message ],
            'requestAttributes': intent_request['requestAttributes']
            if 'requestAttributes' in intent_request else None
            }

def build_validation_result(isvalid, violated_slot, slot_elicitation_style,
 message_content):
    return {'isValid': isvalid,
            'violatedSlot': violated_slot,
            'slotElicitationStyle': slot_elicitation_style,
            'message': {'contentType': 'PlainText',
            'content': message_content}
```

```
            }

def GetItemInDatabase(postal_code):
    """
    Perform database check for transcribed postal code. This is a no-op
    check that shows that postal_code can't be found in the database.
    """
    return None

def validate_postal_code(intent_request):

    postal_code = get_slot(intent_request, 'PostalCode')

    if GetItemInDatabase(postal_code) is None:
        return build_validation_result(
            False,
            'PostalCode',
            'SpellByWord',
            "Sorry, I can't find your information. " +
            "To try again, spell out your postal " +
            "code using words, like a as in apple."
        )
    return {'isValid': True}

def check_account(intent_request):
    """
    Performs dialog management and fulfillment for checking an account
    with a postal code. Besides fulfillment, the implementation for this
    intent demonstrates the following:
    1) Use of elicitSlot in slot validation and re-prompting.
    2) Use of sessionAttributes to pass information that can be used to
        guide a conversation.
    """
    slots = get_slots(intent_request)
    postal_code = get_slot(intent_request, 'PostalCode')
    session_attributes = get_session_attributes(intent_request)

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Validate the PostalCode slot. If any aren't valid,
        # re-elicit for the value.
        validation_result = validate_postal_code(intent_request)
        if not validation_result['isValid']:
            slots[validation_result['violatedSlot']] = None
            return elicit_slot(
```

```
                session_attributes,
                intent_request,
                slots,
                validation_result['violatedSlot'],
                validation_result['slotElicitationStyle'],
                validation_result['message']
            )

        return close(
            intent_request,
            session_attributes,
            'Fulfilled',
            {'contentType': 'PlainText',
             'content': 'Thanks'
             }
        )

def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': 'xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
        },
        'messages': [ message ],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }

# --- Intents ---

def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """
    intent_name = intent_request['sessionState']['intent']['name']
    response = None

    # Dispatch to your bot's intent handlers
```

```
    if intent_name == 'CheckAccount':
        response = check_account(intent_request)

    return response

# --- Main handler ---

def lambda_handler(event, context):
    """
    Route the incoming request based on the intent.

    The JSON body of the request is provided in the event slot.
    """

    # By default, treat the user request as coming from
    # Eastern Standard Time.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()

    logger.debug('event={}'.format(json.dumps(event)))
    response = dispatch(event)
    logger.debug("response={}".format(json.dumps(response)))

    return response
```

Monitoraggio delle prestazioni dei bot

Il monitoraggio è importante per mantenere l'affidabilità, la disponibilità e le prestazioni dei chatbot Amazon Lex V2. Questo argomento descrive l'utilizzo dei log delle conversazioni per monitorare le conversazioni tra gli utenti e i chatbot, l'utilizzo delle statistiche sugli enunciati per determinare gli enunciati rilevati e non rilevati dai bot e come utilizzare Amazon Logs e monitorare Amazon Lex CloudWatch V2. AWS CloudTrail Descrive inoltre i parametri di runtime e associazione di canale di Amazon Lex V2.

Usa questi strumenti e metriche per capire quali direzioni e azioni puoi intraprendere per migliorare le prestazioni dei tuoi bot.

Argomenti

Misurazione delle prestazioni aziendali con Analytics

Con Analytics, puoi valutare le prestazioni del tuo bot con metriche correlate ai tassi di successo e fallimento delle interazioni dei bot con i clienti. Puoi anche visualizzare i modelli dei flussi di conversazione tra il tuo bot e i clienti. Analytics semplifica la tua esperienza riassumendo queste metriche in grafici e diagrammi. Analytics fornisce strumenti che consentono di filtrare i risultati per identificare problemi e problematiche riguardanti intenti, intervalli, espressioni e conversazioni. Puoi utilizzare questi dati per iterare e migliorare il tuo bot per creare un'esperienza cliente migliore.

> **Note**
>
> Affinché un utente possa accedere ad Analytics, al suo ruolo IAM deve essere associata la policy Policy gestita da AWS: AmazonLexFullAccess o una policy personalizzata che includa le autorizzazioni dell'API di analisi. Vedi Gestione delle autorizzazioni di accesso per l'analisi i dettagli su come gestire le autorizzazioni degli utenti con una policy personalizzata. Se Policy gestita da AWS: AmazonLexReadOnly è associato al ruolo IAM di un cliente, viene

visualizzato un errore che mostra le autorizzazioni mancanti che devi aggiungere al ruolo IAM dell'utente per consentirgli di accedere alle dashboard di Analytics.

Per accedere ad Analytics

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Nel riquadro di navigazione sotto Bot, seleziona il bot che desideri visualizzare nelle analisi.
3. Seleziona la sezione in Analisi che desideri visualizzare.

Argomenti

Definizioni chiave

Questo argomento fornisce le definizioni chiave che vi aiuteranno a interpretare l'analisi dei bot. Queste definizioni sono correlate alle prestazioni del bot in quattro contesti: intenti, slot, conversazioni ed enunciati. I seguenti campi sono rilevanti per molte delle metriche delle prestazioni:

- Il `state`campo dell'`Intent`oggetto.
- Il `type`campo dell'`dialogAction`oggetto all'interno dell'SessionStateoggetto.

Intenti

Amazon Lex V2 classifica gli intenti nei seguenti modi:

- Successo: il bot ha raggiunto con successo l'intento. Si verifica una delle seguenti situazioni:
  - L'intento `state` è `ReadyForFulfillment` e il `type` di `dialogAction` è`Close`.
  - L'intento `state` è `Fulfilled` e il `type` di `dialogAction` è. `Close`
- Fallito: il bot non è riuscito a soddisfare l'intento. Lo stato dell'intento. Si verifica una delle seguenti situazioni:
  - L'intento `state` è `Failed` e l'`type`of `dialogAction` è `Close` (ad esempio, l'utente ha rifiutato la richiesta di conferma).
  - Il bot passa alla modalità `AMAZON.FallbackIntent` prima che l'intento sia completato.
- Cambiato : il bot riconosce un intento diverso e passa invece a tale intento, prima che l'intento originale venga classificato come riuscito o fallito.
- Eliminato : il cliente non risponde prima che l'intento venga classificato come riuscito o fallito.

Slot

Amazon Lex V2 classifica gli slot nei seguenti modi:

- Operazione riuscita: il bot ha riempito lo slot ed è passato con successo a un altro slot o alla fase di conferma.
- Fallito: il bot non è riuscito a riempire lo slot, anche dopo aver raggiunto il numero massimo di tentativi.
- Eliminato : il cliente non risponde o passa a un altro intento prima che lo slot venga classificato come riuscito o non riuscito.

Conversazioni

Quando un cliente effettua una chiamata di runtime ad Amazon Lex V2, fornisce una `sessionId`e Amazon Lex V2 genera una. `originatingRequestId` Se il cliente non risponde entro il timeout di sessione (`idleSessionTTLInSeconds`) impostato per il bot, la sessione scade. Se un cliente torna alla sessione utilizzando la stessa`sessionId`, Amazon Lex V2 ne genera una nuova`originatingRequestId`.

Per l'analisi, una conversazione è una combinazione unica di a `sessionId` e un`originatingRequestId`. Amazon Lex V2 classifica le conversazioni nei seguenti modi:

- Successo: l'intento finale della conversazione viene classificato come successo.

- Fallito: l'intento finale della conversazione è fallito. La conversazione ha esito negativo anche se Amazon Lex V2 ha come impostazione predefinita il. AMAZON.FallbackIntent
- Interrotta : il cliente non risponde prima che la conversazione venga classificata come riuscita o fallita.

Enunciazioni

Amazon Lex V2 classifica gli enunciati nei seguenti modi:

- Rilevato: Amazon Lex V2 riconosce l'enunciato come un tentativo di richiamare un intento configurato per un bot.
- Mancato: Amazon Lex V2 non riconosce l'enunciato.

Filtrare i risultati

Nella parte superiore di ogni pagina, puoi filtrare i risultati per l'analisi dei bot.
Opzioni di filtraggio per l'analisi.

È possibile filtrare in base ai seguenti parametri:

- Tempo: puoi filtrare i risultati in base a un intervallo di tempo relativo o assoluto. Quando selezioni un'ora di inizio e di fine, Amazon Lex V2 recupera le conversazioni iniziate dopo l'ora di inizio e terminate prima dell'ora di fine.
  - Intervallo relativo: seleziona 1d per visualizzare i risultati dell'ultimo giorno, 1w per la settimana precedente o 1m per l'ultimo mese.

    Per ulteriori opzioni, seleziona Personalizzato e scegli una durata nel menu Intervallo relativo. Per un maggiore controllo sulla durata, seleziona Intervallo personalizzato, inserisci un numero nel campo Durata e scegli un'unità di tempo dal menu a discesa.
  - Intervallo assoluto: seleziona Personalizzato e scegli il menu Intervallo assoluto per filtrare le conversazioni entro un intervallo di tempo specificato. Puoi scegliere una data di inizio e di fine sul calendario o inserirla nel formato YYYY/MM/DD.

> **Note**
>
> Il periodo di tempo massimo per cui puoi visualizzare i risultati è di 30 giorni.

- Filtri bot: per filtrare per locale, alias e versione del bot, seleziona i menu a discesa denominati Tutte le impostazioni locali, Tutti gli alias e Tutte le versioni.
- Modalità : seleziona l'icona a forma di ingranaggio e scegli il menu a discesa Modalità per scegliere se visualizzare i risultati per Speech o Text.
- Canale: seleziona l'icona a forma di ingranaggio e scegli il menu a discesa Canale per scegliere il canale per il quale desideri visualizzare i risultati. Per ulteriori informazioni sull'integrazione dei canali, consulta Integrazione di un bot Amazon Lex V2 con una piattaforma di messaggistica i centri di contatto Amazon Connect

Panoramica: un riepilogo delle prestazioni del tuo bot

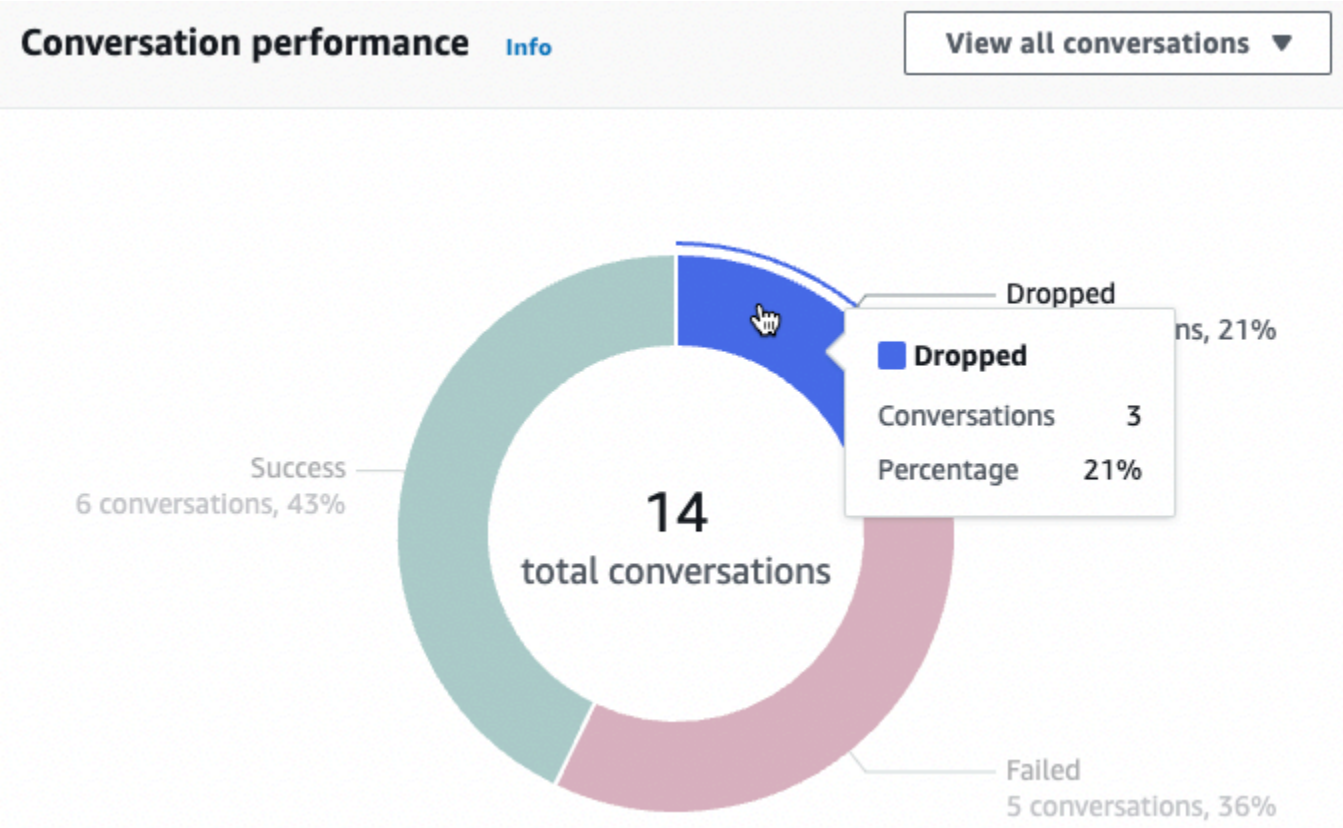
La pagina panoramica riassume le prestazioni del bot in termini di conversazioni, riconoscimento degli enunciati e utilizzo delle intenzioni. La panoramica è composta dalle seguenti sezioni:

- Prestazioni di conversazione
- Tasso di riconoscimento delle espressioni
- Cronologia delle prestazioni delle conversazioni
- I 5 principali intenti utilizzati
- I 5 principali intenti falliti

Prestazioni di conversazione

Usa questo grafico per tenere traccia del numero e della percentuale di conversazioni classificate come riuscite, non riuscite e interrotte. Per accedere a un elenco di conversazioni, seleziona Visualizza tutte le conversazioni per visualizzare un menu a discesa. Puoi scegliere di visualizzare un elenco di tutte le conversazioni degli utenti con il bot o filtrare le conversazioni con un risultato specifico (successo, non riuscito o interrotto). Questi link portano alla sottosezione Conversazioni del pannello di controllo Conversazioni. Per ulteriori informazioni, consulta Conversazioni.

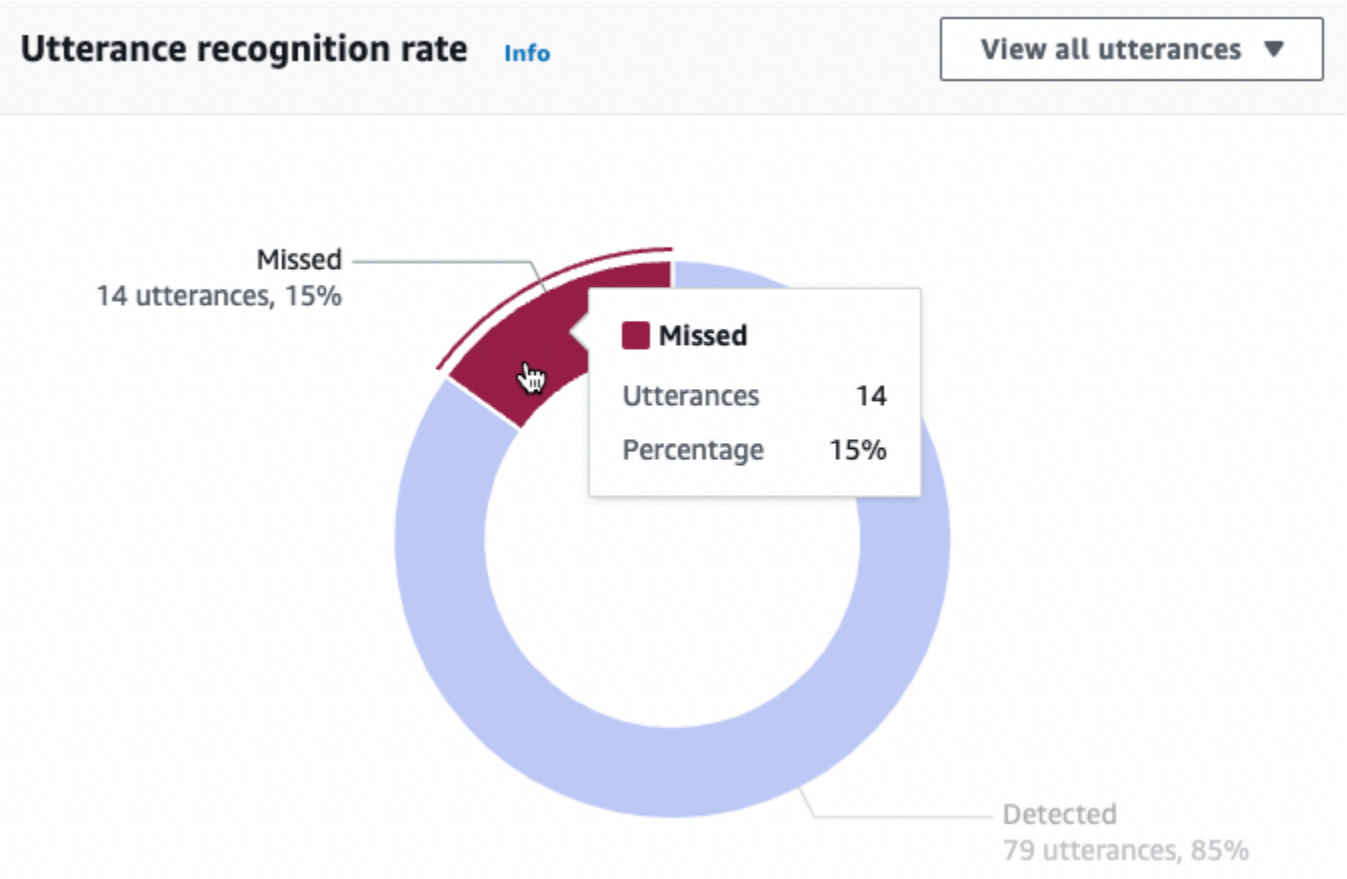
Per visualizzare un riquadro con il conteggio e la percentuale di conversazioni con quel risultato, passa il mouse su un segmento del grafico, come nell'immagine seguente.



Tasso di riconoscimento delle espressioni

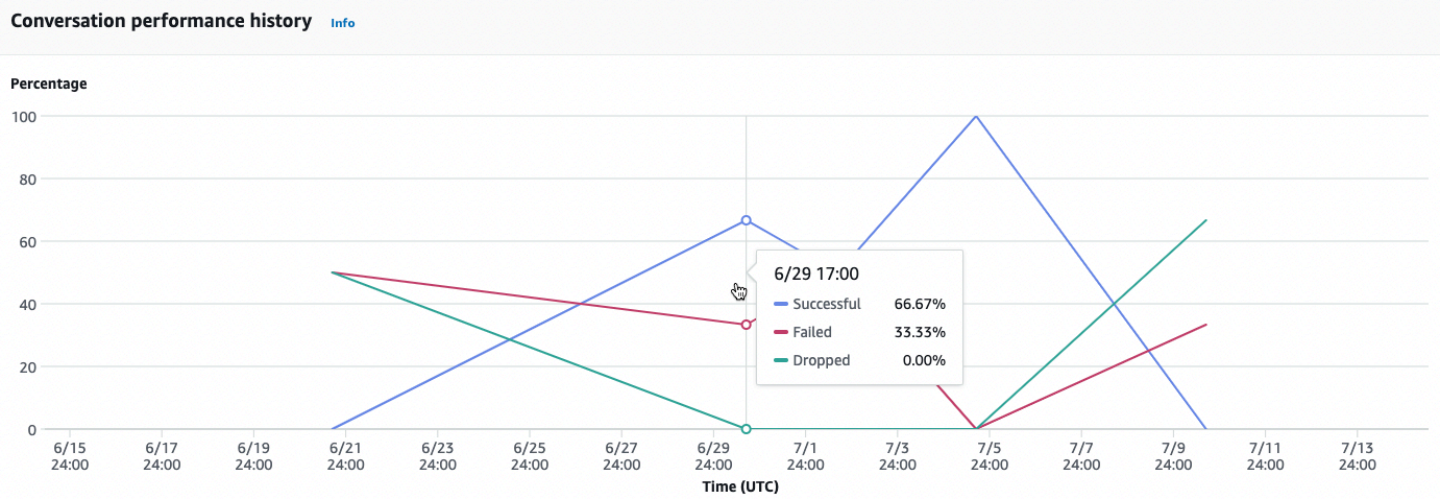
Usa questo grafico per tenere traccia del numero e della percentuale di enunciati rilevati e non rilevati dal bot. Per accedere a un elenco di enunciati, seleziona Visualizza enunciati per visualizzare un menu a discesa. Puoi scegliere di visualizzare un elenco di tutte le espressioni dell'utente o filtrare le espressioni con un risultato specifico (mancate o rilevate). Questi link rimandano alla sottosezione di riconoscimento di Utterance del pannello di controllo delle prestazioni. Per ulteriori informazioni, consulta Visualizza enunciati a cui accedere. Riconoscimento delle espressioni

Per visualizzare un riquadro con il conteggio e la percentuale di enunciati, passa il mouse su un segmento del grafico, come mostrato nell'immagine seguente.



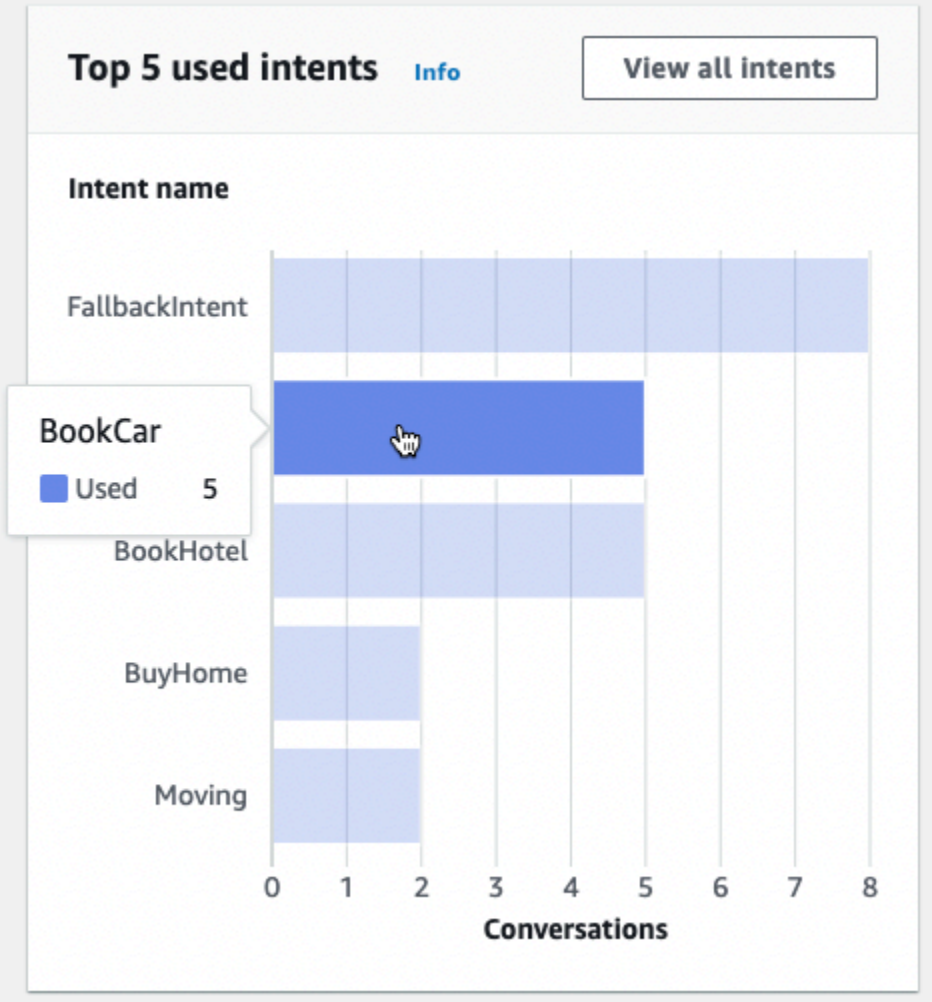
Cronologia delle prestazioni delle conversazioni

Usa questo grafico per tenere traccia della percentuale di conversazioni classificate come riuscite, non riuscite e interrotte nell'intervallo di tempo impostato nei filtri. Per visualizzare la percentuale di conversazioni con un risultato specifico in un intervallo di tempo, posiziona il mouse su quell'intervallo, come nell'immagine seguente.



I 5 principali intenti utilizzati

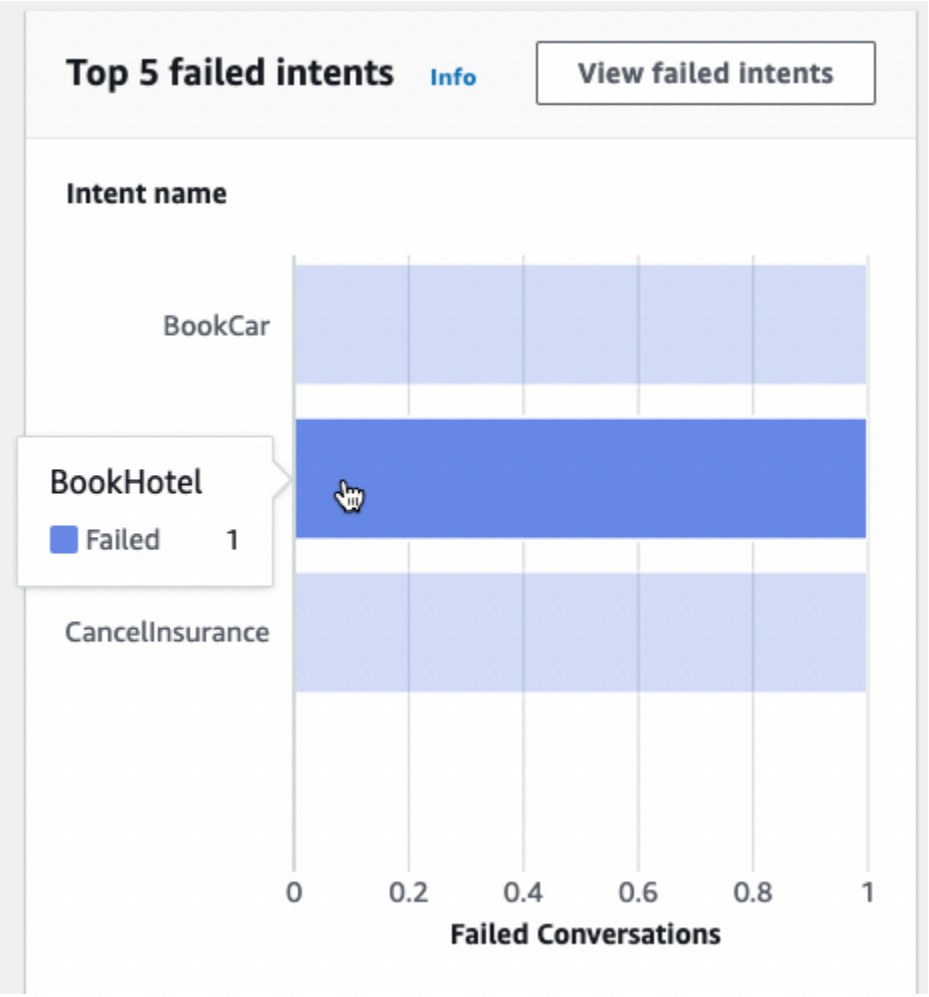
Usa questo grafico per identificare i cinque principali intenti utilizzati dai clienti con il tuo bot. Passa il mouse su una barra per vedere quante volte il bot ha riconosciuto quell'intento, come nell'immagine seguente.



Seleziona Visualizza tutti gli intenti per accedere alla sottosezione relativa alle prestazioni degli intenti della dashboard Performance, dove puoi visualizzare le metriche relative alle prestazioni del bot nell'adempimento degli intenti. Per ulteriori informazioni, consulta Prestazioni degli intenti.

I 5 principali intenti falliti

Usa questo grafico per identificare i cinque principali intenti che il tuo bot non è riuscito a soddisfare (vedi Intenti per la definizione di intento fallito). Passa il mouse su una barra per vedere quante volte il bot non è riuscito a soddisfare quell'intento, come nell'immagine seguente.



Seleziona Visualizza gli intenti falliti per accedere alla sottosezione relativa alle prestazioni degli intenti della dashboard Performance, dove puoi visualizzare le metriche relative agli intenti che il bot non è riuscito a soddisfare. Per ulteriori informazioni, consulta Prestazioni degli intenti.

Dashboard delle conversazioni: un riepilogo delle conversazioni con i bot

La dashboard delle conversazioni visualizza le metriche relative alle conversazioni dei clienti (vedi Conversazioni per la definizione di conversazione) con il tuo bot.

Il riepilogo contiene le seguenti informazioni sulle conversazioni degli utenti con il bot. I numeri vengono calcolati in base alle impostazioni del filtro.

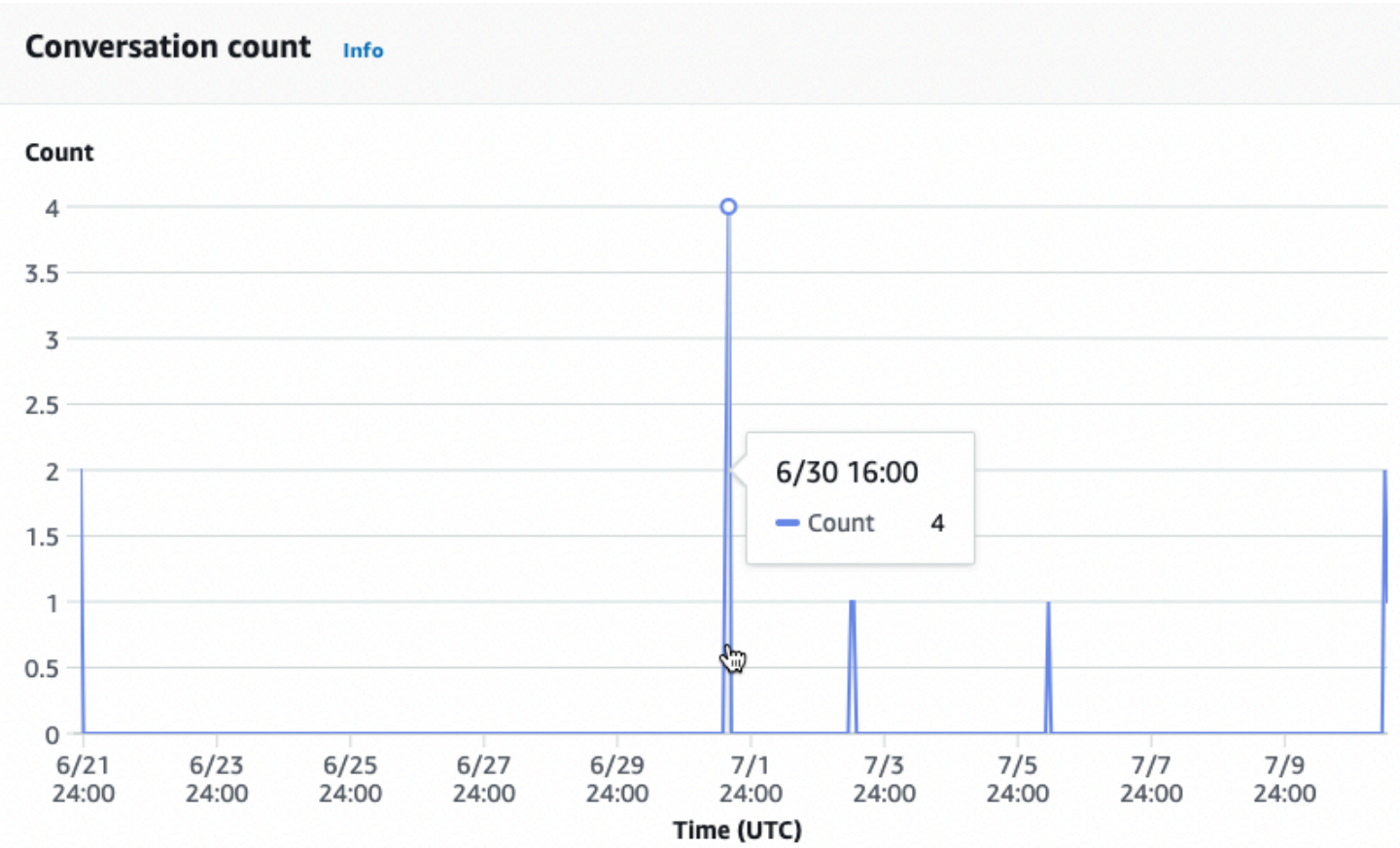
- Conversazioni totali: il numero totale di conversazioni con il bot.
- Durata media delle conversazioni: il tempo medio delle conversazioni degli utenti con il bot, espresso in minuti e secondi. Il formato è mm:ss.

- Numero medio di turni per conversazione: il numero medio di turni della durata di una conversazione.

Le sezioni Numero conversazioni e Numero messaggi contengono ciascuna un grafico che mostra il numero di conversazioni e messaggi, rispettivamente, nell'intervallo di tempo specificato nei filtri. Passa il mouse su un intervallo di tempo per visualizzare il numero di conversazioni o messaggi in quel segmento. La dimensione del segmento temporale dipende dall'intervallo di tempo specificato:

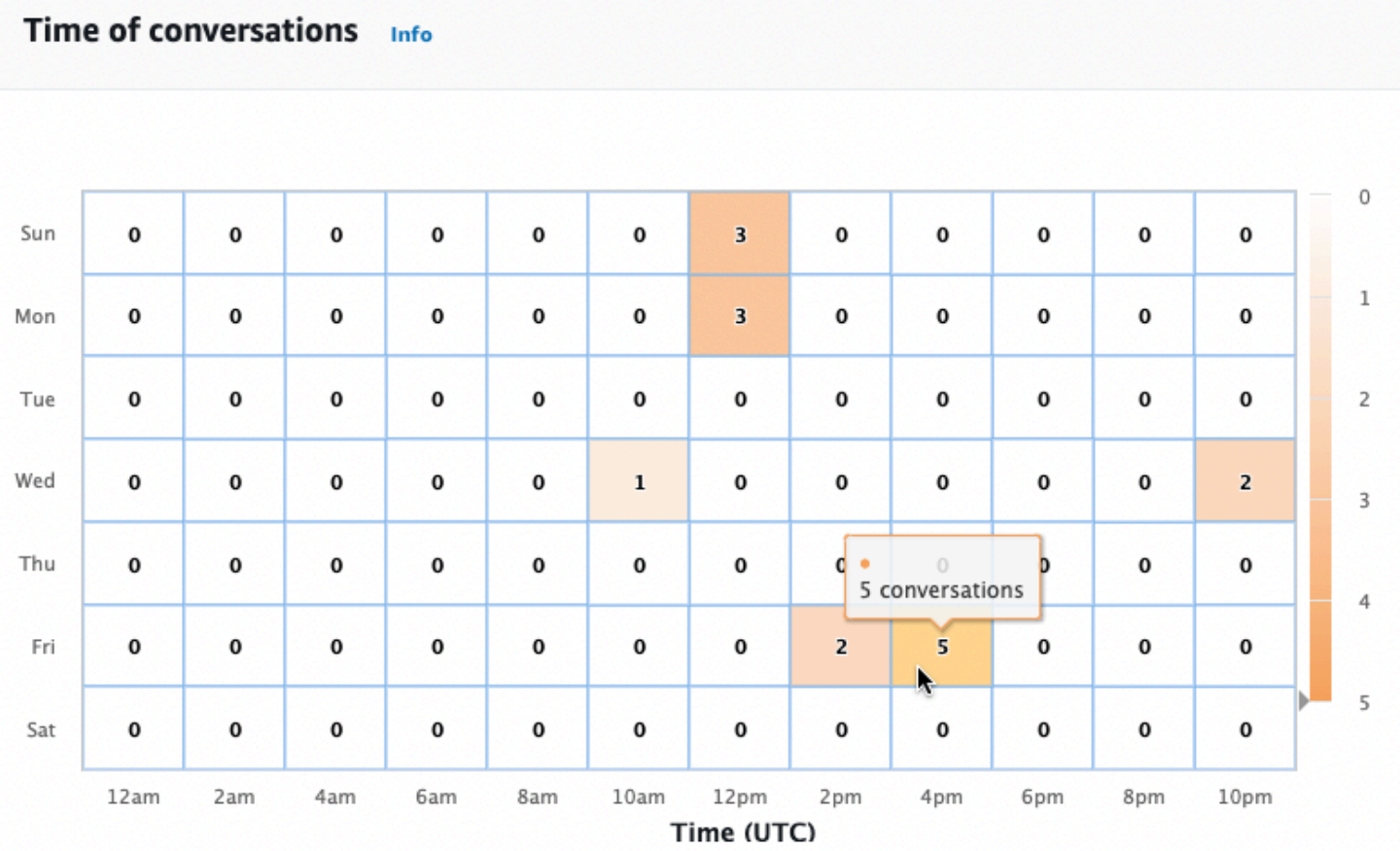
- Meno di 1 settimana: il conteggio viene visualizzato per ogni ora.
- 1 settimana o più: il conteggio viene visualizzato per ogni giorno.

Guarda un esempio del comportamento del passaggio del mouse nell'immagine seguente.



La sezione Ora delle conversazioni mostra il numero di conversazioni che hanno avuto luogo tra il bot e i clienti in ogni intervallo di due ore in ogni giorno della settimana, entro l'intervallo di tempo specificato nei filtri. Celle più scure indicano gli orari in cui si sono svolte più conversazioni. Passa il mouse su una cella per visualizzare il numero di conversazioni nelle 2 ore a partire da quella

fascia oraria. Ad esempio, l'azione nell'immagine seguente mostra il numero di conversazioni che si verificano tra le 16:00 e le 18:00 UTC.



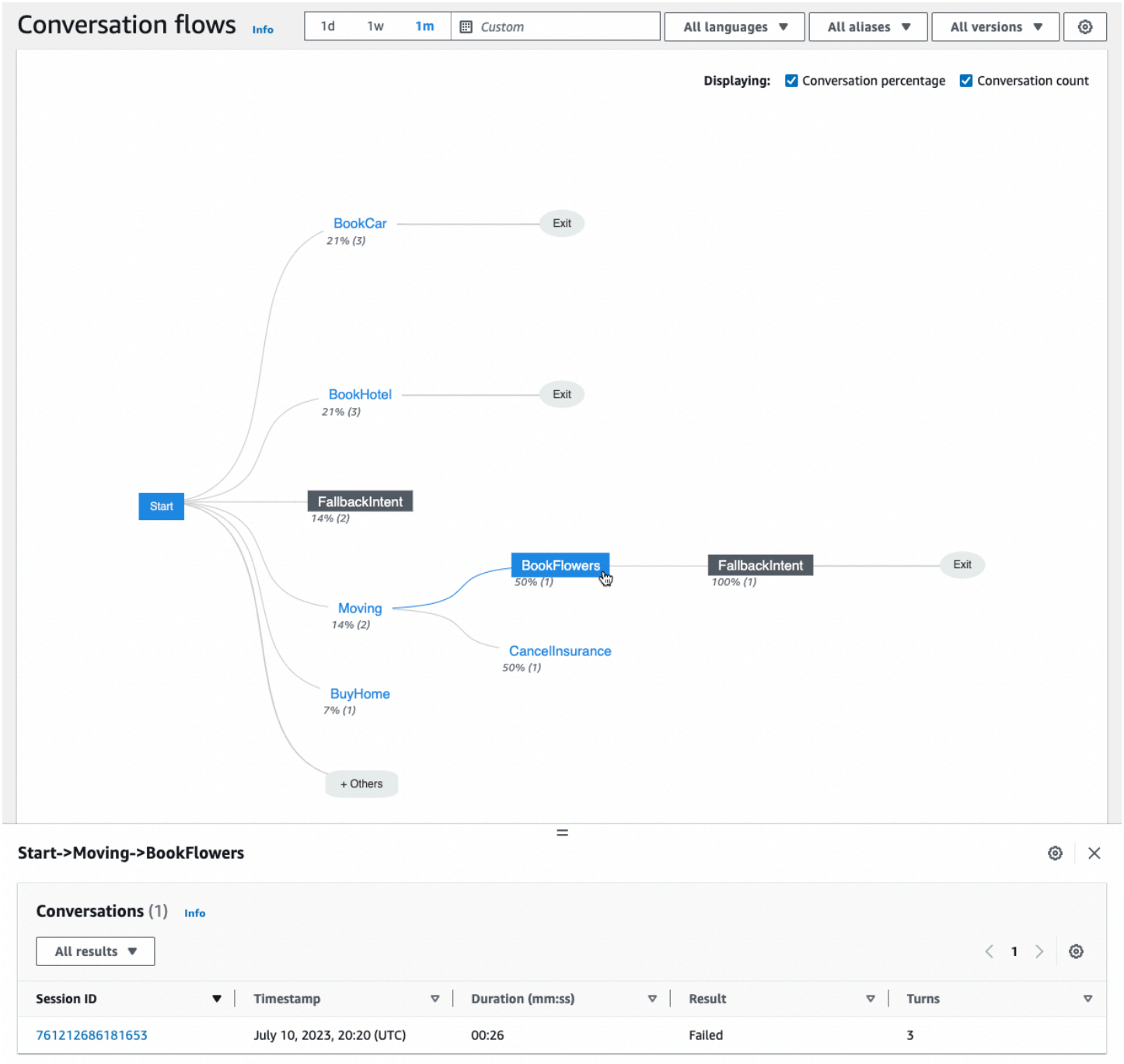
La dashboard Conversation contiene due strumenti, Flussi di conversazione e Conversazioni. Accedi a uno strumento selezionandolo nella dashboard di Conversazione nel riquadro di navigazione a sinistra.

Flussi di conversazione

Utilizza i flussi di conversazione per visualizzare gli ordini di intenti che i clienti seguono nelle conversazioni con il tuo bot. Al di sotto di ogni intento ci sono la percentuale e il numero di conversazioni che hanno invocato quell'intento in quel momento della conversazione. Puoi cambiare la percentuale e il conteggio alla rovescia selezionando Percentuale di conversazione e Numero di conversazioni in alto. Per impostazione predefinita, i cinque intenti più comuni in quel momento della conversazione vengono visualizzati in ordine decrescente di frequenza. Seleziona + Altri per visualizzare tutti gli intenti.

Scegli un intento da espandere in una nuova colonna di rami che mostri un elenco degli intenti presi in quel momento della conversazione, ordinati in ordine decrescente.

Quando selezioni un nodo nel flusso di conversazione, puoi espandere la finestra sottostante per visualizzare un elenco di conversazioni che hanno seguito quell'ordine di intenti. Scegli l'ID di sessione corrispondente a una conversazione per visualizzare i dettagli di quella conversazione. L'immagine seguente mostra un flusso di conversazione e una finestra Conversazioni espansa nella parte inferiore.



Conversazioni

Lo strumento Conversazioni mostra un elenco di conversazioni per il tuo bot. Puoi selezionare una colonna da ordinare in base a quella colonna in ordine crescente o decrescente.

Per filtrare le conversazioni in base al risultato, seleziona Tutti i risultati e scegli Operazione riuscita, Non riuscita o Eliminata.

Per filtrare le conversazioni in base alla durata

1. Seleziona la barra di ricerca contrassegnata Filtra le conversazioni per durata
2. Definisci il filtro in uno dei seguenti modi:
   - Utilizzate le opzioni predefinite.
     a. Seleziona Durata.
     b. Scegliete tra gli operatori = (uguale), > (maggiore di) e < (minore di).
     c. Scegli un periodo di tempo.
   - Inserisci un input nel formato «Duration {operator} {number} sec» Ad esempio, per cercare tutte le conversazioni che durano più di 30 secondi, inserisci**`Duration > 30 sec`**. Specificate la durata in secondi.

Per visualizzare informazioni dettagliate sulla sessione, inclusi metadati, utilizzo delle intenzioni e una trascrizione, seleziona l'ID di sessione di una conversazione.

> **Note**
>
> Poiché una conversazione è una combinazione unica di un `sessionId` e`originatingRequestId`, la stessa `sessionId` può apparire più volte nella tabella.

La sezione Dettagli contiene i seguenti metadati:

- Timestamp: specifica la data e l'ora di inizio della conversazione. L'ora è nel formato hh:mm:ss.
- Durata: specifica la durata della conversazione in formato mm:ss. La durata non include la durata del timeout della sessione (). `idleSessionTTLInSeconds`
- Risultato : specifica se la conversazione è stata classificata come riuscita, non riuscita o interrotta. ConversazioniPer ulteriori dettagli su questi risultati, vedere.

- Modalità: specifica se la conversazione è stata `SpeechText`, o `DTMF` (pressioni del tastierino a toni tattili). Una conversazione composta da più modalità è. `Multimode`
- Canale: specifica il canale su cui si è svolta la conversazione, se applicabile. Per informazioni, consulta Integrazione di un bot Amazon Lex V2 con una piattaforma di messaggistica.
- Lingua: specifica la lingua del bot.

Gli intenti che il bot ha suscitato nella conversazione sono mostrati sotto Dettagli. Seleziona Vai all'intento per passare a quell'intento nell'editor degli intenti. Seleziona Snap to transcript per far scorrere automaticamente la trascrizione fino alla prima istanza in cui il bot ha suscitato l'intento.

Seleziona la freccia destra accanto al nome dell'intento per visualizzare i dettagli sugli slot utilizzati per l'intento, inclusi i nomi degli slot, il valore che il bot ha generato per ogni slot e il numero di volte in cui il bot ha tentato di suscitare ogni slot.

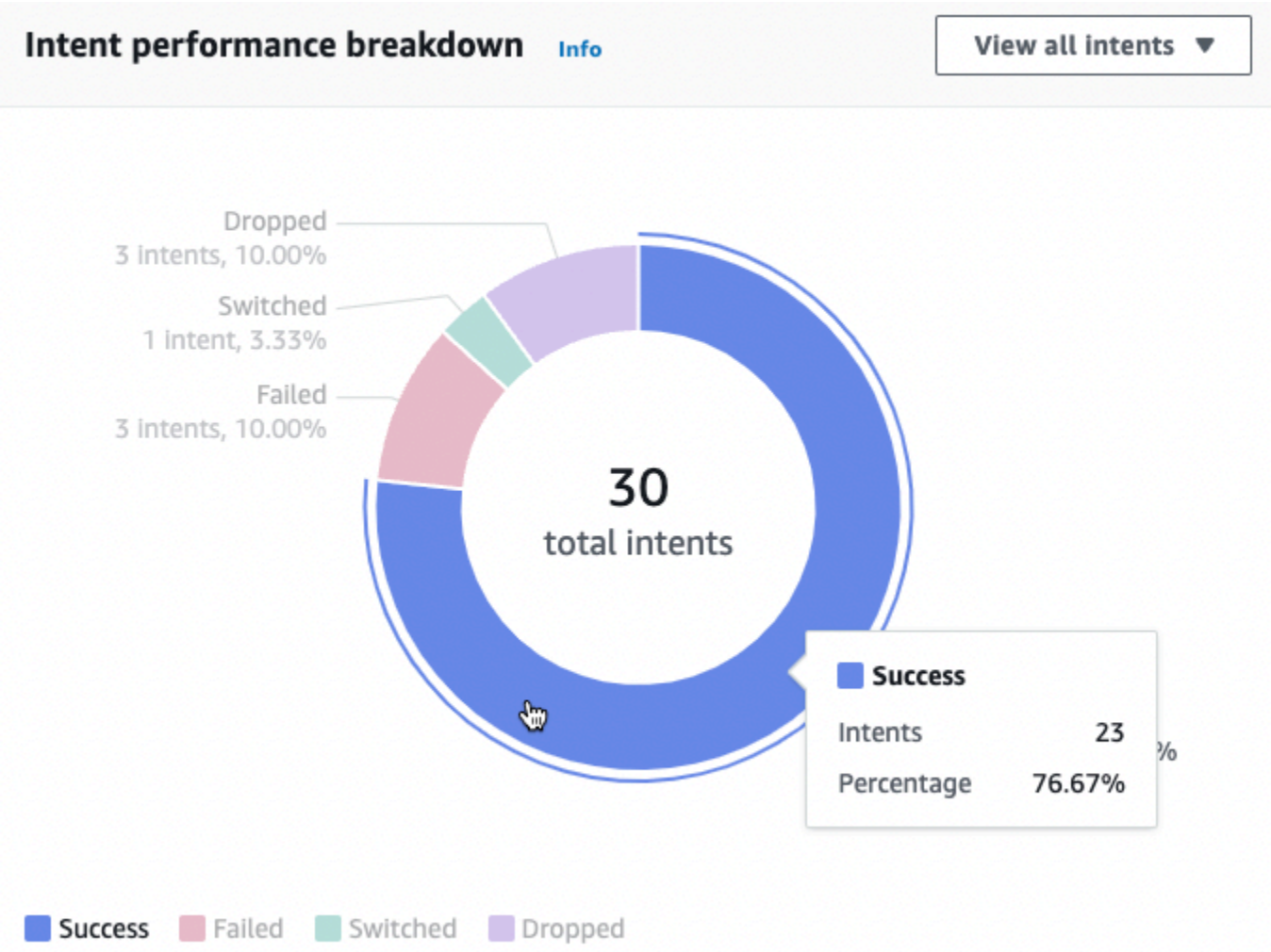
La Trascrizione consente di esaminare le espressioni della conversazione e il comportamento del bot nel suscitare intenti e slot. Gli enunciati degli utenti vengono visualizzati a sinistra e gli enunciati dei bot vengono visualizzati a destra. Utilizza la barra di ricerca denominata Filtra le trascrizioni in questa sessione per trovare il testo nella trascrizione. Accanto a Visualizzazione: ci sono tre informazioni mostrate in ogni turno di conversazione che puoi scegliere se visualizzare o meno:

- Timestamp: specifica l'ora dell'enunciato.
- Stato dell'intento: specifica l'intento che il bot suscita durante un'enunciazione e il risultato dell'intento, se applicabile. Sono possibili i seguenti stati di intento:
  - Intento invocato: *`nome dell'intento`*: il bot ha identificato un intento che il cliente sta invocando.
  - Intento cambiato: *`nome dell'intento: il bot è passato a un intento`* diverso in base all'enunciato.
  - *`nome dell'intento: successo: il`* bot ha raggiunto l'intento.
- Stato dello slot: specifica lo slot che il bot richiama durante un'enunciazione, se applicabile, e il valore fornito dal cliente.

Dashboard delle prestazioni: un riepilogo delle metriche relative alle intenzioni e alle espressioni del bot

Nella dashboard delle prestazioni, puoi visualizzare i dettagli sulle prestazioni relative alla realizzazione degli intenti del bot e al riconoscimento degli enunciati.

La sezione di analisi delle prestazioni degli intenti mostra il numero totale di volte in cui il bot ha invocato un intento e analizza il numero e la percentuale di volte in cui l'intento è stato classificato come riuscito, fallito, abbandonato e cambiato. Per una spiegazione di queste definizioni, vediIntenti. Passa il mouse su un segmento del grafico per visualizzare una casella con il conteggio e la percentuale di conversazioni con quel risultato, come nell'immagine seguente.



Seleziona Visualizza tutti gli intenti per visualizzare un menu a discesa, dal quale puoi scegliere di visualizzare un elenco di intenti suscitati dal bot. Puoi anche scegliere di visualizzare gli intenti con un risultato specifico (riuscito, fallito, abbandonato o cambiato). Questi collegamenti portano alla

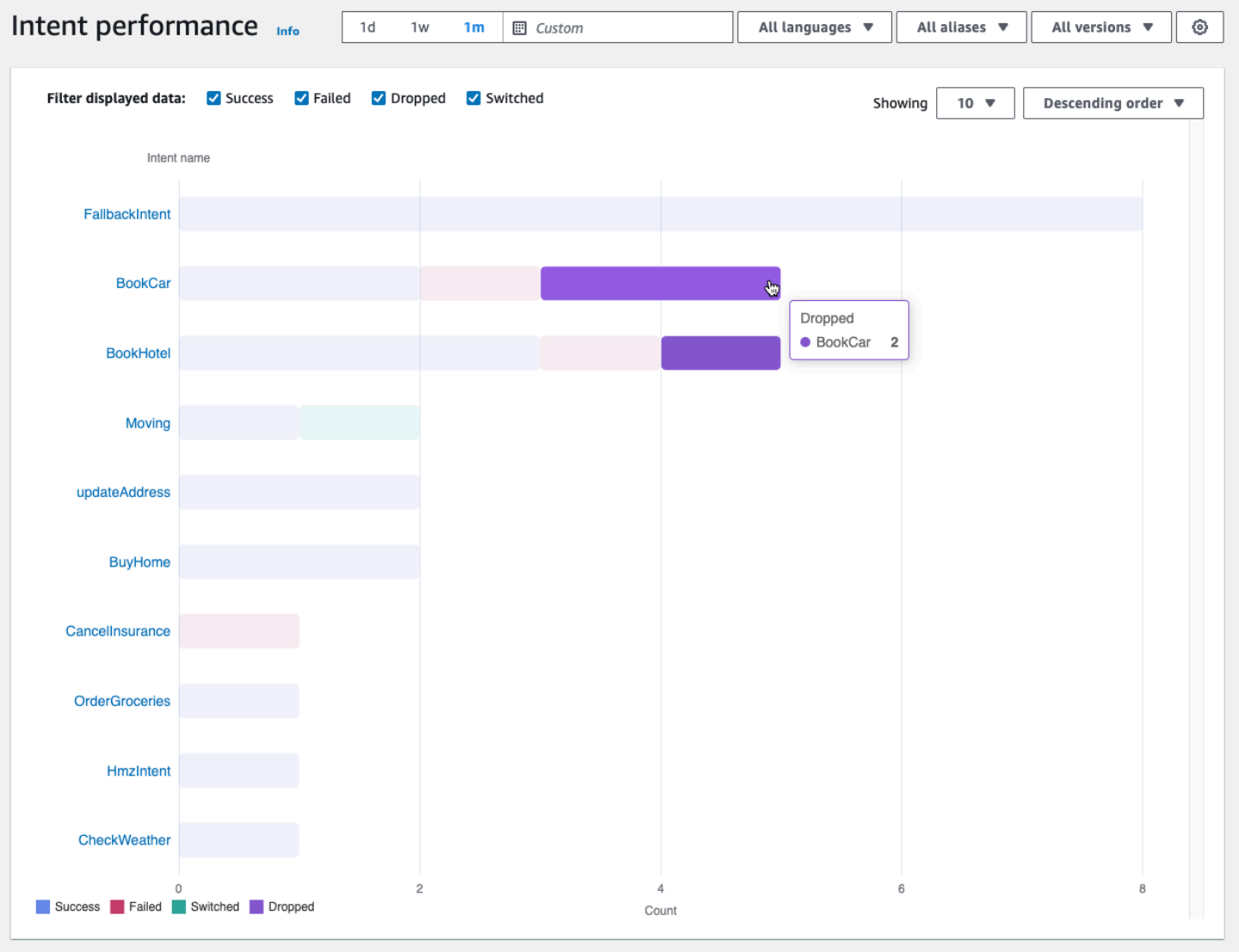
sottosezione relativa alle prestazioni degli Intent del pannello di controllo Performance. Per ulteriori informazioni, consulta Prestazioni degli intenti.

La sezione sul riconoscimento degli enunciati riassume il numero di enunciati che sono stati persi e rilevati. Seleziona Visualizza dettagli per accedere a un elenco di enunciati per il bot. Scegli il numero in Enunciati persi per visualizzare un elenco di enunciati persi e il numero in Enunciati rilevati per visualizzare un elenco degli enunciati rilevati per il bot. Per ulteriori informazioni, consulta Riconoscimento delle espressioni.

Seleziona Prestazioni degli intenti e Riconoscimento degli enunciati nella dashboard Prestazioni nella barra laterale sinistra per visualizzare i dettagli sugli intenti e gli enunciati nel tuo bot.

Prestazioni degli intenti

Questa dashboard riassume le prestazioni degli intenti utilizzati con il bot in ordine decrescente di frequenza. La barra accanto a ciascun intento visualizza il numero di volte in cui l'intento è stato classificato come riuscito, fallito, abbandonato e cambiato. Vedi Intenti per una spiegazione di queste definizioni. Passa il mouse su un segmento della barra per vedere il numero di conversazioni che utilizzano quell'intento con quel risultato, come nell'immagine seguente:



> **Note**
>
> La dashboard mostra i primi 1.000 risultati per una serie di impostazioni di filtro. Per ottenere risultati più mirati, configura le impostazioni granulari del filtro.

Nella parte superiore del grafico, puoi attivare o disattivare gli stati di intento che desideri visualizzare con le caselle di controllo Operazione riuscita, Non riuscita, Eliminata e Cambiata.

Seleziona i menu a discesa a destra di Showing per regolare il numero di intenti da visualizzare e se visualizzare gli intenti in ordine crescente o decrescente di frequenza.

Seleziona il nome di un intento per accedere a una pagina che mostra tre grafici: Intent performance breakdown, Slot performance e Intent switches.

La sezione di analisi delle prestazioni degli intenti mostra il numero totale di volte in cui il bot ha utilizzato l'intento e analizza il numero e la percentuale di volte in cui l'adempimento dell'intento è stato classificato come riuscito, fallito, interrotto e cambiato. Per una spiegazione di queste definizioni, vedere. Intenti Passa il mouse su un segmento del grafico per vedere il numero e la percentuale di volte in cui l'adempimento dell'intento ha prodotto quel risultato.

La sezione Prestazioni degli slot mostra le metriche relative agli slot che appartengono all'intento corrente. Per ordinare in base a una colonna, selezionate quella colonna una volta per ordinarla in ordine crescente e due volte per ordinarla in ordine decrescente. Puoi utilizzare la barra di ricerca per trovare uno slot specifico o utilizzare i pulsanti dei numeri di pagina per navigare tra gli slot.

> **Note**
>
> La dashboard mostra i primi 1.000 risultati per una serie di impostazioni di filtro. Per ottenere risultati più mirati, configura le impostazioni granulari del filtro.

La sezione Intent switches elenca i casi in cui il bot è passato dall'intento corrente a un altro con le seguenti informazioni:

- Fase: la fase della conversazione in cui il bot ha cambiato l'intento.
- Intento cambiato in: l'intento a cui il bot ha cambiato l'intento attuale.
- Numero di sessioni: il numero di sessioni in cui lo Stage e l'Intent sono passati alla combinazione.

> **Note**
>
> La dashboard mostra i primi 1.000 risultati per una serie di impostazioni di filtro. Per ottenere risultati più mirati, configura le impostazioni granulari del filtro.

Riconoscimento delle espressioni

Questa pagina elenca tutti gli enunciati persi e rilevati dal bot e fornisce gli strumenti per aggiungere esempi di enunciati agli intenti di addestramento del bot. Vedi Enunciazioni per una spiegazione di

queste definizioni. Usa le schede in alto per passare da un elenco di enunciati persi a uno di enunciati rilevati.

> **Note**
>
> La dashboard mostra i primi 1.000 risultati per una serie di impostazioni di filtro. Per ottenere risultati più mirati, configura le impostazioni granulari del filtro.

Per aggiungere enunciati a un intento:

1. Scegliete la casella di controllo accanto agli enunciati che desiderate aggiungere come enunciati di esempio per un intento.
2. Seleziona Aggiungi all'intento e scegli l'intento a cui desideri aggiungere gli enunciati nel menu a discesa sotto Intento.
3. Selezionare Aggiungi.

Utilizzo delle API per l'analisi

Questa sezione descrive le operazioni API utilizzate per recuperare le analisi per un bot.

> **Note**
>
> Per utilizzare ListUtteranceMetrics and ListUtteranceAnalyticsData, il tuo ruolo IAM deve disporre delle autorizzazioni per eseguire l'operazione ListAggregatedUtterances, che fornisce l'accesso alle analisi relative agli enunciati. Visualizzazione delle statistiche sugli enunciatiPer ulteriori dettagli, consulta la policy IAM da applicare al ruolo IAM.

- Le seguenti operazioni API recuperano le metriche di riepilogo per un bot:
  - ListSessionMetriche
  - ListIntentMetriche
  - ListIntentStageMetrics
  - ListUtteranceMetriche
- Le seguenti operazioni API recuperano un elenco di metadati per sessioni ed enunciati:
  - ListSessionAnalyticsData

  - ListUtteranceAnalyticsData
- L'operazione ListIntentPaths recupera le metriche relative a un ordine di intenti che i clienti adottano nelle conversazioni con un bot.

Filtrare i risultati

Le richieste dell'API Analytics richiedono di specificare e. `startTime` `endTime` L'API restituisce sessioni, intenti, fasi di intento o enunciati iniziati dopo *`startTime`* e terminati prima del. `endTime`

`filters`è un campo opzionale nelle richieste dell'API Analytics. Viene mappato a un elenco di AnalyticsSession oggetti AnalyticsIntentFilter AnalyticsIntentStageFilter, AnalyticsUtterance Filtero Filter. In ogni oggetto, utilizzate i campi per creare un'espressione in base alla quale filtrare. Ad esempio, se aggiungi il seguente filtro all'elenco, il bot cerca le conversazioni che durano più di 30 secondi.

```
{
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

Recupero delle metriche per un bot

Utilizza le `ListUtteranceMetrics` operazioni`ListSessionMetrics`, `ListIntentMetrics``ListIntentStageMetrics`, e per recuperare le metriche di riepilogo per sessioni, intenti, fasi degli intenti ed enunciati.

Per queste operazioni, compila i seguenti campi obbligatori:

- Fornisci `startTime` e `endTime` definisci un intervallo di tempo per il quale desideri recuperare i risultati.
- Specificate le metriche in cui desiderate calcolare`metrics`, un elenco di oggetti AnalyticsSessionmetrici, AnalyticsIntentmetrici o metrici. AnalyticsIntentStageMetricAnalyticsUtterance In ogni oggetto, utilizzate il `name` campo per specificare la metrica per calcolare il `statistic` campo per specificare se calcolare il `Sum``Average`, o il `Max` numero e il `order` campo per specificare se ordinare o ordinare i risultati. `Ascending` `Descending`

> **Note**
>
> Entrambi gli `binBy` oggetti `metrics` e contengono un `order` campo. È possibile specificare l'ordinamento solo `order` in uno dei due oggetti.

I campi rimanenti della richiesta sono facoltativi. È possibile filtrare e organizzare i risultati nei seguenti modi:

- Filtraggio dei risultati: utilizza il `filters` campo per filtrare i risultati. Per ulteriori dettagli, consulta Filtrare i risultati.
- Raggruppamento dei risultati per categoria: specifica il `groupBy` campo, un elenco contenente un singolo oggetto AnalyticsSessionResult AnalyticsIntentStageResult, AnalyticsIntentResult o AnalyticsUtteranceResult. Nell'oggetto, specificate il `name` campo con la categoria in base alla quale desiderate raggruppare i risultati.

  Se specificate un `groupBy` campo nella richiesta, l'`results`oggetto nella risposta contiene `groupByKeys` un elenco di oggetti AnalyticsSessionGroupByAnalyticsIntentGroupByKey AnalyticsIntentStageGroupByKey, AnalyticsUtteranceGroupByKey o Key, ciascuno con `name` quello specificato nella richiesta e un membro di quella categoria nel `value` campo.
- Risultati di binning per ora: specifica il `binBy` campo, un elenco contenente un singolo AnalyticsBinBySpecificationoggetto. Nell'oggetto, specifica il `name` campo con `ConversationStartTime` cui raggruppare i risultati entro quando è iniziata la conversazione o in cui `UtteranceTimestamp` racchiudere i risultati entro quando ha avuto luogo l'enunciato. Specificate l'intervallo di tempo in base al quale desiderate raggruppare i risultati nel `interval` campo e se ordinare in base `Ascending` o `Descending` in base al tempo nel `order` campo.

  Se specificate un `binBy` campo nella richiesta, l'`results`oggetto nella risposta contiene `binKeys` un elenco di oggetti AnalyticsBinKey, ciascuno con `name` quello specificato nella richiesta e l'intervallo di tempo che definisce il `value` contenitore nel campo.

> **Note**
>
> Entrambi gli `binBy` oggetti `metrics` e contengono un `order` campo. È possibile specificare l'ordinamento solo `order` in uno dei due oggetti.

Utilizzate i seguenti campi per gestire la visualizzazione della risposta:

- Specificate un numero compreso tra 1 e 1.000 nel `maxResults` campo per limitare il numero di risultati da restituire in una singola risposta.
- Se il numero di risultati è maggiore del numero specificato nel `maxResults` campo, la risposta contiene un`nextToken`. Ripeti la richiesta, ma usa questo valore nel `nextToken` campo per restituire il successivo batch di risultati.

Se si utilizza`ListUtteranceMetrics`, è possibile specificare gli attributi da restituire nel `attributes` campo. Questo campo è mappato a un elenco contenente un singolo oggetto AnalyticsUtteranceAttribute. Specificare `LastUsedIntent` nel `name` campo per restituire l'intento utilizzato da Amazon Lex V2 al momento dell'enunciato.

Nella risposta, il `results` campo viene mappato a un elenco di oggetti AnalyticsSessionResult AnalyticsIntentStageResult, AnalyticsIntentResult o AnalyticsUtterance Result. Ogni oggetto contiene un `metrics` campo che restituisce il valore di una statistica riassuntiva per una metrica richiesta, oltre a eventuali contenitori o gruppi creati con i metodi specificati.

Recupero dei metadati per sessioni ed enunciati in un bot

Utilizzate le ListUtteranceAnalyticsDataoperazioni ListSessionAnalyticsDataand per recuperare i metadati relativi a singole sessioni ed enunciati.

Compila i `endTime` campi obbligatori `startTime` e per definire un intervallo di tempo per il quale desideri recuperare i risultati.

I campi rimanenti della richiesta sono facoltativi. Per filtrare e ordinare i risultati:

- Filtraggio dei risultati: utilizza il `filters` campo per filtrare i risultati. Per ulteriori dettagli, consulta Filtrare i risultati.
- Ordinamento dei risultati: ordina i risultati in base al `sortBy` campo, che contiene un oggetto SessionDataSortByor UtteranceDataSortBy. Specificate il valore in base al quale desiderate ordinare nel `name` campo e se eseguire l'ordinamento `Ascending` o l'`Descending`ordinamento nel `order` campo.

Utilizzate i seguenti campi per gestire la visualizzazione della risposta:

- Specificate un numero compreso tra 1 e 1.000 nel `maxResults` campo per limitare il numero di risultati da restituire in una singola risposta.
- Se il numero di risultati è maggiore del numero specificato nel `maxResults` campo, la risposta contiene un`nextToken`. Ripeti la richiesta, ma usa questo valore nel `nextToken` campo per restituire il successivo batch di risultati.

Nella risposta, il `utterances` campo `sessions` or viene mappato a un elenco di UtteranceSpecificationoggetti SessionSpecificationor. Ogni oggetto contiene metadati per una singola sessione o enunciato.

Recupero dei metadati per sessioni ed enunciati in un bot

Utilizzate l'operazione ListIntentPaths per recuperare le metriche relative a un ordine di intenti che i clienti adottano durante una conversazione con un bot.

Per questa operazione, compila i seguenti campi obbligatori:

- Fornisci `startTime` e `endTime` definisci un intervallo di tempo per il quale desideri recuperare i risultati.
- Fornisci un `intentPath` campo per definire un ordine di intenti per il quale desideri recuperare le metriche. Separa gli intenti nel percorso con una barra in avanti. Ad esempio, compila il `intentPath` campo con **`/BookCar/BookHotel`** per visualizzare i dettagli su quante volte gli utenti hanno richiamato gli `BookCar` e gli `BookHotel` intenti in quell'ordine.

Utilizzate il `filters` campo opzionale per filtrare i risultati. Per ulteriori dettagli, consulta Filtrare i risultati.

Visualizzazione delle statistiche sugli enunciati

Puoi utilizzare le statistiche sugli enunciati per determinare gli enunciati che gli utenti inviano al tuo bot. Puoi vedere sia gli enunciati che Amazon Lex V2 rileva con successo sia quelli che non rileva. Puoi utilizzare queste informazioni per ottimizzare il tuo bot.

Ad esempio, se scopri che i tuoi utenti stanno inviando un enunciato che manca Amazon Lex V2, puoi aggiungere l'enunciato a un intento. La versione bozza dell'intento viene aggiornata con il nuovo enunciato e puoi testarla prima di distribuirla al bot.

Un'enunciazione viene rilevata quando Amazon Lex V2 riconosce l'enunciato come un tentativo di richiamare un intento configurato per un bot. Un enunciato viene perso quando Amazon Lex V2 non lo riconosce e lo richiama al suo posto. `AMAZON.FallbackIntent`

Le statistiche sugli enunciati possono essere visualizzate utilizzando l'API e l'API. `ListUtteranceMetrics` `ListAggregatedUtterance`

Le statistiche sugli enunciati non vengono generate utilizzando l'`ListUtteranceMetrics`API nelle seguenti condizioni:

- L'impostazione del Child Online Privacy Protection Act era impostata su Sì quando il bot è stato creato con la console, oppure il `childDirected` campo era impostato su true quando il bot è stato creato con l'`CreateBot`operazione.

L'`ListUtteranceMetrics`API offre funzionalità aggiuntive, tra cui:

- Sono disponibili ulteriori informazioni, ad esempio l'intento mappato per gli enunciati rilevati.
- Maggiore capacità di filtraggio (inclusi canale e modalità).
- Intervallo di date di conservazione più lungo (30 giorni).
- Puoi utilizzare l'API anche se hai disattivato l'archiviazione dei dati. La funzionalità della console per gli enunciati persi e rilevati si baserà sull'`ListUtteranceMetrics`API.

Le statistiche sugli enunciati non vengono generate utilizzando l'`ListAggregatedUtterance`API nelle seguenti condizioni:

- L'impostazione del Child Online Privacy Protection Act era impostata su Sì quando il bot è stato creato con la console, oppure il `childDirected` campo era impostato su true quando il bot è stato creato con l'`CreateBot`operazione.
- Stai utilizzando l'offuscamento degli slot con uno o più slot.
- Hai scelto di non partecipare al miglioramento di Amazon Lex.

L'`ListAggregatedUtterance`API offre funzionalità tra cui:

- Sono disponibili informazioni meno dettagliate (nessun intento mappato per gli enunciati).
- Capacità di filtraggio limitata (esclusi canale e modalità).
- Intervallo di date di conservazione breve (15 giorni).

Utilizzando le statistiche sugli enunciati, è possibile vedere se un enunciato specifico è stato rilevato o perso, oltre all'ultima volta che l'enunciato è stato utilizzato in un'interazione con un bot.

Amazon Lex V2 archivia gli enunciati in modo continuo mentre gli utenti interagiscono con il bot. Puoi interrogare le statistiche utilizzando la console o l'`ListAggregatedUtterances`operazione. Ha una conservazione dei dati di 15 giorni e non è disponibile se l'utente ha disattivato l'archiviazione dei dati. È possibile eliminare gli enunciati utilizzando l'`DeleteUtterances`operazione o disattivando l'archiviazione dei dati. Tutti gli enunciati vengono eliminati se si chiude l'account. AWS Gli enunciati memorizzati sono crittografati con una chiave gestita dal server.

Quando elimini una versione bot, le statistiche sugli enunciati sono disponibili per la versione per un massimo di 30 giorni con e 15 giorni di `ListUtteranceMetrics` utilizzo. `ListAggregatedUtterances` Non puoi visualizzare le statistiche relative alla versione eliminata nella console Amazon Lex V2. Per visualizzare le statistiche relative alle versioni eliminate, puoi utilizzare entrambe `ListAggregatedUtterances` le `ListUtteranceMetrics` operazioni.

Sia con le API che con `ListAggregatedUtterances` le `ListUtteranceMetrics` API, gli enunciati vengono aggregati in base al testo dell'enunciato. Ad esempio, tutti i casi in cui il cliente ha usato la frase «Voglio ordinare una pizza» vengono aggregati nella stessa riga in una risposta. Quando si utilizza l'RecognizeUtteranceoperazione, il testo utilizzato è la trascrizione di input.

Per utilizzare le `ListUtteranceMetrics` API `ListAggregatedUtterances` and, applica la seguente politica a un ruolo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
            "Resource": "*"
        }
    ]
}
```

Gestione delle autorizzazioni di accesso per l'analisi

Per fornire a un utente l'accesso all'analisi, collega una policy a un ruolo IAM che consenta al ruolo di richiamare le operazioni API per l'analisi. Puoi collegarlo Policy gestita da AWS:

AmazonLexFullAccess al ruolo IAM per fornire l'accesso completo alle operazioni dell'API Amazon Lex oppure puoi creare una policy personalizzata che consenta solo le autorizzazioni per l'analisi e collegarla a un ruolo IAM.

Per creare una policy personalizzata contenente le autorizzazioni per l'analisi

1. Se devi prima creare un ruolo IAM, segui i passaggi riportati in Creazione di un ruolo per delegare le autorizzazioni a un utente IAM.
2. Segui i passaggi descritti in Creazione di policy IAM per creare una policy utilizzando il seguente oggetto JSON. Per consentire l'accesso analitico a bot specifici per il ruolo IAM, aggiungi l'ARN di ciascun bot al `Resource` campo. Sostituisci la *`regione`*, l'*`account-id`* e il *`BOTID`* con i valori corrispondenti ai bot. Puoi anche sostituire l'identificatore della dichiarazione con un nome *`AnalyticsActions`*a tua scelta.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AnalyticsActions",
            "Effect": "Allow",
            "Action": [
                "lex:ListAggregatedUtterances",
                "lex:ListIntentMetrics",
                "lex:ListSessionAnalyticsData",
                "lex:ListIntentPaths",
                "lex:ListIntentStageMetrics",
                "lex:ListSessionMetrics"
            ],
            "Resource": [
                "arn:aws:lex:region:account-id:bot/BOTID"
            ]
        }
    ]
}
```

3. Allega la policy che hai creato al ruolo a cui desideri concedere le autorizzazioni di analisi seguendo i passaggi riportati in Aggiungere e rimuovere le autorizzazioni di identità IAM.
4. Il ruolo dovrebbe ora avere le autorizzazioni per visualizzare le analisi per i bot che hai specificato.

Abilitazione dei registri delle conversazioni

Usa i registri delle conversazioni per archiviare le conversazioni degli utenti con il tuo bot. Esamina questi log per identificare i problemi relativi alle interazioni del bot con gli utenti e modifica il comportamento del bot con queste informazioni. Questa sezione descrive anche come offuscare i valori degli slot per proteggere la privacy degli utenti.

Argomenti

Registrazione con i registri 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 dell'RecognizeTextoperazione. È possibile registrare sia il testo che l'audio per l'RecognizeUtteranceoperazione. Abilitando i registri delle conversazioni, ottieni una visione dettagliata delle conversazioni che gli utenti hanno con il tuo 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 registri di testo archiviano l'input di testo, le trascrizioni dell'input audio e i metadati associati nei registri. CloudWatch I log audio archiviano 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, usa la console o l'operazione Alias o CreateBotAlias. UpdateBot Dopo aver abilitato i registri delle conversazioni per un alias, utilizzando l'RecognizeUtteranceoperazione RecognizeTexto relativa a quell'alias vengono registrati gli enunciati di testo o audio nel gruppo di log Logs configurato o nel bucket S3. CloudWatch

Argomenti

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

Politiche IAM per i registri delle conversazioni

A seconda del tipo di registrazione selezionato, Amazon Lex V2 richiede l'autorizzazione per utilizzare i bucket Amazon CloudWatch Logs e Amazon Simple Storage Service (S3) per archiviare i log. È necessario creare AWS Identity and Access Management ruoli e autorizzazioni per consentire ad Amazon Lex V2 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 abiliti la crittografia degli oggetti per i tuoi oggetti S3, devi concedere l'autorizzazione di accesso alle AWS KMS chiavi utilizzate per crittografare gli oggetti.

Puoi utilizzare la console IAM, l'API IAM o AWS Command Line Interface per creare il ruolo e le policy. Queste istruzioni utilizzano il AWS CLI per creare il ruolo e le politiche.

> **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 registri 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 V2 come entità attendibile al ruolo. Ciò consente ad Amazon Lex di assumere il ruolo di fornire log alle risorse configurate per i log delle conversazioni.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. In AWS CLI, esegui 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
```

Successivamente, crea e allega una policy al ruolo che consenta ad Amazon Lex V2 di scrivere nei CloudWatch log.

Per creare una policy IAM per la registrazione del testo della conversazione in Logs CloudWatch

1. Crea un documento nella directory corrente denominata**`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. Nel AWS 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. Allega la policy al ruolo IAM che hai 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 V2 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. In AWS CLI, crea la policy IAM che conceda 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 utilizzi la console AWS Command Line Interface, il o un AWS SDK per specificare un ruolo IAM da utilizzare per i log delle conversazioni, l'utente che specifica il ruolo IAM dei log di conversazione deve avere l'autorizzazione per passare il ruolo ad Amazon Lex V2. Per consentire all'utente di passare il ruolo ad Amazon Lex V2, devi concedere l'`PassRole`autorizzazione all'utente, al ruolo o al gruppo IAM dell'utente.

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 il passaggio di un ruolo a un AWS servizio e IAM e AWS STS Condition Context Keys nella Guida per l'AWS Identity and Access Management utente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lexv2.amazonaws.com"
                },
```

```
                "StringLike": {
                    "iam:AssociatedResourceARN": "arn:aws:lex:region:account-
id:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

Configurazione dei registri delle conversazioni

È possibile abilitare e disabilitare i registri delle conversazioni utilizzando la console o il `conversationLogSettings` campo dell'operazione `CreateBotAlias` o`UpdateBotAlias`. 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 tuo AWS account. È possibile utilizzare qualsiasi gruppo di log valido. Il gruppo di log deve trovarsi nella stessa regione del bot Amazon Lex V2. Per ulteriori informazioni sulla creazione di un gruppo di log CloudWatch Logs, consulta Working with Log Groups and Log Streams nella Amazon CloudWatch Logs User Guide.

Per archiviare i log audio, usa un bucket Amazon S3 nel tuo account. AWS È possibile utilizzare qualsiasi bucket S3 valido. Il bucket deve trovarsi nella stessa regione del bot Amazon Lex V2. Per ulteriori informazioni sulla creazione di un bucket S3, consulta Creating a bucket nella Amazon Simple Storage Service Getting Started Guide.

Quando gestisci i log delle conversazioni utilizzando la console, la console aggiorna il tuo ruolo di servizio in modo che abbia accesso al gruppo di log e al bucket S3.

Se non utilizzi la console, devi fornire un ruolo IAM con policy che consentano ad Amazon Lex V2 di scrivere nel gruppo di log o nel bucket configurato. Se crei un ruolo collegato al servizio utilizzando il AWS Command Line Interface, devi aggiungere un suffisso personalizzato al ruolo utilizzando l'`custom-suffix`opzione come nell'esempio seguente. Per ulteriori informazioni, consulta Creazione di un ruolo IAM e delle policy per i log delle conversazioni.

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.amazon.aws.com \
    --custom-suffix suffix
```

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

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

Attivazione dei registri delle conversazioni

Per attivare i log utilizzando la console

1. Apri la console Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dall'elenco, scegliere un bot.
3. Dal menu a sinistra, scegli Alias.
4. Nell'elenco degli alias, scegli l'alias per il quale desideri configurare i registri delle conversazioni.
5. Nella sezione Registri delle conversazioni, scegli Gestisci i registri delle conversazioni.
6. Per i log di testo, scegli Abilita, quindi inserisci il nome del gruppo di log di Amazon CloudWatch Logs.
7. Per i log audio, scegli Enable, quindi inserisci le informazioni sul bucket S3.
8. Facoltativo. Per crittografare i log audio, scegli la AWS KMS chiave da usare per la crittografia.
9. Scegliere Save (Salva) per iniziare a registrare le conversazioni. Se necessario, Amazon Lex V2 aggiornerà il tuo ruolo di servizio con le autorizzazioni per accedere al gruppo di log CloudWatch Logs e al bucket S3 selezionato.

Disattivazione dei registri delle conversazioni

Per disattivare i log utilizzando la console

1. Apri la console Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dall'elenco, scegliere un bot.

3. Dal menu a sinistra, scegli Alias.
4. Nell'elenco degli alias, scegli l'alias per il quale desideri configurare i registri delle conversazioni.
5. Nella sezione Registri delle conversazioni, scegli Gestisci i registri delle conversazioni.
6. Disattiva la registrazione di testo, la registrazione audio o entrambe per disattivare la registrazione.
7. Scegliere Save (Salva) per interrompere la registrazione delle conversazioni.

Visualizzazione dei log di testo in Amazon CloudWatch Logs

Amazon Lex V2 archivia i log di testo per le tue conversazioni in Amazon CloudWatch Logs. Per visualizzare i log, usa la console o l'API CloudWatch Logs. Per ulteriori informazioni, consulta Search Log Data Using Filter Patterns e CloudWatch Logs Insights Query Syntax nella Amazon CloudWatch Logs User Guide.

Per visualizzare i log utilizzando la console Amazon Lex V2

1. Apri la console Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dall'elenco, scegliere un bot.
3. Dal menu a sinistra, scegli Analytics, quindi scegli CloudWatch Metriche.
4. Visualizza le metriche per il tuo bot nella pagina delle CloudWatch metriche.

Puoi anche utilizzare la CloudWatch console o l'API per visualizzare le 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 V2 o utilizzando l'DescribeBotoperazione Alias.

Le voci di registro relative all'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:

versione del messaggio

La versione dello schema dei messaggi.

bot

Dettagli sul bot con cui il cliente interagisce.

messaggi

La risposta che il bot ha inviato all'utente.

UtteranceContext

Informazioni sull'elaborazione di questo enunciato.

- `runtimeHints`—contesto di runtime utilizzato per trascrivere e interpretare l'input dell'utente. Per ulteriori informazioni, consulta Miglioramento del riconoscimento dei valori degli slot con suggerimenti di runtime.
- `slotElicitationStyle`—Stile di elicitazione degli slot utilizzato per interpretare l'input dell'utente. Per ulteriori informazioni, consulta Acquisizione dei valori degli slot con stili di ortografia.

Stato della sessione

Lo stato attuale della conversazione tra l'utente e il bot. Per ulteriori informazioni, consulta Gestire le conversazioni.

interpretazioni

Un elenco di intenti che Amazon Lex V2 ha stabilito potrebbero soddisfare l'enunciato dell'utente. Utilizzo dei punteggi di confidenza.

Fonte di interpretazione

Indica se uno slot viene risolto da Amazon Lex o Amazon Bedrock. Valori: Lex | Bedrock

sessionId

L'identificatore della sessione utente in cui è in corso la conversazione.

inputTranscript

Una trascrizione dell'input dell'utente.

- Per l'immissione di testo, si tratta del testo digitato dall'utente. Per l'input DTMF, questa è la chiave immessa dall'utente.
- Per l'input vocale, questo è il testo in cui Amazon Lex V2 converte l'enunciato dell'utente per richiamare un intento o riempire uno slot.

disegnare InputTranscript

La trascrizione non elaborata dell'input dell'utente prima dell'applicazione di qualsiasi elaborazione del testo. Nota: l'elaborazione del testo è disponibile solo per le versioni locali en-US e en-GB.

trascrizioni

Un elenco di potenziali trascrizioni dell'input dell'utente. Per ulteriori informazioni, consulta Utilizzo dei punteggi di confidenza nella trascrizione vocale.

Trascrizione RAW

Utilizzo dei punteggi di confidenza nella trascrizione vocale. Per ulteriori informazioni, consulta Utilizzo dei punteggi di confidenza nella trascrizione vocale.

Enunciato perso

Indica se Amazon Lex V2 è stato in grado di riconoscere l'enunciato dell'utente.

requestId

Amazon Lex V2 ha generato l'ID della richiesta per l'input dell'utente.

timestamp

Il timestamp dell'input dell'utente.

DeveloperOverride

Indica se il flusso di conversazione è stato aggiornato utilizzando un hook di codice di dialogo. Per ulteriori informazioni sull'utilizzo di un hook di codici di dialogo, vedereAbilitazione della logica personalizzata con AWS Lambda funzioni.

InputMode

Indica il tipo di input. Può essere audio, DTMF o testo.

requestAttributes

Gli attributi della richiesta utilizzati durante l'elaborazione dell'input dell'utente.

Proprietà audio

Se i registri delle conversazioni audio sono abilitati e l'input dell'utente era in formato audio, include la durata totale dell'ingresso audio, la durata della voce e la durata del silenzio nell'audio. Include anche un collegamento al file audio.

Bargein

Indica se l'input dell'utente ha interrotto la precedente risposta del bot.

Motivo della risposta

Il motivo per cui è stata generata una risposta. Può essere uno dei seguenti:

- `UtteranceResponse`— risposta all'input dell'utente
- `StartTimeout`— risposta generata dal server quando l'utente non ha fornito alcun input
- `StillWaitingResponse`— risposta generata dal server quando l'utente richiede al bot di attendere
- `FulfillmentInitiated`— risposta generata dal server che indica che l'adempimento sta per essere avviato
- `FulfillmentStartedResponse`— risposta generata dal server che indica che l'adempimento è iniziato
- `FulfillmentUpdateResponse`— risposta periodica generata dal server mentre l'adempimento è in corso
- `FulfillmentCompletedResponse`— risposta generata dal server quando l'adempimento è completo.

operationName

L'API utilizzata per interagire con il bot. Può essere uno dei `PutSessionRecognizeText`,`RecognizeUtterance`, o`StartConversation`.

```
{
    "message-version": "2.0",
    "bot": {
        "id": "string",
        "name": "string",
        "aliasId": "string",
        "aliasName": "string",
        "localeId": "string",
        "version": "string"
    },
    "messages": [
        {
            "contentType": "PlainText | SSML | CustomPayload | ImageResponseCard",
            "content": "string",
            "imageResponseCard": {
                "title": "string",
                "subtitle": "string",
                "imageUrl": "string",
                "buttonsList": [
                    {
```

```
                    "text": "string",
                    "value": "string"
                }
            ]
        }
    }
],
"utteranceContext": {
    "activeRuntimeHints": {
        "slotHints": {
            "string": {
                "string": {
                    "runtimeHintValues": [
                        {
                            "phrase": "string"
                        },
                        {
                            "phrase": "string"
                        }
                    ]
                }
            }
        }
    },
    "slotElicitationStyle": "string"
},
"sessionState": {
    "dialogAction": {
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
        "slotToElicit": "string"
    },
    "intent": {
        "name": "string",
        "slots": {
            "string": {
                "value": {
                   "interpretedValue": "string",
                   "originalValue": "string",
                   "resolvedValues": [ "string" ]
                }
             },
            "string": {
                "shape": "List",
                "value": {
```

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

```
                            {
                                "phrase": "string"
                            }
                        ]
                    }
                }
            }
        }
    },
   "dialogEventLogs": [
        {
   // only for conditional
     "conditionalEvaluationResult":[
      // all the branches until true

     {
     "conditionalBranchName": "string",
     "expressionString": "string",
     "evaluatedExpression": "string",
     "evaluationResult": "true | false"
     }
    ],
  "dialogCodeHookInvocationLabel": "string",
  "response": "string",
  "nextStep": {
        "dialogAction": {
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
            "slotToElicit": "string"
        },
       "intent": {
                           "name": "string",
           "slots": {
               }
        }
       }
    ]
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "nluConfidence": "string",
            "intent": {
                "name": "string",
                "slots": {
                    "string": {
```

```
                        "value": {
                            "originalValue": "string",
                            "interpretedValue": "string",
                            "resolvedValues": [ "string" ]
                        }
                    },
                    "string": {
                        "shape": "List",
                        "value": {
                            "interpretedValue": "string",
                            "originalValue": "string",
                            "resolvedValues": [ "string" ]
                        },
                        "values": [
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }
                            },
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }

                            }
                        ]
                    }
                },
                "kendraResponse": {
                    // Only present when intent is KendraSearchIntent. For details, see

                    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                    },
                "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
                "confirmationState": "Confirmed | Denied | None"
                },
            "sentimentResponse": {
```

```
                "sentiment": "string",
                "sentimentScore": {
                    "positive": "string",
                    "negative": "string",
                    "neutral": "string",
                    "mixed": "string"
                }
            }
        }
    ],
    "sessionId": "string",
    "inputTranscript": "string",
    "rawInputTranscript": "string",
    "transcriptions": [
        {
            "transcription": "string",
            "rawTranscription": "string",
            "transcriptionConfidence": "number",
            },
            "resolvedContext": {
                "intent": "string"
            },
            "resolvedSlots": {
                "string": {
                    "name": "slotName",
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "resolvedValues": [
                            "string"
                        ]
                    }
                }
            }
        }
    ],
    "missedUtterance": "bool",
    "requestId": "string",
    "timestamp": "string",
    "developerOverride": "bool",
    "inputMode": "DTMF | Speech | Text",
    "requestAttributes": {
        "string": "string"
    },
```

```
    "audioProperties": {
        "contentType": "string",
        "s3Path": "string",
        "duration": {
            "total": "integer",
            "voice": "integer",
            "silence": "integer"
        }
    },
    "bargeIn": "string",
    "responseReason": "string",
    "operationName": "string"
}
```

Il contenuto della voce di registro 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 `developerOverride` 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.

Accesso ai log audio in Amazon S3

Amazon Lex V2 archivia i registri audio delle tue conversazioni in un bucket S3.

Puoi utilizzare la console o l'API di Amazon S3 per accedere ai log audio. Puoi visualizzare il prefisso della chiave oggetto S3 dei file audio nella console Amazon Lex V2 o nel campo nella `DescribeBotAlias` risposta `conversationLogSettings` dell'operazione.

Monitoraggio dello stato del registro delle conversazioni tramite metriche CloudWatch

Usa Amazon CloudWatch per monitorare i parametri 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 V2 fornisce quattro parametri nello spazio dei `AWS/Lex` nomi per i log delle conversazioni:

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

Le metriche di successo mostrano che Amazon Lex V2 ha scritto correttamente i tuoi log audio o di testo nelle rispettive destinazioni.

Le metriche di errore mostrano che Amazon Lex V2 non è in grado di inviare log 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 V2 sia un'entità affidabile per il ruolo IAM.
- Per la registrazione del testo, assicurati che esista il gruppo di CloudWatch log Logs. Per la registrazione dell'audio, verificare l'esistenza del bucket S3.

- Assicurati che il ruolo IAM utilizzato da Amazon Lex V2 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 V2 e appartenga al tuo account.

Oscurare i valori degli slot nei registri delle conversazioni

Amazon Lex V2 consente di offuscare o nascondere il contenuto degli slot in modo che il contenuto non sia visibile. Per proteggere i dati sensibili acquisiti come valori degli slot, puoi abilitare l'offuscamento degli slot per mascherare questi valori nella registrazione di log.

Quando scegli di offuscare i valori dello slot, Amazon Lex V2 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:
    My name is John Stiles
After:
    My name is {full_name}
```

Se un enunciato contiene caratteri tra parentesi quadre ({}) Amazon Lex V2 elimina i caratteri tra parentesi con due barre rovesciate (\\). Ad esempio, il testo `{John Stiles}` viene offuscato come segue:

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

I valori degli slot vengono offuscati nei log delle conversazioni. I valori degli slot sono ancora disponibili nella risposta delle `RecognizeUtterance` operazioni `RecognizeText` e i valori degli slot sono disponibili per le funzioni Lambda di convalida e 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 V2 offusca i valori degli slot se riconosce uno slot e un valore di slot nell'enunciato. Se non viene riconosciuto alcun valore dello slot, Amazon Lex V2 non offusca l'enunciato.

Nella seconda e nelle fasi successive, Amazon Lex V2 conosce lo slot da attivare e se il valore dello slot deve essere offuscato. Se Amazon Lex V2 riconosce il valore dello slot, il valore viene offuscato. Se Amazon Lex V2 non riconosce un valore, l'intera espressione viene offuscata. Qualsiasi valore di slot in enunciazioni perse non verrà offuscato.

Amazon Lex V2, inoltre, non offusca i valori degli 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 V2 non offusca il valore dello slot nell'audio. Offusca il valore dello slot nella trascrizione audio.

Puoi scegliere quali slot offuscare utilizzando la console o l'API Amazon Lex V2. Nella console, selezionare Slot obfuscation (Offuscamento dello slot) nelle impostazioni di uno slot. Se utilizzi l'API, imposta il `obfuscationSetting` campo dello slot su `DEFAULT_OBFUSCATION` quando chiami l'operazione or. CreateSlotUpdateSlot

Acquisizione selettiva del registro delle conversazioni

L'acquisizione selettiva dei registri delle conversazioni consente all'utente di selezionare il modo in cui i registri delle conversazioni vengono acquisiti con dati di testo e audio delle conversazioni dal vivo.

Per abilitare e acquisire l'output della funzionalità di acquisizione selettiva dei log di conversazione, è necessario attivare la funzionalità nella console Amazon Lex V2 e abilitare gli attributi di sessione richiesti nelle impostazioni API per acquisire l'output selezionato dai log.

È possibile selezionare le seguenti opzioni per l'acquisizione selettiva dei log delle conversazioni:

- solo testo
- solo audio
- testo e audio

È possibile registrare parti specifiche della conversazione e scegliere se registrare l'audio, il testo o entrambi per il registro delle conversazioni.

> **Note**
>
> L'acquisizione selettiva dei log delle conversazioni funziona solo per Amazon Lex V2.

Argomenti

Gestisci l'acquisizione selettiva dei registri delle conversazioni

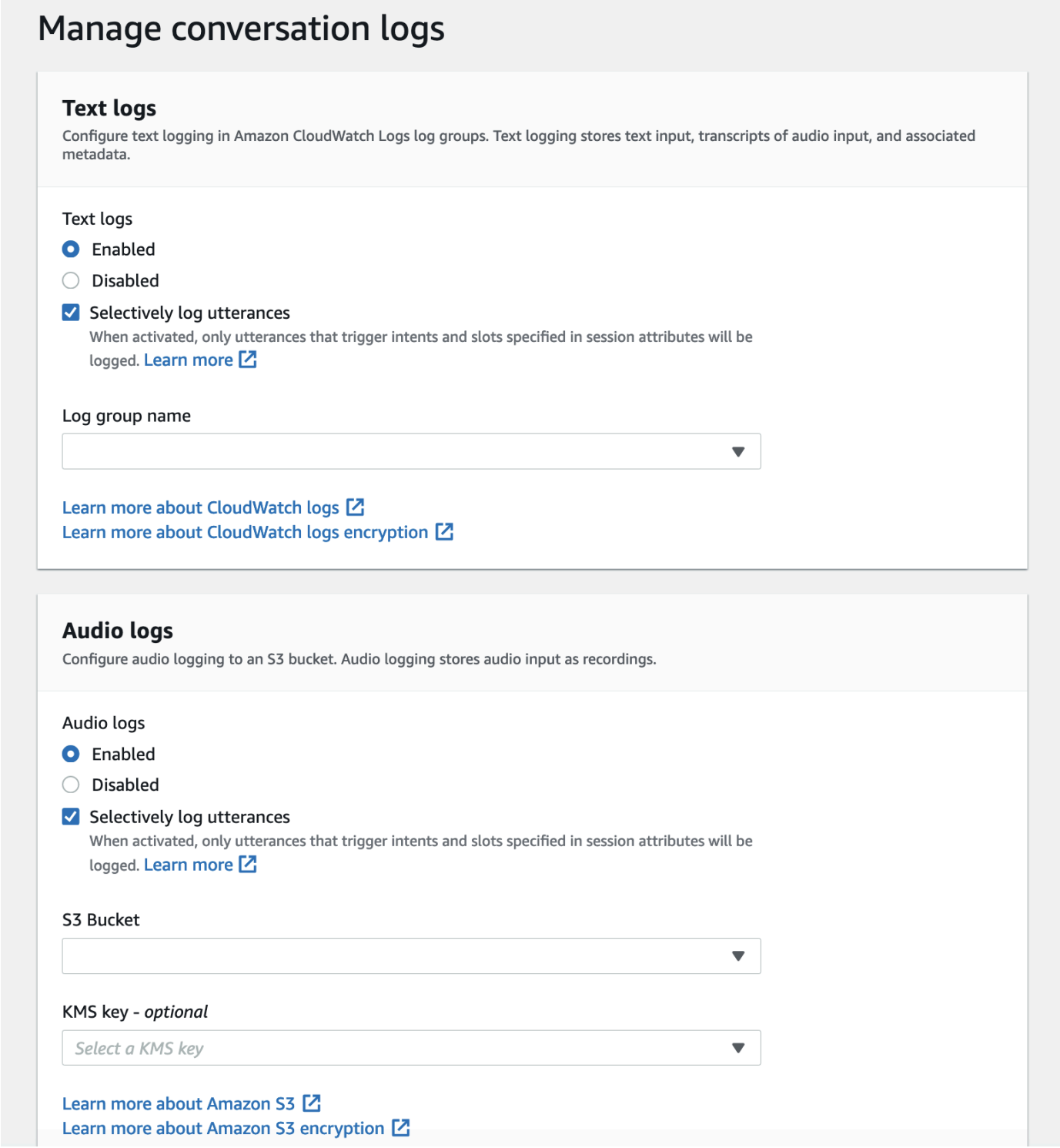
Utilizzando la console Lex, è possibile abilitare le impostazioni di acquisizione selettiva dei registri delle conversazioni e scegliere per quali slot abilitare l'acquisizione selettiva dei registri delle conversazioni.

Attiva l'acquisizione selettiva dei log delle conversazioni nella console Amazon Lex V2:

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Seleziona Bot dal pannello laterale sinistro e scegli il bot a cui desideri abilitare l'acquisizione selettiva dei log delle conversazioni. Usa un bot esistente o creane uno nuovo.
3. Scegli Alias per il bot selezionato nella sezione Distribuzione nel pannello laterale sinistro.
4. Scegli l'alias del bot, quindi seleziona Gestisci i registri delle conversazioni.
5. Nel pannello Gestisci i registri delle conversazioni, per i registri di testo, scegli se i registri di testo sono abilitati o disabilitati selezionando il pulsante di opzione. Se scegli Abilitato per i registri di testo, dovrai inserire un nome per il gruppo di log o scegliere un nome per il gruppo di log esistente dal menu a discesa. Seleziona la casella di controllo Registra selettivamente gli enunciati se stai registrando selettivamente i file di testo.

   > **Note**
   >
   > Abilita i registri di testo e/o audio selezionando la casella di controllo Registra selettivamente gli enunciati nelle impostazioni dei registri delle conversazioni (testo e/o audio) nelle impostazioni del tempo di compilazione. BotAlias È necessario configurare il gruppo di CloudWatch log e il bucket Amazon S3 per selezionare questa opzione.

6. Nella sezione Registri audio, scegli se abilitare o disabilitare i registri audio selezionando il pulsante di opzione. Se scegli Enabled for audio logs, devi specificare la posizione del bucket Amazon S3 e (opzionale) la chiave KMS per crittografare i dati audio. Seleziona la casella di controllo Registra selettivamente gli enunciati se stai registrando in modo selettivo i file audio.

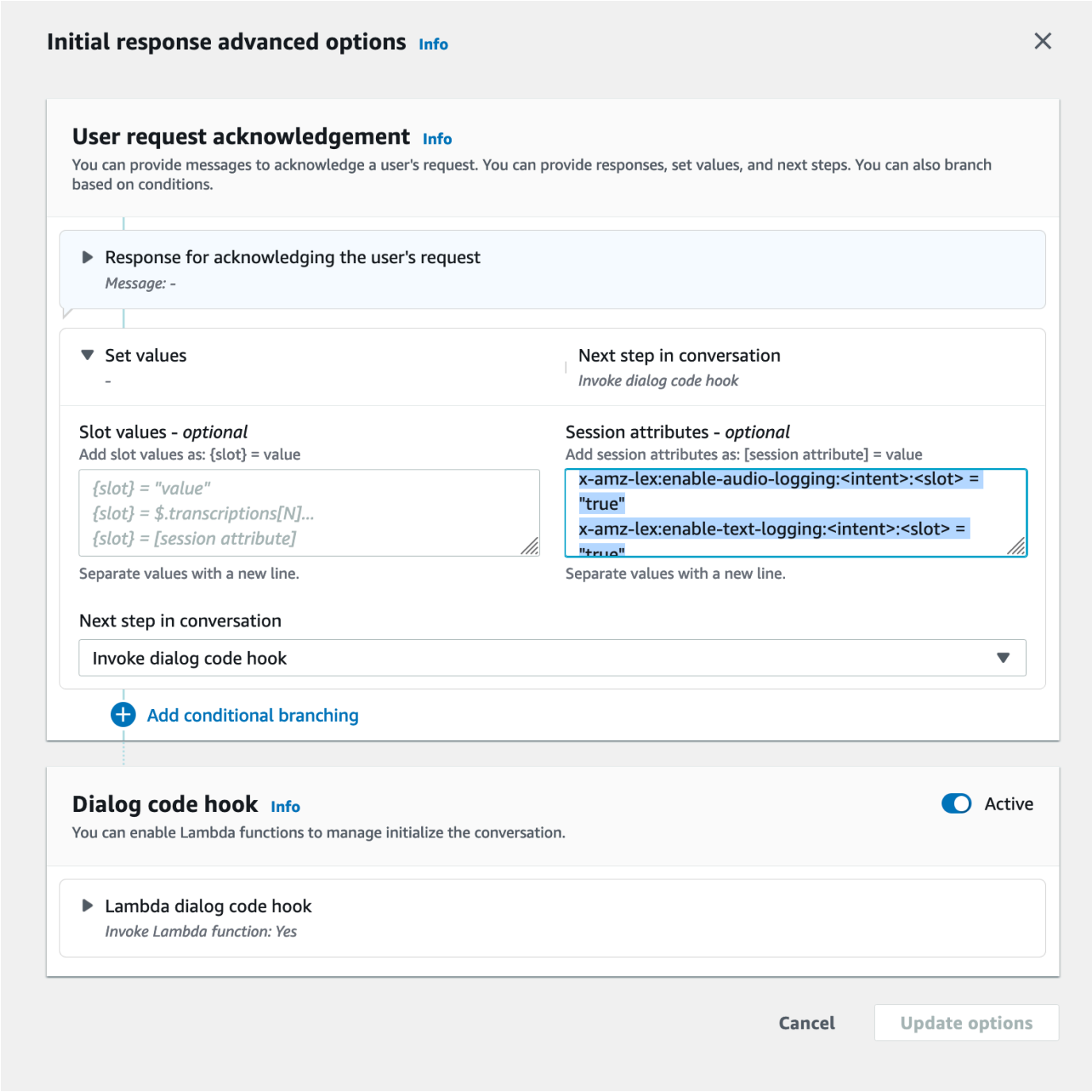


7. Seleziona Salva nell'angolo in basso a destra del pannello per salvare le impostazioni selettive di acquisizione dei registri delle conversazioni.

Attiva l'acquisizione selettiva del registro delle conversazioni nella console Lex:

1. Vai a Intents e seleziona il nome dell'intento, la risposta iniziale, le impostazioni avanzate, i valori impostati, gli attributi della sessione.
2. Imposta i seguenti attributi in base agli intenti e agli slot per i quali desideri abilitare l'acquisizione selettiva dei registri delle conversazioni:

   - `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`*` = "true"`
   - `x-amz-lex:enable-text-logging:`*`intent`*`:`*`slot`*` = "true"`



Note

Imposta `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`* `= "true"` per catturare le espressioni che contengono solo uno spazio specifico nella conversazione. L'azione di registrazione di un enunciato dipende dalla valutazione dell'*`intento`*: *`lo`*

spazio all'interno dell'enunciato, rispetto alle espressioni degli attributi di sessione e il valore del flag corrispondente. Per registrare un enunciato, almeno un'espressione nell'attributo di sessione deve consentirlo, con il flag enable logging impostato su. `true` Anche il valore di *intent* e *slot può esserlo*`"*"`. Se il valore dello slot e/o dell'intento è`"*"`, significa che qualsiasi valore dello slot e/o dell'intento `"*"` corrisponderà ad esso. Analogamente`x-amz-lex:enable-audio-logging`, `x-amz-lex:enable-text-logging` verrà utilizzato un nuovo attributo di sessione chiamato per controllare i log di testo.

3. Seleziona Opzioni di aggiornamento e crea il bot per includere le impostazioni aggiornate.

> **Note**
>
> Il tuo ruolo IAM deve disporre dell'autorizzazione di accesso per consentirti di scrivere dati nel bucket Amazon S3 e utilizzare una chiave KMS per crittografare i dati. Lex aggiornerà il tuo ruolo IAM con le autorizzazioni Lex per accedere al gruppo di log CloudWatch Logs e al bucket Amazon S3 selezionato.

Linee guida per l'utilizzo dell'acquisizione selettiva dei log delle conversazioni:

È possibile abilitare l'acquisizione selettiva dei registri delle conversazioni per i registri di testo e/o audio solo dopo aver abilitato i registri di testo e/o audio nelle impostazioni del registro delle conversazioni. Abilitando l'acquisizione selettiva dei registri delle conversazioni per i registri di testo e/o audio, si disabilita la registrazione per tutti gli intenti e gli slot della conversazione. Per generare registri di testo e/o audio per intenti e slot particolari, è necessario impostare su «true» gli attributi di sessione di acquisizione selettiva del testo e/o dell'audio dei registri delle conversazioni per tali intenti e slot.

- Se l'acquisizione selettiva dei registri delle conversazioni è abilitata e non enable-audio-logging sono presenti attributi di sessione con il prefisso x-amz-lex:, la registrazione sarà disabilitata per impostazione predefinita per tutte le espressioni. Questo scenario è vero anche per quanto riguarda: enable-text-logging. x-amz-lex
- I registri degli enunciati verranno archiviati esclusivamente per i segmenti della conversazione di testo e/o audio se almeno un'espressione nell'attributo session lo consente.
- Le configurazioni per l'acquisizione selettiva del testo e/o dell'audio nei registri delle conversazioni, come definite negli attributi di sessione, saranno efficaci solo quando l'acquisizione selettiva del

registro delle conversazioni per testo e/o audio è abilitata nelle Impostazioni del registro delle conversazioni all'interno dell'alias del bot; in caso contrario, gli attributi di sessione verranno ignorati.

- Quando l'acquisizione selettiva del registro delle conversazioni è abilitata, tutti i valori degli slot in SessionState, Interpretazioni e Trascrizioni per i quali la registrazione non è abilitata utilizzando gli attributi di sessione verranno offuscati nel registro di testo generato.
- La decisione di produrre registri audio e/o di testo viene valutata abbinando lo slot generato dal bot agli attributi selettivi della sessione di acquisizione del registro delle conversazioni, ad eccezione del turno di elicitazione dell'intento in cui l'utente può fornire i valori degli slot insieme all'elicitazione dell'intento. In un turno di evocazione dell'intento, gli slot occupati nel turno corrente vengono confrontati con gli attributi selettivi della sessione di acquisizione del registro delle conversazioni.
- Gli slot considerati occupati derivano dallo stato della sessione alla fine del turno. Pertanto, qualsiasi modifica apportata da Dialog Codehook Lambda agli slot nello stato della sessione influenzerà il comportamento dell'acquisizione selettiva del registro delle conversazioni.
- In un turno di evocazione degli intenti, se l'utente fornisce più valori di slot, il registro di testo e/o audio verrà generato solo se gli attributi della sessione di testo/audio consentono la registrazione per tutti gli slot riempiti in questo turno.
- L'approccio operativo consigliato consiste nell'impostare l'attributo selettivo della sessione di acquisizione del registro delle conversazioni all'inizio della sessione e nell'astenersi dal modificarlo durante la sessione.
- Se uno slot contiene dati sensibili, è necessario abilitare sempre l'offuscamento degli slot.

Esempio di acquisizione selettiva del registro delle conversazioni

Di seguito è riportato un esempio di utilizzo aziendale per l'acquisizione selettiva dei registri delle conversazioni.

Caso d'uso:

Un'azienda fintech utilizza un bot Amazon Lex V2 per supportare il proprio sistema IVR, che consente agli utenti di effettuare il pagamento delle bollette. Per soddisfare i requisiti di conformità e controllo, devono conservare le registrazioni audio del consenso di autorizzazione fornito dall'utente. Tuttavia, l'abilitazione dei registri audio generali non è fattibile in quanto li renderebbe non conformi, poiché non è possibile offuscare slot sensibili come CVV e altre informazioni nei registri audio. CardNumber Possono invece abilitare l'acquisizione selettiva dei registri delle conversazioni per i registri audio e

impostare l'attributo di sessione in modo che produca registri audio solo per le espressioni che hanno ottenuto il consenso dell'autorizzazione.

BotAlias Impostazioni:

- Registri di testo abilitati: true
- Registri di testo Registrazione selettiva abilitata: false
- Registri audio abilitati: true
- Registri audio Registrazione selettiva abilitata: true

Attributi della sessione:

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

Conversazione di esempio:

- Utente (ingresso audio): «Voglio pagare la bolletta con la fattura numero 35XU68".
- Bot: «Qual è l'importo dovuto in dollari?»
- Utente (ingresso audio): «235».
- Bot: «Qual è il numero della tua carta di credito?»
- Utente (ingresso audio): «9239829722200348».
- Bot: «Stai pagando 235 dollari usando il numero della tua carta di credito che termina con 0348. Per favore, dì 'Autorizzo a pagare 235 dollari'».
- Utente (ingresso audio): «Autorizzo a pagare 235 dollari».
- Bot: «La tua fattura è stata pagata».

Output dei registri di conversazione:

In questa situazione, verranno prodotti registri di testo per tutti i turni. Tuttavia, i registri audio verranno registrati solo per quel particolare turno quando è stato attivato lo AuthorizationConsentslot all'interno dell'PayBillintento e non verrà prodotto alcun registro audio per nessun altro turno.

Monitoraggio delle metriche operative

Amazon CloudWatch e 2 AWS CloudTrail sono due AWS servizi che si integrano con Amazon Lex V2 per aiutarti a monitorare le interazioni degli utenti con il tuo bot. Utilizza questi servizi per

registrare le azioni, inviare dati quasi in tempo reale e configurare notifiche e azioni automatizzate quando i criteri vengono soddisfatti.

Argomenti

Misurazione delle metriche operative con Amazon CloudWatch

Puoi monitorare Amazon Lex V2 utilizzando CloudWatch, che raccoglie dati grezzi e li elabora in parametri leggibili quasi in tempo reale. Queste statistiche vengono conservate per un periodo di 15 mesi, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione web. È anche possibile impostare allarmi che controllano determinate soglie e inviare notifiche o intraprendere azioni quando queste soglie vengono raggiunte. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.

Il servizio Amazon Lex V2 riporta i seguenti parametri nel namespace. `AWS/Lex`

Parametro	Descrizione
`AssistedSlotResolutionModelAccessDeniedErrorCount`	Il numero di volte in cui ad Amazon Lex V2 è stato negato l'accesso ad Amazon Bedrock Dimensioni valide per le operazioni `RecognizeUtterance` `eStartConversation` : • BotId, BotAliasId, LocaleId, Operazione, InputMode, ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento, InputMode, ModelType, Modello Dimensioni valide per`RecognizeText` : • BotId, BotAliasId, LocaleId, Operazione ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento ModelType, Modello

Parametro	Descrizione
	Unità: numero
`AssistedSlotResolutionModelInvocationCount`	Il numero di volte in cui Amazon Bedrock è stato richiamato. Dimensioni valide per le operazioni `RecognizeUtterance` and`StartConversation` : • BotId, BotAliasId, LocaleId, Operazione, InputMode, ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento, InputMode, ModelType, Modello Dimensioni valide per`RecognizeText` : • BotId, BotAliasId, LocaleId, Operazione ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento ModelType, Modello Unità: numero

Parametro	Descrizione
`AssistedSlotResolutionModelSystemErrorCount`	Il numero di volte in cui si è verificato un 5xx durante una chiamata ad Amazon Bedrock. Dimensioni valide per le operazioni e`RecognizeUtterance` : `StartConversation` • BotId, BotAliasId, LocaleId, Operazione, InputMode, ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento, InputMode, ModelType, Modello Dimensioni valide per`RecognizeText` : • BotId, BotAliasId, LocaleId, Operazione ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento ModelType, Modello Unità: numero
`AssistedSlotResolutionModelThrottlingErrorCount`	Il numero di volte in cui Amazon Lex è stato limitato da Amazon Bedrock. Dimensioni valide per le operazioni e`RecognizeUtterance` : `StartConversation` • BotId, BotAliasId, LocaleId, Operazione, InputMode, ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento, InputMode, ModelType, Modello Dimensioni valide per`RecognizeText` : • BotId, BotAliasId, LocaleId, Operazione ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento ModelType, Modello Unità: numero

Parametro	Descrizione
`AssistedSlotResolutionResolvedSlotCount`	Il numero di volte in cui Amazon Bedrock ha restituito il valore di uno slot. Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` e: • BotId, BotAliasId, LocaleId, Operazione, InputMode, ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento, InputMode, ModelType, Modello Dimensioni valide per`RecognizeText` : • BotId, BotAliasId, LocaleId, Operazione ModelType, Modello • BotId, BotVersion, LocaleId, Funzionamento ModelType, Modello Unità: numero
`KendraIndexAccessError`	Il numero di volte in cui Amazon Lex V2 non ha potuto accedere al tuo indice Amazon Kendra. • Operazione,, BotId BotAliasId LocaleId Unità: numero
`KendraLatency`	Il tempo impiegato da Amazon Kendra per rispondere a una richiesta proveniente da. `AMAZON.KendraSearchIntent` Dimensioni valide: • Operazione,,, BotId BotVersion LocaleId • Operazione BotId, BotAliasId, LocaleId Unità: millisecondi

Parametro	Descrizione
`KendraSuccess`	Il numero di volte in cui Amazon Lex V2 non è riuscito ad accedere al tuo indice Amazon Kendra. Dimensioni valide: • Operazione,, BotId BotVersion LocaleId • Operazione BotId, BotAliasId, LocaleId Unità: numero
`KendraSystemErrors`	Il numero di volte in cui Amazon Lex V2 non è riuscito a eseguire query sull'indice Amazon Kendra. Dimensioni valide: • Operazione,,, BotId BotAliasId InputMode LocaleId Unità: numero
`KendraThrottledEvents`	Il numero di volte in cui Amazon Kendra ha limitato le richieste provenienti da. `AMAZON.KendraSearchIntent` Dimensioni valide: • Operazione,,. BotId BotAliasId InputMode LocaleId Unità: numero

Parametro	Descrizione
`RuntimeConcurrency`	Il numero di connessioni simultanee nel periodo di tempo specificato. `RuntimeConcurrency` è riportato come. `StatisticSet` Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` o: • Operazione BotId, BotVersion, InputMode, LocaleId • Operazione BotId, BotAliasId, InputMode, LocaleId Dimensioni valide per altre operazioni: • Operazione BotId, BotVersion, LocaleId • Operazione BotId, BotAliasId, LocaleId Unità: numero
`RuntimeInvalidLambdaResponses`	Il numero di AWS Lambda risposte non valide nel periodo specificato. Dimensioni valide: • Operazione, BotId, BotAliasId, InputMode LocaleId Unità: numero
`RuntimeLambdaErrors`	Il numero di errori di runtime Lambda nel periodo di tempo specificato. Dimensioni valide: • Operazione, BotId, BotAliasId, InputMode LocaleId Unità: numero

Parametro	Descrizione
`RuntimePollyErrors`	Il numero di risposte Amazon Polly non valide nel periodo di tempo specificato. Dimensioni valide: • Operazione,, BotId, BotAliasId InputMode LocaleId Unità: numero
`RuntimeRequestCount`	Il numero di richieste di runtime nel periodo di tempo specificato. Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` and: • Operazione BotId, BotVersion, InputMode, LocaleId • Operazione BotId, BotAliasId, InputMode, LocaleId Dimensioni valide per altre operazioni: • Operazione BotId, BotVersion, LocaleId • Operazione BotId, BotAliasId, LocaleId Unità: numero
`RuntimeRequestLength`	Durata totale di una conversazione con un bot Amazon Lex V2. Applicabile solo all'StartConversationoperazione. Dimensioni valide: • BotAliasId, BotId, LocaleId, Operazione • BotId, BotAliasId LocaleId, Operazione Unità: millisecondi

Parametro	Descrizione
`RuntimeSuccessfulRequestLatency` **Important** Questa metrica è `RuntimeSuccessfulRequestLatency` e non `RuntimeSuccessfulRequestLatency` lo è.	La latenza delle richieste riuscite tra il momento in cui è stata effettuata la richiesta e la risposta è stata restituita. Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` and: • Operazione BotId, BotVersion, InputMode, LocaleId • Operazione BotId, BotAliasId, InputMode, LocaleId Dimensioni valide per altre operazioni: • Operazione BotId, BotVersion, LocaleId • Operazione BotId, BotAliasId, LocaleId 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. Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` e: • Operazione BotId, BotVersion, InputMode, LocaleId • Operazione BotId, BotAliasId, InputMode, LocaleId Dimensioni valide per altre operazioni: • Operazione BotId, BotVersion, LocaleId • Operazione BotId, BotAliasId, LocaleId Unità: numero

Parametro	Descrizione
`RuntimeThrottledEvents`	Il numero di eventi limitati. Amazon Lex V2 limita un evento quando riceve più richieste rispetto 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 i limiti dei servizi AWS. Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` e: • Operazione BotId, BotVersion, InputMode, LocaleId • Operazione BotId, BotAliasId, InputMode, LocaleId Dimensioni valide per altre operazioni: • Operazione BotId, BotVersion, LocaleId • Operazione BotId, BotAliasId, LocaleId 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. Dimensioni valide per le `StartConversation` operazioni `RecognizeUtterance` e: • Operazione BotId, BotVersion, InputMode, LocaleId • Operazione BotId, BotAliasId, InputMode, LocaleId Dimensioni valide per altre operazioni: • Operazione BotId, BotVersion, LocaleId • Operazione BotId, BotAliasId, LocaleId Unità: numero

Le seguenti dimensioni sono supportate per i parametri di Amazon Lex V2.

Dimensione	Descrizione
`Operation`	Il nome dell'operazione Amazon Lex V2 —`RecognizeText` ,,`RecognizeUtterance` ,`StartConversation` , `GetSession` `PutSession` , `DeleteSession` — che ha generato la voce.
`BotId`	L'identificatore univoco alfanumerico per il bot.
`BotAliasId`	L'identificatore univoco alfanumerico per l'alias del bot.
`BotVersion`	La versione numerica del bot.
`InputMode`	Il tipo di input per il bot: voce, testo o DTMF.
`LocaleId`	L'identificatore delle impostazioni locali del bot, ad esempio en-US o fr-CA.
`Model`	Indica l'ID del modello di linguaggio di grandi dimensioni di Amazon Bedrock.
`ModelType`	Indica il tipo di modello linguistico di grandi dimensioni richiamato da Amazon Bedrock.

Visualizzazione degli eventi con AWS CloudTrail

Amazon Lex V2 è integrato con AWS CloudTrail un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in Amazon Lex V2. CloudTrail acquisisce le chiamate API per Amazon Lex V2 come eventi. Le chiamate acquisite includono chiamate dalla console Amazon Lex V2 e chiamate in codice alle operazioni dell'API Amazon Lex V2. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon Lex V2. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console in Cronologia eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata ad Amazon Lex V2, l'indirizzo IP da cui è stata effettuata, chi ha effettuato la richiesta, quando è stata effettuata e ulteriori dettagli.

Per ulteriori informazioni CloudTrail, consulta la Guida per l'AWS CloudTrail utente.

Informazioni su Amazon Lex V2 in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Amazon Lex V2, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo AWS account. 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 V2, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- 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 V2 supporta la registrazione per tutte le azioni elencate nell'API Model Building V2.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management IAM.
- 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 AWS servizio.

Per ulteriori informazioni, vedete l'elemento CloudTrail userIdentity.

Informazioni sulle voci dei file di log di Amazon Lex V2

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon 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 mostra una voce di CloudTrail registro che illustra l'azione CreateBotAlias.

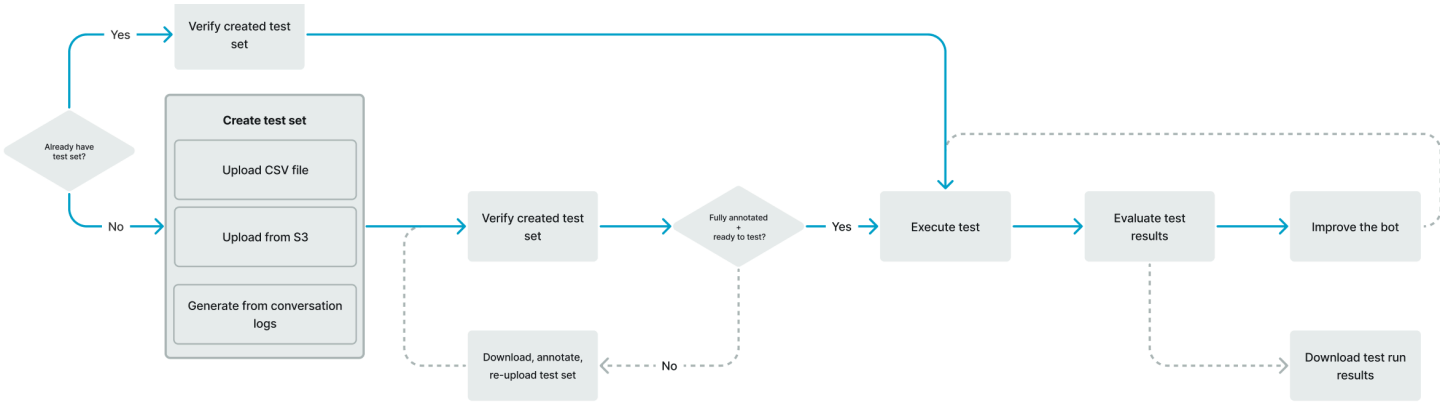
```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ID of caller:temporary credentials",
        "arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ID of caller",
                "arn": "arn:aws:iam::111122223333:role/role name",
                "accountId": "111122223333",
                "userName": "role name"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "creation date"
            }
        }
    },
    "eventTime": "event timestamp",
    "eventSource": "lex.amazonaws.com",
    "eventName": "CreateBotAlias",
    "awsRegion": "Region",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "user agent",
    "requestParameters": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
```

```
                "enabled": true
            }
        },
        "botId": "bot ID",
        "botAliasName": "bot aliase name",
        "botVersion": "1"
    },
    "responseElements": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
            }
        },
        "botAliasId": "bot alias ID",
        "botAliasName": "bot alias name",
        "botId": "bot ID",
        "botVersion": "1",
        "creationDateTime": creation timestamp
    },
    "requestID": "unique request ID",
    "eventID": "unique event ID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Valutazione delle prestazioni dei bot con Test Workbench

Per migliorare le prestazioni dei bot, puoi valutare le prestazioni dei tuoi bot su larga scala. I risultati della valutazione del test vengono visualizzati in semplici tabelle e grafici.

È possibile utilizzare Test Workbench per creare set di test di riferimento che utilizzano dati di trascrizione esistenti. È possibile testare i bot per valutare le prestazioni prima dell'implementazione e visualizzare le suddivisioni dei risultati dei test su larga scala.



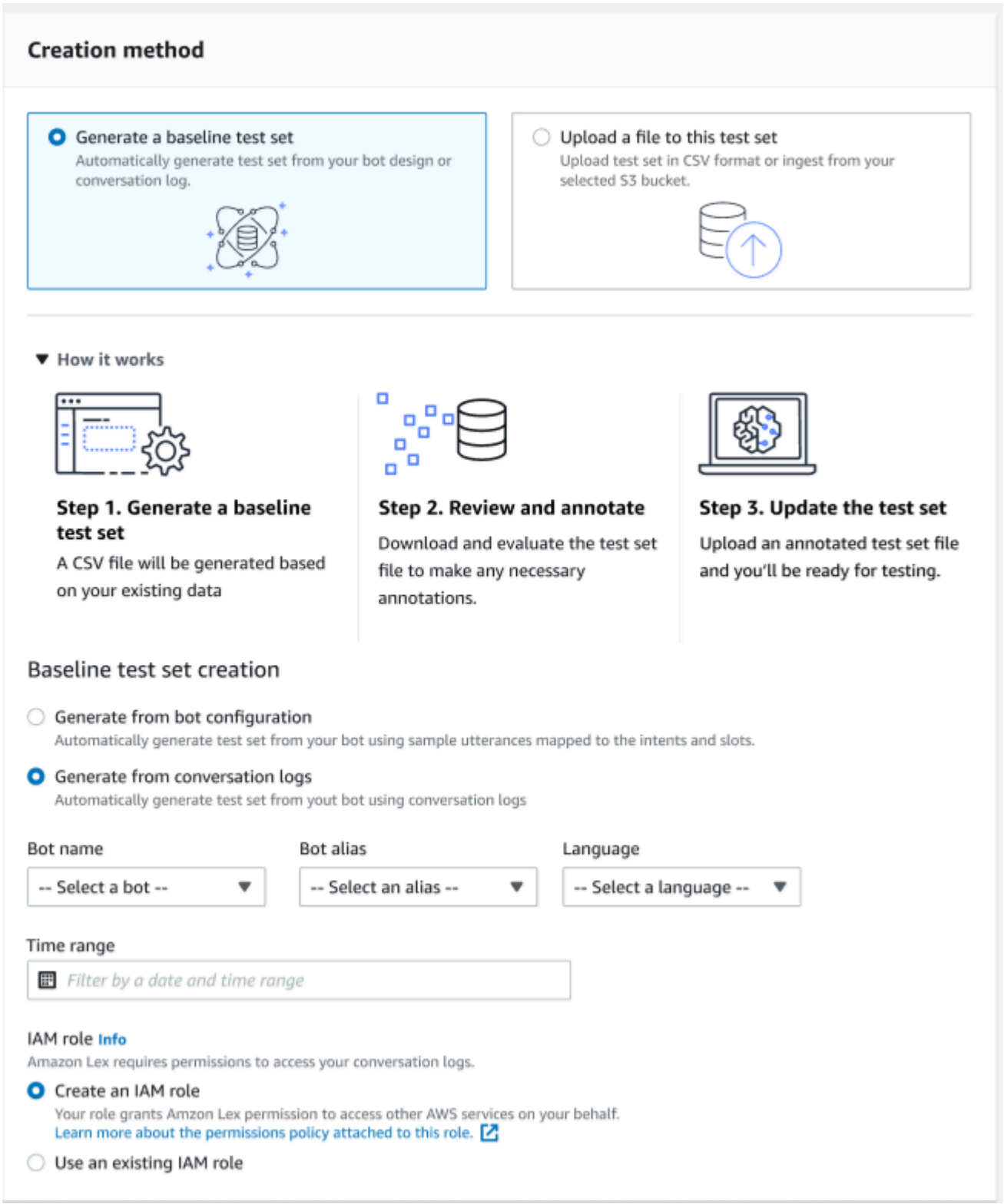
Gli utenti possono utilizzare Test Workbench per stabilire le prestazioni di base per i bot. Ciò riguarda le prestazioni degli intenti e degli slot per le espressioni che si presentano sotto forma di input o conversazioni singoli. Una volta che un set di test è stato caricato correttamente, è possibile eseguirlo sui bot di preproduzione o produzione esistenti. Il Test Workbench consente di identificare le opportunità per migliorare il riempimento degli slot e la classificazione degli intenti.

Argomenti

- Genera un set di test
- Gestisci i set di test
- Esegui un test
- Copertura del set di test
- Visualizza i risultati del test
- Dettagli dei risultati del test

Genera un set di test

Puoi creare un set di test per valutare le prestazioni del tuo bot. Genera un set di test caricando un set di test in formato file CSV o generando un set di test dai registri delle conversazioni. Il set di test può contenere input audio o di testo.



Se un set di test crea errori di convalida, rimuovi il set di test e sostituiscilo con un altro elenco di dati del set di test oppure modifica i dati nel file CSV utilizzando un programma di modifica dei fogli di calcolo.

Per creare un set di test:

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli Test workbench dal pannello laterale sinistro.
3. Seleziona Set di test dalle opzioni in Test workbench.
4. Seleziona il pulsante Crea set di test sulla console.
5. Nei Dettagli, inserisci il nome del set di test e una descrizione opzionale.
6. Seleziona Genera un set di test di base.
7. Seleziona Genera dai registri delle conversazioni.
8. Seleziona il nome del bot, l'alias del bot e la lingua dai menu a discesa.
9. Se stai generando un test di base da un registro delle conversazioni, scegli Time range e IAM role, se necessario. Puoi creare un ruolo con le autorizzazioni di base di Amazon Lex V2 o utilizzare un ruolo esistente.
10. Scegli una modalità di audio o testo per il set di test che stai creando. NOTA: Test Workbench può importare file di testo fino a 50.000 e fino a 5 ore di audio.
11. Seleziona una posizione Amazon S3 per archiviare i risultati dei test e aggiungi una chiave KMS opzionale per crittografare le trascrizioni di output.
12. Seleziona Crea.

Per caricare un set di test esistente in un formato di file CSV o per aggiornare il set di test:

1. Scegli Test workbench dal pannello laterale sinistro.
2. Seleziona Set di test dalle opzioni in Test workbench.
3. Seleziona Carica un file su questo set di test sulla console.
4. Scegli Carica dal bucket Amazon Amazon S3 o Carica dal tuo computer. NOTA: puoi caricare un file CSV creato da un modello. Fai clic su modello CSV per scaricare un file zip che contiene i modelli.
5. Scegli Crea un ruolo con autorizzazioni Amazon Lex di base o Usa un ruolo esistente per Role ARN.
6. Scegli una modalità di audio o testo per il set di test che stai creando. NOTA: Test Workbench può importare file di testo fino a 50.000 e fino a 5 ore di audio.

7. Seleziona una posizione Amazon S3 per archiviare i risultati dei test e aggiungi una chiave KMS opzionale per crittografare le trascrizioni di output.
8. Seleziona Crea.

Se l'operazione ha esito positivo, il messaggio di conferma indicherà che il set di test è pronto per il test e lo stato verrà visualizzato Pronto per il test.

Suggerimenti per creare un set di test di successo

- Puoi creare un ruolo IAM per Test Workbench nella console oppure configurare il tuo ruolo step-by-step IAM. Per ulteriori informazioni, consulta Creare un ruolo IAM per il Test Workbench.
- Prima di eseguire un test, convalida il set di test e la definizione del bot per eventuali incongruenze utilizzando il pulsante Validate discrepancy. Se le convenzioni di denominazione degli intenti e degli slot utilizzate nel set di test sono coerenti con quelle del bot, procedi con l'esecuzione del test. Se vengono identificate anomalie, rivedete il set di test, aggiornate il set di test e scegliete Convalida discrepanza. Ripeti nuovamente questa sequenza fino a quando non vengono rilevate incongruenze, quindi esegui il test.
- Il Test Workbench può eseguire test con diversi formati di valori degli slot nella colonna Expected Output Slot. Per ogni slot integrato, è possibile scegliere il valore fornito nell'input dell'utente (ad esempio, Date = tomorrow) o fornire il valore risolto assoluto (ad esempio, Date = 2023-03-21). Per ulteriori informazioni sugli slot incorporati e sui relativi valori assoluti, vedere Slot integrati.
- Per garantire coerenza e leggibilità nelle colonne Expected Output Slot, seguite la convenzione "SlotName = SlotValue" (ad esempio, AppointmentType = pulizia) con uno spazio prima e dopo il segno di uguale.
- Se il bot include slot compositi, in Expected Output Slot definisci i sottoslot relativi al nome dello slot, separati da un punto (ad esempio, «Car.Color»). Nessun'altra sintassi e punteggiatura funzionerà.
- Se il bot include slot multivalore, in Expected Output Slot fornisci più valori di slot, separati da una virgola (» FlowerType = rose, gigli»). Nessun'altra sintassi e punteggiatura funzionerà.
- Assicurati che il set di test sia creato da registri di conversazione validi.
- Il valore slot:slot sarà nella stessa colonna dopo le colonne degli intenti in formato CSV.
- L'input DTMF proveniente da un turno utente viene interpretato come una trascrizione prevista e non elenca una posizione Amazon S3.

Creazione di un test case all'interno di un set di test

I risultati del Test Workbench dipendono dalla definizione del bot e dal set di test corrispondente. È possibile generare un set di test con le informazioni della definizione del bot per individuare le aree che necessitano di miglioramenti. Crea un set di dati di test con esempi che sospetti (o conosci) possano essere difficili da interpretare correttamente per il bot, considerando l'attuale design del bot e la tua conoscenza delle conversazioni con i clienti.

Rivedi regolarmente le tue intenzioni sulla base delle conoscenze acquisite dal tuo bot di produzione. Continuate ad aggiungere e modificare le espressioni di esempio e i valori degli slot del bot. Valuta la possibilità di migliorare la risoluzione degli slot utilizzando le opzioni disponibili, come i suggerimenti di runtime. La progettazione e lo sviluppo del bot sono un processo iterativo a ciclo continuo.

Ecco alcuni altri suggerimenti per ottimizzare il set di test:

- Seleziona i casi d'uso più comuni con intenti e slot utilizzati di frequente nel set di test.
- Esplora diversi modi in cui un cliente può fare riferimento alle tue intenzioni e ai tuoi slot. Ciò può includere gli input dell'utente sotto forma di dichiarazioni, domande e comandi di lunghezza variabile da minima a estesa.
- Includi input utente con un numero variabile di slot.
- Includi sinonimi o abbreviazioni di uso comune dei valori degli slot personalizzati supportati dal bot (ad esempio, «root canal», «canal» o «RC»).
- Includi variazioni dei valori degli slot incorporati (ad esempio, «domani», «asap» o «il giorno successivo»).
- Esamina la robustezza del bot per la modalità vocale raccogliendo gli input dell'utente che possono essere interpretati erroneamente (ad esempio, «ink», «ankle» o «anchor»).

Creazione di un set di test da un file CSV

Puoi creare un set di test dal modello di file CSV fornito nella console Amazon Lex V2 inserendo i valori direttamente utilizzando un editor di fogli di calcolo CSV. Il set di test è un file con valori separati da virgole (CSV) composto da enunciati da un singolo utente e conversazioni a più turni registrate nelle seguenti colonne:

- Riga #: questa colonna è un contatore incrementale che tiene traccia del totale delle righe riempite da testare.

- Conversazione #: questa colonna tiene traccia del numero di turni in una conversazione. Per input singoli, questa colonna può essere lasciata vuota, riempita con «-» o «N/A». Per quanto riguarda le conversazioni, a ogni turno all'interno di una conversazione verrà assegnato lo stesso numero di conversazione.
- Fonte: questa colonna è impostata su «Utente» o «Agente». Per i singoli ingressi, sarà sempre impostata su «Utente».
- Input: questa colonna include l'enunciato dell'utente o le istruzioni del bot.
- Intento di output previsto: questa colonna riporta l'intento soddisfatto nell'input.
- Intent Expected Output Slot 1: questa colonna cattura il primo slot generato nell'input dell'utente. Il set di test dovrebbe includere una colonna chiamata Expected Output Slot X per ogni slot nell'input dell'utente.

Esempio di set di test con input singoli:

Linea #	Conversazione #	Origine	Input	Intento di output previsto	Slot di uscita previsto 1	Slot di uscita previsto 2
1		Utente	prenota un appuntamento per le pulizie domani	MakeAppointment	AppointmentType = pulizia	Data = domani
2	N/D	Utente	prenota un appuntamento per le pulizie il 15 aprile	MakeAppointment	AppointmentType = pulizia	Data = 15/04/23
3	N/D	Utente	prenota un appuntamento per il primo dicembre	MakeAppointment	Data = primo dicembre	

Linea #	Conversazione #	Origine	Input	Intento di output previsto	Slot di uscita previsto 1	Slot di uscita previsto 2
4	N/D	Utente	prenota un appuntamento per le pulizie	MakeAppointment	AppointmentType = pulizia	
1		Utente	Puoi aiutarmi a prenotare un appuntamento?	MakeAppointment		

Esempio di un set di test con conversazioni

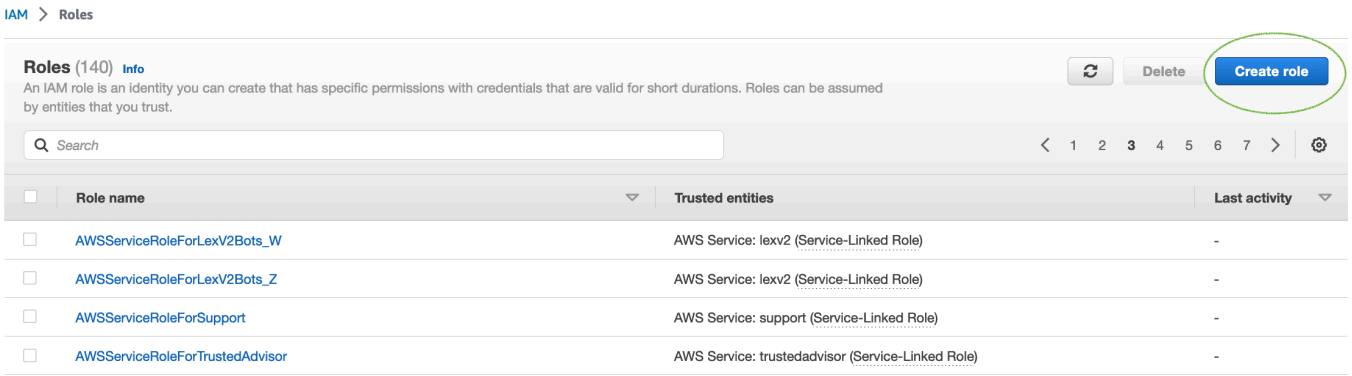
Linea #	Conversazione #	Origine	Input	Intento di output previsto	Slot di uscita previsto 1	Slot di uscita previsto 2	Slot di uscita previsto 3
1	1	Utente	prenotare un appuntamento	MakeAppointment			
2	1	Agente	Che tipo di appuntamento vuoi fissare?	MakeAppointment			
3	1	Utente	pulizia	MakeAppointment	AppointmentType = pulizia		

Linea #	Conversazione #	Origine	Input	Intento di output previsto	Slot di uscita previsto 1	Slot di uscita previsto 2	Slot di uscita previsto 3
4	1	Agente	Quando devo fissare un appuntamento?	MakeAppointment			
5	1	Utente	tomorrow	MakeAppointment		Data = domani	
6	2	Utente	prenota oggi stesso un appuntamento per il canale radicolare	MakeAppointment	AppointmentType = canale radicolare	Data = oggi	
7	2	Agente	A che ora devo fissare il vostro appuntamento?	MakeAppointment			
8	2	Utente	alle undici del mattino	MakeAppointment			Ora = undici del mattino

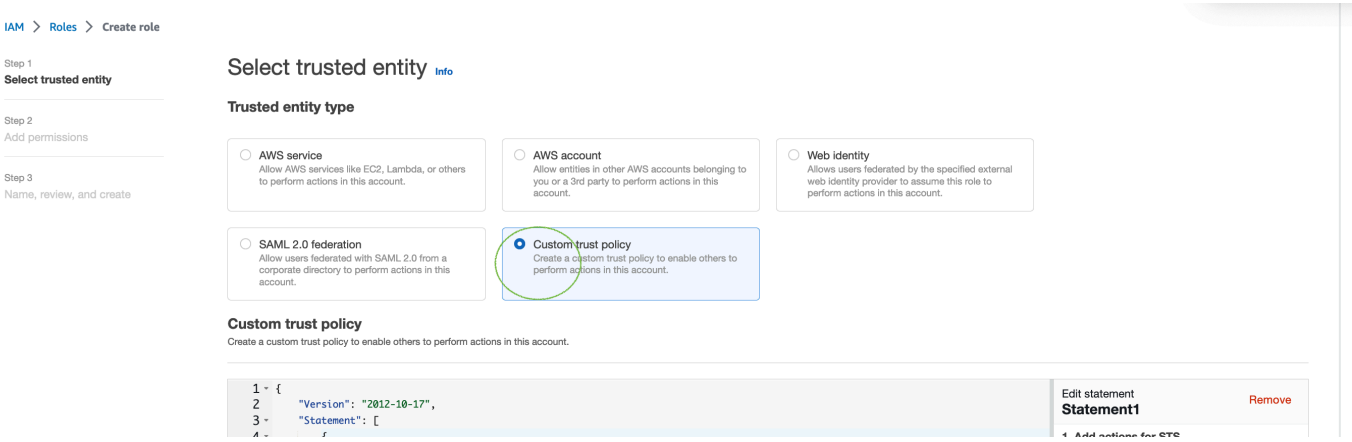
Crea un ruolo IAM per il Test Workbench

Per creare un ruolo IAM per il Test Workbench

1. Segui i passaggi indicati in Creare un utente IAM per creare un utente IAM che può essere utilizzato per accedere alla console test-workbench.
2. Seleziona il pulsante Crea ruolo.



3. Seleziona l'opzione Custom Trust Policy.



4. Inserisci la politica di fiducia di seguito e fai clic su Avanti.

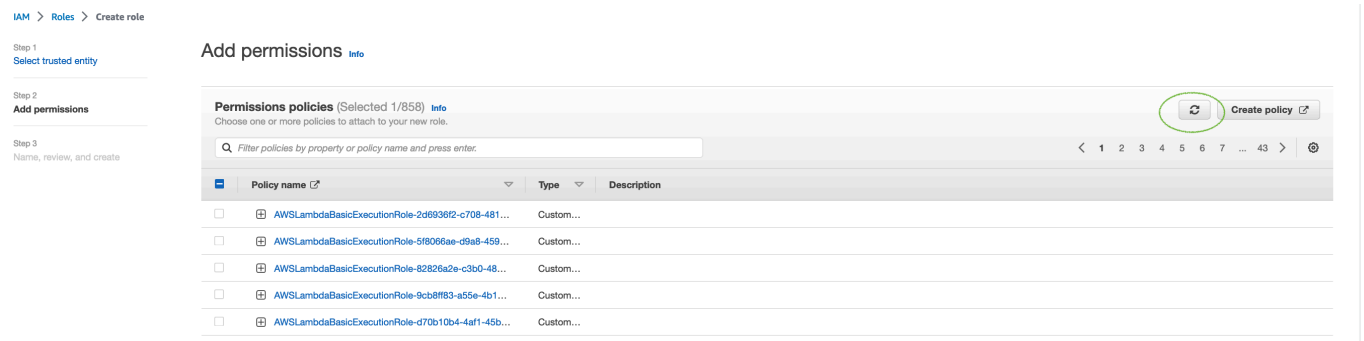
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
```

5. Seleziona il pulsante Crea politica.
6. Nel browser si aprirà una nuova scheda in cui è possibile inserire la politica seguente e fare clic sul pulsante Avanti: Tag.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Inserisci il nome di una politica, ad esempio LexTest WorkbenchPolicy «», quindi fai clic su Crea politica.
8. Torna alla scheda precedente del browser e Aggiorna l'elenco delle politiche facendo clic sul pulsante Aggiorna come mostrato di seguito.



9. Cerca nell'elenco delle politiche inserendo il nome della politica che hai utilizzato nel sesto passaggio e scegli la politica.
10. Seleziona il pulsante Avanti.
11. Inserisci il nome del ruolo, quindi fai clic sul pulsante Crea ruolo.
12. Scegli il tuo nuovo ruolo IAM quando richiesto nella console Amazon Lex V2 per Test Workbench.

Gestisci i set di test

È possibile scaricare, aggiornare ed eliminare i set di test dalla finestra del set di test. Oppure puoi utilizzare l'elenco dei set di test disponibili per modificare o annotare manualmente il file del set di test. Quindi, caricalo di nuovo per riprovare la convalida, a causa di errori o altri problemi di input.

Per scaricare il file del set di test dal record del set di test:

1. Seleziona il nome del set di test dall'elenco dei set di test.
2. Nella finestra di registrazione del set di test, seleziona il pulsante Download sul lato destro dello schermo nella sezione Test Inputs.
3. se nella parte superiore della finestra sono presenti dettagli sugli errori di convalida relativi al set di test, seleziona il pulsante Download. Il file verrà salvato nella cartella Download. È possibile correggere gli errori di convalida nel set di test dai messaggi di errore nel file CSV del set di test. Trova l'errore identificato nella fase di convalida, correggi la riga o rimuovila e carica il file per ripetere la fase di convalida.
4. se scarichi correttamente il set di test, apparirà un banner verde.

Per scaricare un set di test dall'elenco dei set di test:

1. Dall'elenco dei set di test, selezionate il pulsante di opzione accanto alla voce del set di test che desiderate scaricare.
2. Dal menu Azione in alto a destra, scegli Scarica.
3. Un banner verde indicherà se hai scaricato correttamente il set di test. Il file verrà salvato nella cartella Download.

Visualizza gli errori di convalida del test

È possibile correggere i set di test che segnalano errori di convalida. Questi errori di convalida vengono generati quando un set di test non è pronto per essere testato. Il Test Workbench può mostrarvi quali colonne obbligatorie nel file CSV di input del set di test non avevano un valore nel formato previsto.

Per visualizzare gli errori di convalida del test:

1. Dall'elenco dei set di test, selezionate il nome del set di test che riporta l'errore di stato di convalida che desiderate visualizzare. I nomi dei set di test sono collegamenti attivi che consentono di accedere ai dettagli relativi al set di test.
2. Il record del set di test mostra i dettagli dell'errore di convalida nella parte superiore dello schermo. Scegli Visualizza dettagli per visualizzare il rapporto sugli errori di convalida.
3. Nella finestra di segnalazione degli errori, esamina il numero di riga e il tipo di errore per vedere dove si verifica l'errore. Per un lungo elenco di errori, puoi scegliere di scaricare il rapporto sugli errori.
4. Confronta gli errori elencati nel file CSV di input del set di test con il file di test originale per correggere eventuali problemi e carica nuovamente il set di test.

La tabella seguente elenca i messaggi di errore di convalida CSV di input con gli scenari.

Scenario	Messaggio di errore	Note
La dimensione del file del set di test supera	La dimensione del file del set di test è superiore a 200 MB. Fornisci un file più piccolo e riprova la richiesta.	

Scenario	Messaggio di errore	Note
Il set di test supera il numero massimo di record	Il file di input conteneva record superiori al numero massimo supportato di 200.000.	
Carica un set di test vuoto	Il set di test importato è vuoto. Fornisci un set di test non vuoto e riprova la richiesta.	
Nome dell'intestazione della colonna vuota	Column Headers Row: è stato trovato un nome di colonna vuoto nella colonna numero 5.	
Nome dell'intestazione di colonna non riconosciuto	Column Headers Row: impossibile riconoscere il nome di colonna 'dummy' nella colonna numero 2.	
Nome dell'intestazione della colonna duplicato	Column Headers Row: sono state trovate più colonne 'S3 audio link' e 'S3 audio link' che sono uguali o equivalen ti. Rimuovi o rinomina una di quelle colonne.	
Il nome della colonna multivalo re ha superato il limite	Column Headers Row: il numero di colonne per «Expected Output Slot» ha superato il numero massimo supportato: 6. Rimuovi alcune colonne per «Expected Output Slot» e riprova.	Il numero massimo di colonne supportate per la colonna multivalore è 6.

Scenario	Messaggio di errore	Note
L'intestazione della colonna relativa al testo o all'audio non è presente	Impossibile trovare colonne per conversazioni di testo o audio. Per le conversazioni di testo, usa le colonne {'Text input'}. Per le conversazioni audio, usa le colonne {'S3 audio link', 'Expected transcription'}.	Colonne audio obbligatorie: {'S3 audio link', 'Trascrizione prevista'} Colonne obbligatorie del testo: {'Text input'}
Esistono intestazioni di colonna relative sia al testo che all'audio	Sono state trovate colonne per conversazioni di testo e audio. Puoi utilizzare le colonne {'Text input'} per le conversazioni di testo o le colonne {'S3 audio link', 'Expected transcription'} per le conversazioni audio.	Colonne audio obbligatorie: {'S3 audio link', 'Trascrizione prevista'} Colonne obbligatorie del testo: {'Text input'}
Manca la colonna obbligatoria	Impossibile trovare le colonne obbligatorie ["Expected Output Intent"].	Colonne obbligatorie: {"Line #», «Source», «Expected Output Intent"}
Trovato un dato in una colonna senza intestazione	Sono stati trovati dati nella colonna numero 8 per la riga numero 6, ma la colonna corrispondente non aveva un'intestazione di colonna.	
Dati non trovati per le colonne obbligatorie	Row=12: nessun valore trovato per le colonne obbligatorie: {"Source», «Expected Output Intent"}	

Scenario	Messaggio di errore	Note
È stata trovata una conversazione duplicata	il numero di conversazione '19' è stato visualizzato nella conversazione precedente e nella riga numero 39.» Assicurati che non sia stato fornito lo stesso numero di conversazione per due conversazioni. Puoi farlo assicurandoti che tutte le righe relative a un numero di conversazione siano raggruppate.	
È stata fornita una conversazione non valida	È stato trovato il valore non valido 'test_conversation' nella colonna 'Conversation #'. Il valore di questa colonna deve essere numerico o N/A (ossia non applicabile) per una riga utente.	
Valore non numerico fornito per il numero di riga	È stato trovato il valore non numerico 'test_line' nella colonna 'Riga #'. Il suo valore deve essere numerico.	
La conversazione non è stata trovata nella riga dell'agente	Nessun valore trovato per la colonna 'Conversation #'. Deve essere fornito per una riga relativa all'agente.	

Scenario	Messaggio di errore	Note
La conversazione non numerica si trova nella riga dell'agente	È stato trovato il valore non numerico 'test_conversation ' nella colonna 'Conversation #'. Il suo valore deve essere numerico per la riga di un agente.	
Posizione S3 non valida	È stato fornito il valore «bucket/folder» non valido. <keyName>Il formato valido è S3://<bucketName>/.	
Nome del bucket S3 non valido	È stato fornito un nome di bucket s3 «test_bucket» non valido. Controlla il nome del bucket.	
La posizione audio di S3 è la cartella	La posizione audio fornita 'S3: //bucket/folder' non è valida. Punta a una cartella S3.	
Nome dell'intento non valido	Nell'intento 'intent @name 'erano presenti caratteri non validi. Controlla il nome dell'intento.	Controllo Regex: ^ ([0-9a-zA-Z] [_-]?) +$
Nome slot non valido	Nello slot 'Slot @Name 'erano presenti caratteri non validi. Controlla il nome dello slot.	Regex: ^ ([0-9a-zA-Z] [_-]?) + $Non deve iniziare o terminare con dot (.)

Scenario	Messaggio di errore	Note
Valore dello slot fornito per lo slot principale	I valori dello slot sono stati forniti per lo slot secondario «Address.City» e per lo slot principale «Address». I valori devono essere forniti solo per il sottoslot.	Lo slot principale in CST non dovrebbe avere un valore di slot
Carattere non valido nel nome del contesto	Nel nome di contesto 'context @1 'erano presenti caratteri non validi. Controlla il nome del contesto.	Regex: ^ ([a-zA-Z] _?) +$
Stile ortografico degli slot non valido	È stato fornito il valore 'test' non valido. Assicurati che siano tutte maiuscole. I valori validi sono ["Default», "SpellByLetter», "SpellByWord"].	Valori supportati ["Default», "SpellByLetter», "SpellByWord»
Il partecipante o la fonte deve essere agente o utente	È stato fornito il valore «bot» non valido. I valori validi sono ["Agent», «User"].	Enumerazioni supportate: «Agent», «User»
Il numero di riga non deve essere decimale	È stato fornito il valore «10.1» non valido. Deve essere un numero valido senza frazioni.	
Il numero di conversazione non deve essere decimale	È stato fornito il valore «10.1» non valido. Deve essere un numero valido senza frazioni.	
Il numero di riga deve essere compreso nell'intervallo	È stato fornito il valore non valido '92233720368547758 071'. Deve essere maggiore o uguale a 1 e minore o uguale a 9223372036854775807.	

Scenario	Messaggio di errore	Note
La colonna barge-in accetta solo valori booleani	È stato fornito il valore «test» non valido. Deve essere un valore booleano valido come «true» o «false». In alternati va è possibile utilizzare «sì» e «no».	Valori possibili: "True», «true», «T», «Yes», «Y», «1", «1.0", «False», «false», «F», «No», «no», «N», «0", «0.0"
Lo slot previsto, l'attributo di sessione e l'attributo di richiesta devono essere separati per uguale a (=)	Il valore 'slotName:slotValue' non ha '='. <key><value>Tale valore deve essere fornito come coppia chiave-valore nel formato '='.	Ad esempio: slotName = slotType
Lo slot previsto, l'attributo di sessione, l'attributo di richiesta devono avere una coppia di valori chiave	'=SlotValue' non ha una chiave prima di '='. <key><value>Tale valore deve essere fornito come coppia chiave-valore nel formato '='.	Ad esempio: slotName = slotType
Citazione non valida alla fine	Trovata una citazione errata in 'Lenny's Burger'». Inizia con il carattere di virgoletta `"`ma non finisce con lo stesso carattere di virgoletta.	Ad esempio: `«Lenny's Burger», KFC`
Citazione non valida al centro	Trovata una citazione errata in `"Lenny's» Burger, KFC`. Contiene il carattere di virgoletta `"`all'interno del suo contenuto. I valori contenent i virgolette singole devono essere racchiusi tra virgolette doppie e viceversa.	Corretto. Ad esempio: `«Lenny's Burger», KFC`

Scenario	Messaggio di errore	Note
Citazioni obbligatorie	`key = Lenny's Burger` contiene virgolette singole o doppie ma non è racchiusa tra virgolette. I valori contenent i virgolette singole devono essere racchiusi tra virgolette doppie e viceversa.	
Chiave duplicata ripetuta nella colonna	La chiave `key1` è stata ripetuta in due colonne: `Attributo di sessione 3` e `Attributo di sessione 1`.	
Formato non valido nel suggerimento di runtime	Chiave `.Car non valida. BookFlight «`fornito per Runtime Hints. <intentNa me>Per Runtime Hints, la chiave deve essere in formato. <slotName>.	Se '.' deve essere presente al centro della chiave, il nome dell'intento e il nome dello slot non possono essere estratti da tale chiave. esempi di tale formattazione errata: "«BookFlight,». BookFligh t.Auto», ".Auto». BookFlight
Nome di intento non valido nella chiave di suggerimento in fase di esecuzione	Trovato l'intento `intent @name `non valido per i Runtime Hints. Controlla il nome dell'intento.	Controllo Regex: ^ ([0-9a-zA-Z] [_-]?) +$
Nome slot non valido nella chiave di suggerimento in fase di esecuzione	È stato trovato un nome di slot non valido in `Slot @Name `per i Runtime Hints. Controlla il nome dello slot.	Regex: ^ ([0-9a-zA-Z] [_-]?) + $Non deve iniziare o terminare con dot (.)

Eliminare un set di test

È possibile eliminare facilmente un set di test dall'elenco dei set di test.

Per eliminare un set di test:

1. Vai all'elenco dei set di test dal menu a sinistra per vedere l'elenco dei set di test.
2. Dall'elenco dei set di test, seleziona il set di test che desideri eliminare.
3. Vai al menu a discesa Azioni in alto a destra e scegli Elimina.
4. Un messaggio conferma che il set di test è stato eliminato.

Modifica i dettagli del set di test

È possibile modificare il nome e i dettagli di un set di test nell'elenco dei set di test. Il nome o i dettagli possono essere aggiunti o aggiornati in un secondo momento. Tuttavia, dovrai aggiornare il set di test prima di eseguire il test con il bot o i dati di trascrizione.

Per modificare i dettagli del set di test:

1. Vai all'elenco dei set di test dal menu a sinistra per vedere l'elenco dei set di test.
2. Dall'elenco dei set di test, seleziona la casella di controllo relativa al set di test che desideri modificare.
3. Vai al menu a discesa Azioni in alto a destra e scegli Modifica dettagli.
4. Un messaggio conferma che il set di test è stato modificato correttamente.

Aggiorna il set di test

È possibile aggiornare, correggere, modificare o eliminare elementi dal set di test per ottimizzare i risultati di base o per correggere altri errori che potrebbero essersi verificati nel set di test

È possibile scaricare un set di test e correggere gli errori di convalida prima di caricare il set di test corretto. Vedi Visualizza gli errori di convalida del test.

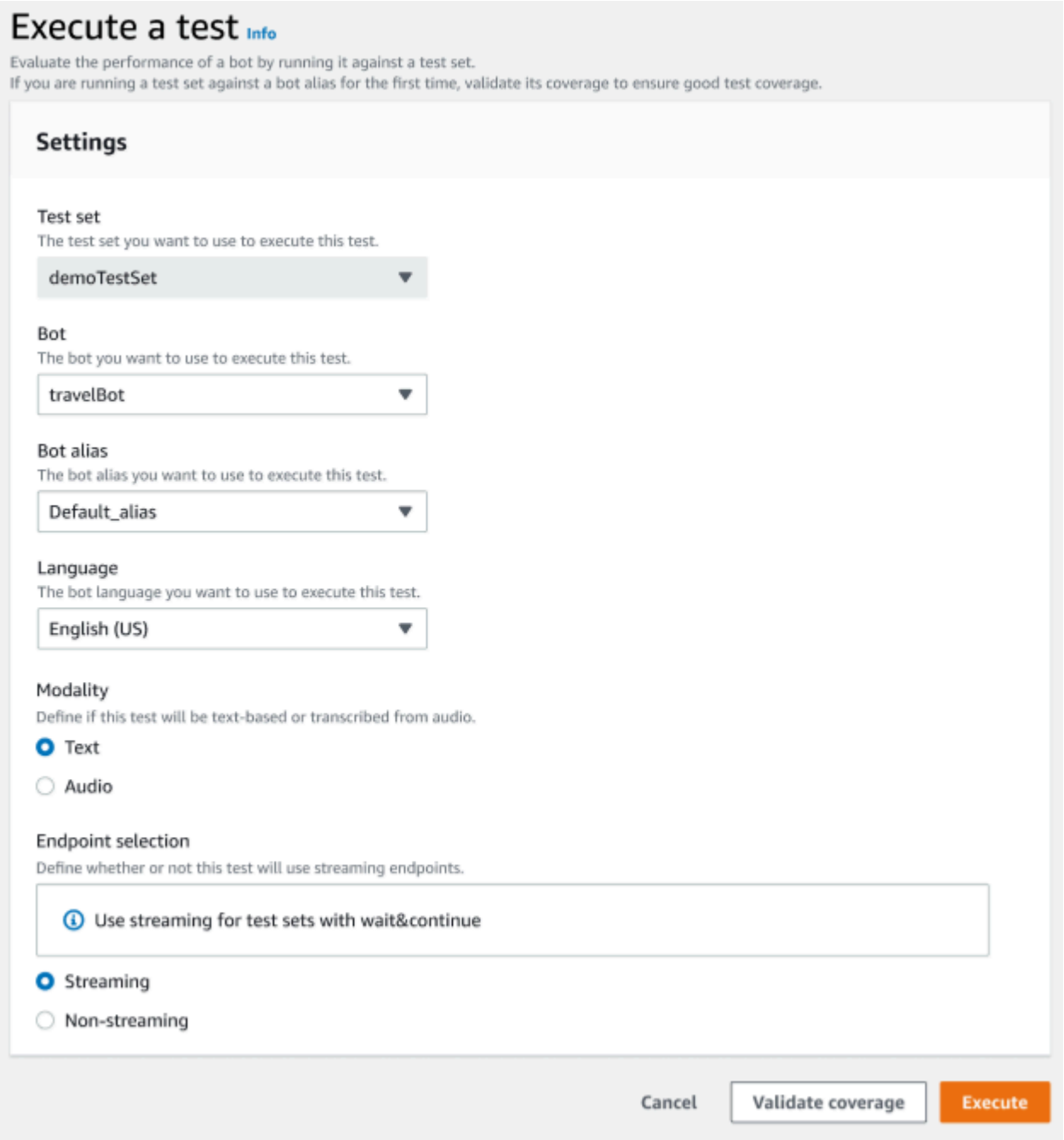
Per aggiornare un set di test:

1. Dal record del set di test, scegli il pulsante Aggiorna set di test in alto a destra.
2. Scegli un file da caricare dal tuo account Amazon S3 o carica un file di test CSV dal tuo computer. NOTA: l'aggiornamento di un set di test sovrascriverà i dati esistenti.
3. Seleziona il pulsante Aggiorna.

4. Un messaggio conferma che il set di test è stato aggiornato correttamente. NOTA: Questa operazione può richiedere alcuni minuti, a seconda della complessità e delle dimensioni del set di test.
5. Un messaggio conferma che il set di test è stato aggiornato correttamente e lo stato visualizza Pronto per il test.

Esegui un test

Per eseguire un set di test, è necessario scegliere il bot appropriato per eseguire il test sul set di test. Puoi scegliere un bot dal tuo AWS account dal menu a discesa sotto Test Set. Questa operazione testerà il bot selezionato rispetto ai dati di test convalidati per riportare le metriche delle prestazioni rispetto ai dati di base del set di test.



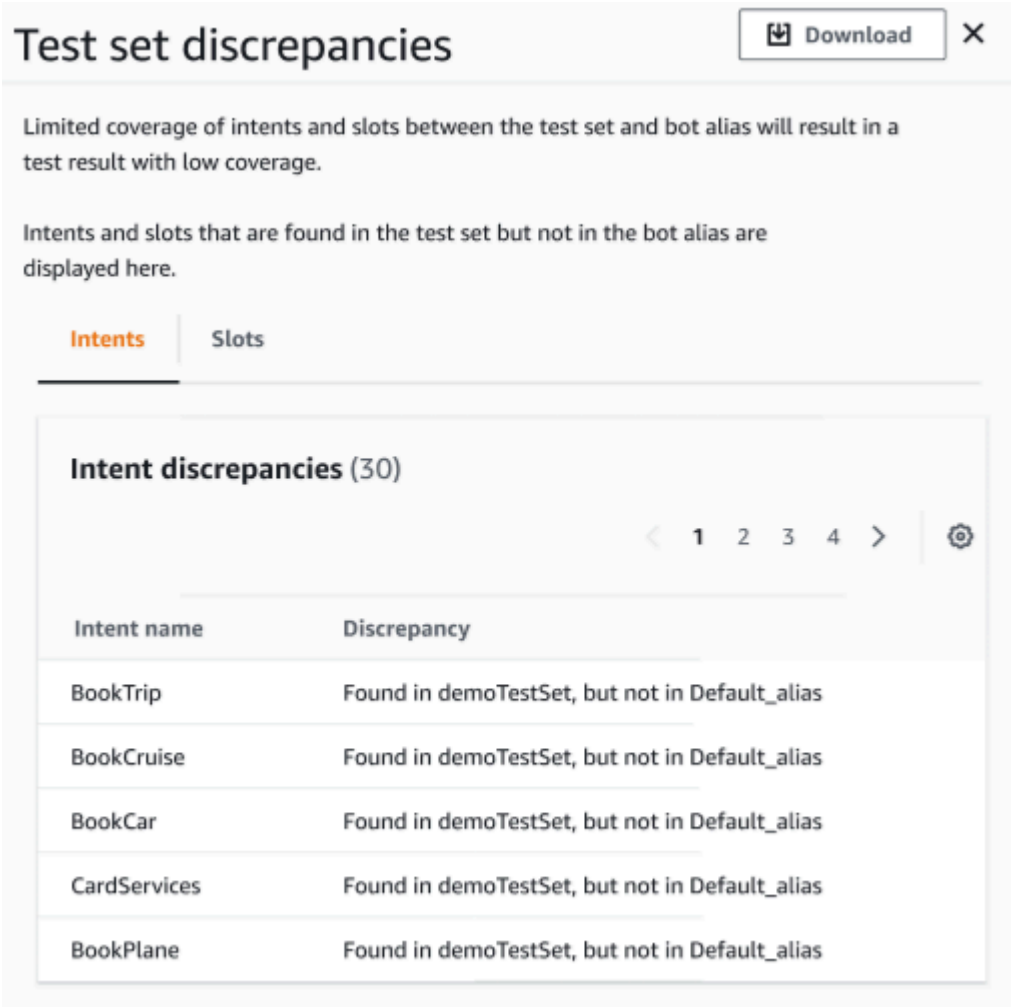
Per eseguire un test nel Test Workbench

1. Nella pagina dei record del set di test, scegli Esegui test.
2. Seleziona il set di test che desideri utilizzare nel test.
3. Seleziona il nome del bot da usare nel test dal menu a discesa Bot.
4. Scegliete un alias del bot, se applicabile, dal menu a discesa Bot alias.
5. Dalla selezione Lingue, scegli una versione dell'inglese.

6. Seleziona Testo o Audio per il tipo di modalità.
7. Scegli la tua sede Amazon S3. (solo audio)
8. Seleziona la tua selezione di endpoint per il bot. (solo streaming)
9. Seleziona il pulsante Convalida copertura per confermare che il test è pronto per l'esecuzione. Se sono presenti errori nella fase di convalida, rivedi i parametri precedenti e apporta le correzioni.
10. Seleziona Esegui per eseguire il test.
11. Un messaggio conferma che il test è stato eseguito correttamente.

Copertura del set di test

Una copertura limitata degli intenti e degli slot tra il set di test e il bot può comportare le misure prestazionali previste. Ti consigliamo di controllare la copertura del set di test prima di eseguire il test.



Per esaminare la copertura di convalida

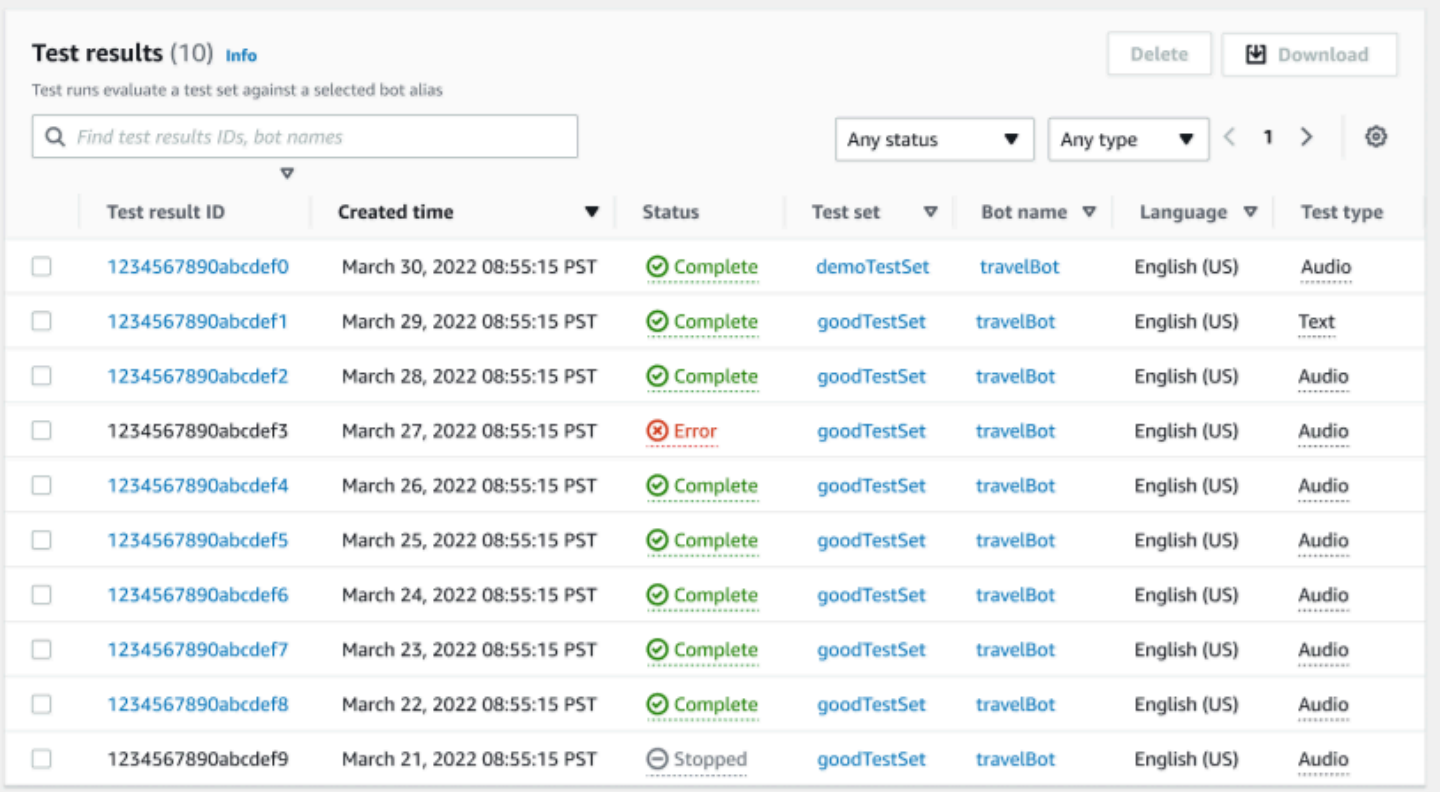
1. Nei record del set di test, scegli il pulsante Convalida copertura.
2. Il messaggio indica che sta convalidando la copertura tra il set di test e il bot selezionato.
3. Una volta completata l'operazione, il messaggio indica che la convalida della copertura è avvenuta con successo.
4. Scegli il pulsante Visualizza dettagli nella parte inferiore della finestra.
5. Visualizza le discrepanze dei set di test per intenti e slot scegliendo la scheda corrispondente. Puoi scaricare questi dati in formato CSV scegliendo il pulsante Download.
6. Esamina i risultati di convalida per i dati del set di test, gli intenti dei bot e gli slot. Identifica i problemi e apporta modifiche all'architettura del set di test dei bot per migliorare i risultati. Carica il set di test modificato e il bot per eseguire il test dopo aver apportato modifiche al file CSV. NOTA: la copertura di convalida è valida per il set di test e non per il bot. Gli intenti presenti nel bot ma non presenti nel set di test non verranno coperti.

Visualizza i risultati del test

Interpreta i risultati dei test del Test Workbench per determinare dove la conversazione tra il bot e il cliente potrebbe non riuscire o richiedere al cliente di fare più tentativi per soddisfare l'intento.

Individuando questi problemi nei risultati del test, è possibile ottimizzare le prestazioni del bot migliorando le prestazioni degli intenti utilizzando diversi dati di allenamento o espressioni più coerenti con i valori di trascrizione del bot in tempo reale.

È possibile ottenere una visione dettagliata degli intenti e degli slot con discrepanze nelle prestazioni. Una volta identificati gli intenti o gli slot che presentano discrepanze, puoi approfondire ulteriormente e rivedere le espressioni e il flusso di conversazione.



Per esaminare i risultati dei test:

1. Vai all'elenco dei set di test dal menu a sinistra per selezionare l'opzione Risultati dei test in Test workbench. NOTA: i risultati dei test indicano lo stato di completamento se hanno avuto esito positivo.
2. Seleziona l'ID del risultato del test per i risultati del test che desideri esaminare.

Dettagli dei risultati del test

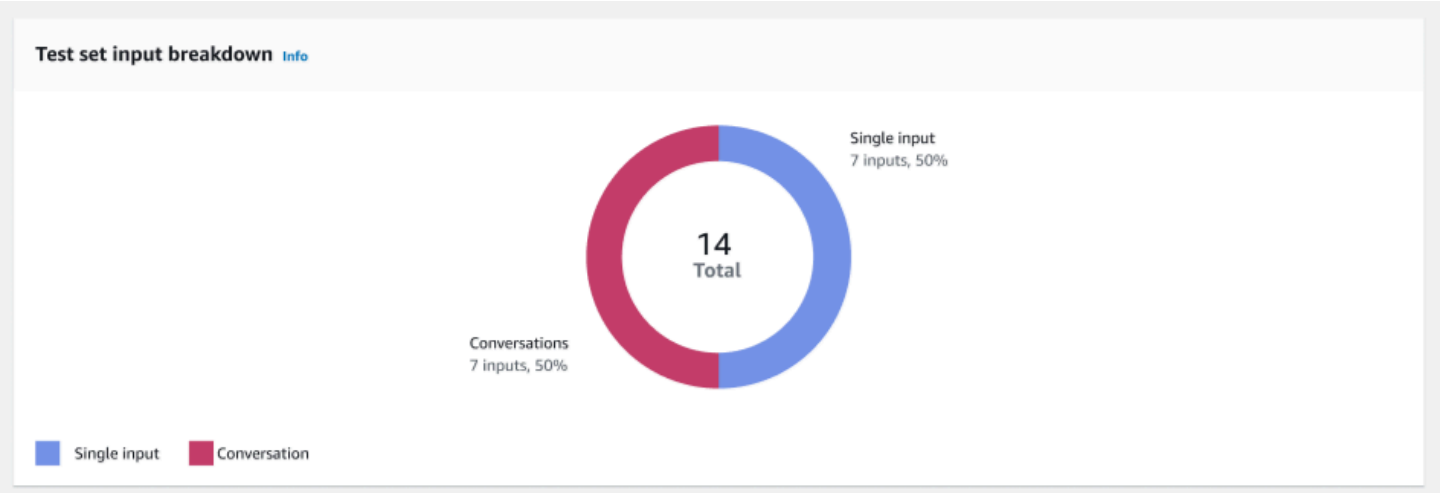
I risultati del test mostrano i dettagli del set di test, gli intenti utilizzati e gli slot utilizzati. Fornisce inoltre la suddivisione complessiva degli input del set di test, che include i risultati complessivi, i risultati della conversazione, l'intento e i risultati degli slot.

I risultati dei test comprendono tutte le informazioni relative ai test, come:

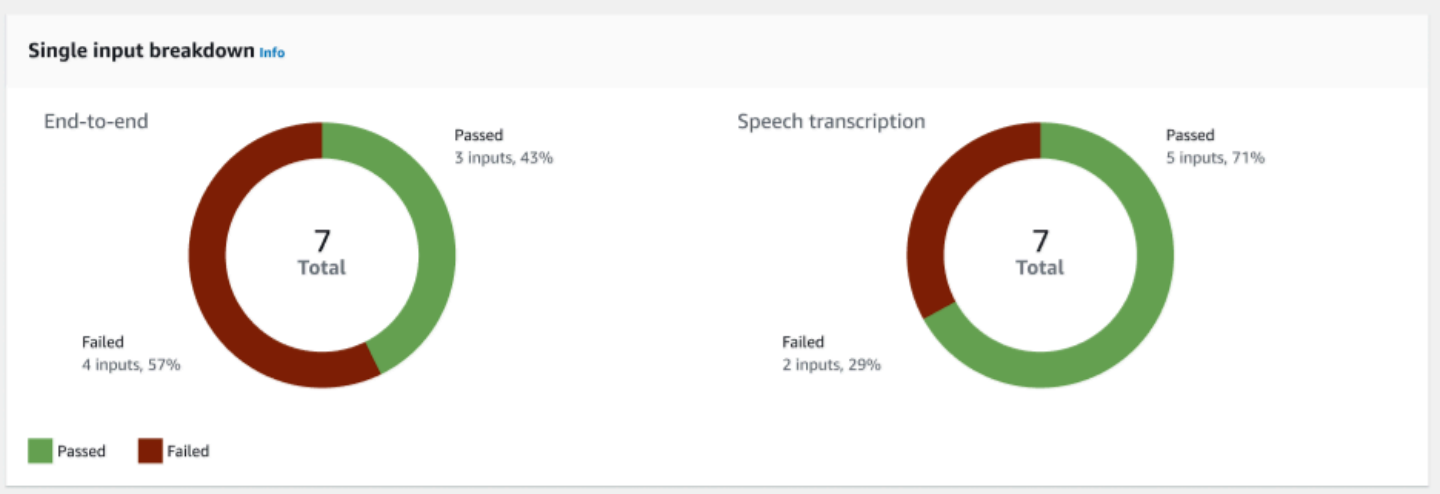
- Dettagli del test (metadati)
- Risultati complessivi
- Risultati della conversazione
- Intento e risultati degli slot

- Risultati dettagliati

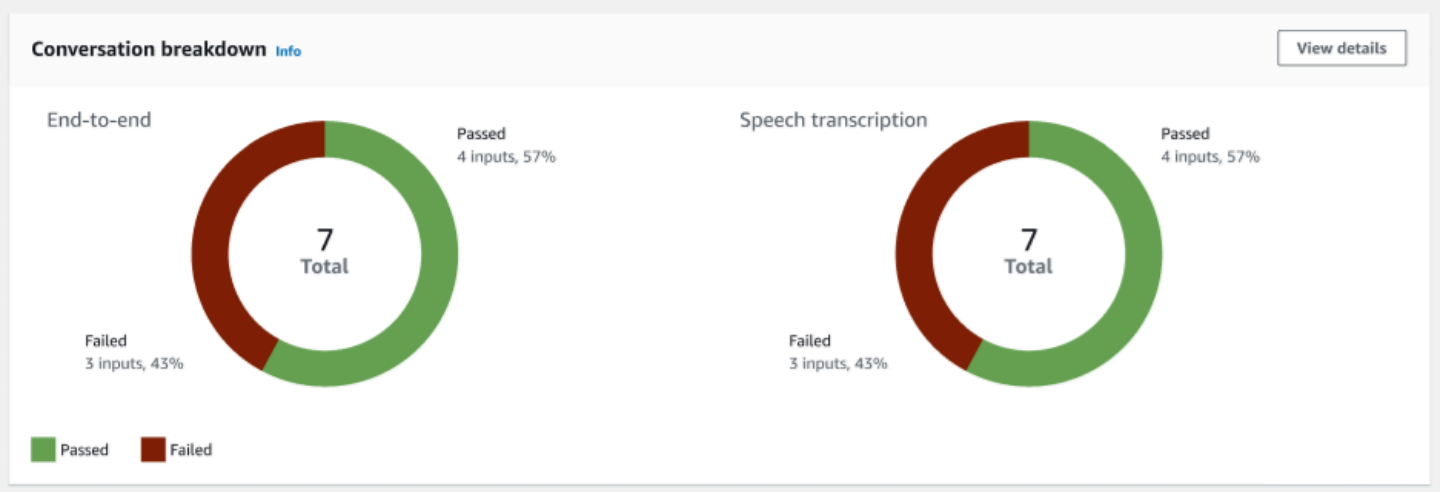
Scheda dei risultati complessivi:



Suddivisione dell'input del set di test: questo grafico mostra la suddivisione del numero di conversazioni e delle singole espressioni di input nel set di test.

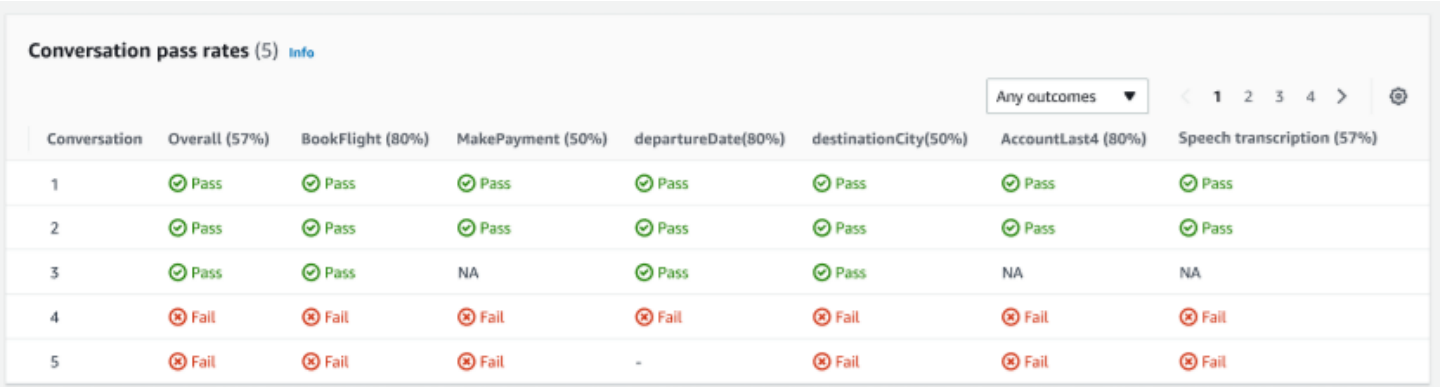


Suddivisione in input singolo: visualizza due grafici che includono end-to-end conversazioni e trascrizioni vocali. Il numero di input passati e non riusciti è indicato su ogni grafico. Nota: la tabella di trascrizione vocale sarà visibile solo per il set di test audio.

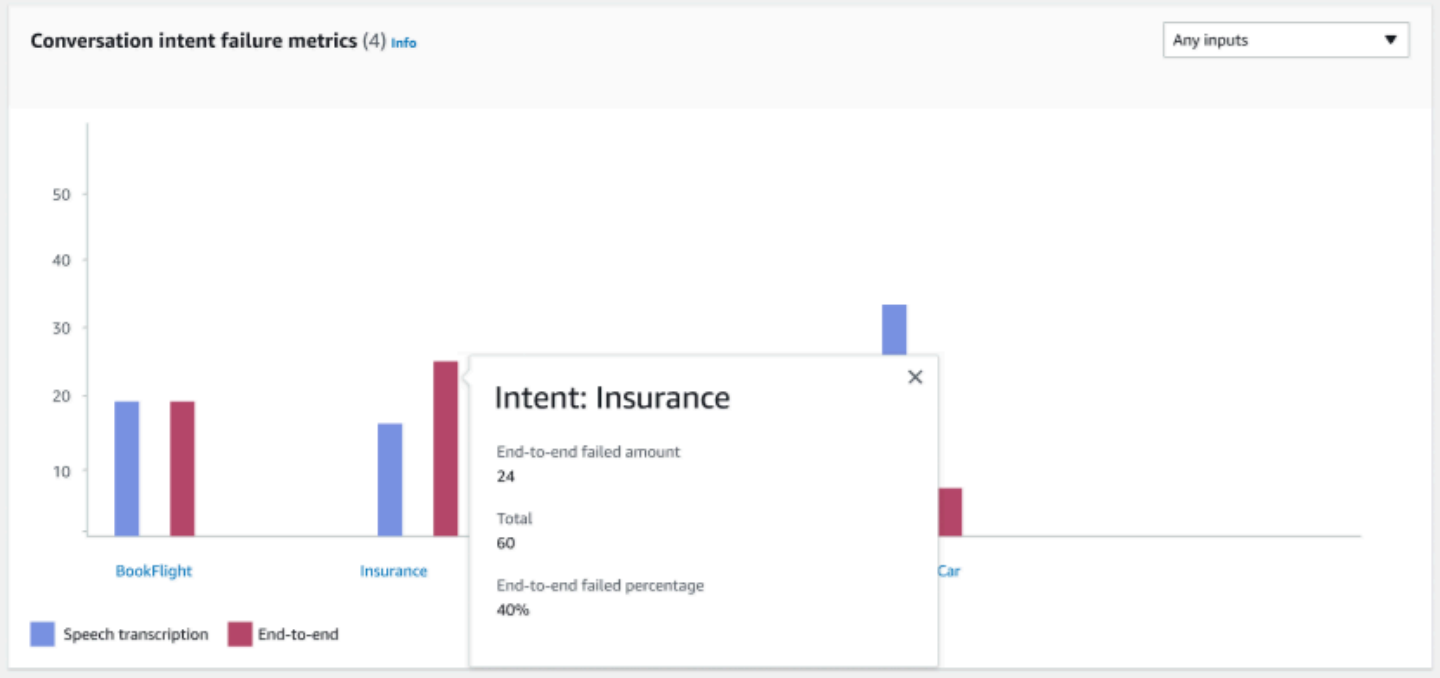


Suddivisione delle conversazioni: visualizza due grafici che includono end-to-end conversazioni e trascrizioni vocali. Il numero di input passati e non riusciti è indicato su ogni grafico. Nota: la tabella di trascrizione vocale sarà visibile solo per il set di test audio.

Scheda dei risultati della conversazione:



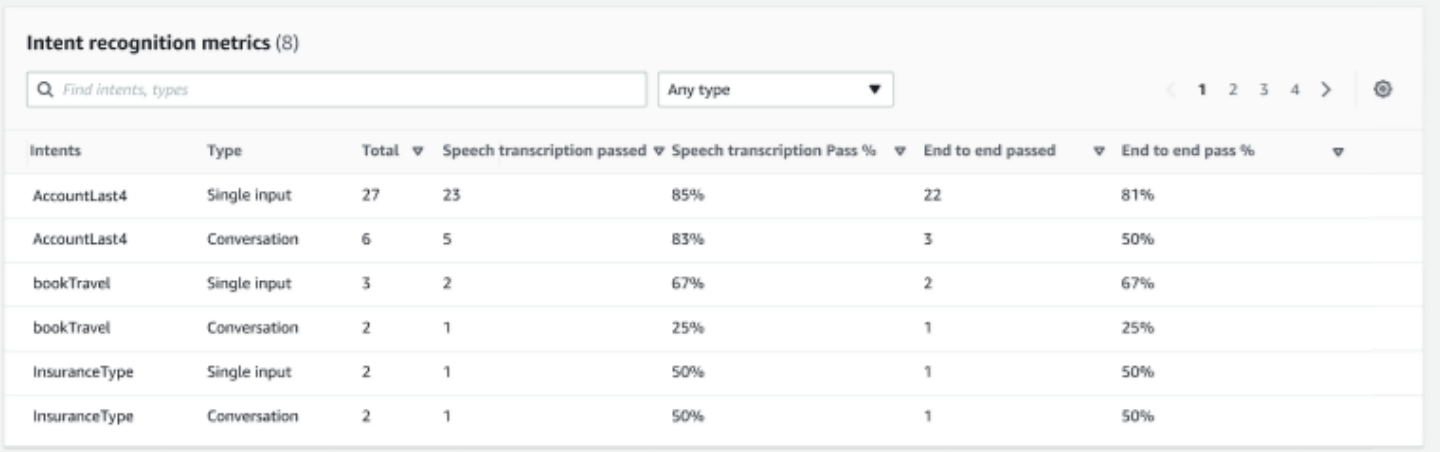
Percentuali di successo delle conversazioni: la tabella delle percentuali di successo delle conversazioni viene utilizzata per vedere quali intenti e quali intervalli vengono utilizzati in ciascuna conversazione del set di test. Puoi visualizzare dove la conversazione ha avuto esito negativo esaminando quali intenti o slot hanno avuto esito negativo, oltre alla percentuale di successo di ogni intento e slot.



Metriche del fallimento dell'intento di conversazione: questa metrica mostra i 5 intenti con le peggiori prestazioni nel set di test. Questo pannello mostra un grafico della percentuale o del numero di intenti che hanno avuto successo o meno, in base ai registri delle conversazioni o alla trascrizione del bot. Un intento riuscito non significa che l'intera conversazione abbia avuto successo. Queste metriche si applicano solo al valore degli intenti, indipendentemente dall'intento precedente o successivo.

Metriche degli errori negli slot di conversazione: questa metrica mostra i 5 slot con le prestazioni peggiori nel set di test. Indicava la percentuale di successo per ogni slot nell'intento. Il grafico a barre mostra sia la trascrizione del parlato che le end-to-end conversazioni per ogni slot dell'intento.

Scheda dei risultati degli intenti e degli slot:

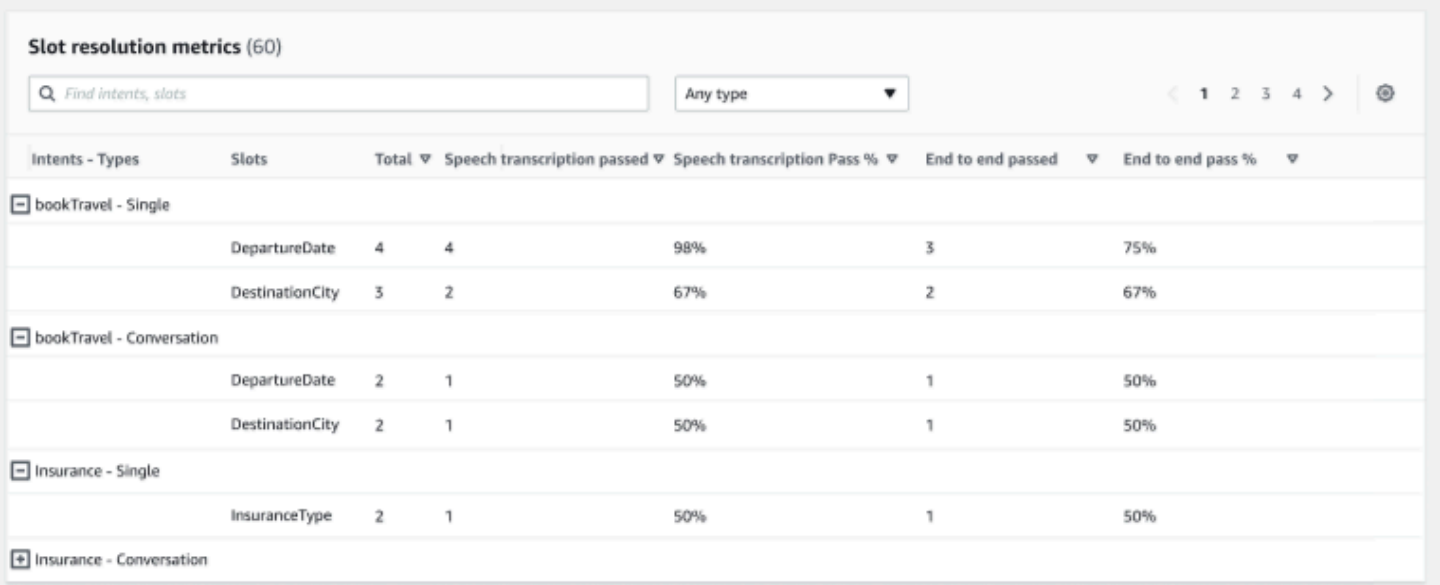


Intent recognition metrics (8)

Find intents, types | Any type | 1 2 3 4

Intents	Type	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
AccountLast4	Single input	27	23	85%	22	81%
AccountLast4	Conversation	6	5	83%	3	50%
bookTravel	Single input	3	2	67%	2	67%
bookTravel	Conversation	2	1	25%	1	25%
InsuranceType	Single input	2	1	50%	1	50%
InsuranceType	Conversation	2	1	50%	1	50%

Metriche di riconoscimento degli intenti: mostra una tabella di quanti intenti sono stati riconosciuti con successo. Visualizza la velocità di superamento della trascrizione vocale e delle conversazioni. end-to-end



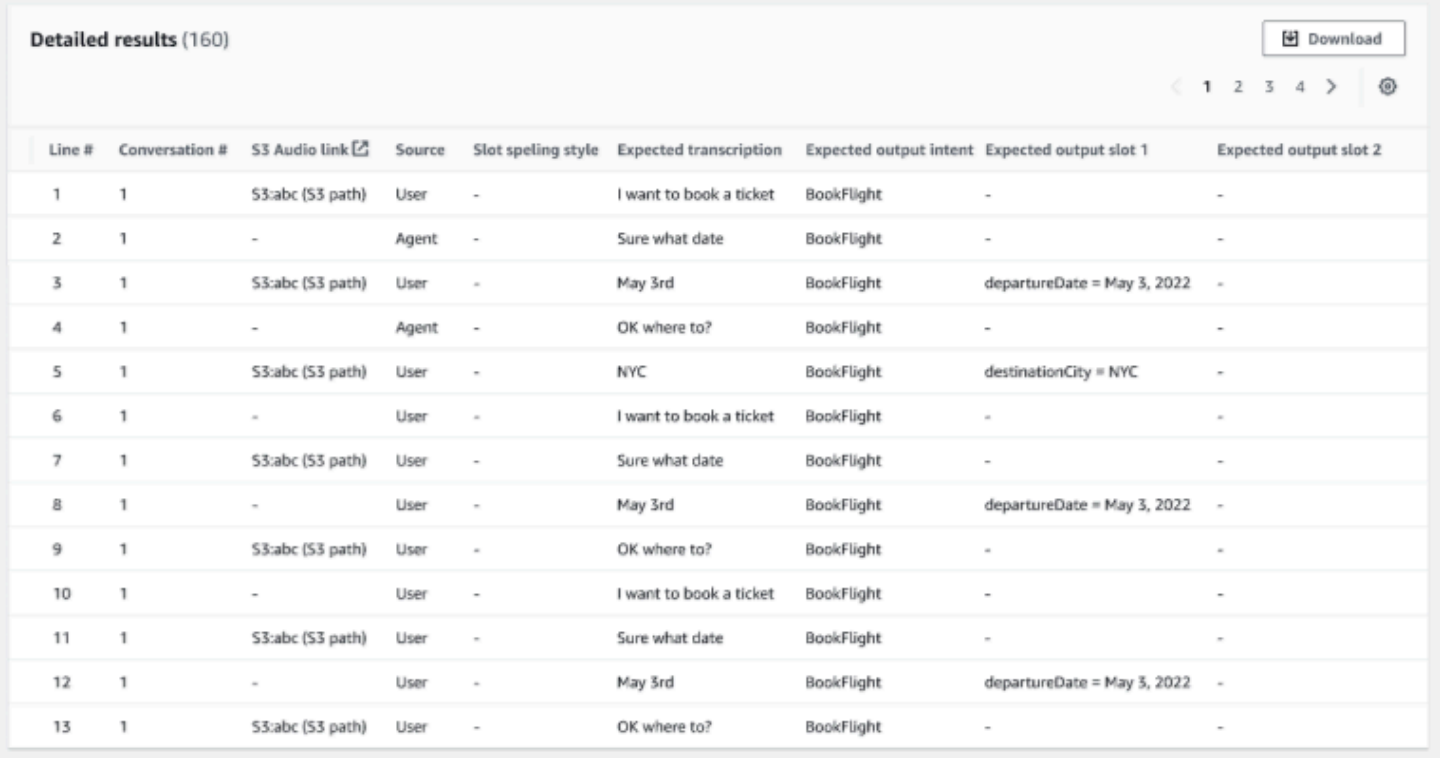
Slot resolution metrics (60)

Find intents, slots | Any type | 1 2 3 4

Intents - Types	Slots	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
bookTravel - Single						
	DepartureDate	4	4	98%	3	75%
	DestinationCity	3	2	67%	2	67%
bookTravel - Conversation						
	DepartureDate	2	1	50%	1	50%
	DestinationCity	2	1	50%	1	50%
Insurance - Single						
	InsuranceType	2	1	50%	1	50%
Insurance - Conversation						

Metriche di risoluzione degli slot: mostra gli intenti e gli slot separatamente e la percentuale di successo e fallimento di ogni slot per ogni intento utilizzato nella conversazione o nel singolo input. Visualizza la velocità di superamento della trascrizione vocale e delle conversazioni. end-to-end

Scheda dettagliata dei risultati:



Detailed results (160)

Download

1 2 3 4

Line #	Conversation #	S3 Audio link	Source	Slot speling style	Expected transcription	Expected output intent	Expected output slot 1	Expected output slot 2
1	1	S3:abc (S3 path)	User	-	I want to book a ticket	BookFlight	-	-
2	1	-	Agent	-	Sure what date	BookFlight	-	-
3	1	S3:abc (S3 path)	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
4	1	-	Agent	-	OK where to?	BookFlight	-	-
5	1	S3:abc (S3 path)	User	-	NYC	BookFlight	destinationCity = NYC	-
6	1	-	User	-	I want to book a ticket	BookFlight	-	-
7	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
8	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
9	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-
10	1	-	User	-	I want to book a ticket	BookFlight	-	-
11	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
12	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
13	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-

Risultati dettagliati: mostra una tabella dettagliata nel registro delle conversazioni con gli enunciati di utenti e agenti e l'output e la trascrizione previsti per ogni slot. Puoi scaricare questo rapporto selezionando il pulsante Download.

La tabella seguente elenca i risultati, i messaggi di errore relativi agli errori con gli scenari.

Scenario	Messaggio di errore	Azione
Mancata corrispondenza degli intenti	BookFlight Intento previsto ma era intento. BookHotel	Salta gli altri turni della conversazione
Mancata corrispondenza tra Slot Elicitation	Era previsto che venisse generato lo slot Departure Date, ma era CabinType.	Salta gli altri turni della conversazione
Mancata corrispondenza del valore dello slot	Mancata corrispondenza tra il valore dello slot previsto e quello effettivo.	Continua con gli altri turni delle conversazioni
Manca il prompt ack-to-back dell'agente B	Si aspettava che il bot restituis se una richiesta dell'agente in	Salta gli altri turni della conversazione

Scenario	Messaggio di errore	Azione
	questo turno, ma non è stata ricevuta.	
Mancata corrispondenza nella trascrizione	La trascrizione prevista non corrisponde alla trascrizione effettiva.	Continua con altri turni nelle conversazioni
Slot opzionale non attivato	Ci si aspettava che venisse generato lo slot CabinType nel prossimo turno, tuttavia l'intento attuale era stato raggiunto prima.	Salta gli altri turni della conversazione
Slot non riconosciuto	Lo slot DepartureDate previsto non è stato riconosciuto in questo turno.	Salta gli altri turni della conversazione
Richiesta aggiuntiva per back-to-back l'agente	Era previsto il turno di un utente, ma era richiesto dall'agente	Salta gli altri turni della conversazione

Streaming su un bot Amazon Lex V2

Puoi utilizzare l'API di streaming Amazon Lex V2 per avviare uno stream bidirezionale tra un bot Amazon Lex V2 e la tua applicazione. L'avvio di uno stream consente al bot di gestire la conversazione tra il bot e l'utente. Il bot risponde all'input dell'utente senza che tu scriva codice per gestire le risposte dell'utente. Il bot può:

- Gestisci le interruzioni dell'utente durante la riproduzione di un prompt. Per ulteriori informazioni, consulta Consentire al bot di essere interrotto dall'utente.
- Attendi che l'utente fornisca un input. Ad esempio, il bot può attendere che l'utente raccolga i dati della carta di credito. Per ulteriori informazioni, consulta Abilitare il bot ad attendere che l'utente fornisca ulteriori informazioni.
- Inserisci sia l'ingresso a frequenza multipla (DTMF) bicolore che l'ingresso audio nello stesso stream.
- Gestisci meglio le pause nell'input dell'utente rispetto a quando gestissi la conversazione dalla tua applicazione.

Il bot Amazon Lex V2 non solo risponde ai dati inviati dall'applicazione, ma invia anche informazioni sullo stato della conversazione all'applicazione. È possibile utilizzare queste informazioni per modificare il modo in cui l'applicazione risponde ai clienti.

Il bot Amazon Lex V2 monitora anche la connessione tra il bot e l'applicazione. Può determinare se la connessione è scaduta.

Per utilizzare l'API per avviare uno stream verso un bot Amazon Lex V2, consultaAvvio di uno stream verso un bot.

Quando avvii lo streaming verso un bot Amazon Lex V2 dalla tua applicazione, puoi configurare il bot in modo che accetti input audio o input di testo dall'utente. Puoi anche scegliere se l'utente riceve audio o testo in risposta al suo input.

Se hai configurato il bot Amazon Lex V2 per accettare input audio dall'utente, non può accettare input di testo. Se hai configurato il bot per accettare l'input di testo, l'utente può utilizzare solo testo scritto per comunicare con esso.

Quando un bot Amazon Lex V2 riceve un ingresso audio in streaming, il bot determina quando un utente inizia a parlare e quando smette di parlare. Gestisce eventuali pause o interruzioni da parte

dell'utente. Può anche accettare input DTMF (multi-frequenza a doppia tonalità) e input vocali nello stesso flusso. Questo aiuta l'utente a interagire con il bot in modo più naturale. Puoi presentare agli utenti messaggi e prompt di benvenuto. Puoi anche consentire agli utenti di interrompere tali messaggi e prompt.

Quando avvii uno stream bidirezionale, Amazon Lex V2 utilizza il protocollo HTTP/2. L'applicazione e il bot si scambiano dati in un unico flusso sotto forma di una serie di eventi. Un evento può essere uno dei seguenti:

- Testo, audio o input DTMF dall'utente.
- Segnali dall'applicazione al bot Amazon Lex V2. Questi includono l'indicazione che la riproduzione audio di un messaggio è stata completata o che l'utente si è disconnesso dalla sessione.

Per ulteriori informazioni sugli eventi di , consulta Avvio di uno stream verso un bot. Per informazioni su come codificare gli eventi, consultaCodifica del flusso di eventi.

Argomenti

Avvio di uno stream verso un bot

Utilizzi l'StartConversationoperazione per avviare uno stream tra l'utente e il bot Amazon Lex V2 nella tua applicazione. La`POST` richiesta dall'applicazione stabilisce una connessione tra l'applicazione e il bot Amazon Lex V2. Ciò consente all'applicazione e al bot di iniziare a scambiarsi informazioni tramite eventi.

L'`StartConversation`operazione è supportata solo nei seguenti SDK:

- SDK AWS per C++
- AWS SDK for Java V2

- SDK AWS per JavaScript v3
- SDK AWS per Ruby V3

Il primo evento che l'applicazione deve inviare al bot Amazon Lex V2 è un ConfigurationEvent. Questo evento include informazioni come il formato del tipo di risposta. Di seguito sono riportati i parametri che è possibile utilizzare in un evento di configurazione:

- responseContentType— Determina se il bot risponde all'input dell'utente con testo o voce.
- SessionState — Informazioni relative alla sessione di streaming con il bot, come l'intento predeterminato o lo stato della finestra di dialogo.
- WelcomeMessages: specifica i messaggi di benvenuto che vengono riprodotti dall'utente all'inizio della conversazione con un bot. Questi messaggi vengono riprodotti prima che l'utente fornisca qualsiasi input. Per attivare un messaggio di benvenuto, è necessario specificare anche i valori per i`dialogAction` parametri`sessionState` and.
- disablePlayback — Determina se il bot deve attendere un segnale dal client prima di iniziare ad ascoltare l'input del chiamante. Per impostazione predefinita, la riproduzione è attivata, quindi il valore di questo campo è`false`.
- requestAttributes: fornisce informazioni aggiuntive per la richiesta.

Per informazioni su come specificare i valori per i parametri precedenti, vedere il tipo di ConfigurationEventdati dell'StartConversationoperazione.

Ogni flusso tra un bot e l'applicazione può avere un solo evento di configurazione. Dopo che l'applicazione ha inviato un evento di configurazione, il bot può ricevere ulteriori comunicazioni dall'applicazione.

Se hai specificato che l'utente utilizza l'audio per comunicare con il bot Amazon Lex V2, l'applicazione può inviare i seguenti eventi al bot durante quella conversazione:

- AudioInputEvent— Contiene un blocco audio con una dimensione massima di 320 byte. L'applicazione deve utilizzare più eventi di input audio per inviare un messaggio dal server al bot. Ogni evento di ingresso audio nello stream deve avere lo stesso formato audio.
- DTMFInputEvent: invia un input DTMF al bot. Ogni pressione di un tasto DTMF corrisponde a un singolo evento.
- PlaybackCompletionEvent— Informa il server che è stata riprodotta una risposta dall'input dell'utente. È necessario utilizzare un evento di completamento della riproduzione se si invia una

risposta audio all'utente. Se `disablePlayback` uno dei tuoi eventi di configurazione è `true`, non puoi usare questa funzione.

- DisconnectionEvent— Informa il bot che l'utente si è disconnesso dalla conversazione.

Se hai specificato che l'utente utilizza il testo per comunicare con il bot, l'applicazione può inviare i seguenti eventi al bot durante quella conversazione:

- TextInputEvent— Testo inviato dalla tua applicazione al bot. È possibile contenere fino a 512 caratteri in un evento di immissione di testo.
- PlaybackCompletionEvent— Informa il server che è stata riprodotta una risposta dall'input dell'utente. È necessario utilizzare questo evento se si sta riproducendo l'audio all'utente. Se `disablePlayback` uno dei tuoi eventi di configurazione è `true`, non puoi usare questa funzione.
- DisconnectionEvent— Informa il bot che l'utente si è disconnesso dalla conversazione.

È necessario codificare ogni evento inviato a un bot Amazon Lex V2 nel formato corretto. Per ulteriori informazioni, consulta Codifica del flusso di eventi.

Ogni evento ha un ID evento. Per aiutare a risolvere eventuali problemi che potrebbero verificarsi nello stream, assegna un ID evento univoco a ciascun evento di input. È quindi possibile risolvere eventuali errori di elaborazione con il bot.

Amazon Lex V2 utilizza anche timestamp per ogni evento. È possibile utilizzare questi timestamp in aggiunta all'ID evento per risolvere eventuali problemi di trasmissione di rete.

Durante la conversazione tra l'utente e il bot Amazon Lex V2, il bot può inviare i seguenti eventi in uscita in risposta all'utente:

- IntentResultEvent— Contiene l'intento che Amazon Lex V2 ha determinato dall'enunciato dell'utente. Ogni evento interno con risultati include:
  - inputMode — Il tipo di espressione dell'utente. I valori validi sono `Speech`, `DTMF` o `Text`.
  - interpretazioni: interpretazioni che Amazon Lex V2 determina dall'enunciato dell'utente.
  - requestAttributes — Se non hai modificato gli attributi della richiesta utilizzando una funzione lambda, questi sono gli stessi attributi passati all'inizio della conversazione.
  - sessionId: identificatore di sessione utilizzato per la conversazione.
  - SessionState: lo stato della sessione dell'utente con Amazon Lex V2.

- TranscriptEvent— Se l'utente fornisce un input all'applicazione, questo evento contiene la trascrizione dell'enunciato dell'utente al bot. La tua applicazione non riceve un messaggio`TranscriptEvent` se non c'è alcun input da parte dell'utente.

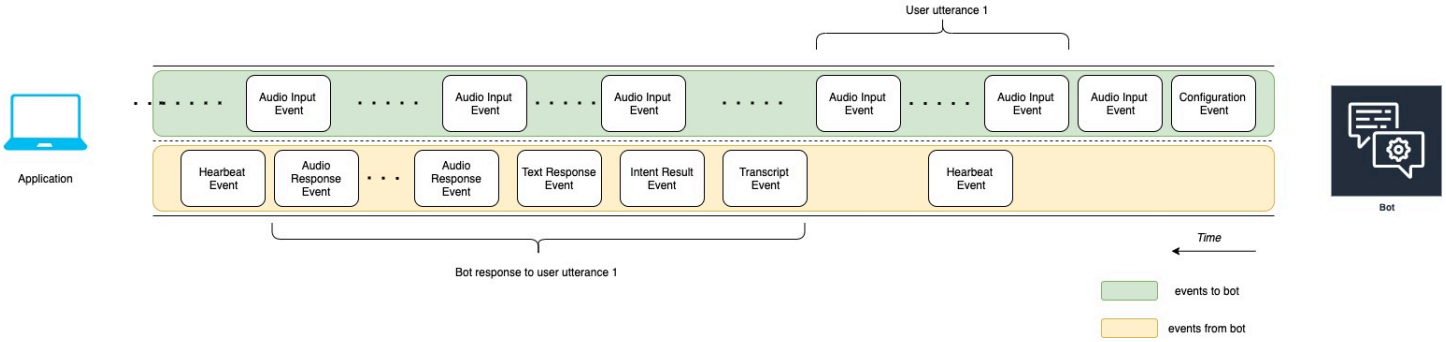
  Il valore dell'evento di trascrizione inviato all'applicazione dipende dal fatto che tu abbia specificato l'audio (voce e DMTF) o il testo come modalità di conversazione:

  - Trascrizione dell'input vocale: se l'utente sta parlando con il bot, l'evento di trascrizione è la trascrizione dell'audio dell'utente. È una trascrizione di tutto il discorso dal momento in cui l'utente inizia a parlare al momento in cui finisce di parlare.
  - Trascrizione dell'input DTMF: se l'utente sta digitando su una tastiera, l'evento di trascrizione contiene tutte le cifre che l'utente ha premuto nell'input.
  - Trascrizione dell'input di testo: se l'utente fornisce un input di testo, l'evento di trascrizione contiene tutto il testo inserito dall'utente.
- TextResponseEvent— Contiene la risposta del bot in formato testo. Per impostazione predefinita, viene restituita una risposta testuale. Se hai configurato Amazon Lex V2 per restituire una risposta audio, questo testo viene utilizzato per generare una risposta audio. Ogni evento di risposta testuale contiene una serie di oggetti di messaggio che il bot restituisce all'utente.
- AudioResponseEvent— Contiene la risposta audio sintetizzata dal testo generato in`TextResponseEvent`. Per ricevere eventi di risposta audio, devi configurare Amazon Lex V2 per fornire una risposta audio. Tutti gli eventi di risposta audio hanno lo stesso formato audio. Ogni evento contiene blocchi audio di non più di 100 byte. Amazon Lex V2 invia un blocco audio vuoto con il`bytes` campo impostato`null` per indicare la fine dell'evento di risposta audio all'applicazione.
- PlaybackInterruptionEvent— Quando un utente interrompe una risposta che il bot ha inviato all'applicazione, Amazon Lex V2 attiva questo evento per interrompere la riproduzione della risposta.
- HeartbeatEvent— Amazon Lex V2 invia periodicamente questo evento per evitare il timeout della connessione tra l'applicazione e il bot.

Sequenza temporale degli eventi per una conversazione audio

I diagrammi seguenti mostrano una conversazione audio in streaming tra un utente e un bot Amazon Lex V2. L'applicazione trasmette continuamente l'audio al bot e il bot cerca l'input dell'utente dall'audio. In questo esempio, sia l'utente che il bot utilizzano la voce per comunicare. Ogni diagramma corrisponde a un'espressione dell'utente e alla risposta del bot a quell'enunciato.

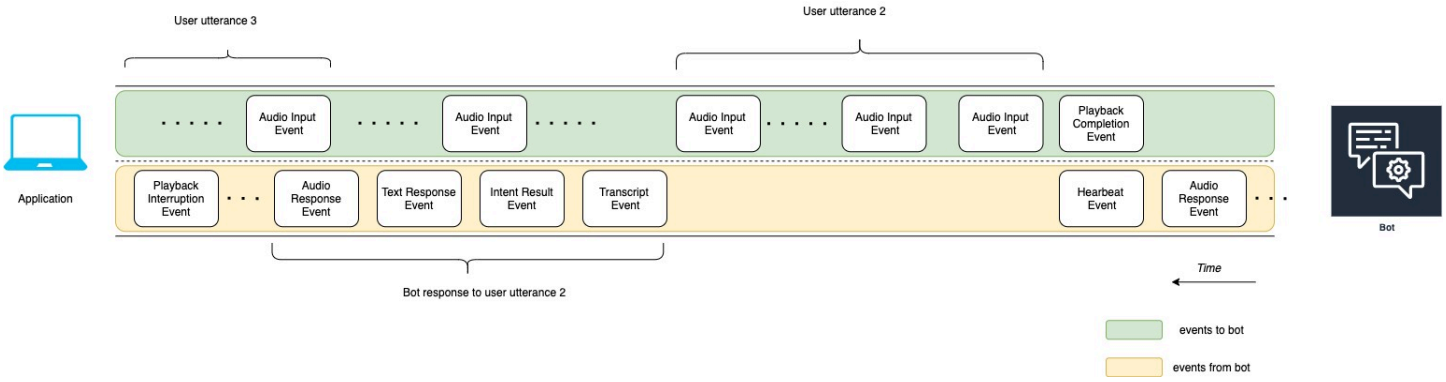
Il diagramma seguente mostra l'inizio di una conversazione tra l'applicazione e il bot. Lo stream inizia al tempo zero (t0).



Nell'elenco seguente vengono descritti gli eventi del diagramma precedente.

- t0: l'applicazione invia un evento di configurazione al bot per avviare lo stream.
- t1: L'applicazione trasmette dati audio. Questi dati vengono suddivisi in una serie di eventi di input dall'applicazione.
- t2: Per l'enunciazione dell'utente 1, il bot rileva un evento di ingresso audio quando l'utente inizia a parlare.
- t2: Mentre l'utente sta parlando, il bot invia un evento di battito cardiaco per mantenere la connessione. Invia gli eventi a intermittenza per assicurarsi che non si verifica il timeout della connessione.
- t3: Il bot rileva la fine dell'enunciato dell'utente.
- t4: Il bot invia all'applicazione un evento di trascrizione che contiene una trascrizione del discorso dell'utente. Questo è l'inizio della risposta del bot all'espressione dell'utente 1.
- t5: il bot invia un evento di intento per indicare l'azione che l'utente desidera eseguire.
- t6: Il bot inizia a fornire la sua risposta come testo in un evento di risposta testuale.
- t7: Il bot invia una serie di eventi di risposta audio all'applicazione da riprodurre per l'utente.
- t8: Il bot invia un altro evento di battito cardiaco per mantenere la connessione in modo intermittente.

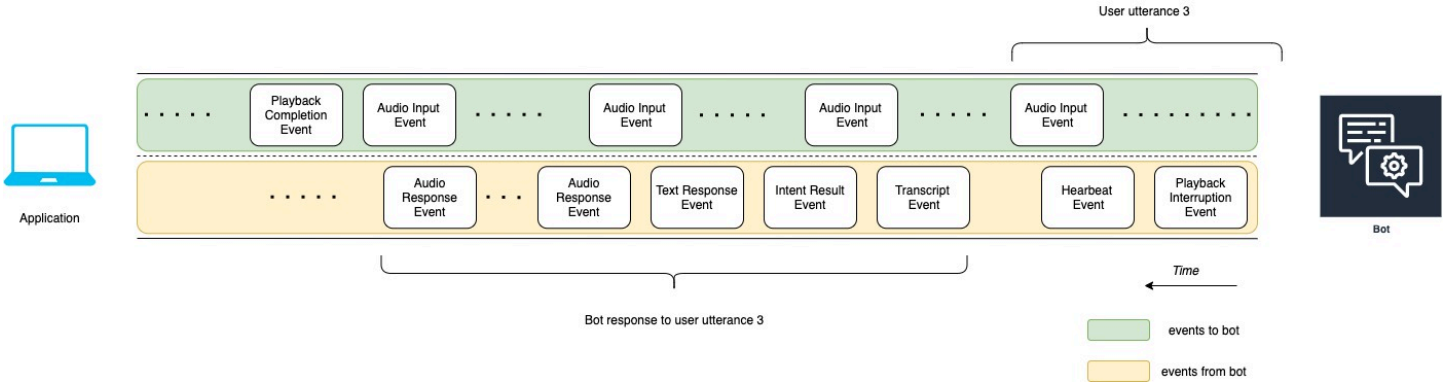
Il diagramma seguente è una continuazione del diagramma precedente. Mostra l'applicazione che invia un evento di completamento della riproduzione al bot per indicare che ha interrotto la riproduzione della risposta audio per l'utente. L'applicazione riproduce all'utente la risposta del bot all'enunciato 1 dell'utente. L'utente risponde alla risposta del bot all'espressione dell'utente 1 con User utterance 2.



Nell'elenco seguente vengono descritti gli eventi del diagramma precedente:

- t10: L'applicazione invia un evento di completamento della riproduzione per indicare che ha terminato la riproduzione del messaggio del bot all'utente.
- t11: L'applicazione invia la risposta dell'utente al bot come User utterance 2.
- t12: Per la risposta del bot all'espressione dell'utente 2, il bot attende che l'utente smetta di parlare e quindi inizia a fornire una risposta audio.
- t13: Mentre il bot invia la risposta del bot all'espressione utente 2 all'applicazione, il bot rileva l'inizio di User utterance 3. Il bot interrompe la risposta del bot all'espressione 2 dell'utente e invia un evento di interruzione della riproduzione.
- t14: Il bot invia un evento di interruzione della riproduzione all'applicazione per segnalare che l'utente ha interrotto la richiesta.

Il diagramma seguente mostra la risposta del bot all'enunciato dell'utente 3 e che la conversazione continua dopo che il bot ha risposto all'enunciato dell'utente.



Utilizzo dell'API per avviare una conversazione di streaming

Quando avvii uno stream verso un bot Amazon Lex V2, esegui le seguenti attività:

1. Crea una connessione iniziale al server.
2. Configura le credenziali di sicurezza e i dettagli del bot. I dettagli del bot includono se il bot accetta DTMF e input audio o input di testo.
3. Invia eventi al server. Questi eventi sono dati di testo o dati audio dell'utente.
4. Elabora gli eventi inviati dal server. In questo passaggio, si determina se l'output del bot viene presentato all'utente come testo o voce.

I seguenti esempi di codice inizializzano una conversazione in streaming con un bot Amazon Lex V2 e il tuo computer locale. È possibile modificare il codice per soddisfare le esigenze specifiche.

Il codice seguente è un esempio di richiesta che utilizza ilAWS SDK for Java per avviare la connessione a un bot e configurare i dettagli e le credenziali del bot.

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

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

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

/**
 * The following code creates a connection with the Amazon Lex bot and configures the
 bot details and credentials.
 * Prerequisite: To use this example, you must be familiar with the Reactive streams
 programming model.
 * For more information, see
 * https://github.com/reactive-streams/reactive-streams-jvm.
 * This example uses AWS SDK for Java for Amazon Lex V2.
 * <p>
 * The following sample application interacts with an Amazon Lex bot with the streaming
 API. It uses the Audio
 * conversation mode to return audio responses to the user's input.
 * <p>
```

```
 * The code in this example accomplishes the following:
 * <p>
 * 1. Configure details about the conversation between the user and the Amazon Lex bot.
 These details include the conversation mode and the specific bot the user is speaking
 with.
 * 2. Create an events publisher that passes the audio events to the Amazon Lex bot
 after you establish the connection. The code we provide in this example tells your
 computer to pick up the audio from
 * your microphone and send that audio data to Amazon Lex.
 * 3. Create a response handler that handles the audio responses from the Amazon Lex
 bot and plays back the audio to you.
 */
public class LexBidirectionalStreamingExample {

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

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

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


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

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

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

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

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

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

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

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

```
        });

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

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

Il codice seguente è un esempio di richiesta che utilizza ilAWS SDK for Java per inviare eventi al bot. Il codice in questo esempio utilizza il microfono del computer per inviare eventi audio.

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

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

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

```
    private AudioEventsSubscription audioEventsSubscription;

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

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

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

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

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

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

Il codice seguente è un esempio di richiesta che utilizza ilAWS SDK for Java per gestire le risposte del bot. Il codice in questo esempio configura Amazon Lex V2 per riprodurti una risposta audio.

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

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

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

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

    private final EventsPublisher eventsPublisher;

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


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

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

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

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

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

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

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

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

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

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

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

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

                           // Inform the Amazon Lex bot that the playback has
finished.
                           eventsPublisher.playbackFinished();
                           if (isDialogStateClosed) {
                               lastBotResponsePlayedBack = true;
                           }
                       }
                   });
                   audioPlayer.play();
               } catch (JavaLayerException e) {
                   throw new RuntimeException("got an exception when using audio
player", e);
               }
           });
```

```
        }

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

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

}
```

Per configurare un bot in modo che risponda agli eventi di input con audio, devi prima iscriverti agli eventi audio di Amazon Lex V2 e quindi configurare il bot per fornire una risposta audio agli eventi di input dell'utente.

Il codice seguente è unAWS SDK for Java esempio di sottoscrizione agli eventi audio da Amazon Lex V2.

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

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lexruntimev2.model.AudioInputEvent;
```

```
import software.amazon.awssdk.services.lexruntimev2.model.ConfigurationEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DisconnectionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackCompletionEvent;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

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

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

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


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

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

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

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

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

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

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

        ConfigurationEvent configurationEvent = StartConversationRequestEventStream
                .configurationEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
```

```
                .responseContentType(RESPONSE_TYPE)
                .build();

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

    public void disconnect() {

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

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

        eventWriter.writeDisconnectEvent(disconnectionEvent);

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

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

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

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

        eventWriter.writePlaybackFinishedEvent(playbackCompletionEvent);
```

```
    }

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

        private boolean conversationConfigured;

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

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

        public void writeConfigurationEvent(ConfigurationEvent configurationEvent) {
            eventQueue.add(configurationEvent);
        }

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

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

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

        @Override
        public void run() {
            try {

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

                    if (currentDemand > 0) {
                        // Try to read from queue of events.
```

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

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

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

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

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

                    AudioInputEvent audioInputEvent =
StartConversationRequestEventStream
                            .audioInputEventBuilder()

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

                    //System.out.println("sending audio event:" + audioInputEvent);
                    subscriber.onNext(audioInputEvent);
```

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

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

L'AWS SDK for Javaesempio seguente configura il bot Amazon Lex V2 per fornire una risposta audio agli eventi di input.

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

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

public class AudioResponse extends InputStream{

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

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

    private volatile boolean closed;

    @Override
    public int read() throws IOException {
```

```
        try {
            Optional<Integer> maybeInt;
            while (true) {
                maybeInt = Optional.ofNullable(this.byteQueue.poll(POLL_INTERVAL_MS,
 TimeUnit.MILLISECONDS));

                // If we get an integer from the queue, return it.
                if (maybeInt.isPresent()) {
                    return maybeInt.get();
                }

                // If the stream is closed and there is nothing queued up, return -1.
                if (this.closed) {
                    return -1;
                }
            }
        } catch (InterruptedException e) {
            throw new IOException(e);
        }
    }

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

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

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

Codifica del flusso di eventi

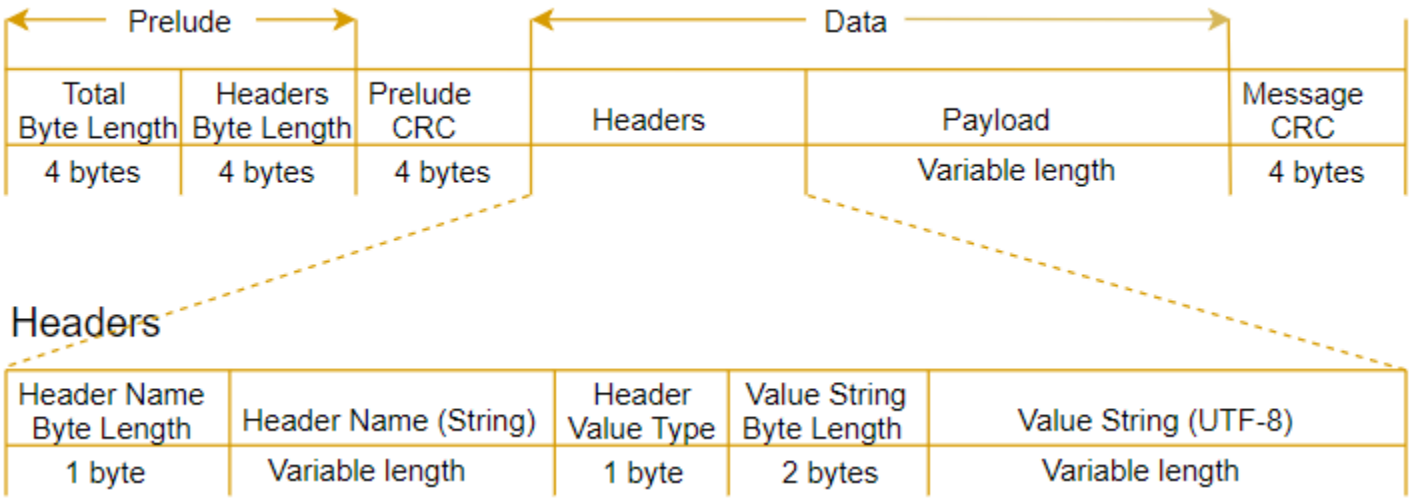
La codifica del flusso di eventi offre la comunicazione bidirezionale tramite messaggi tra un client e un server. I frame di dati inviati al servizio di streaming Amazon Lex V2 sono codificati in questo formato. Anche la risposta di Amazon Lex V2 utilizza questa codifica.

Ogni messaggio è composto da due sezioni: è costituita da due sezioni: l'introduzione e i dati. La sezione Prelude contiene la lunghezza totale in byte del messaggio e la lunghezza in byte combinata di tutte le intestazioni. La sezione dati contiene le intestazioni e un payload.

Ogni sezione termina con un checksum CRC intero big-endian a 4 byte. Il checksum CRC del messaggio include la sezione preludio e la sezione dati. Amazon Lex V2 utilizza CRC32 (spesso indicato come GZIP CRC32) per calcolare entrambi i CRC. Per ulteriori informazioni su CRC32, consultare la specifica versione 4.3 del formato di file GZIP.

La lunghezza totale del messaggio, inclusi introduzione ed entrambi i checksum, è pari a 16 byte.

Il seguente diagramma mostra i componenti che compongono un messaggio e un'intestazione. Sono presenti più intestazioni per messaggio.



Ogni messaggio contiene i seguenti componenti:

- Introduzione: sempre una dimensione fissa di 8 byte, due campi di 4 byte ciascuno.
  - Primi 4 byte: la lunghezza totale dei byte. Si tratta della lunghezza in byte interi big-endian dell'intero messaggio, incluso il campo di 4 byte.

    - Secondi 4 byte: la lunghezza dei byte delle intestazioni. Si tratta della lunghezza in byte interi big-endian della porzione delle intestazioni del messaggio, escluso il campo della lunghezza delle intestazioni.
- CRC dell'introduzione: il checksum del CRC da 4 byte della porzione dell'introduzione del messaggio, escluso il CRC stesso. Il preludio ha un CRC separato dal messaggio CRC per garantire che Amazon Lex V2 sia in grado di rilevare immediatamente le informazioni danneggiate sulla lunghezza dei byte senza causare errori come sovraccarichi del buffer.
- Intestazioni: i metadati per l'annotazione del messaggio, come il tipo di messaggio, il tipo di contenuto e così via. I messaggi hanno più intestazioni, che sono coppie chiave-valore in cui la chiave è una stringa UTF-8. Le intestazioni possono essere visualizzate in qualsiasi ordine nella relativa porzione del messaggio e vengono visualizzate una sola volta. Per i tipi di intestazione richiesti, consulta le seguenti sezioni.
- Payload: il contenuto audio o testuale inviato ad Amazon Lex.
- CRC del messaggio: il checksum CRC da 4 byte dall'inizio del messaggio all'inizio del checksum; Ciò include tutto il messaggio tranne il CRC stesso.

Ogni intestazione contiene i seguenti componenti. Sono presenti più intestazioni per frame.

- Lunghezza in byte del nome dell'intestazione: la lunghezza dei byte del nome dell'intestazione.
- Nome dell'intestazione: il nome dell'intestazione che indica il tipo di intestazione. Per i valori validi, consulta le seguenti descrizioni di frame.
- Tipo di valore dell'intestazione: un'enumerazione che indica il tipo di valore dell'intestazione.
- Lunghezza dei byte della stringa di valori: la lunghezza dei byte della stringa di valori dell'intestazione.
- Valore dell'intestazione: il valore della stringa dell'intestazione. I valori validi per questo campo dipendono dal tipo di intestazione. Per i valori validi, consulta le seguenti descrizioni di frame.

Consentire al bot di essere interrotto dall'utente

Quando avvii un flusso audio bidirezionale tra un bot Amazon Lex V2 e la tua applicazione, puoi configurare il bot in modo che ascolti gli input dell'utente mentre invia un prompt. In questo modo l'utente può interrompere il prompt prima che il bot abbia terminato la riproduzione. È possibile utilizzare questa configurazione in situazioni in cui l'utente potrebbe già conoscere la risposta a una domanda, ad esempio quando gli viene richiesto di fornire un codice CVV.

Un bot sa quando l'utente interrompe un prompt quando rileva l'input dell'utente prima che l'applicazione possa restituire un`PlaybackCompletion` evento. Quando l'utente interrompe un bot, il bot invia un`PlaybackInterruptionEvent`.

Per impostazione predefinita, l'utente può interrompere qualsiasi prompt che il bot sta trasmettendo all'applicazione. Puoi modificare questa impostazione nella console Amazon Lex V2.

È possibile modificare il modo in cui un utente può rispondere a un prompt modificando uno slot. Uno slot fa parte di un intento ed è il mezzo con cui l'utente ti fornisce le informazioni che desideri. In ogni slot viene richiesto all'utente di fornire tali informazioni. Per ulteriori informazioni sugli slot, consultaCome funziona.

Per modificare se l'utente può interrompere un prompt (console)

1. AccediAWS Management Console e apri la console Amazon Lex V2 nella console Amazon Lex V2.
2. In Bot, seleziona un bot.
3. In Lingua, seleziona la lingua del bot.
4. Scegli Visualizza intenti.
5. Seleziona l'intento .
6. Per le slot, scegli uno slot.
7. In Opzioni avanzate, scegli Slot prompts.
8. Scegli Altre opzioni di richiesta.
9. Seleziona o deseleziona Gli utenti possono interrompere il prompt durante la lettura.

Puoi testare questa funzionalità creando un bot con due slot e specificando che gli utenti non possono interrompere un prompt per uno slot. Se si interrompe un prompt interrompibile, il bot invia un evento di interruzione della riproduzione. Se si interrompe un programma ininterrotto, il prompt continua a essere riprodotto.

Abilitare il bot ad attendere che l'utente fornisca ulteriori informazioni

Quando avvii uno stream bidirezionale da un bot Amazon Lex V2 alla tua applicazione, puoi configurare il bot in modo che attenda che l'utente fornisca informazioni aggiuntive. Ci sono

circostanze in cui un utente potrebbe non essere pronto a rispondere a un prompt. Ad esempio, un utente potrebbe non essere pronto a fornire i dati della propria carta di credito perché il portafoglio si trova in un'altra stanza.

Utilizzando il comportamento Wait and continue del bot Amazon Lex V2, gli utenti possono pronunciare frasi come «aspetta un secondo» per far sì che il bot attenda che trovi le informazioni e le fornisca. Quando abiliti questo comportamento, il bot invia promemoria periodici all'utente per fornire le informazioni. Non restituisce gli eventi di trascrizione perché non ci sono espressioni dell'utente da trascrivere.

Il bot Amazon Lex V2 gestisce automaticamente una conversazione di streaming. Non hai bisogno di scrivere codice aggiuntivo per abilitare questa funzionalità. Quando a un bot viene richiesto di attendere dall'utente, il`state` of`Intent` is`Waiting` e il`type` of the`DialogAction` is`ElicitSlot`. È possibile utilizzare queste informazioni per personalizzare l'applicazione in base alle proprie esigenze. Ad esempio, puoi configurare l'applicazione per riprodurre musica quando l'utente cerca la propria carta di credito.

Abilita il comportamento di attesa e continua per un singolo slot. Per ulteriori informazioni sugli slot, consultaCome funziona.

Per abilitare attendi e continua

1. AccediAWS Management Console e apri la console Amazon Lex V2 nella console Amazon Lex V2.
2. In Bot, seleziona un bot.
3. In Lingua, seleziona la lingua del bot.
4. Scegli Visualizza intenti.
5. Seleziona l'intento .
6. In Slot, scegli uno slot.
7. In Opzioni avanzate, scegli Aspetta e continua.
8. In Attendi e continua specifica i seguenti campi:

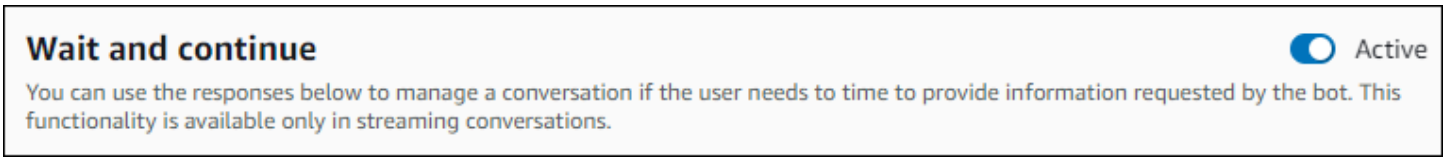
    - Risposta quando l'utente desidera che il bot attenda: questo è il modo in cui il bot risponde quando l'utente gli chiede di attendere le informazioni aggiuntive.
    - Risposta se l'utente ha bisogno che il bot continui ad aspettare: questa è la risposta che il bot invia per ricordare all'utente che è ancora in attesa delle informazioni. Puoi modificare la frequenza con cui il bot ricorda all'utente.

- Risposta quando l'utente desidera continuare: questa è la risposta del bot quando l'utente ha le informazioni richieste.

Per ogni risposta del bot, puoi fornire più varianti della risposta e una viene presentata all'utente a caso. Puoi anche scegliere se queste risposte possono essere interrotte dall'utente.

Per testare la funzionalità wait and continue, configura il bot in modo che attenda l'input dell'utente e avvia uno stream verso un bot Amazon Lex V2. Per informazioni sullo streaming verso un bot, consultaUtilizzo dell'API per avviare una conversazione di streaming.

Potrebbe essere necessario disattivare l'attesa e continuare le risposte. Usa l'interruttore Attivo per impostare se utilizzare o meno le risposte di attesa e continua.



Configurazione degli aggiornamenti sullo stato di avanzamento dell'evasione

Quando viene chiamata la funzione Lambda di adempimento per un intento, il bot non invia una risposta fino al completamento della funzione. Se la funzione Lambda impiega più di qualche secondo per essere completata, l'utente potrebbe pensare che il bot non risponda. Per risolvere questo problema, puoi configurare il bot in modo che invii aggiornamenti all'utente mentre la funzione Lambda di adempimento è in esecuzione, in modo che l'utente sappia che il bot sta ancora lavorando sulla sua richiesta.

Quando aggiungi aggiornamenti di adempimento a un intento, il bot risponde all'inizio dell'evasione e periodicamente mentre l'adempimento è in corso. Quando si configura la risposta di avvio, è possibile specificare un ritardo prima che il bot invii la risposta. In questo modo, puoi supportare i casi in cui l'adempimento non si concluda in tempi relativamente brevi. Quando si configura una risposta di aggiornamento, si specifica la frequenza con cui si desidera che gli aggiornamenti vengano inviati. È inoltre possibile configurare un timeout per limitare il tempo di esecuzione della funzione di adempimento.

Puoi anche aggiungere risposte successive all'evasione a un bot. Ciò consente al bot di inviare una risposta diversa a seconda che l'adempimento abbia un errore o al timeout.

Gli aggiornamenti relativi agli adempimenti vengono utilizzati solo quando si interagisce con un bot che utilizza l'StartConversationoperazione. Puoi utilizzare l'aggiornamento successivo all'evasione quando interagisci con il bot utilizzando le RecognizeUtteranceoperazioni StartConversationRecognizeText, e

Aggiornamenti relativi agli adempimenti

Gli aggiornamenti relativi agli adempimenti vengono inviati mentre la funzione Lambda soddisfa un obiettivo. Quando attivi gli aggiornamenti degli adempimenti, fornisci una risposta iniziale inviata all'inizio dell'evasione e una risposta di aggiornamento inviata periodicamente mentre l'evasione è in corso.

Quando si specifica una risposta di aggiornamento, si specifica anche un timeout che determina per quanto tempo può essere eseguita la funzione di adempimento. È possibile specificare una durata del timeout fino a 15 minuti (900 secondi).

Se disattivi gli aggiornamenti di evasione`active` impostando su false nella console o utilizzando l'UpdateIntentoperazione CreateIntentor, il timeout specificato per gli aggiornamenti di evasione degli ordini non viene utilizzato e viene invece utilizzato il timeout predefinito di 30 secondi.

Se il timeout della funzione di evasione ordini si interrompe, Amazon Lex V2 esegue una delle tre operazioni seguenti:

- La risposta post-evasione è configurata e attiva: restituisce la risposta al timeout.
- La risposta post-evasione è configurata e non attiva: restituisce un'eccezione.
- La risposta post-evasione non è configurata: restituisce un'eccezione.

Inizia la risposta

Amazon Lex V2 restituisce la risposta iniziale quando la funzione di adempimento Lambda viene chiamata durante una conversazione in streaming. In genere indica all'utente che il raggiungimento dell'intento richiede del tempo e che deve attendere. La risposta iniziale non viene restituita quando si utilizzano le`RecognizeUtterance` operazioni`RecognizeText` or.

È possibile specificare fino a cinque messaggi di risposta. Amazon Lex V2 sceglie uno dei messaggi da riprodurre per l'utente.

È possibile configurare un ritardo tra il momento in cui viene chiamata la funzione Lambda e il momento in cui viene restituita la risposta di avvio. La risposta iniziale non viene restituita se la funzione Lambda completa il suo lavoro prima che il ritardo sia completo.

È possibile utilizzare l'`active`interruttore nella console o nella FulfillmentUpdatesSpecificationstruttura per attivare e disattivare la risposta di avvio. Quando`active` è false, la risposta iniziale non viene riprodotta.

Aggiorna risposta

Amazon Lex restituisce periodicamente la risposta all'aggiornamento durante una conversazione in streaming mentre è in esecuzione la funzione di adempimento Lambda. La risposta all'aggiornamento non viene riprodotta quando si utilizzano le`RecognizeUtterance` operazioni`RecognizeText` o. È possibile configurare la frequenza con cui viene riprodotta la risposta all'aggiornamento. Ad esempio, puoi riprodurre una risposta di aggiornamento ogni 30 secondi durante l'esecuzione della funzione di evasione degli ordini per far sapere all'utente che il processo è in esecuzione e che deve continuare ad attendere.

È possibile specificare fino a cinque messaggi di aggiornamento. Amazon Lex V2 sceglie un messaggio da trasmettere all'utente. L'utilizzo di più messaggi evita che gli aggiornamenti siano ripetitivi.

Se l'utente fornisce input tramite voce, DTMF o testo mentre la funzione di gestione Lambda è in esecuzione, Amazon Lex V2 restituisce la risposta di aggiornamento all'utente.

Se la funzione Lambda completa il suo lavoro prima della fine del primo periodo di aggiornamento, la risposta all'aggiornamento non viene restituita.

Puoi utilizzare l'`active`interruttore nella console o nella FulfillmentUpdatesSpecificationstruttura per attivare e disattivare la risposta all'aggiornamento. Quando`active` è false, la risposta all'aggiornamento non viene restituita.

Risposta successiva all'evasione

Amazon Lex V2 restituisce una risposta post-evasione al termine della funzione di evasione ordini. Una risposta successiva all'evasione può essere utilizzata per soddisfare qualsiasi intento, non solo durante lo streaming di conversazioni. La risposta successiva all'evasione consente all'utente di sapere che la funzione è completa e il risultato.

Puoi utilizzare l'`active`interruttore nella console o nella PostFulfillmentStatusSpecificationstruttura per attivare e disattivare la risposta dopo l'evasione. Quando`active` è false, la risposta non viene riprodotta.

Esistono tre tipi di risposte successive all'adempimento:

- Successo: restituito quando la funzione Lambda di adempimento completa il suo lavoro con successo. Se le risposte successive all'evasione non sono attive. Amazon Lex V2 esegue l'azione configurata successiva.
- Timeout: restituito se la funzione Lambda non completa il suo lavoro prima che sia trascorso il periodo di timeout configurato. Se le risposte successive all'evasione non sono attive, Amazon Lex V2 restituisce un'eccezione.
- Errore: restituito quando la funzione Lambda restituisce lo stato`Failed` nella risposta o quando Amazon Lex V2 rileva un errore mentre soddisfa l'intento. Se le risposte successive all'evasione non sono attive, Amazon Lex V2 restituisce un'eccezione.

Puoi specificare un massimo di cinque messaggi per ciascun tipo. Amazon Lex V2 sceglie uno dei messaggi da riprodurre per l'utente.

A differenza delle risposte relative all'adempimento e all'aggiornamento, le risposte successive all'adempimento vengono riprodotte per le conversazioni sia in streaming sia non in streaming.

Puoi anche ignorare questi messaggi configurando la funzione Lambda per restituire un messaggio successivo all'evasione.

> **Note**
>
> Se l'intento ha una risposta conclusiva, viene restituito dopo la risposta successiva all'evasione.

Esempio successivo all'evasione

Per comprendere meglio la risposta post-evasione, prendiamo ad esempio un *`BookTrip`*bot, creato per aiutare a pianificare un viaggio, con un *`BookFlight`*intento, configurato con una funzione Lambda di evasione ordini che prenota il volo del cliente con una compagnia aerea. Una volta attivati gli slot per *`BookFlight`*, Amazon Lex V2 richiama la funzione di adempimento Lambda. Durante questo processo adempimento può verificarsi uno dei seguenti tre risultati:

- Successo: il volo è stato prenotato con successo.
- Timeout: il processo di prenotazione richiede più tempo del tempo di esecuzione di Lambda per l'evasione degli ordini configurato (ad esempio, se la compagnia aerea non può essere contattata entro il tempo assegnato).
- Fallimento: la prenotazione non va a buon fine per un altro motivo.

Puoi sfruttare la risposta post-evasione per fornire una risposta più significativa ai tuoi clienti in ciascuna di queste situazioni. Di seguito sono riportati alcuni esempi per ciascuna situazione:

- Risposta riuscita: «Siamo riusciti a prenotare con successo il tuo biglietto e ti abbiamo inviato un'email di conferma. Non esitate a contattarci utilizzando le informazioni di contatto fornite nell'e-mail in caso di domande.»
- Risposta al timeout: «A causa del traffico intenso sui nostri sistemi, la prenotazione del biglietto richiede più tempo del previsto. Abbiamo la tua richiesta in coda e ti abbiamo inviato un'e-mail con il numero di riferimento corrispondente a questa richiesta. Una volta prenotato il biglietto, ti invieremo una conferma della prenotazione. Non esitate a contattarci utilizzando le informazioni di contatto fornite nell'e-mail in caso di domande.»

  > **Note**
  >
  > Se non si configura un messaggio di timeout, Lex genera un errore 4XX corrispondente al caso d'uso.

- Risposta fallita: «Purtroppo non siamo riusciti a prenotare il tuo biglietto. Abbiamo inviato un'e-mail con i dettagli relativi al problema riscontrato durante la prenotazione.»

Configurazione dei timeout per l'acquisizione dell'input dell'utente

L'API di streaming Amazon Lex V2 consente a un bot di rilevare automaticamente le espressioni nell'input dell'utente. Quando si crea un intento o uno slot, è possibile configurare aspetti di un'enunciazione, come la durata massima di un'enunciazione, il timeout in attesa dell'input dell'utente o il carattere finale per l'input DTMF. Puoi personalizzare il comportamento di un bot in base al tuo caso d'uso. Ad esempio, puoi limitare a 16 il numero di cifre di una carta di credito.

È inoltre possibile configurare i timeout tramite gli attributi di sessione quando si avvia una conversazione con un bot e sovrascriverli nella funzione Lambda, se necessario.

Le chiavi di configurazione di un attributo utilizzano la sintassi seguente:

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

`InputType` può essere **`audio`**, **`dtmf`** o **`text`**.

È possibile configurare le impostazioni predefinite per tutti gli intenti o gli slot in un bot specificando***** come intento o nome dello slot. Qualsiasi impostazione specifica dell'intento o dello slot ha la precedenza sulle impostazioni predefinite.

Amazon Lex V2 fornisce attributi di sessione predefiniti per gestire il modo in cui StartConversationle operazioni funzionano con input di testo, voce o DTMF al bot. Tutti gli attributi predefiniti sono nel namespace `x-amz-lex`.

È possibile configurare le impostazioni predefinite per tutti gli intenti, gli slot o i sottoslot in un bot specificando* come intento o nome dello slot. Qualsiasi impostazione specifica dell'intento o dello slot ha la precedenza sulle impostazioni predefinite. Usa questi schemi per tutti i timeout seguenti.

Per il sottoslot di uno slot composito è possibile separare per`.`. Ad esempio:

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

Espressione	Scenario
Intento: Slot. SubSlot	Applicabile solo allo slot secondario denominat o "SubSlot 'all'interno dello slot composito denominato' Slot '
Intento: Slot. *	Applicabile a qualsiasi slot secondario all'inter no dello slot composito denominato «Slot»
Intento: *. SubSlot	Applicabile solo allo slot secondario denominat o 'SubSlot' all'interno di qualsiasi slot composito
Intento: * . *	Applicabile a qualsiasi slot secondario all'inter no di qualsiasi slot composito

Interrompi il comportamento

Puoi impostare il comportamento di interruzione per il bot. L'attributo è definito da Amazon Lex V2.

Consenti l'interruzione

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

Definisce se l'utente può interrompere il prompt riprodotto dal bot Amazon Lex V2. Puoi disattivarlo in modo selettivo.

Impostazione predefinita: True

Timeout per l'input vocale

Puoi impostare i valori di timeout per l'interazione vocale con il tuo bot utilizzando gli attributi della sessione. Gli attributi sono definiti da Amazon Lex V2. Questi attributi consentono di specificare per quanto tempo Amazon Lex V2 aspetta che un cliente finisca di parlare prima di raccogliere la voce in ingresso.

Tutti questi attributi si trovano nel`x-amz-lex:audio` namespace.

Lunghezza massima dell'enunciato

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

Definisce il tempo di attesa di Amazon Lex V2 prima che l'input vocale venga troncato e il riconoscimento vocale venga restituito all'applicazione. Puoi aumentare la lunghezza dell'input quando prevedi risposte lunghe o se desideri dare ai clienti più tempo per fornire informazioni.

Impostazione predefinita: 13.000 millisecondi (13 secondi). Il valore massimo è 15.000 millisecondi (15 secondi)

Se si imposta l'`max-length-ms`attributo su più di 15.000 millisecondi, il valore predefinito sarà 15.000 millisecondi.

Timeout vocale

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

Quanto tempo aspetta un bot prima di dare per scontato che il cliente non parlerà. È possibile aumentare il tempo in situazioni in cui il cliente potrebbe aver bisogno di più tempo per trovare o richiamare le informazioni prima di parlare. Ad esempio, potresti voler dare ai clienti il tempo di estrarre la carta di credito in modo che possano inserire il numero.

Impostazione predefinita: 4.000 millisecondi (4 secondi)

Timeout silenzio

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

Quanto tempo aspetta un bot dopo che il cliente smette di parlare per supporre che l'enunciato sia terminato. È possibile aumentare il tempo in situazioni in cui sono previsti periodi di silenzio fornendo input.

Impostazione predefinita: 600 millisecondi (0,6 secondi)

Consenti input audio

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

È possibile abilitare questo attributo in modo che il bot accetti l'input dell'utente solo tramite la modalità audio. Il bot non accetterà l'ingresso audio se questo flag è impostato su false. Il valore è impostato su true per impostazione predefinita.

Impostazione predefinita: True

Timeout per l'immissione di testo

Usa il seguente attributo di sessione per specificare come si comporta il tuo bot con la modalità conversazione testuale.

Questo attributo si trova nel`x-amz-lex:text` namespace.

Soglia di timeout di avvio

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

Quanto tempo attende il bot prima di richiedere nuovamente a un cliente l'immissione di testo. Puoi aumentare il tempo in situazioni in cui desideri concedere al cliente più tempo per trovare o

richiamare le informazioni prima di fornire un input di testo. Ad esempio, lo puoi fare in modo che i clienti abbiano più tempo per trovare i dettagli del loro ordine. In alternativa, puoi ridurre la soglia per avvisare i clienti prima.

Impostazione predefinita: 30.000 millisecondi (30 secondi)

Configurazione per l'ingresso DTMF

Utilizza i seguenti attributi di sessione per specificare in che modo il bot Amazon Lex V2 risponde all'ingresso DTMF quando utilizza una conversazione audio.

Tutti questi attributi si trovano nel`x-amz-lex:dtmf` namespace.

Carattere di cancellazione

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

Il carattere DTMF che cancella le cifre DTMF accumulate e termina immediatamente l'input.

Predefinito: *

Carattere finale

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

Il carattere DTMF che termina immediatamente l'input. Se l'utente non preme questo carattere, l'input termina dopo il timeout di fine.

Predefinito: #

Timeout di fine

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

Quanto tempo deve attendere il bot dall'ultimo input di caratteri DTMF prima di presumere che l'input sia stato concluso.

Impostazione predefinita: 5000 millisecondi (5 secondi)

Numero massimo di cifre DTMF per enunciato

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

Il numero massimo di cifre DTMF consentite in un enunciato. Ad esempio, è possibile impostare questo valore su 16 per limitare il numero di caratteri che possono essere inseriti per il numero di una carta di credito. Questo valore non può essere aumentato.

Predefinito: 1024 caratteri

Consenti input DTMF

Puoi impostare il tipo di input che il bot può accettare utilizzando gli attributi della sessione. Gli attributi sono definiti da Amazon Lex V2.

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

È possibile abilitare questo attributo in modo che il bot accetti l'input dell'utente tramite la modalità DTMF. Il bot non accetterà input DTMF se questo flag è impostato su false. Il valore è impostato su true per impostazione predefinita.

Impostazione predefinita: True

Importazione ed esportazione

Puoi esportare una definizione di bot, le impostazioni locali del bot o un vocabolario personalizzato e quindi importarlo nuovamente per creare una nuova risorsa o sovrascrivere una risorsa esistente in un account. AWS Ad esempio, puoi esportare un bot da un account di prova e quindi creare una copia del bot nel tuo account di produzione. Puoi anche copiare un bot da una AWS regione all'altra.

È possibile modificare le risorse della risorsa esportata prima di importarla. Ad esempio, puoi esportare un bot e quindi modificare il file JSON per uno slot per aggiungere o rimuovere le enunciazioni di elicitazione dei valori di slot da uno slot specifico. Dopo aver modificato la definizione, è possibile importare il file modificato.

Argomenti

Exporting (Esportazione)

Puoi esportare un bot, le impostazioni locali del bot o un vocabolario personalizzato utilizzando la console o l'operazione. `CreatExport` Specificate la risorsa da esportare e potete fornire una password opzionale per proteggere il file.zip quando avviate un'esportazione. Dopo aver scaricato il file.zip, è necessario utilizzare la password per accedere al file prima di poterlo utilizzare. Per ulteriori informazioni, consulta Utilizzo di una password durante l'importazione o l'esportazione.

L'esportazione è un'operazione asincrona. Una volta avviata l'esportazione, puoi utilizzare la console o l'`DescribeExport`operazione per monitorare l'avanzamento dell'esportazione. Una volta completata l'esportazione, la console o l'`DescribeExport`operazione mostra lo stato di `COMPLETED` e la console scarica il file.zip di esportazione nel browser. Se utilizzi l'`DescribeExport`operazione, Amazon Lex V2 fornisce un URL Amazon S3 prefirmato in cui puoi scaricare i risultati dell'esportazione. L'URL di download è disponibile solo per cinque minuti, ma puoi ottenere un nuovo URL richiamando nuovamente l'`DescribeExport`operazione.

Puoi visualizzare la cronologia delle esportazioni di una risorsa con la console o con l'`ListExports`operazione. I risultati mostrano le esportazioni insieme al loro stato attuale. Un'esportazione è disponibile nella cronologia per sette giorni.

Quando si esporta la `Draft` versione di un bot o delle impostazioni locali di un bot, è possibile che la definizione nel file JSON non sia coerente perché la `Draft` versione di un bot o delle impostazioni locali del bot può essere modificata mentre è in corso un'esportazione. Se la `Draft` versione viene modificata durante l'esportazione, è possibile che le modifiche non vengano incluse nel file di esportazione.

Quando esporti le impostazioni locali di un bot, Amazon Lex esporta tutte le informazioni che definiscono la lingua, il vocabolario personalizzato, gli intenti, i tipi di slot e gli slot.

Quando esporti un bot, Amazon Lex esporta tutte le impostazioni locali definite per il bot, inclusi gli intenti, i tipi di slot e gli slot. I seguenti elementi non vengono esportati con un bot:

- Alias del bot
- Ruolo ARN associato a un bot
- Tag associati a bot e alias di bot
- Hook del codice Lambda associati a un alias bot

L'ARN e i tag del ruolo vengono inseriti come parametri di richiesta quando si importa un bot. È necessario creare alias per i bot e assegnare i code hook Lambda dopo l'importazione, se necessario.

È possibile rimuovere un'esportazione e il file.zip associato utilizzando la console o l'`DeleteExport`operazione.

Per un esempio di esportazione di un bot tramite la console, vediEsportazione di un bot (console).

Autorizzazioni IAM necessarie per l'esportazione

Per esportare bot, impostazioni locali dei bot e vocabolari personalizzati, l'utente che esegue l'esportazione deve disporre delle seguenti autorizzazioni IAM.

API	• Azioni IAM richieste	Risorsa
CreateExport	• CreateExport	Bot

API	• Azioni IAM richieste	Risorsa
UpdateExport	• UpdateExport	Bot
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	Bot
DescribeExportper vocabolari personalizzati	• DescribeExport • DescribeCustomVocabulary	bot
DeleteExport	• DeleteExport	Bot
ListExports	• ListExports	*

Per un esempio di policy IAM, consultare Consenti a un utente di esportare bot e impostazioni locali dei bot .

Esportazione di un bot (console)

Puoi esportare un bot dall'elenco dei bot, dall'elenco delle versioni o dalla pagina dei dettagli della versione. Quando scegli una versione, Amazon Lex V2 esporta quella versione. Le seguenti istruzioni presuppongono che inizi a esportare il bot dall'elenco dei bot, ma quando inizi con una versione i passaggi sono gli stessi.

Per esportare un bot utilizzando la console

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Dall'elenco dei bot, scegli il bot da esportare.
3. Da Azione, scegli Esporta.
4. Scegli la versione del bot, la piattaforma e il formato di esportazione.
5. (Facoltativo) Inserisci una password per il file.zip. Fornire una password aiuta a proteggere l'archivio di output.
6. Scegli Export (Esporta).

Dopo aver avviato l'esportazione, torni all'elenco dei bot. Per monitorare l'avanzamento dell'esportazione, utilizza l'elenco della cronologia delle importazioni/esportazioni. Quando lo stato dell'esportazione è Completo, la console scarica automaticamente il file.zip sul tuo computer.

Per scaricare nuovamente l'esportazione, nell'elenco di importazione/esportazione, scegli l'esportazione, quindi scegli Scarica. Puoi fornire una password per il file.zip scaricato.

Per esportare una lingua del bot

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Dall'elenco dei bot, scegli il bot di cui desideri esportare la lingua.
3. Da Aggiungi lingue, scegli Visualizza lingue.
4. Dall'elenco Tutte le lingue, scegli la lingua da esportare.
5. Da Azione, scegli Esporta.
6. Scegli la versione, la piattaforma e il formato del bot.
7. (Facoltativo) Inserisci una password per il file.zip. Fornire una password aiuta a proteggere l'archivio di output.
8. Scegli Export (Esporta).

Dopo aver avviato l'esportazione, torni all'elenco delle lingue. Per monitorare l'avanzamento dell'esportazione, utilizza l'elenco della cronologia delle importazioni/esportazioni. Quando lo stato dell'esportazione è Completo, la console scarica automaticamente il file.zip sul tuo computer.

Per scaricare nuovamente l'esportazione, nell'elenco di importazione/esportazione, scegli l'esportazione, quindi scegli Scarica. Puoi fornire una password per il file.zip scaricato.

Importing (Importazione)

Per utilizzare la console per importare un bot esportato in precedenza, le impostazioni locali del bot o un vocabolario personalizzato, devi fornire la posizione del file sul tuo computer locale e la password opzionale per sbloccare il file. Per un esempio, consulta Importazione di un bot (console).

Quando si utilizza l'API, l'importazione di una risorsa è un processo in tre fasi:

1. Crea un URL di caricamento utilizzando l'`CreateUploadUrl`operazione. Non è necessario creare un URL di caricamento quando si utilizza la console.
2. Carica il file.zip che contiene la definizione della risorsa.
3. Avvia l'importazione con l'`StartImport`operazione.

L'URL di caricamento è un URL Amazon S3 prefirmato con autorizzazione di scrittura. L'URL è disponibile per cinque minuti dopo la generazione. Se proteggi con password il file.zip, devi fornire la password all'avvio dell'importazione. Per ulteriori informazioni, consulta Utilizzo di una password durante l'importazione o l'esportazione.

Un'importazione è un processo asincrono. È possibile monitorare l'avanzamento di un'importazione utilizzando la console o l'`DescribeImport`operazione.

Quando importi un bot o un bot a livello locale, potrebbero verificarsi conflitti tra i nomi delle risorse nel file di importazione e i nomi delle risorse esistenti in Amazon Lex V2. Amazon Lex V2 è in grado di gestire il conflitto in tre modi:

- Errore in caso di conflitto: l'importazione si interrompe e non viene importata alcuna risorsa dal file.zip di importazione.
- Sovrascrivi: Amazon Lex V2 importa tutte le risorse dal file.zip di importazione e sostituisce qualsiasi risorsa esistente con la definizione del file di importazione.
- Aggiungi: Amazon Lex V2 importa tutte le risorse dal file.zip di importazione e le aggiunge a qualsiasi risorsa esistente con la definizione del file di importazione. Questa opzione è disponibile solo per le impostazioni locali del bot.

È possibile visualizzare un elenco delle importazioni in una risorsa utilizzando la console o l'`ListImports`operazione. Le importazioni rimangono nell'elenco per sette giorni. È possibile utilizzare la console o l'`DescribeImport`operazione per visualizzare i dettagli su un'importazione specifica.

È inoltre possibile rimuovere un'importazione e il file zip associato utilizzando la console o l'`DeleteImport`operazione.

Per un esempio di importazione di un bot tramite la console, vediImportazione di un bot (console).

Autorizzazioni IAM necessarie per l'importazione

Per importare bot, impostazioni locali dei bot e vocabolari personalizzati, l'utente che esegue l'importazione deve disporre delle seguenti autorizzazioni IAM.

API	Azioni IAM richieste	Risorsa
CreateUploadUrl	• CreateUploadUrl	*
StartImportper bot e bot locali	• StartImport • scopo: PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary	1. Per importare un nuovo bot: bot, bot alias. 2. Per sovrascrivere un bot esistente: bot. 3. Per importare una nuova lingua: bot.

API	Azioni IAM richieste	Risorsa
	• DeleteLocale • DeleteIntent • DeleteSlot • DeleteSlotType	
StartImportper vocabolari personalizzati	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	bot
DescribeImport	• DescribeImport	Bot
DeleteImport	• DeleteImport	Bot
ListImports	• ListImports	*

Per un esempio di policy IAM, consultare Consenti a un utente di importare bot e impostazioni locali dei bot .

Importazione di un bot (console)

Per importare un bot utilizzando la console

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Da Azione, scegli Importa.
3. Nel file di input, assegna un nome al bot e quindi scegli il file.zip che contiene i file JSON che definiscono il bot.
4. Se il file.zip è protetto da password, inserisci la password per il file.zip. La protezione dell'archivio con password è facoltativa, ma aiuta a proteggere i contenuti.
5. Crea o inserisci il ruolo IAM che definisce le autorizzazioni per il tuo bot.
6. Indica se il tuo bot è soggetto al Children's Online Privacy Protection Act (COPPA).

7. Fornisci un'impostazione di timeout di inattività per il tuo bot. Se non fornisci un valore, viene utilizzato il valore del file zip. Se il file.zip non contiene un'impostazione di timeout, Amazon Lex V2 utilizza l'impostazione predefinita di 300 secondi (cinque minuti).
8. (Facoltativo) Aggiungi tag per il tuo bot.
9. Scegli se avvisare della sovrascrittura dei bot esistenti con lo stesso nome. Se attivi gli avvisi, se il bot che stai importando sovrascrive un bot esistente, ricevi un avviso e il bot non viene importato. Se disabiliti gli avvisi, il bot importato sostituisce il bot esistente con lo stesso nome.
10. Seleziona Import (Importa).

Dopo aver avviato l'importazione, torni all'elenco dei bot. Per monitorare l'avanzamento dell'importazione, utilizza l'elenco della cronologia delle importazioni/esportazioni. Quando lo stato dell'importazione è Completo, puoi scegliere il bot dall'elenco dei bot per modificare o creare il bot.

Per importare un linguaggio bot

1. Accedi AWS Management Console e apri la console Amazon Lex V2 all'indirizzo https://console.aws.amazon.com/lexv2/home.
2. Dall'elenco dei bot, scegli il bot in cui desideri importare una lingua.
3. Da Aggiungi lingue, scegli Visualizza lingue.
4. Da Azione, scegli Importa.
5. In File di input, scegli il file che contiene la lingua da importare. Se hai protetto il file.zip, inserisci la password in Password.
6. In Lingua, scegli la lingua con cui importare. La lingua non deve necessariamente corrispondere a quella del file di importazione. È possibile copiare gli intenti da una lingua all'altra.
7. In Voice, scegli la voce di Amazon Polly da utilizzare per l'interazione vocale o scegli Nessuno per un bot di solo testo.
8. In Soglia del punteggio di confidenza, inserisci la soglia in cui Amazon Lex V2 inserisce il `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`, il o entrambi quando restituisce intenti alternativi.
9. Scegli se inviare un avviso sulla sovrascrittura di una lingua esistente. Se abiliti gli avvisi, se la lingua che stai importando sovrascrive una lingua esistente, ricevi un avviso e la lingua non viene importata. Se disabiliti gli avvisi, la lingua importata sostituisce la lingua esistente.
10. Scegliete Importa per iniziare a importare la lingua.

Dopo aver avviato l'importazione, si torna all'elenco delle lingue. Per monitorare l'avanzamento dell'importazione, utilizza l'elenco della cronologia delle importazioni/esportazioni. Quando lo stato dell'importazione è Completo, puoi scegliere la lingua dall'elenco dei bot per modificare o creare il bot.

Utilizzo di una password durante l'importazione o l'esportazione

Amazon Lex V2 può proteggere con password gli archivi di esportazione o leggere gli archivi di importazione protetti utilizzando la compressione standard di file.zip. Dovresti sempre proteggere con password i tuoi archivi di importazione ed esportazione.

Amazon Lex V2 invia il tuo archivio di esportazione a un bucket S3 ed è disponibile con un URL S3 prefirmato. L'URL è disponibile solo per cinque minuti. L'archivio è disponibile per chiunque abbia accesso all'URL di download. Per proteggere i dati nell'archivio, inserisci una password quando esporti la risorsa. Se devi recuperare l'archivio dopo la scadenza dell'URL, puoi utilizzare la console o l'`DescribeExport`operazione per ottenere un nuovo URL.

Se si perde la password per un archivio di esportazione, è possibile creare una nuova password per un file esistente scegliendo Scarica dalla tabella della cronologia di importazione/esportazione o utilizzando l'`UpdateExport`operazione. Se scegli Scarica nella tabella della cronologia per un'esportazione e non fornisci una password, Amazon Lex V2 scarica un file zip non protetto.

Formato JSON per l'importazione e l'esportazione

Puoi importare ed esportare bot, versioni locali dei bot o vocabolari personalizzati da Amazon Lex V2 utilizzando un file.zip che contiene strutture JSON che descrivono le parti della risorsa. Quando esporti una risorsa, Amazon Lex V2 crea il file.zip e lo rende disponibile utilizzando un URL prefirmato Amazon S3. Quando importi una risorsa, devi creare un file.zip che contenga le strutture JSON e caricarlo su un URL prefirmato S3.

Amazon Lex crea la seguente struttura di directory nel file.zip quando esporti un bot. Quando esporti le impostazioni locali di un bot, viene esportata solo la struttura sotto la lingua. Quando si esporta un vocabolario personalizzato, viene esportata solo la struttura sotto il vocabolario personalizzato.

```
BotName_BotVersion_ExportID_LexJson.zip
            -or-
BotName_BotVersion_LocaleId_ExportId_LEX_JSON.zip
        --> manifest.json
```

```
        --> BotName
        ----> Bot.json
        ----> BotLocales
        ------> Locale_A
        --------> BotLocale.json
        --------> Intents
        ----------> Intent_A
        ------------> Intent.json
        ------------> Slots
        --------------> Slot_A
        ----------------> Slot.json
        --------------> Slot_B
        ----------------> Slot.json
        ----------> Intent_B
                       ...
        --------> SlotTypes
        ----------> SlotType_A
        ------------> SlotType.json
        ----------> SlotType_B
                       ...
        --------> CustomVocabulary
        ------------> CustomVocabulary.json

        ------> Locale_B
                       ...
```

Struttura del file manifest

Il file manifest contiene i metadati per il file di esportazione.

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

Struttura dei file bot

Il file bot contiene le informazioni di configurazione per il bot.

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

Struttura dei file locali dei bot

Il file locale del bot contiene una descrizione delle impostazioni locali o della lingua di un bot. Quando esporti un bot, può esserci più di un file locale del bot nel file.zip. Quando esporti le impostazioni locali di un bot, nel file zip è presente una sola lingua.

```
{
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

Struttura del file di intento

Il file di intento contiene le informazioni di configurazione per un intento. Nel file.zip è presente un file di intenti per ogni intento in una lingua specifica.

Di seguito è riportato un esempio di struttura JSON per l'BookCarintento nel bot di esempioBookTrip. Per un esempio completo della struttura JSON per un intento, vedere l'CreateIntentoperazione.

```
{
    "name": "BookCar",
    "identifier": "891RWHHICO",
    "description": "Intent to book a car.",
```

```
    "parentIntentSignature": null,
    "sampleUtterances": [
        {
            "utterance": "Book a car"
        },
        {
            "utterance": "Reserve a car"
        },
        {
            "utterance": "Make a car reservation"
        }
    ],
    "intentConfirmationSetting": {
        "confirmationPrompt": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have you down for a {CarType} hire in
{PickUpCity} from {PickUpDate} to {ReturnDate}.  Should I book the reservation?"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "declinationResponse": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have cancelled your reservation in
progress."
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
```

```
            ]
        }
    },
    "intentClosingSetting": null,
    "inputContexts": null,
    "outputContexts": null,
    "kendraConfiguration": null,
    "dialogCodeHook": null,
    "fulfillmentCodeHook": null,
    "slotPriorities": [
        {
            "slotName": "DriverAge",
            "priority": 4
        },
        {
            "slotName": "PickUpDate",
            "priority": 2
        },
        {
            "slotName": "ReturnDate",
            "priority": 3
        },
        {
            "slotName": "PickUpCity",
            "priority": 1
        },
        {
            "slotName": "CarType",
            "priority": 5
        }
    ]
}
```

Struttura del file Slot

Il file slot contiene le informazioni di configurazione per uno slot in un intento. C'è un file slot nel file.zip per ogni slot definito per un intento in una lingua specifica.

L'esempio seguente è la struttura JSON di uno slot che consente al cliente di scegliere il tipo di auto che desidera noleggiare nell'BookCarintento del bot di BookTrip esempio. Per un esempio completo della struttura JSON per uno slot, vedere l'CreateSlotoperazione.

```
{
```

```
    "name": "CarType",
    "identifier": "KDHJWNGZGC",
    "description": "Type of car being reserved.",
    "multipleValuesSetting": {
        "allowMutlipleValues": false
    },
    "slotTypeName": "CarTypeValues",
    "obfuscationSetting": null,
    "slotConstraint": "Required",
    "defaultValueSpec": null,
    "slotValueElicitationSetting": {
        "promptSpecification": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "What type of car would you like to rent?  Our
 most popular options are economy, midsize, and luxury"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "sampleValueElicitingUtterances": null,
        "waitAndContinueSpecification": null,
    }
}
```

L'esempio seguente mostra la struttura JSON di uno slot composito.

```
{
  "name": "CarType",
  "identifier": "KDHJWNGZGC",
  "description": "Type of car being reserved.",
  "multipleValuesSetting": {
      "allowMutlipleValues": false
  },
  "slotTypeName": "CarTypeValues",
```

```
  "obfuscationSetting": null,
  "slotConstraint": "Required",
  "defaultValueSpec": null,
  "slotValueElicitationSetting": {
      "promptSpecification": {
          "messageGroupList": [
              {
                  "message": {
                      "plainTextMessage": {
                          "value": "What type of car would you like to rent?  Our most
popular options are economy, midsize, and luxury"
                      },
                      "ssmlMessage": null,
                      "customPayload": null,
                      "imageResponseCard": null
                  },
                  "variations": null
              }
          ],
          "maxRetries": 2
      },
      "sampleValueElicitingUtterances": null,
      "waitAndContinueSpecification": null,
  },
  "subSlotSetting": {
    "slotSpecifications": {
      "firstname": {
        "valueElicitationSetting": {
          "promptSpecification": {
            "allowInterrupt": false,
            "messageGroupsList": [
              {
                "message": {
                  "imageResponseCard": null,
                  "ssmlMessage": null,
                  "customPayload": null,
                  "plainTextMessage": {
                    "value": "please provide firstname"
                  }
                },
                "variations": null
              }
            ],
            "maxRetries": 2,
```

```
            "messageSelectionStrategy": "Random"
          },
          "defaultValueSpecification": null,
          "sampleUtterances": [
            {
              "utterance": "my name is {firstName}"
            }
          ],
          "waitAndContinueSpecification": null
        },
        "slotTypeId": "AMAZON.FirstName"
      },
      "eyeColor": {
        "valueElicitationSetting": {
          "promptSpecification": {
            "allowInterrupt": false,
            "messageGroupsList": [
              {
                "message": {
                  "imageResponseCard": null,
                  "ssmlMessage": null,
                  "customPayload": null,
                  "plainTextMessage": {
                    "value": "please provide eye color"
                  }
                },
                "variations": null
              }
            ],
            "maxRetries": 2,
            "messageSelectionStrategy": "Random"
          },
          "defaultValueSpecification": null,
          "sampleUtterances": [
            {
              "utterance": "eye color is {eyeColor}"
            },
            {
              "utterance": "I have eyeColor eyes"
            }
          ],
          "waitAndContinueSpecification": null
        },
        "slotTypeId": "7FEVCB2PQE"
```

```
      }
    },
    "expression": "(firstname OR eyeColor)"
  }
}
```

Struttura del file di tipo slot

Il file del tipo di slot contiene le informazioni di configurazione per un tipo di slot personalizzato utilizzato in una lingua o in una lingua. C'è un file di tipo di slot nel file.zip per ogni tipo di slot personalizzato in una lingua specifica.

Di seguito è riportata la struttura JSON per il tipo di slot che elenca i tipi di auto disponibili nel bot di BookTrip esempio. Per un esempio completo della struttura JSON per un tipo di slot, vedere l'CreateSlotTypeoperazione.

```
{
    "name": "CarTypeValues",
    "identifier": "T1YUHGD9ZR",
    "description": "Enumeration representing possible types of cars available for
 hire",
    "slotTypeValues": [{
        "synonyms": null,
        "sampleValue": {
            "value": "economy"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "standard"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "midsize"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "full size"
        }
    }, {
```

```
        "synonyms": null,
        "sampleValue": {
            "value": "luxury"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "minivan"
        }
    }],
    "parentSlotTypeSignature": null,
    "valueSelectionSetting": {
        "resolutionStrategy": "TOP_RESOLUTION",
        "advancedRecognitionSetting": {
            "audioRecognitionStrategy": "UseSlotValuesAsCustomVocabulary"
        },
        "regexFilter": null
    }
}
```

L'esempio seguente mostra la struttura JSON per un tipo di slot composito.

```
{
  "name": "CarCompositeType",
  "identifier": "TPA3CC9V",
  "description": null,
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": {
    "regexFilter": null,
    "resolutionStrategy": "CONCATENATION"
  },
  "compositeSlotTypeSetting": {
    "subSlots": [
      {
        "name": "model",
        "slotTypeId": "MODELTYPEID" # custom slot type Id for model
      },
      {
        "name": "city",
        "slotTypeId": "AMAZON.City"
      },
      {
```

```
        "name": "country",
        "slotTypeId": "AMAZON.Country"
      },
      {
        "name": "make",
        "slotTypeId": "MAKETYPEID" # custom slot type Id for make
      }
    ]
  }
}
```

Quello che segue è un tipo di slot che utilizza una grammatica personalizzata per comprendere le espressioni del cliente. Per ulteriori informazioni, consulta Tipo di slot grammaticale.

```
{
  "name": "custom_grammar",
  "identifier": "7KEAQIQKPX",
  "description": "Slot type using a custom grammar",
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": null,
  "externalSourceSetting": {
    "grammarSlotTypeSetting": {
      "source": {
        "kmsKeyArn": "arn:aws:kms:Region:123456789012:alias/customer-grxml-key",
        "s3BucketName": "grxml-test",
        "s3ObjectKey": "grxml_files/grammar.grxml"
      }
    }
  }
}
```

Struttura dei file di vocabolario personalizzata.

Il file di vocabolario personalizzato contiene le voci di un vocabolario personalizzato per una sola lingua o locale. C'è un file di vocabolario personalizzato nel file.zip per ogni lingua con un vocabolario personalizzato.

Quello che segue è un file di vocabolario personalizzato per un bot che accetta ordini al ristorante. C'è un file per ogni lingua nel bot.

```
{
```

```
    "customVocabularyItems": [
        {
            "weight": 3,
            "phrase": "wafers"
        },
        {
            "weight": null,
            "phrase": "extra large"
        },
        {
            "weight": null,
            "phrase": "cremini mushroom soup"
        },
        {
            "weight": null,
            "phrase": "ramen"
        },
        {
            "weight": null,
            "phrase": "orzo"
        }
    ]
}
```

Assegnazione di tag alle risorse

Per semplificare la gestione di bot e alias di bot Amazon Lex V2, è possibile assegnare metadati a ciascuna risorsa sotto forma di tag. Un tag è un'etichetta che assegni a una risorsa AWS. Ciascun tag è formato da una chiave e da un valore,

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

- Identificazione e organizzazione delle risorse AWS. MolteAWS risorse supportano l'assegnazione di tag, perciò è possibile assegnare lo stesso tag a risorse in diversi servizi per indicare che le risorse sono le stesse. Ad esempio, è possibile assegnare lo stesso tag a un bot e alle funzioni Lambda che utilizza.
- 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 V2, puoi allocare i costi per ogni alias utilizzando tag specifici per l'alias. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella AWS Billing and Cost ManagementGuida per l'utente.
- Controllare l'accesso alle risorse di . Puoi utilizzare i tag con Amazon Lex V2 per creare policy per controllare l'accesso alle risorse di Amazon Lex V2. Queste politiche possono essere associate a un ruolo IAM o a un utente per abilitare il controllo degli accessi basato su tag.

Puoi lavorare con i tag utilizzando laAWS Management ConsoleAWS Command Line Interface, l'o l'API Amazon Lex V2.

Tagging delle risorse

Se utilizzi la console Amazon Lex V2, è possibile assegnare tag ai tipi di risorse al momento della loro creazione o aggiungere i tag in un secondo momento. È inoltre possibile utilizzare la console per aggiornare o rimuovere i tag esistenti.

Se utilizzi l'AWS CLIAPI Amazon Lex V2, utilizzi le seguenti operazioni per gestire i tag per la tua risorsa:

- CreateBote CreateBotAlias— applica i tag quando crei un bot o un alias bot.
- ListTagsForResource— visualizzare i tag associati a una risorsa.

- TagResource— aggiungere e modificare tag su una risorsa esistente.
- UntagResource: rimuovere i tag da una risorsa.

Le seguenti risorse in Amazon Lex V2 supportano l'assegnazione di tag:

- Bot: utilizzare un nome della risorsa Amazon (ARN) come il seguente:
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Alias dei bot: usa un ARN come il seguente:
  - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

`bot-alias-id`I valori`bot-id` e sono stringhe alfanumeriche in maiuscolo lunghe 10 caratteri.

Limitazioni applicate ai tag

Le seguenti restrizioni di base si applicano ai tag sulle risorse Amazon Lex V2:

- Numero massimo di chiavi: 50 utilizzando la console, 200 utilizzando l'API
- 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'AWSutilizzo di questo prefisso è esclusivo di

Assegnazione di tag alle risorse (console)

Puoi usare la console per gestire i tag su un bot o un alias bot. È possibile aggiungere tag durante la creazione della risorsa oppure aggiungere, modificare o rimuovere tag da risorse esistenti.

Per aggiungere un tag quando si crea un bot

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli Crea bot.
3. Nella sezione Impostazioni avanzate di Configura le impostazioni del bot, scegli Aggiungi nuovo tag. Puoi aggiungere i tag al bot e all'`TestBotAlias`alias.

4. Scegli Avanti per continuare a creare il tuo bot.

Per aggiungere un tag quando si crea un alias bot

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegliere il bot a cui si desidera aggiungere l'alias bot.
3. Dal menu a sinistra, scegli Alias, quindi scegli Crea alias.
4. In Informazioni generali, scegli Aggiungi nuovo tag da Tag.
5. Seleziona Create (Crea).

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

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot da modificare.
3. Dal menu a sinistra, scegli Impostazioni, quindi scegli Modifica.
4. In Tag, apporta le modifiche.
5. Scegli Salva per salvare le modifiche al bot.

Per aggiungere, rimuovere o modificare un tag su un alias esistente

1. Accedere alAWS Management Console e aprire la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli il bot da modificare.
3. Dal menu a sinistra, scegli Alias, quindi dall'elenco degli alias, scegli l'alias da modificare.
4. Dai dettagli dell'alias, in Tag, scegli Modifica tag.
5. In Gestisci i tag, apporta le modifiche.
6. Scegli Salva per salvare le modifiche all'alias.

Sicurezza in Amazon Lex V2

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di data center e architetture di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra te e te. AWS Il modello di responsabilità condivisa descrive questo aspetto 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. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per ulteriori informazioni sui programmi di conformità applicabili ad Amazon Lex V2, consulta AWS Services in Scope by Compliance Program .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Sei anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della tua azienda e le leggi e normative vigenti.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa quando usi Amazon Lex V2. I seguenti argomenti mostrano come configurare Amazon Lex V2 per soddisfare i tuoi obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri servizi AWS che ti aiutano a monitorare e proteggere le tue risorse Amazon Lex V2.

Argomenti

Protezione dei dati in Amazon Lex V2

Amazon Lex V2 è conforme al modello di responsabilità AWS condivisa Modello di , che include regolamenti e linee guida per la protezione dei dati. AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i AWS servizi. AWS mantiene il controllo sui dati ospitati su questa infrastruttura, compresi i controlli di configurazione di sicurezza per la gestione dei contenuti e dei dati personali dei clienti. AWS i clienti e i partner APN, che agiscono in qualità di titolari o incaricati del trattamento dei dati, sono responsabili di tutti i dati personali che inseriscono nel AWS Cloud.

Per gli scopi di protezione dei dati, ti consigliamo di proteggere le credenziali dell'account AWS e configurare singoli account utente con AWS Identity and Access Management (IAM), in modo che a ogni utente vengano fornite solo le autorizzazioni necessarie per soddisfare i compiti del processo. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS

- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno AWS dei servizi.
- Utilizza i servizi di sicurezza gestiti avanzati, ad esempio Amazon Macie, che aiutano a individuare e proteggere i dati personali archiviati in Amazon S3.

Ti consigliamo di non inserire mai informazioni identificative sensibili, ad esempio i numeri di account dei clienti, in campi a formato libero, ad esempio un campo Name (Nome). Ciò include quando lavori con Amazon Lex V2 o altri AWS servizi utilizzando la console, l'API o gli AWS SDK. AWS CLI Tutti i dati che inserisci in Amazon Lex V2 o in altri servizi potrebbero essere raccolti per essere inclusi nei log di diagnostica. Quando fornisci un URL a un server esterno, non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta a tale server.

Per ulteriori informazioni sulla protezione dei dati, consulta il post del blog AWS Modello di responsabilità condivisa e GDPR su AWS Security Blog.

Crittografia a riposo

Amazon Lex V2 crittografa le espressioni degli utenti e altre informazioni archiviate.

Argomenti

Esempi di enunciati

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 inutilizzate e vengono utilizzate solo per creare il bot e creare l'esperienza del cliente.

Attributi sessione

Gli attributi di sessione contengono informazioni specifiche dell'applicazione che vengono trasmesse tra Amazon Lex V2 e le applicazioni client. Amazon Lex V2 trasmette gli attributi di sessione a tutte le AWS Lambda funzioni configurate per un bot. Se una funzione Lambda aggiunge o aggiorna gli attributi di sessione, Amazon Lex V2 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 della 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 V2 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 V2 utilizza il protocollo HTTPS per comunicare con l'applicazione client. Utilizza HTTPS e AWS firme per comunicare con altri servizi, come Amazon Polly, AWS Lambda e per conto della tua applicazione.

Gestione delle identità e degli accessi per Amazon Lex V2

AWS Identity and Access Management (IAM) è uno strumento Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (dispone delle autorizzazioni) a utilizzare le risorse Amazon Lex V2. IAM è uno strumento Servizio AWS 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 V2.

Utente del servizio: se utilizzi il servizio Amazon Lex V2 per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità di Amazon Lex V2 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 V2, consultaRisoluzione dei problemi di identità e accesso ad Amazon Lex V2.

Amministratore del servizio: se sei responsabile delle risorse Amazon Lex V2 della tua azienda, probabilmente hai pieno accesso ad Amazon Lex V2. È tuo compito determinare a quali funzionalità e risorse di Amazon Lex V2 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 V2, consultaCome funziona Amazon Lex V2 con IAM.

Amministratore IAM: se sei un amministratore IAM, potresti voler conoscere i dettagli su come scrivere policy per gestire l'accesso ad Amazon Lex V2. Per visualizzare esempi di policy basate sull'identità di Amazon Lex V2 che puoi utilizzare in IAM, consulta. Esempi di policy basate sull'identità per Amazon Lex V2

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 presenti 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 sapere 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 V2 con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon Lex V2, scopri quali funzionalità IAM sono disponibili per l'uso con Amazon Lex V2.

Funzionalità IAM che puoi utilizzare con Amazon Lex V2

Funzionalità IAM	Supporto per Amazon Lex V2
Policy basate su identità	Sì
Policy basate su risorse	Sì
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	No
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	No
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	Parziale

Per avere una visione di alto livello di come Amazon Lex V2 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 V2

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 V2

Per visualizzare esempi di policy basate sull'identità di Amazon Lex V2, consulta. Esempi di policy basate sull'identità per Amazon Lex V2

Policy basate sulle risorse all'interno di Amazon Lex V2

Supporta le policy basate su risorse	Sì

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 utenti, ruoli, utenti federati o servizi AWS.

Non puoi utilizzare politiche tra account o aree geografiche con Amazon Lex. Se crei una politica per una risorsa con un ARN tra account o più regioni, Amazon Lex restituisce un errore.

Il servizio Amazon Lex supporta policy basate sulle risorse denominate bot policy e bot alias policy, che sono collegate a un bot o a un alias bot. Queste politiche definiscono quali principali possono eseguire azioni sul bot o sull'alias del bot.

Le azioni possono essere utilizzate solo su risorse specifiche. Ad esempio, l'`UpdateBot`azione può essere utilizzata solo su risorse bot, l'`UpdateBotAlias`azione può essere utilizzata solo su risorse alias bot. Se specifichi un'azione in una politica che non può essere utilizzata sulla risorsa specificata nella politica, Amazon Lex restituisce un errore. Per l'elenco delle azioni e delle risorse con cui possono essere utilizzate, consulta la tabella seguente.

Azione	Supporta politiche basate sulle risorse	Risorsa
BuildBotLocale	Supportato	BotId
CreateBot	No	
CreateBotAlias	No	
CreateBotChannel [solo autorizzazione]	Supportato	BotId
CreateBotLocale	Supportato	BotId
CreateBotVersione	Supportato	BotId
CreateExport	Supportato	BotId
CreateIntent	Supportato	BotId
CreateResourcePolitica	Supportato	BotId, BotAliasId
CreateSlot	Supportato	BotId
CreateSlotTipo	Supportato	BotId
CreateUploadUrl	No	

Azione	Supporta politiche basate sulle risorse	Risorsa
DeleteBot	Supportato	BotId, BotAliasId
DeleteBotAlias	Supportato	BotAliasId
DeleteBotChannel [solo autorizzazione]	Supportato	BotId
DeleteBotLocale	Supportato	BotId
DeleteBotVersione	Supportato	BotId
DeleteExport	Supportato	BotId
DeleteImport	Supportato	BotId
DeleteIntent	Supportato	BotId
DeleteResourcePolitica	Supportato	BotId, BotAliasId
DeleteSession	Supportato	BotAliasId
DeleteSlot	Supportato	BotId
DeleteSlotTipo	Supportato	BotId
DescribeBot	Supportato	BotId
DescribeBotPseudonimo	Supportato	BotAliasId
DescribeBotChannel [solo autorizzazione]	Supportato	BotId
DescribeBotLocale	Supportato	BotId
DescribeBotVersione	Supportato	BotId
DescribeExport	Supportato	BotId
DescribeImport	Supportato	BotId

Azione	Supporta politiche basate sulle risorse	Risorsa
DescribeIntent	Supportato	BotId
DescribeResourcePolitica	Supportato	BotId, BotAliasId
DescribeSlot	Supportato	BotId
DescribeSlotTipo	Supportato	BotId
GetSession	Supportato	BotAliasId
ListBotAlias	Supportato	BotId
ListBotChannels [solo autorizzazione]	Supportato	BotId
ListBotLocali	Supportato	BotId
ListBots	No	
ListBotVersioni	Supportato	BotId
ListBuiltInIntents	No	
ListBuiltIntSlotTipologie	No	
ListExports	No	
ListImports	No	
ListIntents	Supportato	BotId
ListSlots	Supportato	BotId
ListSlotTipologie	Supportato	BotId
PutSession	Supportato	BotAliasId
RecognizeText	Supportato	BotAliasId

Azione	Supporta politiche basate sulle risorse	Risorsa
RecognizeUtterance	Supportato	BotAliasId
StartConversation	Supportato	BotAliasId
StartImport	Supportato	BotId, BotAliasId
TagResource	No	
UpdateBot	Supportato	BotId
UpdateBotPseudonimo	Supportato	BotAliasId
UpdateBotLocale	Supportato	BotId
UpdateBotVersione	Supportato	BotId
UpdateExport	Supportato	BotId
UpdateIntent	Supportato	BotId
UpdateResourcePolitica	Supportato	BotId, BotAliasId
UpdateSlot	Supportato	BotId
UpdateSlotTipo	Supportato	BotId
UntagResource	No	

Per informazioni su come allegare una policy basata sulle risorse a un bot o a un alias di bot, consulta. Esempi di policy basate sulle risorse per Amazon Lex V2

Esempi di policy basate sulle risorse all'interno di Amazon Lex V2

Per visualizzare esempi di policy basate sulle risorse di Amazon Lex V2, consulta. Esempi di policy basate sulle risorse per Amazon Lex V2

Azioni politiche per Amazon Lex V2

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 V2, consulta Azioni definite da Amazon Lex V2 nel Service Authorization Reference.

Le azioni politiche in Amazon Lex V2 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"
         ]
```

Per visualizzare esempi di policy basate sull'identità di Amazon Lex V2, consulta. Esempi di policy basate sull'identità per Amazon Lex V2

Risorse relative alle policy per Amazon Lex V2

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": "*"
```

Per visualizzare un elenco dei tipi di risorse Amazon Lex V2 e dei relativi ARN, consulta Resources defined by Amazon Lex V2 nel Service Authorization Reference. Per sapere con quali azioni è possibile specificare l'ARN di ogni risorsa, consulta Azioni definite da Amazon Lex V2.

Per visualizzare esempi di policy basate sull'identità di Amazon Lex V2, consulta. Esempi di policy basate sull'identità per Amazon Lex V2

Chiavi delle condizioni delle politiche per Amazon Lex V2

Supporta le chiavi di condizione delle policy specifiche del servizio	No

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 V2, consulta Chiavi di condizione per Amazon Lex V2 nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Azioni definite da Amazon Lex V2.

Per visualizzare esempi di policy basate sull'identità di Amazon Lex V2, consulta. Esempi di policy basate sull'identità per Amazon Lex V2

Elenchi di controllo degli accessi (ACL) in Amazon Lex V2

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.

Controllo degli accessi basato sugli attributi (ABAC) con Amazon Lex V2

Supporta ABAC (tag nelle policy)	Sì

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.

Utilizzo di credenziali temporanee con Amazon Lex V2

Supporta le credenziali temporanee	No

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 accedi AWS utilizzando il link Single Sign-On (SSO) della tua 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.

Autorizzazioni principali per più servizi per Amazon Lex V2

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 V2

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 V2. Modifica i ruoli di servizio solo quando Amazon Lex V2 fornisce indicazioni in tal senso.

Ruoli collegati ai servizi per Amazon Lex V2

Supporta i ruoli collegati ai servizi	Parziale

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 ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate sull'identità per Amazon Lex V2

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Amazon Lex V2. 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 V2, incluso il formato degli ARN per ciascun tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon Lex V2 nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della console Amazon Lex V2
- Consenti agli utenti di aggiungere funzioni a un bot
- Consenti agli utenti di aggiungere canali a un bot
- Consenti agli utenti di creare e aggiornare i bot
- Consenti agli utenti di utilizzare Automated Chatbot Designer
- Consenti agli utenti di utilizzare una AWS KMS chiave per crittografare e decrittografare i file
- Consenti agli utenti di eliminare i bot
- Consenti agli utenti di conversare con un bot

- Consenti a un utente specifico di gestire le politiche basate sulle risorse
- Consenti a un utente di esportare bot e impostazioni locali dei bot
- Consenti a un utente di esportare un vocabolario personalizzato
- Consenti a un utente di importare bot e impostazioni locali dei bot
- Consenti a un utente di importare un vocabolario personalizzato
- Consenti a un utente di migrare un bot da Amazon Lex ad Amazon Lex V2
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Consenti a un utente di tracciare il flusso di conversazione con Visual Conversation Builder in Amazon Lex V2
- Consenti agli utenti di creare e visualizzare le repliche dei bot, ma non di eliminarle

Best practice per le policy

Le policy basate sull'identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon Lex V2 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 V2

Per accedere alla console Amazon Lex V2, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Amazon Lex V2 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'API. AWS Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano continuare a utilizzare la console Amazon Lex V2, gli utenti devono disporre dell'accesso alla console. Per ulteriori informazioni sulla creazione di un utente con accesso alla console, consulta Creating an IAM user in your AWS account nella IAM User Guide.

Consenti agli utenti di aggiungere funzioni a un bot

Questo esempio mostra una policy che consente agli utenti IAM di aggiungere Amazon Comprehend, sentiment analysis e autorizzazioni per le query Amazon Kendra a un bot Amazon Lex V2.

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

```
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
        }
    ]
}
```

Consenti agli utenti di aggiungere canali a un bot

Questo esempio è una policy che consente agli utenti IAM di aggiungere un canale di messaggistica a un bot. Un utente deve disporre di questa policy prima di poter implementare un bot su una piattaforma di messaggistica.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        }
    ]
}
```

Consenti agli utenti di creare e aggiornare i bot

Questo esempio mostra una policy di esempio che consente agli utenti IAM di creare e aggiornare qualsiasi bot. La policy include le autorizzazioni per completare questa azione sulla console o utilizzando l' AWS API AWS CLI o.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateBot",
        "lex:UpdateBot".
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*]
    }
  ]
}
```

Consenti agli utenti di utilizzare Automated Chatbot Designer

Questo esempio mostra un esempio di policy che consente agli utenti IAM di eseguire Automated Chatbot Designer.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<customer-bucket>/<bucketName>",
                # Resource should point to the bucket or an explicit folder.
                # Provide this to read the entire bucket
                "arn:aws:s3:::<customer-bucket>/<bucketName>/*",
                # Provide this to read a specifc folder
                "arn:aws:s3:::<customer-bucket>/<bucketName>/<pathFormat>/*"
```

```
            ]
        },
        {
            # Use this if your S3 bucket is encrypted with a KMS key.
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:<Region>:<customerAccountId>:key/<kmsKeyId>"
            ]
    ]
}
```

Consenti agli utenti di utilizzare una AWS KMS chiave per crittografare e decrittografare i file

Questo esempio mostra una policy di esempio che consente agli utenti IAM di utilizzare una chiave gestita AWS KMS dal cliente per crittografare e decrittografare i dati.

```
{
    "Version": "2012-10-17",
    "Id": "sample-policy",
    "Statement": [
        {
            "Sid": "Allow Lex access",
            "Effect": "Allow",
            "Principal": {
                "Service": "lexv2.amazonaws.com"
            },
            "Action": [
                # If the key is for encryption
                "kms:Encrypt",
                "kms:GenerateDataKey"
                # If the key is for decryption
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
   ]
}
```

Consenti agli utenti di eliminare i bot

Questo esempio mostra una policy di esempio che consente agli utenti IAM di eliminare qualsiasi bot. La policy include le autorizzazioni per completare questa azione sulla console o utilizzando l' AWS API AWS CLI o.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "lex:DeleteBot",
          "lex:DeleteBotLocale",
          "lex:DeleteBotAlias",
          "lex:DeleteIntent",
          "lex:DeleteSlot",
          "lex:DeleteSlottype"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*",
                   "arn:aws:lex:Region:123412341234:bot-alias/*"]
    }
  ]
}
```

Consenti agli utenti di conversare con un bot

Questo esempio mostra un esempio di policy che consente agli utenti IAM di conversare con qualsiasi bot. La policy include le autorizzazioni per completare questa azione sulla console o utilizzando l' AWS API AWS CLI o.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "lex:StartConversation",
          "lex:RecognizeText",
          "lex:RecognizeUtterance",
          "lex:GetSession",
          "lex:PutSession",
          "lex:DeleteSession"
```

```
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lex:Region:123412341234:bot-alias/*"
    }
  ]
}
```

Consenti a un utente specifico di gestire le politiche basate sulle risorse

L'esempio seguente concede l'autorizzazione a un utente specifico per gestire le politiche basate sulle risorse. Consente l'accesso da console e API alle politiche associate ai bot e agli alias dei bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ResourcePolicyEditor",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/ResourcePolicyEditor"
            },
            "Action": [
                "lex:CreateResourcePolicy",
                "lex:UpdateResourcePolicy",
                "lex:DeleteResourcePolicy",
                "lex:DescribeResourcePolicy"
            ]
        }
    ]
}
```

Consenti a un utente di esportare bot e impostazioni locali dei bot

La seguente politica di autorizzazione IAM consente a un utente di creare, aggiornare e ottenere un'esportazione per un bot o un bot locale.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
```

```
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeBot",
                "lex:DescribeBotLocale",
                "lex:ListBotLocales",
                "lex:DescribeIntent",
                "lex:ListIntents",
                "lex:DescribeSlotType",
                "lex:ListSlotTypes",
                "lex:DescribeSlot",
                "lex:ListSlots",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Consenti a un utente di esportare un vocabolario personalizzato

La seguente politica di autorizzazione IAM consente a un utente di esportare un vocabolario personalizzato da una locale bot.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Consenti a un utente di importare bot e impostazioni locali dei bot

La seguente politica di autorizzazione IAM consente a un utente di importare un bot o una versione locale di un bot e di controllare lo stato di un'importazione.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateBot",
                "lex:UpdateBot",
                "lex:DeleteBot",
                "lex:CreateBotLocale",
                "lex:UpdateBotLocale",
                "lex:DeleteBotLocale",
                "lex:CreateIntent",
                "lex:UpdateIntent",
                "lex:DeleteIntent",
                "lex:CreateSlotType",
                "lex:UpdateSlotType",
                "lex:DeleteSlotType",
                "lex:CreateSlot",
                "lex:UpdateSlot",
                "lex:DeleteSlot",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "iam:PassRole",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*",
                "arn:aws:lex:Region:123456789012:bot-alias/*"
            ]
        }
    ]
}
```

Consenti a un utente di importare un vocabolario personalizzato

La seguente politica di autorizzazione IAM consente a un utente di importare un vocabolario personalizzato in una versione locale del bot.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary"

            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*"
            ]
        }
    ]
}
```

Consenti a un utente di migrare un bot da Amazon Lex ad Amazon Lex V2

La seguente politica di autorizzazione IAM consente a un utente di iniziare la migrazione di un bot da Amazon Lex ad Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "startMigration",
            "Effect": "Allow",
            "Action": "lex:StartMigration",
            "Resource": "arn:aws:lex:>Region<:>123456789012<:bot:*"
        },
        {
            "Sid": "passRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::>123456789012<:role/>v2 bot role<"
        },
        {
            "Sid": "allowOperations",
            "Effect": "Allow",
```

```
            "Action": [
                "lex:CreateBot",
                "lex:CreateIntent",
                "lex:UpdateSlot",
                "lex:DescribeBotLocale",
                "lex:UpdateBotAlias",
                "lex:CreateSlotType",
                "lex:DeleteBotLocale",
                "lex:DescribeBot",
                "lex:UpdateBotLocale",
                "lex:CreateSlot",
                "lex:DeleteSlot",
                "lex:UpdateBot",
                "lex:DeleteSlotType",
                "lex:DescribeBotAlias",
                "lex:CreateBotLocale",
                "lex:DeleteIntent",
                "lex:StartImport",
                "lex:UpdateSlotType",
                "lex:UpdateIntent",
                "lex:DescribeImport",
                "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": "*"
        }
    ]
}
```

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 politica include le autorizzazioni per completare questa azione sulla console o utilizzando programmaticamente l'API o. 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": "*"
        }
    ]
}
```

Consenti a un utente di tracciare il flusso di conversazione con Visual Conversation Builder in Amazon Lex V2

La seguente politica di autorizzazione IAM consente a un utente di disegnare il flusso di conversazione con Visual Conversation Builder in Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:UpdateIntent ",
                "lex:DescribeIntent "
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Consenti agli utenti di creare e visualizzare le repliche dei bot, ma non di eliminarle

Puoi assegnare le seguenti autorizzazioni a un ruolo IAM per consentirgli di creare e visualizzare solo repliche di bot. Omettendo`lex:DeleteBotReplica`, si impedisce al ruolo di eliminare le repliche dei bot. Per ulteriori informazioni, consulta Autorizzazioni per replicare i bot e gestire le repliche dei bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "lex:CreateBotReplica",
            "lex:DescribeBotReplica",
            "lex:ListBotReplica",
            "lex:ListBotVersionReplicas",
            "lex:ListBotAliasReplicas",
         ],
         "Resource": [
             "arn:aws:lex:*:*:bot/*",
             "arn:aws:lex:*:*:bot-alias/*"
         ]
```

```
     },
      {
         "Effect": "Allow",
         "Action": [
             "iam:GetRole"
         ],
         "Resource": [
             "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
             "iam:CreateServiceLinkedRole",
         ],
         "Resource": [
             "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
         ],
         "Condition": {
             "StringEquals": {
                 "iam:AWSServiceName": "lexv2.amazonaws.com"
             }
         }
      }
    ]
}
```

Esempi di policy basate sulle risorse per Amazon Lex V2

Una policy basata sulle risorse è associata a una risorsa, ad esempio un bot o un alias di bot. Con una politica basata sulle risorse è possibile specificare chi ha accesso alla risorsa e le azioni che può eseguire su di essa. Ad esempio, è possibile aggiungere politiche basate sulle risorse che consentono a un utente di modificare un bot specifico o di consentire a un utente di utilizzare le operazioni di runtime su uno specifico alias di bot.

Quando utilizzi una politica basata sulle risorse, puoi consentire ad altri AWS servizi di accedere alle risorse del tuo account. Ad esempio, puoi consentire ad Amazon Connect di accedere a un bot Amazon Lex.

Per informazioni su come creare un bot o un alias di bot, consultaCostruire bot.

Argomenti

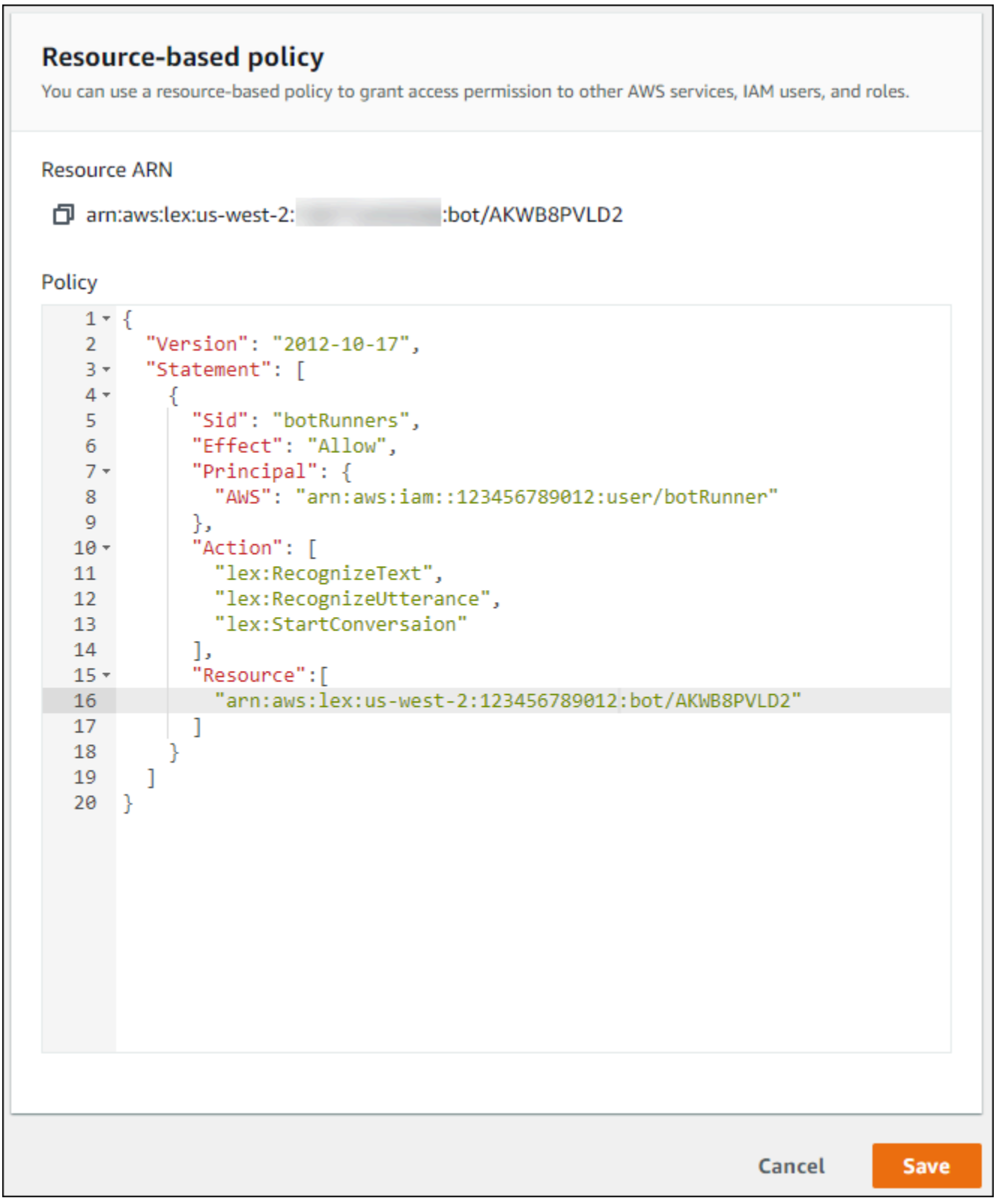
- Usa la console per specificare una policy basata sulle risorse
- Utilizza l'API per specificare una politica basata sulle risorse
- Consenti a un ruolo IAM di aggiornare un bot ed elencare gli alias dei bot
- Consenti a un utente di conversare con un bot
- Consenti a un AWS servizio di utilizzare un bot Amazon Lex V2 specifico

Usa la console per specificare una policy basata sulle risorse

Puoi utilizzare la console Amazon Lex per gestire le policy basate sulle risorse per i bot e gli alias dei bot. Si accede alla struttura JSON di una policy e la console la associa alla risorsa. Se esiste già una politica associata a una risorsa, puoi utilizzare la console per visualizzare e modificare la politica.

Quando si salva una politica con l'editor delle politiche, la console verifica la sintassi della politica. Se la politica contiene errori, ad esempio un utente inesistente o un'azione non supportata dalla risorsa, restituisce un errore e non salva la politica.

Di seguito viene illustrato l'editor di policy basato sulle risorse per un bot nella console. L'editor di policy per un alias bot è simile.


Resource-based policy
You can use a resource-based policy to grant access permission to other AWS services, IAM users, and roles.
Resource ARN
arn:aws:lex:us-west-2: :bot/AKWB8PVLD2
Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "botRunners",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::123456789012:user/botRunner"
},
"Action": [
"lex:RecognizeText",
"lex:RecognizeUtterance",
"lex:StartConversaion"
],
"Resource":[
"arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"
]
}
]
}
Cancel
Save

Per aprire l'editor delle politiche per un bot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli il bot di cui desideri modificare la politica.
3. Nella sezione Politica basata sulle risorse, scegli Modifica.

Per aprire l'editor delle politiche per un alias bot

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dall'elenco dei bot, scegli il bot che contiene l'alias che desideri modificare.
3. Dal menu a sinistra, scegli Alias, quindi scegli l'alias da modificare.
4. Nella sezione Politica basata sulle risorse, scegli Modifica.

Utilizza l'API per specificare una politica basata sulle risorse

Puoi utilizzare le operazioni API per gestire le politiche basate sulle risorse per i bot e gli alias dei bot. Esistono operazioni per creare, aggiornare ed eliminare le politiche.

CreateResourcePolitica

Aggiunge una nuova politica delle risorse con le istruzioni di policy specificate a un bot o a un alias di bot.

CreateResourcePolicyStatement

Aggiunge una nuova dichiarazione sulla politica delle risorse a un bot o a un alias di bot.

DeleteResourcePolitica

Rimuove una politica delle risorse da un bot o da un alias di bot.

DeleteResourcePolicyStatement

Rimuove una dichiarazione sulla politica delle risorse da un bot o da un alias di bot.

DescribeResourcePolitica

Ottiene una politica delle risorse e la revisione della politica.

UpdateResourcePolitica

Sostituisce la politica delle risorse esistente per un bot o un alias di bot con una nuova.

Esempi

Java

L'esempio seguente mostra come utilizzare le operazioni delle politiche basate sulle risorse per gestire una politica basata sulle risorse.

```
        /*
         * Create a new policy for the specified bot alias
         * that allows a role to invoke lex:UpdateBotAlias on it.
         * The created policy will have revision id 1.
         */

        CreateResourcePolicyRequest createPolicyRequest =
                CreateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Allow\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.createResourcePolicy(createPolicyRequest);

        /*
         * Overwrite the policy for the specified bot alias with a new policy.
         * Since no expectedRevisionId is provided, this request overwrites the
 current revision.
         * After this update, the revision id for the policy is 2.
         */
        UpdateResourcePolicyRequest updatePolicyRequest =
        UpdateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Deny\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
```

```
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.updateResourcePolicy(updatePolicyRequest);

        /*
         * Creates a statement in an existing policy for the specified bot alias
         * that allows a role to invoke lex:RecognizeText on it.
         * This request expects to update revision 2 of the policy. The request will
 fail
         * if the current revision of the policy is no longer revision 2.
         * After this request, the revision id for this policy will be 3.
         */

        CreateResourcePolicyStatementRequest createStatementRequest =
                CreateResourcePolicyStatementRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .effect("Allow")

 .principal(Principal.builder().arn("arn:aws:iam::123456789012:role/
BotRunner").build())
                        .action("lex:RecognizeText")
                        .statementId("BotRunnerStatement")
                        .expectedRevisionId(2)
                        .build();

        lexmodelsv2Client.createResourcePolicyStatement(createStatementRequest);

        /*
         * Deletes a statement from an existing policy for the specified bot alias
 by statementId.
         * Since no expectedRevisionId is supplied, the request will remove the
 statement from
         * the current revision of the policy for the bot alias.
         * After this request, the revision id for this policy will be 4.
         */
        DeleteResourcePolicyRequest deleteStatementRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .statementId("BotRunnerStatement")
                        .build();
```

```
        lexmodelsv2Client.deleteResourcePolicy(deleteStatementRequest);

        /*
         * Describe the current policy for the specified bot alias
         * It always returns the current revision.
         */
        DescribeResourcePolicyRequest describePolicyRequest =
                DescribeResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .build();

        lexmodelsv2Client.describeResourcePolicy(describePolicyRequest);


        /*
         * Delete the current policy for the specified bot alias
         * This request expects to delete revision 3 of the policy. Since the
 revision id for
         * this policy is already at 4, this request will fail.
         */
        DeleteResourcePolicyRequest deletePolicyRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .expectedRevisionId(3);
                        .build();

        lexmodelsv2Client.deleteResourcePolicy(deletePolicyRequest);
```

Consenti a un ruolo IAM di aggiornare un bot ed elencare gli alias dei bot

L'esempio seguente concede le autorizzazioni per un ruolo IAM specifico per chiamare le operazioni API di creazione di modelli Amazon Lex V2 per modificare un bot esistente. L'utente può elencare gli alias per un bot e aggiornarlo, ma non può eliminare il bot o gli alias del bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botBuilders",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/BotBuilder"
            },
            "Action": [
                "lex:ListBotAliases",
                "lex:UpdateBot"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/MYBOT"
            ]
        }
    ]
}
```

Consenti a un utente di conversare con un bot

L'esempio seguente concede l'autorizzazione a un utente specifico di chiamare le operazioni dell'API di runtime di Amazon Lex V2 su un singolo alias di un bot.

All'utente viene specificamente negata l'autorizzazione ad aggiornare o eliminare l'alias del bot.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botRunners",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation",
                "lex:DeleteSession",
                "lex:GetSession",
                "lex:PutSession"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        },
        {
```

```
            "Sid": "botRunners",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:UpdateBotAlias",
                "lex:DeleteBotAlias"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        }
    ]
}
```

Consenti a un AWS servizio di utilizzare un bot Amazon Lex V2 specifico

L'esempio seguente concede l'autorizzazione AWS Lambda e Amazon Connect a richiamare le operazioni dell'API di runtime di Amazon Lex V2.

Il blocco condition è necessario per i responsabili del servizio e deve utilizzare le chiavi di contesto globali e. `AWS:SourceAccount` `AWS:SourceArn`

`AWS:SourceAccount`È l'ID dell'account che chiama il bot Amazon Lex V2.

`AWS:SourceArn`È l'ARN della risorsa dell'istanza del servizio Amazon Connect o della funzione Lambda da cui proviene la chiamata all'alias bot di Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "connect-bot-alias",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "connect.amazonaws.com"
                ]
            },
            "Action": [
                "lex:RecognizeText",
                "lex:StartConversation"
```

```
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ],
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "AWS:SourceArn":
 "arn:aws:connect:Region:123456789012:instance/instance-id"
                }
            }
        },
        {
            "Sid": "lambda-function",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "lambda.amazonaws.com"
                ]
            },
            "Action": [
                "lex:RecognizeText",
                "lex:StartConversation"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ],
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "AWS:SourceArn":
 "arn:aws:lambda:Region:123456789012:function/function-name"
                }
            }
        }
    ]
}
```

AWS politiche gestite per Amazon Lex V2

Una politica AWS gestita è una politica autonoma creata e amministrata da. AWS AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Ti 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 politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWSnella Guida per l'utente di IAM.

Policy gestita da AWS: AmazonLexReadOnly

È possibile allegare la policy `AmazonLexReadOnly`alle identità IAM.

Questa policy concede autorizzazioni di sola lettura che consentono agli utenti di visualizzare tutte le azioni nel servizio di creazione di modelli Amazon Lex V2 e Amazon Lex.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `lex`— Accesso in sola lettura alle risorse Amazon Lex V2 e Amazon Lex nel servizio di creazione di modelli.

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

```
            "Sid": "AmazonLexReadOnlyStatement1",
            "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:DescribeBotRecommendation",
                "lex:DescribeBotReplica",
                "lex:DescribeBotVersion",
                "lex:DescribeExport",
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBots",
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotAliasReplicas",
                "lex:ListBotChannels",
                "lex:ListBotRecommendations",
                "lex:ListBotReplicas",
                "lex:ListBotVersions",
                "lex:ListBotVersionReplicas",
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
```

```
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListRecommendedIntents",
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource",
                "lex:SearchAssociatedTranscripts",
                "lex:ListCustomVocabularyItems"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy gestita da AWS: AmazonLexRunBotsOnly

È possibile allegare la policy `AmazonLexRunBotsOnly`alle identità IAM.

Questa policy concede autorizzazioni di sola lettura che consentono l'accesso all'esecuzione di bot conversazionali Amazon Lex V2 e Amazon Lex.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `lex`— Accesso in sola lettura a tutte le azioni nel runtime Amazon Lex V2 e Amazon Lex.

```
{
    "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 policy concede autorizzazioni amministrative che consentono all'utente di creare, leggere, aggiornare ed eliminare risorse Amazon Lex V2 e Amazon Lex e di eseguire bot conversazionali Amazon Lex V2 e Amazon Lex.

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 di modelli e runtime di Amazon Lex V2 e Amazon Lex.
- `cloudwatch`— Consente ai dirigenti di visualizzare le CloudWatch metriche e gli allarmi di Amazon.
- `iam`— Consente ai responsabili di creare ed eliminare ruoli collegati ai servizi, trasferire ruoli e allegare e scollegare le politiche 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 mandanti di elencare gli indici Amazon Kendra.
- `kms`— Consente ai mandanti di descrivere chiavi e alias. AWS KMS
- `lambda`— Consente ai responsabili di elencare AWS Lambda le funzioni e gestire le 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": [
        {
            "Sid": "AmazonLexFullAccessStatement1",
            "Effect": "Allow",
            "Action": [
```

```
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lambda:ListAliases",
                "lambda:ListVersionsByFunction"
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement2",
            "Effect": "Allow",
            "Action": [
                "bedrock:ListFoundationModels"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": "arn:aws:bedrock:*::foundation-model/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
```

```
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement3",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "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*",
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement4",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement5",
```

```
            "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"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement6",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement7",
            "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"
                }
```

```
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement8",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "replication.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement9",
            "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*",
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement10",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
```

```
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement11",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement12",
            "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"
```

```
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement13",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Policy gestita da AWS: AmazonLexReplicationPolicy

Non è possibile collegare `AmazonLexReplicationPolicy`alle entità IAM. Questa policy è associata a un ruolo collegato al servizio che consente ad Amazon Lex V2 di eseguire azioni per tuo conto. Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon Lex V2.

Questa policy concede autorizzazioni amministrative che consentono ad Amazon Lex V2 di replicare AWS le risorse tra regioni per tuo conto. Puoi allegare questa policy per consentire a un ruolo di replicare facilmente le risorse, inclusi bot, impostazioni locali, versioni, alias, intenti, tipi di slot, slot e vocabolari personalizzati.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `lex`— Consente ai responsabili di replicare le risorse in altre regioni.
- `iam`— Consente ai dirigenti di trasferire ruoli da IAM. Ciò è necessario affinché Amazon Lex V2 disponga delle autorizzazioni per replicare risorse in altre regioni.

```
{
    "Version": "2012-10-17",
    "Statement": [
 {
  "Sid": "ReplicationPolicyStatement1",
  "Effect": "Allow",
  "Action": [
   "lex:BuildBotLocale",
   "lex:ListBotLocales",
   "lex:CreateBotAlias",
   "lex:UpdateBotAlias",
   "lex:DeleteBotAlias",
   "lex:DescribeBotAlias",
   "lex:CreateBotVersion",
   "lex:DeleteBotVersion",
   "lex:DescribeBotVersion",
   "lex:CreateExport",
   "lex:DescribeBot",
   "lex:UpdateExport",
   "lex:DescribeExport",
   "lex:DescribeBotLocale",
   "lex:DescribeIntent",
   "lex:ListIntents",
   "lex:DescribeSlotType",
   "lex:ListSlotTypes",
   "lex:DescribeSlot",
   "lex:ListSlots",
   "lex:DescribeCustomVocabulary",
   "lex:StartImport",
   "lex:DescribeImport",
   "lex:CreateBot",
   "lex:UpdateBot",
   "lex:DeleteBot",
   "lex:CreateBotLocale",
   "lex:UpdateBotLocale",
   "lex:DeleteBotLocale",
   "lex:CreateIntent",
```

```
    "lex:UpdateIntent",
    "lex:DeleteIntent",
    "lex:CreateSlotType",
    "lex:UpdateSlotType",
    "lex:DeleteSlotType",
    "lex:CreateSlot",
    "lex:UpdateSlot",
    "lex:DeleteSlot",
    "lex:CreateCustomVocabulary",
    "lex:UpdateCustomVocabulary",
    "lex:DeleteCustomVocabulary",
    "lex:DeleteBotChannel",
    "lex:DeleteResourcePolicy"
   ],
   "Resource": [
    "arn:aws:lex:*:*:bot/*",
    "arn:aws:lex:*:*:bot-alias/*"
   ]
  },
  {
   "Sid": "ReplicationPolicyStatement2",
   "Effect": "Allow",
   "Action": [
    "lex:CreateUploadUrl",
    "lex:ListBots"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ReplicationPolicyStatement3",
   "Effect": "Allow",
   "Action": [
    "iam:PassRole"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "iam:PassedToService": "lexv2.amazonaws.com"
    }
   }
  }
     ]
}
```

Amazon Lex V2 si aggiorna alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon Lex V2 da quando questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS sulla pagina Amazon Lex Cronologia dei documenti per Amazon Lex V2 V2.

Modifica	Descrizione	Data
AmazonLexReadOnly: aggiornamento a una policy esistente	Amazon Lex V2 ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura alle repliche delle risorse bot.	10 maggio 2024
AmazonLexFullAccess: aggiornamento a una policy esistente	Amazon Lex V2 ha aggiunto nuove autorizzazioni per consentire la replica delle risorse bot in altre regioni.	16 aprile 2024
AmazonLexFullAccess: aggiornamento a una policy esistente	Amazon Lex V2 ha aggiunto nuove autorizzazioni per consentire la replica delle risorse bot in altre regioni.	31 gennaio 2024
AmazonLexReplicationPolicy: nuova policy	Amazon Lex V2 ha aggiunto una nuova policy per consentir e la replica delle risorse dei bot in altre regioni.	31 gennaio 2024
AmazonLexReadOnly: aggiornamento a una policy esistente	Amazon Lex V2 ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura agli elenchi di vocaboli personalizzati.	29 novembre 2022

Modifica	Descrizione	Data
AmazonLexFullAccess: aggiornamento a una policy esistente	Amazon Lex V2 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 V2 ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura alle operazioni di Amazon Lex V2 Automated Chatbot Designer.	1° dicembre 2021
AmazonLexFullAccess: aggiornamento a una policy esistente	Amazon Lex V2 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 V2 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
AmazonLexRunBotsSolo: esegui l'aggiornamento a una policy esistente	Amazon Lex V2 ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura alle operazioni del servizio di runtime Amazon Lex V2.	18 agosto 2021

Modifica	Descrizione	Data
Amazon Lex V2 ha iniziato a tracciare le modifiche	Amazon Lex V2 ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	18 agosto 2021

Utilizzo di ruoli collegati ai servizi per Amazon Lex V2

Amazon Lex V2 utilizza ruoli collegati ai servizi AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon Lex V2. I ruoli collegati ai servizi sono predefiniti da Amazon Lex V2 e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato al servizio semplifica la configurazione di Amazon Lex V2 perché non è necessario aggiungere manualmente le autorizzazioni necessarie. Amazon Lex V2 definisce le autorizzazioni dei suoi ruoli collegati ai servizi e, se non diversamente definito, solo Amazon Lex V2 può assumerne i ruoli. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere collegata a nessun'altra entità IAM.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM e cerca i servizi che riportano Sì nella colonna Ruolo associato ai servizi. Scegli un Sì con un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

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.

È possibile eliminare un ruolo collegato al servizio solo dopo aver eliminato per la prima volta le risorse correlate. In questo modo proteggi le tue risorse Amazon Lex V2 perché non puoi rimuovere inavvertitamente le autorizzazioni per accedere alle risorse.

Argomenti

- Creazione di un ruolo collegato ai servizi per Amazon Lex V2
- Modifica di un ruolo collegato al servizio per Amazon Lex V2
- Eliminazione di un ruolo collegato al servizio per Amazon Lex V2
- Autorizzazioni di ruolo collegate ai servizi per Amazon Lex V2

- Regioni supportate per i ruoli collegati ai servizi Amazon Lex V2

Creazione di un ruolo collegato ai servizi per Amazon Lex V2

Non è necessario creare manualmente un ruolo collegato al servizio, perché Amazon Lex V2 crea il ruolo collegato al servizio per te quando esegui l'azione pertinente (vedi Autorizzazioni di ruolo collegate ai servizi per Amazon Lex V2 per ulteriori informazioni) nell' AWS Management Console API, o. AWS CLI AWS

Se elimini questo ruolo collegato al servizio e poi devi crearne uno di nuovo, puoi utilizzare la stessa procedura per creare un nuovo ruolo nel tuo account.

Modifica di un ruolo collegato al servizio per Amazon Lex V2

Amazon Lex V2 non consente di modificare i ruoli collegati ai servizi. Dopo aver creato un ruolo collegato al servizio, non puoi modificarne il nome, perché potrebbero farvi riferimento diverse entità. Tuttavia, puoi modificare la descrizione di un ruolo 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 al servizio per Amazon Lex V2

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, 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 al servizio prima di poterlo eliminare manualmente.

> **Note**
>
> Se il servizio Amazon Lex V2 utilizza il ruolo quando tenti di eliminare le risorse, l'eliminazione potrebbe non riuscire. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per visualizzare i passaggi per l'eliminazione di risorse per ruoli specifici collegati ai servizi in Amazon Lex V2, consulta la sezione specifica del ruolo in. Autorizzazioni di ruolo collegate ai servizi per Amazon Lex V2

Per eliminare manualmente un ruolo collegato al servizio utilizzando IAM

Dopo aver eliminato le risorse relative a un ruolo collegato al servizio, utilizza la console IAM, l'o l' AWS CLI AWS API per eliminare il ruolo. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Autorizzazioni di ruolo collegate ai servizi per Amazon Lex V2

Amazon Lex V2 utilizza ruoli collegati ai servizi con i seguenti prefissi.

Argomenti

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

Il ruolo AWSServiceRoleForLexV2Bots _ fornisce le autorizzazioni per connettere il bot ad altri servizi richiesti. Questo ruolo include una politica di fiducia per consentire al servizio lexv2.amazonaws.com di assumere il ruolo e include le autorizzazioni per eseguire le seguenti azioni.

- Usa Amazon Polly per sintetizzare il parlato su tutte le risorse Amazon Lex V2 supportate dall'azione.
- Se un bot è configurato per utilizzare l'analisi del sentiment di Amazon Comprehend, rileva il sentiment su tutte le risorse Amazon Lex V2 supportate dall'azione.
- Se un bot è configurato per archiviare i log audio in un bucket S3, inserisci gli oggetti in un bucket specificato.
- Se un bot è configurato per archiviare log audio e di testo, crea un flusso di log e inserisci i log in un gruppo di log specificato.
- Se un bot è configurato per utilizzare una AWS KMS chiave per crittografare i dati, genera una chiave dati specifica.
- Se un bot è configurato per utilizzare l'`KendraSearchIntent`intento, richiedi l'accesso a un indice Amazon Kendra specificato.

Per creare il ruolo

Amazon Lex V2 crea un nuovo ruolo AWSServiceRoleForLexV2Bots _ con un suffisso casuale nel tuo account ogni volta che crei un bot. Amazon Lex V2 modifica il ruolo quando aggiungi funzionalità

aggiuntive a un bot. Ad esempio, se aggiungi l'analisi del sentiment di Amazon Comprehend a un bot, Amazon Lex V2 aggiunge l'autorizzazione per l'`lex:DetectSentiment`azione al ruolo di servizio.

Per eliminare il ruolo

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dal riquadro di navigazione a sinistra, seleziona Bot e scegli il bot di cui desideri eliminare il ruolo collegato al servizio.
3. Seleziona una versione qualsiasi del bot.
4. Il ruolo di runtime delle autorizzazioni IAM si trova nei dettagli della versione.
5. Torna alla pagina Bot e scegli il pulsante di opzione accanto al bot da eliminare.
6. Seleziona Azione, quindi scegli Elimina.
7. Segui i passaggi indicati in Eliminazione di un ruolo collegato al servizio per eliminare il ruolo IAM.

AWSServiceRoleForLexV2Channels_

Il ruolo AWSServiceRoleForLexV2Channels _ consente di elencare i bot in un account e di chiamare le API di conversazione per un bot. Questo ruolo include una politica di fiducia per consentire al servizio channels.lexv2.amazonaws.com di assumere il ruolo. Se un bot è configurato per utilizzare un canale per comunicare con un servizio di messaggistica, la policy di autorizzazione AWSServiceRoleForLexV2Channels _ role consente ad Amazon Lex V2 di completare le seguenti azioni.

- Elenca le autorizzazioni per tutti i bot di un account.
- Riconosci il testo, ottieni la sessione e inserisci le autorizzazioni di sessione su un alias bot specificato.

Per creare il ruolo

Quando crei un'integrazione di canale per distribuire un bot su una piattaforma di messaggistica, Amazon Lex V2 crea un nuovo ruolo collegato al servizio nel tuo account per ogni canale con un suffisso casuale.

Per eliminare il ruolo

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Dal riquadro di navigazione a sinistra, seleziona Bot.
3. Scegli un bot.
4. Dal riquadro di navigazione a sinistra, scegli Integrazioni di canale in Implementazioni.
5. Seleziona un canale di cui desideri eliminare il ruolo collegato al servizio.
6. Il ruolo di runtime delle autorizzazioni IAM si trova nella configurazione generale
7. Scegli Elimina, quindi scegli nuovamente Elimina per eliminare il canale.
8. Segui i passaggi indicati in Eliminazione di un ruolo collegato al servizio per eliminare il ruolo IAM.

AWSServiceRoleForLexV2Replication

Il AWSServiceRoleForLexV2Replication ruolo consente di replicare i bot in una seconda regione. Questo ruolo include una politica di fiducia per consentire al servizio replication.lexv2.amazonaws.com di assumere il ruolo e include anche la politica gestita, che consente le autorizzazioni per le AmazonLexReplicationPolicy AWS seguenti azioni.

- Passa i ruoli IAM del bot al bot di replica per duplicare le autorizzazioni appropriate per il bot di replica.
- Crea e gestisci bot e risorse bot (versioni, alias, intenti, slot, vocabolari personalizzati, ecc.) in altre regioni.

Per creare il ruolo

Quando abiliti Global Resiliency per un bot, Amazon Lex V2 crea il ruolo AWSServiceRoleForLexV2Replication collegato al servizio nel tuo account. Assicurati di disporre delle autorizzazioni corrette per concedere al servizio Amazon Lex V2 le autorizzazioni per creare il ruolo collegato al servizio.

Per eliminare le risorse Amazon Lex V2 utilizzate da AWSServiceRoleForLexV2Replication in modo da poter eliminare il ruolo

1. Accedi AWS Management Console e apri la console Amazon Lex all'indirizzo https://console.aws.amazon.com/lex/.
2. Scegli un bot per cui è abilitata Global Resiliency.
3. Seleziona Global Resiliency in Deployment.
4. Seleziona Disabilita la resilienza globale.
5. Ripeti la procedura per tutti i bot con Global Resiliency abilitato.
6. Segui i passaggi indicati in Eliminazione di un ruolo collegato al servizio per eliminare il ruolo IAM.

Regioni supportate per i ruoli collegati ai servizi Amazon Lex V2

Amazon Lex V2 supporta l'utilizzo di ruoli collegati al servizio in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta AWS Regioni ed endpoint di .

Risoluzione dei problemi di identità e accesso ad Amazon Lex V2

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Amazon Lex V2 e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in Amazon Lex V2
- Non sono autorizzato a eseguire iam: PassRole
- Sono un amministratore e desidero consentire ad altri di accedere ad Amazon Lex V2
- Concedi l'accesso programmatico a un utente
- Desidero consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon Lex V2

Non sono autorizzato a eseguire un'azione in Amazon Lex V2

Se ti AWS Management Console dice che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso.

L'errore di esempio seguente si verifica quando l'utente `mateojackson` IAM 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, Mateo richiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa *`my-example-widget`* utilizzando l'azione `lex:`*`GetWidget`*.

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 policy devono essere aggiornate per consentirti di trasferire un ruolo ad Amazon Lex V2.

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 V2. 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.

Sono un amministratore e desidero consentire ad altri di accedere ad Amazon Lex V2

Per consentire ad altri di accedere ad Amazon Lex V2, devi creare un'entità IAM (utente o ruolo) per la persona o l'applicazione che necessita dell'accesso. Tale utente o applicazione utilizzerà le credenziali dell'entità per accedere ad AWS. È quindi necessario allegare una policy all'entità che concede loro le autorizzazioni corrette in Amazon Lex V2.

Per iniziare immediatamente, consulta Creazione dei primi utenti e gruppi delegati IAM nella Guida per l'utente di IAM.

Concedi l'accesso programmatico a un utente

Gli utenti necessitano dell'accesso programmatico se desiderano interagire con l' AWS esterno di. AWS Management Console Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche agli AWS CLI AWS SDK o alle API. AWS	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta Configurazione dell'uso AWS IAM Identity Center nella Guida AWS CLI per l'utente.AWS Command Line Interface • Per AWS SDK, strumenti e AWS API, consulta l'autenticazione IAM Identity Center nella Guida di riferimento agli AWS SDK e agli strumenti.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche agli SDK o alle API AWS CLI. AWS AWS	Segui le istruzioni in Uso delle credenziali temporanee con AWS risorse nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare le richieste	Segui le istruzioni per l'interfa ccia che desideri utilizzare.

Quale utente necessita dell'accesso programmatico?	Per	Come
	programmatiche agli AWS CLI AWS SDK o alle API. AWS	• Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli AWS SDK e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli SDK e agli AWS strumenti. • Per le AWS API, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Desidero consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon Lex V2

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 V2 supporta queste funzionalità, consultaCome funziona Amazon Lex V2 con IAM.
- Per sapere come fornire l'accesso alle risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM di un altro Account AWS utente di tua proprietà nella IAM User Guide. Account AWS
- 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.

Registrazione e monitoraggio in Amazon Lex V2

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon Lex V2 e delle altre soluzioni AWS. AWS fornisce i seguenti strumenti di monitoraggio per monitorare Amazon Lex V2, segnalare quando qualcosa non va e intraprendere azioni automatiche quando necessario:

- Amazon CloudWatch monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale. Puoi raccogliere i parametri e tenerne traccia, creare pannelli di controllo personalizzati e impostare allarmi per inviare una notifica o intraprendere azioni quando un parametro specificato raggiunge una determinata soglia. Ad esempio, puoi tenere CloudWatch traccia dell'utilizzo della CPU o di altri parametri delle tue istanze Amazon EC2 e avviare automaticamente nuove istanze quando necessario. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.
- AWS CloudTrailacquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo AWS account e invia i file di log a un bucket Amazon S3 da te specificato. Puoi identificare quali utenti e account hanno chiamato AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.

Convalida della conformità per Amazon Lex V2

I revisori di terze parti valutano la sicurezza e la conformità di Amazon Lex V2 nell'ambito di diversi programmi di AWS conformità. Amazon Lex V2 è un servizio idoneo alla normativa HIPAA. È conforme a PCI, SOC e ISO.

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono i passaggi per l'implementazione di ambienti di base incentrati sulla AWS sicurezza e la conformità.
- Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services: questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni idonee all'HIPAA.

  > **Note**
  >
  > Non Servizi AWS tutte sono idonee all'HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- AWS Risorse per la per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe essere valida per il tuo settore e la tua località.
- AWS Guide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- Valutazione delle risorse con regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida e alle normative del settore.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- Amazon GuardDuty: Servizio AWS rileva potenziali minacce ai tuoi carichi di lavoro Account AWS, ai contenitori e ai dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty

può aiutarti a soddisfare vari requisiti di conformità, come lo standard PCI DSS, soddisfacendo i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità.

- AWS Audit Manager— Ciò Servizio AWS consente di verificare continuamente l' AWS utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Resilienza in Amazon Lex V2

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, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure.AWS

Oltre all'infrastruttura AWS globale, Amazon Lex V2 offre diverse funzionalità per supportare le esigenze di resilienza e backup dei dati.

> **Note**
>
> Per ulteriori informazioni sulla resilienza globale in Amazon Lex V2, che consente di creare un bot replicato in una seconda regione in coppie predeterminate, consulta Global Resiliency.

Sicurezza dell'infrastruttura in Amazon Lex V2

In quanto servizio gestito, Amazon Lex V2 è protetto dalle procedure di sicurezza di rete AWS globali descritte nel white paper Amazon Web Services: Overview of Security Processes.

Utilizzi chiamate API AWS pubblicate per accedere ad Amazon Lex V2 attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.0 o versioni successive. 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 ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Amazon Lex V2 e endpoint VPC di interfaccia ()AWS PrivateLink

Puoi stabilire una connessione privata tra il tuo VPC e Amazon Lex V2 creando un endpoint VPC di interfaccia. Gli endpoint di interfaccia sono alimentati da AWS PrivateLink, una tecnologia che consente di accedere in modo privato alle API di Amazon Lex V2 senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione Direct Connect. AWS Le istanze nel tuo VPC non necessitano di indirizzi IP pubblici per comunicare con le API di Amazon Lex V2. Il traffico tra il tuo VPC e Amazon Lex V2 non esce dalla rete Amazon.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle tue sottoreti.

Per ulteriori informazioni, consulta Interface VPC endpoints (AWS PrivateLink) nella Amazon VPC User Guide.

Considerazioni sugli endpoint VPC Amazon Lex V2

Prima di configurare un endpoint VPC di interfaccia per Amazon Lex V2, assicurati di esaminare le proprietà e le limitazioni degli endpoint dell'interfaccia nella Amazon VPC User Guide.

Amazon Lex V2 supporta le chiamate a tutte le sue azioni API dal tuo VPC.

Creazione di un endpoint VPC di interfaccia per Amazon Lex V2

Puoi creare un endpoint VPC per il servizio Amazon Lex V2 utilizzando la console Amazon VPC o il (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Creazione di un endpoint dell'interfaccia nella Guida per l'utente di Amazon VPC.

Crea un endpoint VPC per Amazon Lex V2 utilizzando il seguente nome di servizio:

- com.amazonaws.*region*.models-v2-lex
- com.amazonaws.*region*.runtime-v2-lex

Se abiliti il DNS privato per l'endpoint, puoi effettuare richieste API ad Amazon Lex V2 utilizzando il nome DNS predefinito per la regione, ad esempio. `runtime-v2-lex.us-east-1.amazonaws.com`

Per ulteriori informazioni, consulta Accesso a un servizio tramite un endpoint dell'interfaccia in Guida per l'utente di Amazon VPC.

Creazione di una policy sugli endpoint VPC per Amazon Lex V2

Puoi allegare una policy per gli endpoint al tuo endpoint VPC che controlla l'accesso ad Amazon Lex V2. La policy specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Esempio: policy degli endpoint VPC per le azioni di Amazon Lex V2

Di seguito è riportato un esempio di policy sugli endpoint per Amazon Lex V2. Se collegata a un endpoint, questa policy garantisce l'accesso alle azioni Amazon Lex V2 elencate per tutti i principali su tutte le risorse.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lex:RecognizeText",
            "lex:RecognizeUtterance",
            "lex:StartConversation",
            "lex:DeleteSession",
            "lex:GetSession",
            "lex:DeleteSession"
         ],
         "Resource":"*"
      }
```

```
    ]
}
```

Linee guida e best practice

Fai riferimento alle seguenti linee guida e best practice per ottimizzare il comportamento e le interazioni del tuo bot con i clienti.

Firma delle richieste

Tutte le richieste di creazione di modelli e runtime di Amazon Lex V2 nell'API Reference utilizzano la firma V4 per l'autenticazione delle richieste. Per ulteriori informazioni sull'autenticazione delle richieste, consulta la procedura di firma Signature Version 4 nel. Riferimenti generali di AWS

Protezione delle informazioni riservate

Le operazioni dell'API di runtime RecognizeTexte RecognizeUtterancerichiedono un ID di sessione come parametro obbligatorio. Gli sviluppatori possono impostare questo parametro su qualsiasi valore che soddisfi i vincoli descritti nell'API. Ti consigliamo di non utilizzare questo parametro per inviare informazioni riservate, come accessi utente, e-mail o numeri di previdenza sociale. Questo ID viene utilizzato principalmente per identificare in modo univoco una conversazione con un bot.

Acquisizione dei valori degli slot dalle espressioni dell'utente

Amazon Lex V2 utilizza i valori di enumerazione forniti in una definizione del tipo di slot per addestrare i suoi modelli di machine learning. Supponiamo di definire un intento chiamato `GetPredictionIntent` con il seguente enunciato di esempio:

```
"Tell me the prediction for {sign}"
```

dove {sign} è uno slot con il tipo personalizzato `ZodiacSign` che ha 12 valori di enumerazione: attraverso. `Aries` `Pisces` Supponiamo ora che l'utente dica «Dimmi la previsione per la Terra»:

- Amazon Lex V2 deduce che «terra» è un ZodiacSign valore se esegui una delle seguenti azioni:
  - Imposta il `valueSelectionStrategy` campo sull'`ORIGINAL_VALUE`utilizzo dell'CreateSlotTypeoperazione
  - Seleziona Espandi valori nella console
- Amazon Lex V2 non riconosce il valore «earth» se limiti il riconoscimento ai valori definiti per il tipo di slot eseguendo una delle seguenti azioni:
  - Imposta il `valueSelectionStrategy` campo sull'`TOP_RESOLUTIO`utilizzo dell'`CreateSlotType`operazione

- Seleziona Limita ai valori e ai sinonimi degli slot nella console

Quando si definiscono i sinonimi per i valori degli slot, questi vengono riconosciuti come uguali a un valore di slot. Tuttavia, viene restituito il valore dello slot anziché il sinonimo.

Poiché Amazon Lex V2 trasmette questo valore all'applicazione client o alla funzione Lambda, è necessario verificare che i valori degli slot siano valori validi prima di utilizzarli nelle attività di evasione degli ordini.

Quando Amazon Lex V2 richiama una funzione Lambda o restituisce il risultato di un'interazione vocale con il client, i valori degli slot non sono garantiti. 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`.

Acronimi nei valori degli slot

Quando definisci i valori degli slot che contengono acronimi, utilizza i seguenti modelli:

- Lettere maiuscole separate da punti (D.V.D.)
- Lettere maiuscole separate da spazi (D V D)

Slot integrati per data e ora

I Amazon.date tipi di slot Amazon.time integrati registrano date e ore (assolute e relative). Le date e gli orari relativi vengono risolti alla data e all'ora in cui Amazon Lex V2 riceve la richiesta e nella regione in cui la elabora.

Per il tipo di slot `AMAZON.Time` integrato, se l'utente non specifica che un orario è prima o dopo mezzogiorno, l'ora è ambigua. In tal caso, Amazon Lex V2 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."

Evitare l'ambiguità nei dati di addestramento per il tuo bot

Fornire dati di formazione confusi nel bot riduce la capacità di Amazon Lex V2 di comprendere gli input degli utenti. Supponiamo che tu abbia due intenti (`OrderPizza`e`OrderDrink`) nel tuo bot e includi «Voglio ordinare» come esempio di enunciato. Quando crei il tuo bot, Amazon Lex V2 non è in grado di associare questo enunciato a un intento specifico. Di conseguenza, quando un utente

immette questo enunciato in fase di esecuzione, Amazon Lex V2 non è in grado di definire un intento con un alto grado di sicurezza.

Se hai due intenti con lo stesso enunciato di esempio, utilizza i contesti di input per aiutare Amazon Lex V2 a distinguere tra i due intenti in fase di esecuzione. Per ulteriori informazioni, vedere Impostazione del contesto dell'intento.

Utilizzo dell'alias TSTALIASID

- L'alias TSTALIASID del tuo bot rimanda alla versione Draft e deve essere usato solo per i test manuali. Amazon Lex limita il numero di richieste di runtime che puoi effettuare all'alias TSTALIASID del bot.
- Quando aggiorni la versione Draft del bot, Amazon Lex interrompe tutte le conversazioni in corso per qualsiasi applicazione client utilizzando l'alias TSTALIASID del bot. In genere, non dovresti usare l'alias TSTALIASID di un bot in produzione perché la versione Draft può essere aggiornata. Dovresti pubblicare una versione e un alias e usarli invece.
- Quando aggiorni un alias, Amazon Lex impiega alcuni minuti per rilevare le modifiche. Quando modifichi la versione Draft del bot, la modifica viene rilevata immediatamente dall'alias TSTALIASID.

Quote

Le quote di servizio, note anche come limiti, sono il numero massimo di risorse di servizio consentite per il tuo account. AWS Per ulteriori informazioni, consulta AWS service quotas nel riferimento AWS generale.

Alcune quote di servizio possono essere modificate o aumentate. Fai riferimento alla colonna Adjustable nelle tabelle seguenti per vedere se una quota può essere modificata e alla colonna Self-service per vedere se puoi richiedere un aggiustamento della quota tramite la console Service quotas. Contatta AWS Support per aumentare una quota regolabile, ma non tramite self-service. L'aumento di una quota di servizio può richiedere alcuni giorni. Se intendi aumentare la quota nell'ambito di un progetto più ampio, assicurati di aggiungere questo periodo al tuo piano.

> **Note**
>
> I limiti di caratteri vengono calcolati come il numero di unità di codice Unicode. Nella maggior parte dei casi, un carattere Unicode equivale a un'unità di codice Unicode. Alcuni caratteri speciali potrebbero essere più grandi di un'unità e il conteggio potrebbe differire a seconda delle diverse codifiche. Per ulteriori informazioni sul calcolo della lunghezza delle stringhe, consulta questa documentazione.

Quote relative al tempo di compilazione

Le seguenti quote massime vengono applicate durante la creazione di un bot.

Descrizione	Predefinita	Adattabile	Self-service
Bot per account AWS	100	Sì	Sì
Associazioni di canali bot per account AWS	5.000	No	N/D
Bot per rete di bot	5	No	N/D
Reti di bot per bot	25	No	N/D
Versioni per bot	100	No	N/D

Descrizione	Predefinita	Adattabile	Self-service
Intenti per locale in ogni bot	• 1.000 in en-AU, en-GB e en-US • 250 in tutte le altre località	Sì	No
Slot per locale in ogni bot	• 4.000 in en-AU, en-GB e en-US • 2.000 in tutte le altre località	No	N/D
Tipi di slot personalizzati per locale del bot	• 250 in en-AU, en-GB e en-US • 100 in tutte le altre lingue	No	N/D
Valori e sinonimi dei tipi di slot personalizzati per locale in ogni bot	50.000	No	N/D
Caratteri totali negli enunciati di esempio per locale in ogni bot	• 2.000.000 in en-AU, en-GB e en-US • 200.000 in tutte le altre località	No	N/D
Associazioni di canale per alias bot	10	No	N/D
Slot per intento	100	No	N/D
Esempi di enunciati per intento	1.500	Sì	Sì
Caratteri per enunciato di esempio	500	No	N/D

Descrizione	Predefinita	Adattabile	Self-service
Lunghezza della risposta testuale	4.000	No	N/D
Esempi di enunciati per slot	10	Sì	Sì
Caratteri per slot di esempio (enunciato)	500	No	N/D
Richieste per slot	30	No	N/D
Valori e sinonimi per tipo di slot personalizzato	10.000	No	N/D
Caratteri per valore del tipo di slot personalizzato	500	No	N/D
Caratteri nel nome di un'associazione di canale	100	No	N/D
Numero di lavori di analisi simultanei di Automated Chatbot Designer su tutti i bot del tuo account per regione	10	No	N/D
Dimensione del file XML di tipo slot grammaticale personalizzato	100 KB	No	N/D

Quote di runtime

Le seguenti quote massime vengono applicate in fase di esecuzione.

Descrizione	Predefinita	Adattabile	Self-service
Dimensione del testo di input per e Recognize TextRecognize Utterance	1024 caratteri	No	N/D
Lunghezza dell'input vocale per il `RecognizeUtterance` funzionamento	15 secondi	Sì	No
Dimensione delle `RecognizeUtterance` intestazioni	16 KB	No	N/D
Dimensione delle intestazioni combinate di richiesta e sessione per `RecognizeUtterance`	12 KB	No	N/D
Numero massimo di conversazioni simultanee in modalità testo per `RecognizeText` `RecognizeUtterance` , o per `StartConv`	2	No	N/D

Descrizione	Predefinita	Adattabile	Self-service
`ersation` TestBotAlias			
Numero massimo di conversazioni simultanee in modalità testo per o per altri alias `Recognize Text` `Recognize Utterance` `StartConv ersation`	50	Sì	No
Numero massimo di conversazioni simultanee in modalità vocale per `Recognize Utterance` TestBotAlias	2	No	N/D
Numero massimo di conversazioni simultanee in modalità vocale per altri alias `Recognize Utterance`	125	Sì	No
Numero massimo di conversazioni simultanee in modalità vocale per `StartConv ersation` TestBotAlias	2	No	N/D

Descrizione	Predefinita	Adattabile	Self-service
Numero massimo di conversazioni simultanee in modalità vocale per altri alias `StartConversation`	200	Sì	No
Numero massimo di operazioni simultanee di gestione delle sessioni (`PutSession` ,, `GetSession` o) quando si utilizza `DeleteSession` TestBotAlias	2	No	N/D
Numero massimo di operazioni simultanee di gestione delle sessioni (`PutSession` ,`GetSession` , o`DeleteSession`) quando si utilizzano altri alias	50	Sì	No
Dimensione massima di input per una funzione Lambda	12 KB	No	N/D
Dimensione massima di output di una funzione Lambda	50 KB	No	N/D

Descrizione	Predefinita	Adattabile	Self-service
Dimensione massima degli attributi di sessione nell'outp ut della funzione Lambda (dopo la codifica in base 64)	12 KB	No	N/D
Timeout massimo di una funzione Lambda	30 secondi	Sì	No

Guida per la migrazione Amazon Lex V1 verso V2

La console e le API di Amazon Lex V2 semplificano la creazione e la gestione dei bot. Usa questa guida per scoprire i miglioramenti apportati all'API Amazon Lex V2 durante la migrazione dei bot.

Esegui la migrazione di un bot utilizzando la console o l'API di Amazon Lex. Per ulteriori informazioni, consulta Migrating a bot nella guida per sviluppatori di Amazon Lex.

Panoramica Amazon Lex V2

È possibile aggiungere più lingue a un bot in modo da poterle gestire come un'unica risorsa. Un'architettura informativa semplificata consente di gestire in modo efficiente le versioni dei bot. Funzionalità come il «flusso di conversazione», il salvataggio parziale della configurazione del bot e il caricamento in blocco degli enunciati offrono maggiore flessibilità.

Più lingue in un bot

Puoi aggiungere più lingue con l'API Amazon Lex V2. Puoi aggiungere, modificare e creare ogni lingua in modo indipendente. Le risorse, come i tipi di slot, sono classificate a livello linguistico. Puoi passare rapidamente da una lingua all'altra per confrontare e perfezionare le conversazioni. Puoi utilizzare una dashboard nella console per esaminare gli enunciati in tutte le lingue per analisi e iterazioni più rapide. Un operatore bot può gestire le autorizzazioni e le operazioni di registrazione per tutte le lingue con un'unica configurazione del bot. È necessario fornire una lingua come parametro di runtime per conversare con un bot Amazon Lex V2. Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.

Architettura delle informazioni semplificata

L'API Amazon Lex V2 segue un'architettura informativa semplificata (IA) con intenti e tipi di slot riferiti a una lingua. La versione viene eseguita a livello di bot in modo che le risorse come gli intenti e i tipi di slot non vengano versionate singolarmente. Per impostazione predefinita, un bot viene creato con una versione Draft che è mutabile e utilizzata per testare le modifiche. È possibile creare istantanee numerate a partire dalla versione bozza. Scegli le lingue da includere in una versione. Tutte le risorse all'interno del bot (lingue, intenti, tipi di slot) vengono archiviate come parte della creazione di una versione del bot. Per ulteriori informazioni, consulta Versioni.

Produttività dei costruttori

Disponi di strumenti e funzionalità aggiuntivi per la produttività dei costruttori che ti offrono maggiore flessibilità e controllo del processo di progettazione dei bot.

Salva configurazione parziale

L'API Amazon Lex V2 consente di salvare modifiche parziali durante lo sviluppo. Ad esempio, è possibile salvare uno slot che fa riferimento a un tipo di slot eliminato. Questa flessibilità consente di salvare il lavoro e di riprenderlo in un secondo momento. Puoi risolvere queste modifiche prima di creare il bot. In Amazon Lex V2 il salvataggio parziale può essere applicato a slot, versioni e alias.

Ridenominazione delle risorse

Con Amazon Lex V2 puoi rinominare una risorsa dopo averla creata. Utilizza un nome di risorsa per associare metadati intuitivi a ciascuna risorsa. L'API Amazon Lex V2 assegna a ogni risorsa un ID di risorsa univoco di 10 caratteri. Tutte le risorse hanno un nome di risorsa. Puoi rinassegnare le seguenti risorse:

- Bot
- Intento
- Tipo di slot
- Slot
- Alias

Puoi utilizzare gli ID risorsa per leggere e modificare le risorse per leggere e modificare le risorse per cercare le risorse per cercare le risorse Se stai utilizzando l'AWS Command Line InterfaceAPI Amazon Lex V2 in combinazione con Amazon Lex V2, l'uso degli ID risorsa è obbligatorio per determinati comandi.

Gestione semplificata delle funzioni Lambda

Nell'API Amazon Lex V2 definisci una funzione Lambda per lingua anziché una funzione per ogni intento. La funzione Lambda è configurata nell'alias della lingua e viene utilizzata sia per la finestra di dialogo che per l'hook del codice di evasione. Puoi comunque scegliere di abilitare o disabilitare la finestra di dialogo e gli hook del codice di adempimento indipendentemente per ogni intento. Per ulteriori informazioni, consulta Abilitazione della logica personalizzata con AWS Lambda funzioni.

Impostazioni granulari

L'API Amazon Lex V2 sposta la soglia del punteggio di affidabilità della classificazione vocale e degli intenti dal bot all'ambito linguistico. Il flag di analisi del sentiment si sposta dall'ambito del bot all'ambito degli alias. Le impostazioni di timeout della sessione e della privacy nell'ambito del bot e i registri delle conversazioni nell'ambito dell'alias rimangono invariati.

Intento di fallback predefinito

L'API Amazon Lex V2 aggiunge un intento di fallback predefinito quando crei una lingua. Usalo per configurare la gestione degli errori per il tuo bot invece di specifici prompt di gestione degli errori.

Aggiornamento ottimizzato delle variabili di sessione

Con l'API Amazon Lex V2 puoi aggiornare lo stato della sessione direttamente con le RecognizeUtteranceoperazioni RecognizeTextand senza alcuna dipendenza dalle API di sessione.

Creazione di risorse Amazon Lex V2AWS CloudFormation

Amazon Lex V2AWS CloudFormationAWS Puoi creare un modello che descrive tutte leAWS risorse desiderate (come iAWS CloudFormation chatbot Amazon Lex V2

Quando si utilizzaAWS CloudFormation, è possibile riutilizzare il modello per configurare le risorse Amazon Lex V2 Basta descrivere le risorse una volta sola, dopodiché si può effettuare il provisioning di tali risorse quante volte si vuole in più Account AWS e regioni.

Amazon Lex V2AWS CloudFormation

Prima di poter effettuare il provisioning e la configurazione delle risorse per Amazon Lex V2AWS CloudFormation I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui intendi effettuare il provisioning negli stack AWS CloudFormation. Se non hai familiarità con JSON o YAML, puoi usare AWS CloudFormation Designer per iniziare a utilizzare i modelli AWS CloudFormation. Per ulteriori informazioni, consulta Che cos'è AWS CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation.

Amazon Lex V2 supporta la creazione delle seguenti risorse inAWS CloudFormation:

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

Per ulteriori informazioni, inclusi esempi di modelli JSON e YAML per le risorse, consulta Riferimento dei tipi di risorse Amazon Lex V2AWS CloudFormation

Ulteriori informazioni su AWS CloudFormation

Per ulteriori informazioni su AWS CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- AWS CloudFormationguida per l'utente
- Documentazione di riferimento dell'API AWS CloudFormation
- AWS CloudFormationGuida per l'utente dell'interfaccia a riga di comando

Cronologia dei documenti per Amazon Lex V2

- Ultimo aggiornamento della documentazione: 10 maggio 2024

La tabella seguente descrive le modifiche importanti in ogni versione di Amazon Lex V2. Per ricevere notifiche sugli aggiornamenti di questa documentazione, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Aggiornamento della politica AWS gestita	Amazon Lex V2 ha aggiunto nuove autorizzazioni alla policy AmazonLex ReadOnlygestita per consentire l'accesso in lettura alle risorse bot che sono state replicate in altre regioni.	10 maggio 2024
Aggiornamento della politica AWS gestita	Amazon Lex V2 ha aggiunto nuove autorizzazioni alla policy AmazonLexFullAccesssgestita per consentire l'aggiornamento delle risorse bot replicate in altre regioni.	15 aprile 2024
Espansione regionale	Amazon Lex V2 è ora disponibile in AWS GovCloud (Stati Uniti occidentali) (us-gov-west-1).	22 marzo 2024
Aggiornamento alla politica AWS gestita	Amazon Lex V2 ha aggiunto nuove autorizzazioni alla policy AmazonLexReplicationPolicygestita per consentire l'aggiornamento delle risorse bot replicate in altre regioni.	7 marzo 2024

Nuova funzione	Puoi utilizzare la funzione fn.length () per determinare la lunghezza del valore di un valore di stringa in Amazon Lex V2. Per ulteriori informazioni, consulta Conditional Branching - Functions.	4 marzo 2024
Aggiornamento alla funzionalità	Lo slot integrato QnA per le funzionalità di intelligenza artificiale generativa in Amazon Lex V2 è ora GA. Per ulteriori informazioni, consulta Ottimizzare la creazione e le prestazioni dei bot con l'intelligenza artificiale generativa.	28 febbraio 2024
Aggiornamento della politica AWS gestita	Amazon Lex V2 ha aggiunto nuove autorizzazioni alla policy AmazonLexReplicationPolicygestita per consentire l'aggiornamento delle risorse bot replicate in altre regioni.	28 febbraio 2024
Aggiornamento della politica AWS gestita	Amazon Lex V2 ha aggiunto nuove autorizzazioni alla policy AmazonLexFullAccessgestita per consentire la replica delle risorse dei bot in altre regioni.	8 febbraio 2024

Nuova politica gestita	Amazon Lex V2 ha aggiunto una policy gestita che fornisce le autorizzazioni per replicare le risorse dei bot in altre regioni. Per ulteriori informazioni, consulta. AmazonLexReplicationPolicy	8 febbraio 2024
Nuova caratteristica	Puoi utilizzare la resilienza globale per replicare il tuo bot in una seconda regione AWS in Amazon Lex V2. Per ulteriori informazioni, consulta Resilienza globale.	8 febbraio 2024
Nuova caratteristica	Ora puoi sfruttare le funzionalità di intelligenza artificiale generativa di Amazon Lex V2. Per ulteriori informazioni, consulta Ottimizzare la creazione e le prestazioni dei bot con l'intelligenza artificiale generativa.	29 novembre 2023
Nuova caratteristica	Amazon Lex V2 ora può utilizzare la registrazione selettiva per acquisire testo e/o audio a livello di intento o di slot. Per ulteriori informazioni, consulta Registrazione selettiva.	8 novembre 2023

Nuova caratteristica	Amazon Lex V2 ora può utilizzare uno slot integrato per determinare le risposte Sì/No/Forse/Non so utilizzando. `AMAZON.Confirmation` Per ulteriori informazioni, consulta Tipi di slot integrati.	17 agosto 2023
Nuova caratteristica	Puoi visualizzare le metriche prestazionali degli intenti e degli slot, oltre ad altre metriche conversazionali, utilizzando la dashboard di analisi. Per ulteriori informazioni, consulta Analytics.	18 luglio 2023
Nuova caratteristica	Puoi migliorare la precisione e il successo di esecuzione del tuo bot con Test Workbench. Per ulteriori informazioni, consulta Test Workbench.	6 giugno 2023
Nuova caratteristica	Ora puoi creare bot partendo da un modello per alcuni mercati verticali aziendali più diffusi. Per ulteriori informazioni, consulta Modelli di bot.	23 febbraio 2023
Nuova caratteristica	Ora puoi combinare più bot in una rete di bot per creare un'esperienza cliente integrata. Per ulteriori informazioni, consulta Rete di bot.	9 febbraio 2023

Nuova caratteristica	Amazon Lex V2 ora supporta le versioni locali del Golfo Arabico (Emirati Arabi Uniti), cantonese (Hong Kong), finlandese (Finlandia), norvegese (Norvegia), polacca (Polonia) e svedese (Svezia). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	6 dicembre 2022
Aggiornato alla politica AWS gestita	Amazon Lex V2 ha aggiunto nuove autorizzazioni alla policy AmazonLex ReadOnlygestita per consentire la visualizzazione di voci di vocabolario personalizzate.	29 novembre 2022
Nuova caratteristica	Amazon Lex V2 può visualizzare una rappresentazione alternativa di una frase o di una parola utilizzando la console o le API per personalizzare l'output da voce a testo. Per ulteriori informazioni, consulta Creazione di un vocabolario personalizzato per il riconoscimento vocale.	7 novembre 2022

Nuova caratteristica	Amazon Lex V2 può aggiunger e un attributo weight a un elemento item che rappresen terà il grado di potenziam ento della frase durante il riconoscimento vocale. Per ulteriori informazioni, consulta Grammar Weights.	28 ottobre 2022
Nuova caratteristica	Amazon Lex V2 può essere utilizzato per acquisire input in formato libero dall'utente finale composto da parole o caratteri utilizzando`AMAZON.Fr eeFormInput` . Per ulteriori informazioni, consulta Tipi di slot integrati.	19 ottobre 2022
Nuova caratteristica	Amazon Lex V2 può visualizz are una rappresentazione alternativa di una frase o di una parola nell'output finale del discorso rispetto al testo. Per ulteriori informazioni, consulta Creazione di un vocabolario personalizzato per il riconoscimento vocale.	19 ottobre 2022
Nuova caratteristica	Amazon Lex V2 ora supporta le versioni locali hindi (India) e olandese (Paesi Bassi). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	14 ottobre 2022

Nuova caratteristica	È stato aggiornato il modo in cui Amazon Lex V2 gestisce l'input degli utenti. Ora puoi scegliere se Amazon Lex V2 accetta input di testo, audio o DTMF in qualsiasi momento del flusso di conversazione. Per ulteriori informazioni, consulta Attributi configurabili.	22 settembre 2022
Nuova caratteristica	È stato aggiornato il modo in cui Amazon Lex V2 gestisce i flussi di conversazione. Visual Conversation Builder è un generatore di conversazioni drag and drop per progettar e e visualizzare facilmente i percorsi di conversazione. Per ulteriori informazioni, consulta Visual Conversation Builder.	14 settembre 2022
Nuova caratteristica	È stato aggiornato il modo in cui Amazon Lex V2 crea slot complessi. Ora puoi creare sottoslot complessi all'inter no degli slot per gestire gli intenti nella progettazione di conversazioni complesse. Per ulteriori informazioni, consulta Creazione di slot compositi.	9 settembre 2022

Nuova caratteristica	È stato aggiornato il modo in cui Amazon Lex V2 gestisce il flusso di percorsi di conversazione con gli utenti. Ora puoi creare percorsi di conversazione complessi ordinando la fase successiv a della conversazione. Per ulteriori informazioni, consulta Creazione di percorsi di conversazione.	17 agosto 2022
Nuova caratteristica	È stato aggiornato il modo in cui Amazon Lex V2 gestisce il flusso di conversazioni con gli utenti. Ora puoi creare conversazioni complesse ordinando le istruzioni. Per ulteriori informazioni, consulta Configurazione dei prompt.	5 luglio 2022
Nuova caratteristica	È stato aggiornato il modo in cui Amazon Lex V2 gestisce il flusso di conversazioni con gli utenti. Ora puoi creare conversazioni complesse utilizzando le condizioni. Per ulteriori informazioni, consulta Comprendere i nuovi flussi di conversazione.	3 maggio 2022
Nuova caratteristica	Sono stati aggiunti esempi di grammatiche di settore per il tipo di slot grammatic ale integrato. Per ulteriori informazioni, consulta Grammatiche di settore.	22 marzo 2022

Nuova caratteristica	È stata aggiunta documentazione sull'integrazione di Amazon Lex V2 con l'SDK Amazon Chime. Per ulteriori informazioni, consulta Amazon Chime SDK.	18 marzo 2022
Nuova caratteristica	Amazon Lex V2 ora fornisce punteggi di affidabilità per le trascrizioni vocali. Usa il punteggio per determinare la risposta corretta dell'utente. Per ulteriori informazioni, vedere Utilizzo dei punteggi di confidenza nella trascrizione vocale.	27 gennaio 2022
Nuova caratteristica	Ora puoi aggiungere suggerimenti contestuali e dinamici agli slot per migliorare la precisione del tuo bot. Per ulteriori informazioni, consulta Usare i suggerimenti per migliorare la precisione.	13 gennaio 2022
Nuova caratteristica	Amazon Lex V2 aggiunge il supporto per vocabolari personalizzati per migliorare il riconoscimento vocale per l'ingresso audio. Per ulteriori informazioni, consulta Creare un vocabolario personalizzato per migliorare il riconoscimento vocale.	12 gennaio 2022

Nuova caratteristica	Amazon Lex V2 ora supporta AWS PrivateLink. Per ulteriori informazioni, consulta Endpoint VPC (AWS). PrivateLink	7 gennaio 2022
Nuova caratteristica	Amazon Lex V2 ora supporta la versione locale catalana (Spagna). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	3 gennaio 2022
Nuova caratteristica	Ora puoi creare tipi di slot usando la tua grammatica personalizzata. Per ulteriori informazioni, consulta Utilizzo di un tipo di slot grammaticale personalizzato.	20 dicembre 2021
Nuova caratteristica	AWS CloudFormation ora supporta Amazon Lex V2. Per ulteriori informazioni, consulta AWS CloudFormation le risorse.	20 dicembre 2021
Nuova caratteristica	Amazon Lex V2 ora supporta le versioni locali portoghese (Brasile), portoghese (Portogallo) e mandarino (RPC). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	16 dicembre 2021

Nuova caratteristica	Amazon Lex V2 ora offre un'anteprima di Automated Chatbot Designer per aiutarti a iniziare a creare un chatbot partendo dalle trascrizioni dei contact center. Per ulteriori informazioni, consulta Utilizzo di Automated Chatbot Designer (anteprima).	1° dicembre 2021
Nuova caratteristica	Ora puoi utilizzare spell-by-word stili spell-by-letter e stili per l'immissione di valori di slot che Amazon Lex V2 ha difficoltà a comprendere. Per ulteriori informazioni, consulta Uso degli stili ortografici per acquisire i valori degli slot.	19 novembre 2021
Nuova caratteristica	Ora puoi usare le voci di sintesi vocale neurale (NTTS) di Amazon Polly per conversazioni audio con i tuoi utenti. Per ulteriori informazioni, consulta Voices in Amazon Polly (Voci in Amazon Polly).	19 novembre 2021
Nuova caratteristica	Amazon Lex V2 ora supporta la versione locale in inglese (Sudafrica). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	9 novembre 2021

Nuova caratteristica	Amazon Lex V2 ora supporta la versione locale tedesca (Austria). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	5 novembre 2021
Nuova caratteristica	Ora puoi fornire agli utenti messaggi di aggiornamento che vengono riprodotti all'inizio di una funzione di adempimento e periodicamente durante l'esecuzione della funzione. Puoi anche creare messaggi che informano l'utente dello stato di adempimento quando la funzione è completa. Per ulteriori informazioni, consulta Configurazione degli aggiornamenti sullo stato di avanzamento dell'evasione degli ordini.	7 ottobre 2021
Espansione regionale	Amazon Lex V2 è ora disponibile in Africa (Città del Capo) (af-south-1) e Asia Pacifico (Seoul) (ap-northeast-2).	22 settembre 2021
Nuova caratteristica	Ora puoi visualizzare le statistiche relative agli enunciati che gli utenti inviano al tuo bot. Per ulteriori informazioni, vedete Visualizzazione delle statistiche sugli enunciati.	22 settembre 2021

Nuova caratteristica	Amazon Lex V2 ora supporta la versione locale coreana (Corea). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	9 settembre 2021
Nuova caratteristica	Amazon Lex V2 ora offre un tipo di slot integrato per i codici postali del Regno Unito. Per ulteriori informazioni, consulta PostalCodeAMAZON.UK.	27 luglio 2021
Nuova caratteristica	Amazon Lex V2 ora supporta la versione locale inglese (indiana). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	15 luglio 2021
Nuova caratteristica	Amazon Lex V2 ora fornisce uno strumento per migrare un bot da Amazon Lex V1 all'API Amazon Lex V2. Per ulteriori informazioni, consulta Migrazione di un bot nella Amazon Lex Developer Guide.	13 luglio 2021
Nuova caratteristica	Amazon Lex V2 ora consente di accettare più valori per un singolo slot in lingua inglese (Stati Uniti). Per ulteriori informazioni, consulta Utilizzo di più valori in uno slot.	15 giugno 2021

Nuova caratteristica	Ora puoi creare bot più grandi per le lingue inglesi. Per ulteriori informazioni, consulta Quote .	11 giugno 2021
Nuova caratteristica	Utilizza le policy basate sulle risorse di Amazon Lex V2 per gestire l'accesso ai bot e agli alias dei bot. Per ulteriori informazioni, consulta le politiche basate sulle risorse all'interno di Amazon Lex V2.	20 maggio 2021
Nuova caratteristica	Amazon Lex V2 ora consente di importare ed esportare bot e bot locali. Puoi utilizzare questa funzionalità per copiare bot e impostazioni locali dei bot tra account e regioni. AWS Per ulteriori informazioni, consulta Importazione ed esportazione.	18 maggio 2021
Espansione regionale	Amazon Lex V2 è ora disponibile in Canada (Central) (ca-central-1).	17 maggio 2021
Nuova caratteristica	Amazon Lex V2 ora supporta la versione locale giapponese (Giappone). Per ulteriori informazioni, consulta Lingue e impostazioni locali supportate da Amazon Lex V2.	1 aprile 2021

Nuova caratteristica	Amazon Lex V2 ora supporta tre nuovi tipi di slot integrati: `AMAZON.City` `AMAZON.Country` , e`AMAZON.State` .	12 marzo 2021
Nuova guida	Questa è la prima versione della guida per l'utente di Amazon Lex V2.	21 gennaio 2021

Riferimento API

L'API Reference è ora un documento separato.

Glossario per AWS

Per la terminologia AWS più recente, consultare il glossario AWS nella documentazione di riferimento per Glossario AWS.

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.