aws

do Amazon Lex V2

Guia do desenvolvedor

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

Guia do desenvolvedor: do Amazon Lex V2

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

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é Amazon Lex V2?

O Amazon Lex V2 é um serviço da AWS para a criação de interfaces de conversa para qualquer aplicação que usa voz e texto. O Amazon Lex V2 fornece a funcionalidade e a flexibilidade avançadas de compreensão de linguagem natural (NLU) e o reconhecimento automático de fala (ASR) para permitir a criação de experiências do usuário muito interessantes com interações por conversa realistas e a criação de novas categorias de produtos.

O Amazon Lex V2 permite que qualquer desenvolvedor crie bots de conversa rapidamente. Com o Amazon Lex V2 não é necessária nenhuma experiência em aprendizado profundo. Para criar um bot, você só precisa especificar o fluxo de conversa básico no console do Amazon Lex V2. O Amazon Lex V2 gerencia o diálogo e ajusta dinamicamente as respostas na conversa. Usando o console, você pode criar, testar e publicar o chatbot de texto ou voz. Em seguida, você pode adicionar as interfaces de conversa aos bots em dispositivos móveis, aplicativos Web e plataformas de bate-papo (por exemplo, Facebook Messenger).

O Amazon Lex V2 fornece integração com o AWS Lambda e você pode se integrar a muitos outros serviços na plataforma da AWS, incluindo Amazon Connect, Amazon Comprehend e Amazon Kendra. A integração com o Lambda fornece aos bots acesso a conectores empresariais com tecnologia sem servidor, pré-criados para vinculação a dados em aplicações SaaS, como o Salesforce.

Para bots criados após 17 de agosto de 2022, você pode usar a ramificação condicional para controlar o fluxo de conversas com seu bot. Com a ramificação condicional, você pode criar conversas complexas sem precisar gravar código Lambda.

O Amazon Lex V2 fornece os seguintes benefícios:

- Simplicidade: o Amazon Lex V2 orienta você durante o uso do console para criar seu próprio chatbot em minutos. Você fornece apenas algumas frases de exemplo e o Amazon Lex V2 cria um modelo completo de linguagem natural por meio do qual o bot pode interagir usando voz e texto para fazer perguntas, obter respostas e concluir tarefas sofisticadas.

- Tecnologias de aprendizado profundo democratizadas: o Amazon Lex V2 fornece as tecnologias ASR e NLU para criar um sistema de compreensão de linguagem falada (SLU). Por meio da SLU, o Amazon Lex V2 analisa a entrada de linguagem natural falada e de texto, compreende a

intenção por trás da entrada e atende à intenção do usuário invocando a função apropriada do negócio.

O reconhecimento de fala e a compreensão da linguagem natural são alguns dos problemas mais difíceis de resolver em ciência da computação e exigem o treinamento de sofisticados algoritmos de aprendizado profundo em grandes volumes de dados e infraestrutura. O Amazon Lex V2 coloca as tecnologias de aprendizado profundo ao alcance de todos os desenvolvedores. Os bots do Amazon Lex V2 convertem fala em texto e entendem a intenção do usuário para gerar uma resposta inteligente, para que você possa se concentrar na criação de seu bots com valor agregado diferenciado para seus clientes e definir categorias de produtos totalmente novas por meio de interfaces de conversa.

- Implantação e escalabilidade simples: com o Amazon Lex V2, você pode criar, testar e implantar seus bots diretamente no console do Amazon Lex V2. O Amazon Lex V2 permite que você publique facilmente seus bots de voz ou texto para uso em dispositivos móveis, aplicações Web e serviços de bate-papo (por exemplo, o Facebook Messenger). O Amazon Lex V2 é escalado automaticamente. Você não precisa se preocupar com o provisionamento de hardware e o gerenciamento da infraestrutura para potencializar sua experiência de bot.

- Integração incorporada com a plataforma da AWS: o Amazon Lex V2 opera nativamente com outros serviços da AWS, como o AWS Lambda e o Amazon CloudWatch. Você pode aproveitar o poder da plataforma da AWS para segurança, monitoramento, autenticação do usuário, lógica de negócios, armazenamento e desenvolvimento de aplicativos móveis.

- Custo-benefício: com o Amazon Lex V2 não há custos iniciais nem taxas mínimas. Você será cobrado apenas pelas solicitações de texto ou fala feitas. A definição de preço conforme o uso e o baixo custo por solicitação fazem do serviço uma maneira econômica de criar interfaces de conversa. Com o nível gratuito do Amazon Lex V2, você pode testar o Amazon Lex V2 com facilidade e sem nenhum investimento inicial.

Pagar pelo Amazon Lex

O Amazon Lex V2 cobra somente pelas solicitações de texto ou fala que você faz. Este modelo fornece a você um serviço com custo variável que pode aumentar com seus negócios enquanto proporciona a você as vantagens de custos de infraestrutura da AWS. Para obter mais informações, consulte Preços do Amazon Lex.

Ao cadastrar-se na AWS, sua conta da AWS é automaticamente cadastrada em todos os produtos da AWS, inclusive no Amazon Lex. Entretanto, você será cobrado apenas pelos serviços que usar. Se você for um novo cliente do Amazon Lex, você pode começar a usar o Amazon Lex gratuitamente. Para obter mais informações, consulte Nível gratuito da AWS.

Para ver sua fatura, acesse o Painel de Billing and Cost Management no console da AWS Billing and Cost Management. Para saber mais sobre o faturamento da Conta da AWS, consulte o Guia do usuário doAWS Billing. Se tiver dúvidas sobre o faturamento da AWS e as Contas da AWS, entre em contato com o AWS Support.

Você é um usuário iniciante do Amazon Lex V2?

Se você estiver usando o Amazon Lex V2 pela primeira vez, recomendamos que leia as seções a seguir nesta ordem:

1. Como funciona: esta seção apresenta o Amazon Lex V2 e os atributos que você usa para criar um chatbot.
2. Conceitos básicos do Amazon Lex V2: nesta seção, você configura sua conta e testa o Amazon Lex V2.
3. Referência da API: esta seção contém detalhes sobre as operações da API.

Atributos mais recentes

Explore os atributos mais recentes do Amazon Lex V2 abaixo:

Tópicos

Suporte regional para AWS GovCloud (Oeste dos EUA)

O Amazon Lex V2 agora está disponível em AWS GovCloud (Oeste dos EUA).

- Endpoints e cotas do Amazon Lex

Recursos de IA generativa para o Amazon Lex V2

O Amazon Lex V2 agora permite que você aproveite os recursos de IA generativa do Amazon Bedrock para seu bot.

- Construtor de bots descritivo

  - Post de novidades
  - Documentação
- Resolução assistida de slots
  - Post de novidades
  - Documentação
- Geração de enunciados
  - Post de novidades
  - Documentação
- `AMAZON.QnAIntent` (Perguntas frequentes sobre conversação)
  - Post de novidades
  - Documentação
- AWS Postagem no blog sobre Machine Learning

Slot embutido AMAZON.Confirmation para desambiguidade Sim/Não/Talvez/Não sei.

O Amazon Lex V2 agora oferece o slot integrado `AMAZON.Confimation` para melhorar a precisão da confirmação do slot e das respostas Sim/Não/Talvez/Não sei.

- Documentação

Medir o desempenho dos negócios com o Analytics

O Amazon Lex V2 agora oferece aos usuários a capacidade de visualizar o desempenho de intenções e slots no painel do Analytics.

- Post de novidades
- Documentação

Avaliar o desempenho do bot com o Test Workbench

O Amazon Lex V2 agora oferece aos usuários a capacidade de criar e executar conjuntos de testes para medir o desempenho dos bots e melhorar as métricas dos bots.

- Post de novidades
- Documentação
- AWS Postagem no blog sobre Machine Learning

Modelos de bots verticais específicos

O Amazon Lex V2 agora oferece aos usuários modelos de bots pré-criados com fluxos de ready-to-use conversação junto com dados de treinamento e solicitações de diálogo, tanto para as modalidades de voz quanto para chat.

- Post de novidades
- Documentação

Rede de bots

O Amazon Lex V2 agora oferece aos usuários a capacidade de combinar vários bots em uma única rede e a capacidade de encaminhar solicitações para o bot apropriado com base na entrada do usuário.

- Post de novidades
- Documentação

Visual Conversation Builder

O Amazon Lex V2 agora oferece um criador de conversas de arrastar e soltar para criar e visualizar facilmente caminhos de conversação usando intenções em um ambiente visual avançado.

- Post de novidades
- Documentação
- AWS Postagem no blog sobre Machine Learning

Tipo de slot composto

O Amazon Lex V2 agora oferece aos usuários a capacidade de combinar vários slots em um slot composto usando expressões lógicas.

- Post de novidades
- Documentação

Ramificação condicional

O Amazon Lex V2 agora oferece aos usuários a capacidade de escrever condições para controlar melhor o caminho que os clientes percorrem em uma conversa com seu bot.

- Post de novidades
- Documentação

Criador de chatbot automatizado

O Amazon Lex V2 agora oferece aos usuários a opção de criar automaticamente um chatbot a partir de transcrições de conversas. Leia os exemplos de uso.

- Post de novidades
- Documentação
- AWS Postagem no blog sobre Machine Learning
- Página do Designer de Chatbot Automatizado do Amazon Lex

Dicas de runtime

O Amazon Lex V2 agora oferece aos usuários a opção de configurar dicas de runtime para melhorar o reconhecimento de frases e melhorar a elicitação dos valores dos slots.

- Post de novidades
- Documentação

Vocabulário personalizado

O Amazon Lex V2 agora oferece aos usuários a opção de criar um vocabulário personalizado, uma lista de frases que pode incluir nomes próprios ou palavras específicas do domínio, para o Amazon Lex V2 reconhecer na entrada de áudio.

- Post de novidades
- Documentação
- AWS Postagem no blog sobre Machine Learning

Tipo de slot de gramática

O Amazon Lex V2 agora oferece aos usuários a capacidade de criar gramáticas no formato XML seguindo a Speech Recognition Grammar Specification (SRGS) para coletar informações em uma conversa.

- Post de novidades
- Documentação
- Publicação no blog do AWS Machine Learning

Como funciona

Com o Amazon Lex V2, você cria aplicativos usando uma interface de texto ou fala para uma conversa com um usuário. Veja a seguir as etapas comuns para trabalhar com o Amazon Lex V2:

1. Crie um bot e adicione um ou mais idiomas. Configure o bot para que ele entenda o objetivo do usuário, inicie uma conversa com o usuário para obter informações e cumpra a intenção do usuário.
2. Teste o bot. Você pode usar o cliente da janela de teste fornecido pelo console do Amazon Lex V2.
3. Publique uma versão e crie um alias.
4. Implante o bot. Você pode implantar o bot nas suas próprias aplicações ou plataformas de mensagens, como Facebook Messenger ou Slack.

Antes de começar a usar, familiarize-se com os seguintes conceitos principais e a terminologia do Amazon Lex V2:

- Bot – um bot executa tarefas automatizadas, como pedir uma pizza, reservar um hotel, encomendar flores, e assim por diante. Um bot do Amazon Lex V2 possui os recursos de reconhecimento automático de voz (ASR) e compreensão de linguagem natural (NLU).

  Os bots do Amazon Lex V2 podem compreender a entrada do usuário fornecida por texto ou fala e conversar em linguagem natural.
- Idioma: um bot do Amazon Lex V2 pode ter conversas em um ou mais idiomas. Cada idioma é independente dos outros. Você pode configurar o Amazon Lex V2 para conversar com um usuário usando palavras e frases nativas. Para mais informações, consulte Idiomas e locais aceitos pelo Amazon Lex V2.
- Intenção – Uma intenção representa uma ação que o usuário deseja executar. Crie um bot para oferecer suporte a uma ou mais intenções relacionadas. Por exemplo, você pode criar um intent que peça pizza e bebidas. Para cada intenção, forneça as seguintes informações obrigatórias:
  - Nome do intent – um nome descritivo para o intent. Por exemplo, **`OrderPizza`**.
  - Utterances de amostra – Como um usuário pode transmitir o intent. Por exemplo, um usuário pode dizer "Posso pedir uma pizza" ou "Quero pedir uma pizza".

- Como cumprir o intent – como você deseja cumprir o intent depois que o usuário fornecer todas as informações necessárias. Recomendamos criar uma função do Lambda para atender ao intent.

  Você também pode configurar o intent para que o Amazon Lex V2 simplesmente retorne as informações de volta ao aplicativo cliente para executar o atendimento necessário.

Além de intents personalizados, o Amazon Lex V2 fornece intents integrados para configurar seu bot rapidamente. Para mais informações, consulte Intenções integradas.

O Amazon Lex sempre inclui um intent alternativo para cada bot. O intent alternativo é usado sempre que o Amazon Lex não consegue deduzir a intenção do usuário. Para mais informações, consulte AMAZON.FallbackIntent.

- Slot – um intent pode exigir zero ou mais slots ou parâmetros. Você adiciona slots como parte da configuração de intenção. Em runtime, o Amazon Lex V2 solicita ao usuário valores específicos do slot. O usuário deve fornecer valores para todos os slots necessários para que o Amazon Lex V2 possa atender à intenção.

  Por exemplo, o intent `OrderPizza` requer slots como tamanho da pizza, tipo de massa e número de pizzas. Para cada slot, você fornece o tipo de slot e uma ou mais solicitações para o Amazon Lex V2 enviar ao cliente para obter os valores do usuário. Um usuário pode responder com um valor de slot que contém palavras adicionais, como "pizza grande, por favor" ou "vamos querer a pequena". O Amazon Lex V2 ainda entende o valor do slot.

- Tipo de slot – Cada slot possui um tipo. Você pode criar seu próprio tipo de slot ou usar os integrados. Por exemplo, você pode criar e usar os seguintes tipos de slot para a intenção `OrderPizza`:

  - Tamanho – Com valores de enumeração `Small`, `Medium` e `Large`.
  - Massa – Com valores de enumeração `Thick` e `Thin`.

  O Amazon Lex V2 também fornece tipos de slot integrados. Por exemplo, `AMAZON.Number` é um tipo de slot integrado que você pode usar para o número de pizzas pedidas. Para mais informações, consulte Intenções integradas.

- Versão – É um snapshot numerado de seu trabalho que você pode publicar para uso em diferentes partes de seu fluxo de trabalho, como desenvolvimento, implementação beta e produção. Depois de criar uma versão, você pode usar um bot como ele existia quando a versão foi criada. Depois que você cria uma versão, ela permanece a mesma enquanto você continua a trabalhar em seu aplicativo.

- Alias – É um ponteiro para uma versão específica de um bot. Com um alias, você pode atualizar a versão que seus aplicativos cliente estão usando. Por exemplo, você pode apontar um alias para a versão 1 do seu bot. Quando estiver pronto para atualizar o bot, você publica a versão 2 e altera o alias para apontar para a nova versão. Como suas aplicações usam o alias ao invés de uma versão específica, todos os seus clientes obtêm a nova funcionalidade sem a necessidade de atualizações.

Para uma lista das regiões da AWS em que o Amazon Lex V2 está disponível, consulte Endpoints e cotas do Amazon Lex V2 na Referência geral da Amazon Web Services.

Idiomas e locais aceitos pelo Amazon Lex V2

O Amazon Lex V2 oferece suporte a uma série de idiomas e locais. Este tópico trata dos idiomas aceitos, dos recursos que trabalham com eles e das orientações específicas do idioma para melhorar o desempenho do seu bot.

Idiomas e locais compatíveis

O Amazon Lex V2 oferece suporte aos seguintes idiomas e locais.

Código	Idioma e local
ar_AE	Golfo Árabe (Emirados Árabes Unidos)
ca_ES	Catalão (Espanha)
de_AT	Alemão (Áustria)
de_DE	Alemão (Alemanha)
en_AU	Inglês (Austrália)
en_GB	Inglês (Reino Unido)
pt_IN	Inglês (Índia)
pt_BR	Inglês (EUA)
en_ZA	Inglês (África do Sul)

Código	Idioma e local
es_419	Espanhol (América Latina)
es_ES	Espanhol (Espanha)
es_US	Espanhol (EUA)
fi_FI	Finlandês (Finlândia)
fr_CA	Francês (Canadá)
fr_FR	Francês (França)
hi_IN	Hindi (Índia)
it_IT	Italiano (Itália)
ja_JP	Japonês (Japão)
ko_KR	Coreano (Coreia)
nl_NL	Holandês (Países Baixos)
no_NO	Norueguês (Noruega)
pl_PL	Polonês (Polônia)
pt_BR	Português (Brasil)
pt_PT	Português (Portugal)
sv_SE	Sueco (Suécia)
zh_CN	Mandarim (RPC)
zh_HK	Cantonês (Hong Kong)

Idiomas e locais aceitos pelos atributos do Amazon Lex V2

A tabela a seguir lista os atributos do Amazon Lex V2 limitados a determinados idiomas e locais. Todos os outros atributos do Amazon Lex V2 trabalham com todos os idiomas e locais.

Recurso	Idiomas e locais compatíveis
AMAZÔNIA. AlphaNumeric	Todos os idiomas e locais, exceto coreano (ko_KR)
AMAZON.KendraSearchIntent	Inglês (EUA) (en_US)
Melhorar o reconhecimento de fala com um vocabulário personalizado	Inglês (UK) (es_GB) Inglês (EUA) (en_US)
Designer de Chatbot Automatizado	Inglês (EUA) (en_US)
Disponibilidade de regiões	Os seguintes idiomas e locais não estão disponíveis nas regiões Ásia-Pacífico (Singapura) (ap-southeast-1) e África (Cidade do Cabo) (ap-south-1): • Golfo Árabe (Emirados Árabes Unidos) (ar_AE) • Catalão (Espanha) (ca_ES) • Finlandês (Finlândia) (fi_FI) • Hindi (Índia) (hi_IN) • Holandês (Países Baixos) (nl_NL) • Norueguês (Noruega) (no_NO) • Polonês (pl_PL) • Português (Brasil) (pt_BR) • Português (Portugal) (pt_PT) • Sueco (sv_SE) • Mandarim (PRC) (zh_CN) • Cantonês (Hong Kong) (zh_HK)

Recurso	Idiomas e locais compatíveis
Definir o contexto da intenção	Inglês (EUA) (en_US)
Tipo de slot de gramática	Inglês (Austrália) (en_AU) Inglês (UK) (es_GB) Inglês (EUA) (en_US)
Uso de vários valores em um slot	Inglês (EUA) (en_US)
Melhorar o reconhecimento dos valores de slots com sugestões de runtime	Inglês (UK) (es_GB) Inglês (EUA) (en_US)
Capturar valores de slots com estilos de soletração	Inglês (Austrália) (en_AU) Inglês (UK) (es_GB) Inglês (EUA) (en_US)
Usar pontuações de confiança	Inglês (UK) (es_GB) Inglês (EUA) (en_US)

Orientações de idiomas para o Amazon Lex V2

Para melhorar o desempenho do seu bot, siga essas diretrizes para os seguintes idiomas.

Árabe

A variedade de árabe na qual o Amazon Lex V2 é treinado é o árabe usado na região do Golfo Árabe. Lembre-se disso ao fornecer exemplos de declarações para seu bot. O árabe é escrito da direita para a esquerda.

Hindi

O Amazon Lex V2 atende usuários finais em hindi que alternam livremente entre hindi e inglês. Se você planeja criar um bot que faça essa troca de idiomas, recomendamos o seguinte:

- Na definição do bot, escreva palavras em inglês em alfabeto latino.

- Pelo menos 50% das declarações de exemplo devem representar a troca de idioma na mesma frase. Nesses enunciados, use a escrita Devanágari para palavras em hindi e a escrita latina para palavras em inglês (por exemplo, "मैं ticket book करना चाहता हूं।").
- Se você espera que os usuários se comuniquem com o bot usando palavras em hindi no alfabeto latino ou palavras em inglês na escrita Devanágari, inclua exemplos de palavras em hindi no alfabeto latino (por exemplo, “ticket book mujhe ek karni hai”) e palavras em inglês na escrita Devanágari (por exemplo,"मेझ टकिट की बुकंगि में मदद चाहएि") nas declarações de exemplo.
- Se você espera que os usuários se comuniquem com o bot usando frases totalmente em hindi ou em inglês, inclua exemplos de declarações totalmente em um dos idiomas (por exemplo, “I want to book a ticket”).

Regiões

Para uma lista das regiões AWS em que o Amazon Lex V2 está disponível, consulte Regiões e endpoints da AWS na Referência geral da AWS.

Conceitos básicos do Amazon Lex V2

O Amazon Lex V2 fornece operações de API que você pode integrar com os aplicativos existentes. Para uma lista de operações aceitas, consulte a Referência de APIs. Você pode usar qualquer uma das opções a seguir:

- SDK da AWS: ao usar os SDKs, suas solicitações ao Amazon Lex V2 são automaticamente assinadas e autenticadas usando as credenciais que você fornecer. Recomendamos que você use um SDK para compilar seu aplicativo.
- AWS CLI — Você pode usar o AWS CLI para acessar qualquer recurso do Amazon Lex V2 sem precisar escrever nenhum código.
- Console da AWS: o console é a maneira mais fácil de começar a testar e usar o Amazon Lex V2

Se você começou a usar o Amazon Lex V2 recentemente, recomendamos que leia Como funciona antes de continuar.

Tópicos

Etapa 1: configurar uma AWS conta e criar um usuário administrador

Antes de usar o Amazon Lex V2 pela primeira vez, conclua as seguintes tarefas:

1. Inscreva-se para AWS
2. Criar um usuário do IAM

Inscreva-se para AWS

Se você já tiver uma AWS conta, pule essa tarefa.

Quando você se inscreve no Amazon Web Services (AWS), sua AWS conta é automaticamente cadastrada em todos os serviços AWS, incluindo o Amazon Lex V2. Você será cobrado apenas pelos serviços que usar.

Com o Amazon Lex V2, você paga apenas pelos recursos que usa. Se você é um novo AWS cliente, pode começar a usar o Amazon Lex V2 gratuitamente. Para mais informações, consulte Nível de uso gratuito da AWS.

Se você já tiver uma AWS conta, vá para a próxima tarefa. Se você não tiver uma AWS conta, use o procedimento a seguir para criar uma.

Para criar uma AWS conta

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e digitar um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

Anote o ID AWS da sua conta porque você precisará dele para a próxima tarefa.

Criar um usuário do IAM

Serviços em AWS, como o Amazon Lex V2, exigem que você forneça credenciais ao acessá-los para que o serviço possa determinar se você tem permissões para acessar os recursos de propriedade desse serviço.

Crie uma conta de usuário do IAM para acessar sua conta do Amazon Lex V2:

- Use AWS Identity and Access Management (IAM) para criar um usuário do IAM
- Adicione o usuário a um grupo do IAM com permissões de administrador
- Conceda permissões de administrador ao usuário do IAM que você criou.

Em seguida, você pode acessar AWS usando uma URL especial e as credenciais do usuário do IAM.

Os exercícios de conceitos básicos deste guia pressupõem que você tenha um usuário (`adminuser`) com privilégios de administrador. Siga o procedimento para criar `adminuser` na conta.

Para criar um usuário administrador e fazer login no console

1. Crie um usuário administrador chamado `adminuser` em sua AWS conta. Para instruções, consulte Criar o primeiro usuário do IAM e o grupo de administradores no Guia do usuário do IAM.
2. Como usuário, você pode fazer login no AWS Management Console usando um URL especial. Para obter mais informações, consulte Como os usuários fazem login em sua conta no Guia do usuário do IAM.

Para mais informações sobre IAM, consulte o seguinte:

- AWS Identity and Access Management (IAM)
- Conceitos básicos
- Guia do usuário do IAM

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte

Qual usuário precisa de acesso programático?	Para	Por
		a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Próxima etapa

Etapa 2: conceitos básicos (console)

Etapa 2: conceitos básicos (console)

A maneira mais fácil de aprender a usar o Amazon Lex V2 é usando o console. Para começar, criamos os seguintes exercícios que usam o console:

- Exercício 1: criar um bot do Amazon Lex V2 usando um esquema, um bot predefinido que fornece toda a configuração de bot necessária. Você faz apenas um mínimo de trabalho para testar a end-to-end configuração.
- Exercício 2 — Analisar as estruturas JSON enviadas entre seu aplicativo cliente e um bot do Amazon Lex V2.

Tópicos

Exercício 1: criar um bot a partir de um exemplo

Neste exercício, você vai criar seu primeiro bot do Amazon Lex V2 e testá-lo no console do Amazon Lex V2. Para este exercício, você usará o exemplo OrderFlowers.

Visão geral do exemplo

Use o exemplo OrderFlowers para criar um bot do Amazon Lex V2. Para mais informações sobre a estrutura de um bot, consulte Como funciona.

- Intenção – OrderFlowers
- Tipos de slot – Um tipo de slot personalizado chamado `FlowerTypes` com valores de enumeração: `roses`, `lilies` e `tulips`.
- Slots – a intenção requer as seguintes informações (slots) antes de o bot cumprir a intenção.
  - `PickupTime` (tipo integrado AMAZON.TIME)
  - `FlowerType` (tipo personalizado FlowerTypes)
  - `PickupDate` (tipo integrado AMAZON.DATE)
- Utterance – os seguintes utterances de amostra indicam a intenção do usuário:
  - "Gostaria de escolher flores."
  - "Gostaria de pedir algumas flores."

- Prompts – Após o bot identificar a intenção, ele usa os seguintes prompts para preencher slots:
  - Prompt do slot `FlowerType` – "Que tipo de flores você deseja pedir?"
  - Prompt do slot `PickupDate` – "Em que dia você deseja que {FlowerType} seja selecionada?"
  - Prompt do slot `PickupTime` – "Em que hora você deseja que {FlowerType} seja selecionada?"
  - Declaração de confirmação – "OK, {FlowerType} estará pronto para entrega às {PickupTime} em {PickupDate}. Tudo bem?"

Criar um bot Amazon Lex V2 (console)

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha Criar bot.
3. Para o Método de criação, escolha Começar com um exemplo.
4. Na seção Exemplos de bots, escolha OrderFlowers na lista.
5. Na seção Configuração do bot, dê ao bot um nome e uma descrição opcional. O nome deve ser exclusivo na conta.
6. Na seção Permissões, escolha Criar um novo perfil com permissões básicas do Amazon Lex. Isso criará um perfil (IAM) do AWS Identity and Access Management com as permissões que o Amazon Lex V2 precisa para executar seu bot.
7. Na seção Lei de Proteção à Privacidade Online para Crianças (COPPA), escolha a resposta apropriada.
8. Nas seções Tempo limite da sessão e Configurações avançadas, deixe as configurações padrão.
9. Escolha Próximo. O Amazon Lex V2 cria seu bot.

Depois de criar seu bot, adicione um ou mais idiomas compatíveis com o bot. Um idioma contém os intents, os tipos de slots e os slots que o bot usa para conversar com os usuários.

Adicionar um idioma a um bot

1. Na seção Idioma, escolha um idioma compatível e adicione uma descrição.
2. Deixe os campos Interação de voz e Limite de pontuação de confiança da classificação do intent nos valores padrão.
3. Escolha Concluído para adicionar o idioma ao bot.

Depois de escolher Concluído, o console abrirá o editor de intents. Você pode usar o editor de intents para examinar os intents usados pelo bot. Quando terminar de examinar o bot, você pode testá-lo.

Testar o bot OrderFlowers

1. Na parte de cima da página, escolha Versão. Aguarde o término da compilação do bot.
2. Quando a compilação estiver concluída, escolha Testar para abrir a janela de teste.
3. Teste o bot. Comece a conversa com uma das declarações de exemplo, como “Eu gostaria de algumas flores”.

Próximas etapas

Agora que você criou seu primeiro bot usando um modelo, use o console para criar seu próprio bot. Para instruções sobre como criar um bot personalizado e sobre a criação de bots, consulte Compilar bots.

Exercício 2: revisar o fluxo da conversação

Neste exercício, você revisa as estruturas JSON que são enviadas entre seu aplicativo cliente e o bot do Amazon Lex V2 que você criou no Exercício 1: criar um bot a partir de um exemplo. A conversa usa a operação RecognizeText para gerar as estruturas JSON. O RecognizeUtterance traz as mesmas informações dos cabeçalhos HTTP na resposta.

As estruturas JSON são divididas por cada turno da conversa. Um turno é uma solicitação do aplicativo cliente e uma resposta do bot.

Turno 1

Durante o primeiro turno da conversa, o aplicativo cliente inicia a conversa com seu bot. O URI e o corpo da solicitação fornecem informações sobre a solicitação.

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

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

- O URI identifica o bot com o qual o aplicativo cliente está se comunicando. Também inclui um identificador de sessão gerado pelo aplicativo cliente que identifica uma conversa específica entre um usuário e o bot.
- O corpo da solicitação contém o texto que o usuário digitou no aplicativo cliente. Nesse caso, somente o texto é enviado, mas seu aplicativo pode enviar outras informações, como atributos da solicitação ou estado da sessão. Para mais informações, consulte a operação RecognizeText.

A partir de `text`, o Amazon Lex V2 detecta a intenção do usuário, que é pedir flores. O Amazon Lex V2 escolhe um dos slots da intenção (`FlowerType`) e uma das solicitações para o slot e, em seguida, envia a resposta abaixo ao aplicativo cliente. O cliente exibe a mensagem na resposta ao usuário.


```
{
    "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

No turno 2, o usuário responde à solicitação do bot do Amazon Lex V2 no turno 1 com um valor que preenche o slot de `FlowerType`.

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

A resposta para o turno 2 mostra o slot de `FlowerType` preenchido e fornece uma solicitação para obter o próximo valor do slot.

```
{
    "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

No turno 3, o usuário responde à solicitação do bot do Amazon Lex V2 no turno 2 com um valor que preenche o slot de `PickupDate`.

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

A resposta para o turno 3 mostra os slots de `FlowerType` e `PickupDate` preenchidos e fornece uma solicitação para obter o último valor do 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

No turno 4, o usuário fornece o valor final do slot para a intenção, a saber, a hora em que as flores são retiradas.

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

Na resposta, o Amazon Lex V2 envia uma solicitação de confirmação ao usuário para confirmar que o pedido está correto. A `dialogAction` fica definida como `ConfirmIntent` e o `confirmationState` fica como `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

No turno final, o usuário responde com a solicitação de confirmação.

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

Na resposta, os envios do Amazon Lex V2 indicam que o intent foi cumprido definindo `confirmationState` como `Confirmed` e `dialogAction` para fechar. Todos os valores de slot estão disponíveis para o aplicativo cliente.

```
{
    "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"
    }
}
```

Compilar bots

Você cria um bot Amazon Lex V2 para interagir com seus usuários e obter informações para realizar uma tarefa. Por exemplo, é possível criar um bot que reúna as informações necessárias para pedir um buquê de flores ou reservar um quarto de hotel.

Para compilar um bot, as seguintes informações são necessárias:

1. O idioma que o bot usa para interagir com o cliente. Você pode escolher um ou mais idiomas, cada idioma contém intenções, slots e tipos de slots independentes.
2. As intenções ou metas que o bot ajuda o usuário a realizar a atividade. Um bot pode conter uma ou mais intenções, como pedir flores ou reservar um hotel e alugar um carro. Você precisa decidir quais declarações, ou enunciados, o usuário faz para iniciar a intenção.
3. As informações, ou espaços, que você precisa coletar do usuário para cumprir uma intenção. Por exemplo, talvez você precise obter o tipo de flores do usuário ou a data de início de uma reserva de hotel. Você precisa definir uma ou mais solicitações que o Amazon Lex V2 usa para instigar o valor do slot do usuário.
4. O tipo de slots que você precisa do usuário. Talvez seja necessário criar um tipo de slot personalizado, como uma lista de flores que um usuário pode pedir, ou usar um tipo de slot integrado, como usar o tipo de slot `AMAZON.Date` para a data de início de uma reserva.
5. A interação do usuário flui dentro e entre as intenções. Você pode configurar o fluxo de conversação para definir a interação entre o usuário e o bot quando a intenção for invocada. Você pode criar uma função do Lambda para validar e atender à intenção.

Tópicos

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para obter mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Compreender o gerenciamento do fluxo de conversas

Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa.

Antes da mudança, o Amazon Lex V2 gerenciava a conversa selecionando vagas com base em suas prioridades de intenção. Você pode modificar esse comportamento dinamicamente e alterar o caminho da conversa com base nas entradas do usuário usando DialogAction na função do Lambda. Isso pode ser feito acompanhando o estado atual da conversa e decidindo programaticamente o que fazer a seguir com base no estado da sessão.

Com essa alteração, você pode criar caminhos conversacionais e ramificações condicionais usando o console Amazon Lex V2 ou APIs sem usar uma função do Lambda. O Amazon Lex V2 monitora o estado da conversa e controla o que fazer a seguir com base nas condições definidas quando o bot é criado. Isso permite que você crie facilmente conversas complexas ao projetar seu bot.

Essas mudanças oferecem controle total sobre a conversa com seu cliente. No entanto, você não precisa definir um caminho. Se você não especificar um caminho de conversação, o Amazon Lex V2 cria um caminho padrão com base na prioridade dos espaços em sua intenção. Você pode continuar usando as funções do Lambda para definir caminhos de conversação dinamicamente. Nesse cenário, a conversa é retomada com base no estado da sessão configurado na função do Lambda.

Esta atualização fornece o seguinte:

- Uma nova experiência de console para a criação de bots com fluxos de conversação complexos.
- Atualizações nas APIs existentes para criar bots para apoiar os novos fluxos de conversação.

- Uma resposta inicial para enviar uma mensagem sobre invocação de intenção.
- Novas respostas para elicitação de slots, invocação do Lambda como hook de código de diálogo e confirmação.
- Capacidade de especificar as próximas etapas em cada turno da conversa.
- Avaliação das condições para projetar vários caminhos de conversação.
- Configuração dos valores dos slots e dos atributos da sessão em qualquer momento da conversa.

Observe o seguinte para bots mais antigos:

- Os bots criados antes de 17 de agosto de 2022 continuam usando o mecanismo antigo para gerenciar fluxos de conversas. Os bots criados após essa data usam a nova forma de gerenciamento do fluxo de conversas.
- Novos bots criados por meio de importações após 17 de agosto de 2022 usam o novo gerenciamento de fluxo de conversas. As importações de bots existentes continuam usando a forma antiga de gerenciamento de conversas.
- Para ativar o novo gerenciamento do fluxo de conversas para um bot criado antes de 17 de agosto de 2022, exporte o bot e, em seguida, importe o bot usando um novo nome de bot. O bot recém-criado da importação usa o novo gerenciamento do fluxo de conversas.

Observe o seguinte para novos bots criados após 17 de agosto de 2022:

- O Amazon Lex V2 segue o fluxo de conversação definido exatamente como projetado para oferecer a experiência desejada. Você deve configurar todas as ramificações de fluxo para evitar caminhos de conversação padrão durante o runtime.
- As etapas de conversação que seguem um hook de código devem ser totalmente configuradas, pois etapas incompletas podem levar à falha do bot. Recomendamos que você valide os bots criados antes de 17 de agosto de 2022, porque, para esses bots, não há validação automática das etapas da conversa seguindo um hook de código.

Criar um bot

É possível criar um bot com o Amazon Lex V2 das seguintes maneiras:

1. Use o console do Amazon Lex V2 para criar um bot usando um esquema. Para obter mais informações, consulte Criar um bot usando o console Amazon Lex V2.

2. Use o Construtor de bots descritivo para criar um bot usando os recursos de IA generativa do Amazon Bedrock. Para obter mais informações, consulte Usar o construtor de bots descritivo.
3. Use modelos de bot para criar um bot pré-configurado que corresponda aos casos de uso comerciais comuns. Para obter mais informações, consulte Geração de bots predefinidos a partir de modelos de bots.
4. Use um AWSSDK para criar um bot usando operações de API.
5. Use o designer de chatbot automatizado para criar um bot usando as transcrições de bate-papo existentes entre atendentes e clientes. Para obter mais informações, consulte Uso do Automated Chatbot Designer.
6. Importe uma definição de bot existente. Para obter mais informações, consulte Importação.
7. Use AWS CloudFormation para criar um novo bot. Para obter mais informações, consulte Criar recursos do Amazon Lex V2 com o AWS CloudFormation.

Tópicos

- Criar um bot usando o console Amazon Lex V2
- Geração de bots predefinidos a partir de modelos de bots
- Uso do Automated Chatbot Designer

Criar um bot usando o console Amazon Lex V2

Comece a criar seu bot definindo o nome, a descrição e algumas informações básicas.

Para criar um bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha Criar bot.
3. Na seção Método de criação, selecione Criar.
4. Na seção Configuração do bot, dê ao bot um nome e uma descrição opcional.
5. Na seção de permissões do IAM, escolha um perfil do AWS Identity and Access Management (IAM) que forneça permissão ao Amazon Lex V2 para acessar outros serviços AWS, como o Amazon CloudWatch. Você pode fazer com que o Amazon Lex V2 crie o perfil ou você pode escolher um existente com as permissões do CloudWatch.

6. Na seção Lei de Proteção à Privacidade Online para Crianças (COPPA), escolha a resposta apropriada.
7. Na seção Tempo limite da sessão ociosa, escolha por quanto tempo o Amazon Lex V2 mantém uma sessão com um usuário aberto. O Amazon Lex V2 mantém variáveis de sessão durante a sessão para que seu bot possa retomar uma conversa com as mesmas variáveis.
8. Na seção Configurações avançadas, adicione tags que ajudem a identificar o bot e possam ser usadas para controlar o acesso e monitorar recursos.
9. Escolha Avançar para criar o bot e começar a adicionar um idioma.

Geração de bots predefinidos a partir de modelos de bots

O Amazon Lex V2 oferece soluções pré-criadas para criar experiências em grande escala e impulsionar o engajamento digital. Os modelos de bots pré-criados automatizam e padronizam as experiências dos clientes. Os modelos de bot fornecem fluxos de conversação prontos para uso, juntamente com dados de treinamento e solicitações de diálogo, tanto para as modalidades de voz quanto para chat. Você pode agilizar a entrega de soluções de bots e, ao mesmo tempo, otimizar os recursos, para poder se concentrar no relacionamento com os clientes.

Você pode criar bots pré-criados com base no seu caso de uso comercial. Você pode usar o console AWS CloudFormation para selecionar as opções pré-criadas para os serviços relacionados, como Amazon S3, Amazon Connect e DynamoDB.

Atualmente, o Amazon Lex V2 oferece suporte aos seguintes setores de negócios:

- serviços financeiros
- Pedidos de varejo
- Seguro automático
- Telecomunicações
- Serviços aéreos
- Mais em breve...

Você pode criar um bot com o modelo de solução comercial fornecido e personalizá-lo de acordo com suas necessidades comerciais.

> **Note**
>
> Os modelos criam recursos fora do Amazon Lex V2 por meio de pilhas AWS CloudFormation. Talvez seja necessário modificar a pilha em outros consoles, como Lambda e DynamoDB.

Pré-requisitos necessários para criar e implantar o modelo de bot:

- Uma conta da AWS
- Acesso aos seguintes serviços AWS:
  - Amazon Lex V2 para criar bots
  - Lambda para as funções de login comercial
  - DynamoDB para criar as tabelas
  - Criar políticas e perfis do IAM
  - AWS CloudFormation para executar a pilha
- Acesso ao IAM e credenciais de chave secreta
- Instância do Amazon Connect (opcional)

> **Note**
>
> O uso de diferentes serviços AWS incorre nos respectivos custos de uso de cada serviço.

Para criar um bot a partir dos modelos do Amazon Lex V2:

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Selecione o botão laranja que diz Criar bots a partir de um modelo.
3. Selecione qual vertical de negócios você deseja usar para o seu modelo de bot. OBSERVAÇÃO: Existem 5 modelos de bots disponíveis atualmente. Mais em breve...
4. Selecione Criar para o modelo que deseja usar. É aberta uma guia em AWS CloudFormation qual você pode editar os parâmetros da pilha AWS CloudFormation. Todas as opções já estão preenchidas para o modelo que você escolheu. Você também pode saber mais sobre como o modelo de bot funciona selecionando Saiba mais.

5. No console AWS CloudFormation, AWS CloudFormation cria uma configuração padrão para cada um dos valores do modelo que você escolheu. Você também pode selecionar seu próprio nome de pilha, parâmetros do AWS CloudFormation, tabela do Amazon DynamoDB e parâmetros (opcionais) do Amazon Connect.
6. Na parte inferior da janela, selecione Criar pilha.
7. AWS CloudFormation processa a solicitação em segundo plano por vários minutos para configurar seu novo bot. OBSERVAÇÃO: O processo cria automaticamente recursos para uma tabela do DynamoDB, um fluxo de contatos do Amazon Connect e uma instância do Amazon Connect. Você pode acompanhar o progresso no console AWS CloudFormation e voltar para o console Amazon Lex V2 quando a criação da pilha do CloudFormation for concluída.
8. Se criada com sucesso, uma mensagem aparece e você pode selecionar Ir para a lista de bots para acessar a página de bots, onde encontrará seu novo bot que está pronto para ser testado e usado.

Configurando seu modelo de bot

Funções do Lambda — O modelo de bot cria automaticamente as funções do Lambda necessárias para sua implantação. Se vários bots fizerem parte da solução de modelo, várias funções do Lambda serão listadas nos parâmetros AWS CloudFormation. Se você tiver funções do Lambda existentes para implantar com seu bot, poderá inserir o nome da sua função do Lambda personalizada.

Amazon DynamoDB — O modelo de bot cria automaticamente a tabela do DynamoDB necessária para carregar seus exemplos de dados de política. Você também pode inserir o nome da sua tabela personalizada do DynamoDB. Sua tabela personalizada do DynamoDB deve ser formatada da mesma forma que a tabela padrão criada pela implantação do modelo de bot.

Amazon Connect — Você pode configurar sua instância do Amazon Connect para funcionar com seu novo modelo de bot inserindo o ConnectInstanceARN e um ContactFlowName exclusivo. Com o uso do Amazon Connect, você pode testar seu bot usando um sistema IVR de ponta a ponta.

Solução de problemas do seu modelo de bot

- Verifique se você tem as permissões adequadas para criar o modelo que está escolhendo. Os usuários precisam da permissão CloudFormation:CreateStack junto com as permissões para os recursos listados no AWS modelo. Uma lista de recursos que precisam de permissões de usuário está na parte inferior da página Criar modelo.
- Se seu modelo de bot não for criado, o banner vermelho no console do Amazon Lex V2 fornecerá um link para a pilha AWS CloudFormation responsável pela criação do modelo. No console AWS

CloudFormation, você pode visualizar a guia de eventos para ver o erro específico que causou a falha do modelo. Depois de analisar o erro AWS CloudFormation, consulte Solução de problemas do CloudFormation para obter mais informações.

- Os modelos de bot funcionam somente com os dados de amostra. Você deve preencher a tabela do DynamoDB com seus dados para que os modelos funcionem com seus dados personalizados.

Uso do Automated Chatbot Designer

> **Note**
>
> Você só pode usar transcrições no idioma inglês (EUA).

O Automated Chatbot Designer ajuda a criar bots a partir de transcrições de conversas existentes. Ele analisa as transcrições e sugere um design inicial com intenções e tipos de slots. Você pode iterar o design do bot, adicionar prompts, criar, testar e implantar o bot.

Depois de criar um novo bot ou adicionar uma linguagem ao seu bot usando o console ou a API do Amazon Lex V2, você pode fazer upload de transcrições de conversas entre duas partes. O designer automatizado do chatbot analisa as transcrições e determina as intenções e os tipos de slots do bot. Também identifica as conversas que influenciaram a criação de uma intenção ou tipo de espaço específico para sua avaliação.

O console Amazon Lex V2 ou a API podem ser usados para analisar transcrições de conversas e sugerir intenções e tipos de slots para um bot.

Você pode revisar as intenções e os tipos de slots sugeridos depois que o designer do chatbot concluir a análise. Depois de adicionar uma intenção ou tipo de slot sugerido, é possível modificá-la ou excluí-la do design do bot usando o console ou a API.

O designer de chatbot automatizado oferece suporte a arquivos de transcrição de conversas usando o esquema Contact Lens for Amazon Connect. Se você estiver usando um aplicativo de contact center diferente, deverá transformar as transcrições das conversas no formato usado pelo designer do chatbot. Para obter mais informações, consulte Formato da transcrição de entrada.

Para usar o designer de chatbot automatizado, você deve permitir que o perfil do IAM que está executando o designer acesse. Para a política do IAM específica, consulte Permitir que os usuários usem o Automated Chatbot Designer. Para permitir que o Amazon Lex V2 criptografe dados de

saída com uma chave AWS KMS opcional, você precisa atualizar a chave com a política exibida em Permita que os usuários usem uma AWS KMS chave para criptografar e descriptografar arquivos.

> **Note**
>
> Se você usar um KMS key, deverá fornecer uma política KMS key, independentemente da função do IAM usada.

Tópicos

Importação de transcrições de conversas

Importar transcrições de conversas é um processo de três etapas:

1. Prepare as transcrições para importação convertendo-as para o formato correto. Se você estiver usando o Contact Lens para o Amazon Connect, as transcrições já estão no formato correto.
2. Faça upload do arquivo para um bucket do Amazon S3. Se você estiver usando o Contact Lens, suas transcrições já estão em um bucket S3.
3. Analise as transcrições usando o console do Amazon Lex V2 ou operações de API. O tempo necessário para concluir o treinamento depende do volume de transcrições e da complexidade da conversa. Normalmente, 500 linhas de transcrições são analisadas a cada minuto.

Cada uma dessas etapas é descrito nas seções a seguir.

Importação de transcrições do Contact Lens para o Amazon Connect

O designer de chatbot automatizado Amazon Lex V2 é compatível com arquivos de transcrição do Contact Lens. Para usar os arquivos de transcrição do Contact Lens, você deve ativar o Contact Lens e anotar a localização dos arquivos de saída.

Para exportar transcrições do Contact Lens

1. Ative a lente de contato na sua instância do Amazon Connect. Para obter instruções, consulte Habilitar o Contact Lens para o Amazon Connect no guia do administrador do Amazon Connect.
2. Observe a localização do bucket S3 que o Amazon Connect está usando para sua instância. Para ver a localização, abra a página Armazenamento de dados no console do Amazon Connect. Para obter instruções, consulte Atualizar configurações da instância no guia do administrador do Amazon Connect.

Depois de ativar o Contact Lens e anotar a localização dos arquivos de transcrição, acesse Analise suas transcrições usando o console do Amazon Lex V2 para obter instruções sobre como importar e analisar suas transcrições.

Preparar transcrições

Prepare suas transcrições criando arquivos de transcrição.

- Crie um arquivo de transcrição por conversa listando a interação entre as partes. Cada interação na conversa pode abranger várias linhas. Você pode fornecer versões editadas e não editadas da conversa.
- O arquivo deve estar no formato JSON especificado em Formato da transcrição de entrada.
- Você deve fornecer pelo menos 1.000 turnos de conversação. Para melhorar a descoberta de suas intenções e tipos de slots, você deve fornecer cerca de 10.000 ou mais turnos de conversação. O designer automatizado do chatbot processará apenas os primeiros 700.000 turnos.
- Não há limite para o número de arquivos de transcrição que você pode carregar, nem há uma restrição de tamanho de arquivo.

Se você planeja filtrar as transcrições importadas por data, os arquivos devem estar na seguinte estrutura de diretórios:

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

O arquivo de transcrição deve conter a data no formato “aaaa-mm-dd” em algum lugar do nome do arquivo.

Para exportar transcrições de outros aplicativos do contact center

1. Use as ferramentas do seu aplicativo do contact center para exportar conversas. A conversa deve conter pelo menos as informações especificadas em Formato da transcrição de entrada.
2. Transforme as transcrições produzidas pelo seu aplicativo de contact center no formato descrito em Formato da transcrição de entrada. Você é responsável por realizar a transformação.

Fornecemos três scripts para preparar as transcrições. Eles são:

- Um script para combinar as transcrições do Contact Lens com os logs de conversas do Amazon Lex V2. As transcrições do Contact Lens não incluem partes das conversas do Amazon Connect que interagem com os bots do Amazon Lex V2. O script exige que os registros de conversação sejam ativados para o Amazon Lex V2 e as permissões apropriadas para consultar o log de conversas nos buckets CloudWatch Logs e Contact Lens S3.
- Um script para transformar Amazon Transcribe Call Analytics no formato de entrada do Amazon Lex V2.
- Um script para transformar a as transcrições de conversa do Amazon Connect no formato de entrada do Amazon Lex V2.

Você pode baixar os scripts deste repositório do GitHub: https://github.com/aws-samples/amazon-lex-bot-recommendation-integration.

Faça upload do arquivo para um bucket S3

Se você estiver usando o Contact Lens, suas transcrições já estão em um bucket S3. Para saber a localização e os nomes dos arquivos de transcrição, consulte Exemplos de arquivos de saída de lentes de contato no guia do administrador do Amazon Connect.

Se estiver usando outro aplicativo de contact center e não tiver configurado um bucket S3 para seus arquivos de transcrição, siga este procedimento. Caso contrário, se já tiver um bucket do S3, depois de fazer login no console do Amazon S3, siga este procedimento a partir da etapa 5.

Para fazer upload de arquivos em um bucket do S3

1. Faça login no AWS Management Console e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Escolha Criar bucket.
3. Dê um nome ao bucket e escolha uma região. A região deve ser a mesma que você usa para o Amazon Lex V2. Configure as outras opções conforme necessário para seu caso de uso.
4. Escolha Criar bucket.
5. Na lista de buckets, escolha um bucket existente ou um que você acabou de criar
6. Escolha Carregar.
7. Adicione os arquivos de transcrição que você deseja carregar.
8. Escolha Carregar.

Analise suas transcrições usando o console do Amazon Lex V2

Você só pode usar o design automatizado de bots em um idioma vazio. Você pode adicionar um novo idioma a um bot existente ou criar um novo bot.

Para criar um novo idioma em um novo bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha Criar bot
3. Escolha Começar com o Automated Chatbot Designer. Preencha as informações para criar seu novo bot.
4. Escolha Avançar
5. Em Adicionar idioma ao bot, preencha as informações do idioma.
6. Na seção Localização do arquivo de transcrição no S3, escolha o bucket do S3 que contém seus arquivos de transcrição e o caminho local para os arquivos, se necessário.
7. Você pode escolher a seguinte opção:

   - Uma chave AWS KMS para criptografar os dados da transcrição durante o processamento. Se você não selecionar uma chave, uma chave AWS KMS de serviço será usada.

- Para filtrar as transcrições para um intervalo de datas específico. Se você optar por filtrar as transcrições, elas deverão estar na estrutura de pastas correta. Para mais informações, consulte Preparar transcrições.

8. Escolha Concluído.

Aguarde até que o Amazon Lex V2 processe a transcrição. Você verá uma mensagem de conclusão quando a análise for concluída.

Como parar de analisar sua transcrição

Caso precise interromper a análise das transcrições enviadas, você pode interromper um trabalho `BotRecommendation` em execução, que tem o status `BotRecommendationStatus` de processamento. Você pode clicar no botão Interromper processamento presente no banner depois de enviar um trabalho do console ou usando o CLI SDK para a API `StopBotRecommendation`. Para obter mais informações, consulte StopBotRecommendation

Depois de chamar `StopBotRecommendation`, o `BotRecommendationStatus` interno é configurado para `Stopping` e você não é cobrado. Para garantir que o trabalho tenha sido interrompido, você pode chamar a API `DescribeBotRecommendation` e verificar se `BotRecommendationStatus` está `Stopped`. Isso geralmente leva de 3 a 4 minutos.

Você não será cobrado pelo processamento após a chamada da API `StopBotRecommendation`.

Criação de intenções e tipos de slots

Depois que o designer do chatbot cria as intenções e os tipos de slots, você seleciona as intenções e os tipos de slots a serem adicionados ao seu bot. Você pode analisar os detalhes de cada intenção e tipo de slot para ajudá-lo a decidir quais recomendações são mais relevantes para seu caso de uso.

Você pode clicar no nome de uma intenção recomendada para ver os exemplos de frases e espaços sugeridos pelo designer do chatbot. Se você selecionar Mostrar transcrições associadas, também poderá percorrer as conversas que forneceu. Essas transcrições influenciam a recomendação do designer do chatbot sobre essa intenção. Se você clicar em um exemplo de enunciado, poderá revisar a conversa principal e o diálogo relevante, que influenciou esse enunciado específico.

Você pode clicar no nome de um tipo específico de slot para ver os valores de slot que foram recomendados. Se você selecionar Mostrar transcrições associadas, poderá revisar as conversas que influenciaram esse tipo de slot, destacando a solicitação do atendente que indica o tipo de slot.

Se você clicar em um valor de tipo de slot específico, poderá revisar a conversa principal e o diálogo relevante que influenciou esse valor.

Para revisar e adicionar intenções e tipo de slot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot com o qual você quer trabalhar.
3. Escolha Exibir idiomas.
4. Na lista de idiomas, escolha o idioma com o qual deseja trabalhar.
5. Em Estrutura da conversa, escolha Revisar.
6. Na lista de intenções e tipos de slots, escolha aqueles a serem adicionados ao bot. Você pode escolher uma intenção ou tipo de slot para ver os detalhes e as transcrições associadas.

As intenções são classificadas pela confiança que o Amazon Lex V2 tem de que a intenção está associada às transcrições processadas.

Formato da transcrição de entrada

A seguir está o formato do arquivo de entrada para gerar intenções e tipos de slots para seu bot. O arquivo de entrada deve conter esses campos. Outros campos são ignorados.

O formato de entrada é compatível com o formato de saída do Contact Lens para Amazon Connect. Se você estiver usando o Contact Lens, não será necessário modificar seus arquivos de transcrição. Para obter mais informações, consulte Exemplos de arquivos de saída do Contact Lens. Se você estiver usando outro aplicativo de contact center, deverá transformar seu arquivo de transcrição nesse 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"
        }
    ]
}
```

Os campos a seguir devem estar presentes no arquivo de entrada:

- Participantes Identifica os participantes da conversa e o papel que eles desempenham.
- Versão A versão do formato do arquivo de entrada. Sempre “1.1.0”.
- ContentMetadata Indica se você removeu informações confidenciais da transcrição. Defina o campo `Output` como “Bruto” se a transcrição contiver informações confidenciais.
- CustomerMetadata Um identificador exclusivo para a conversa.
- Transcrição O texto da conversa entre as partes na conversa. Cada turno da conversa é identificado com um identificador exclusivo.

Formato da transcrição de entrada

O formato da transcrição de saída é quase igual ao formato da transcrição de entrada. No entanto, também inclui alguns metadados do cliente e um campo listando segmentos que influenciaram a sugestão de intenções e tipos de slots. Você pode baixar a transcrição de saída na página Revisão no console ou usando a API Amazon Lex V2. Para mais informações, consulte Formato da transcrição de entrada.

```
{
    "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 — Há dois campos adicionados ao `CustomerMetadata` campo, o nome do arquivo de entrada que contém a conversa e o formato de entrada, que é sempre “Lex”.
- InfluencingSegments — identifica os segmentos da conversa que influenciaram a sugestão de uma intenção ou tipo de espaço. O ID da intenção ou do tipo de slot identifica aquele específico influenciado pela conversa.

Adicionando um idioma

Você adiciona um ou mais idiomas e localidades ao seu bot para permitir que ele se comunique com os usuários em seus idiomas. Você define as intenções, os slots e os tipos de slots separadamente para cada idioma, de forma que os enunciados, as solicitações e os valores dos slots sejam específicos do idioma.

Seu bot deve conter pelo menos um idioma.

Para adicionar um idioma ao seu bot

1. Na seção Novo idioma, escolha o idioma desejado. Você pode adicionar uma descrição para ajudar a identificar o idioma em listas.
2. Se o seu bot for compatível com interação por voz, na seção Interação por voz, escolha a voz do Amazon Polly que o Amazon Lex V2 usa para se comunicar com o usuário. Se o seu bot não for compatível voz, escolha Nenhum.
3. Para o limite da pontuação de confiança da classificação de intenção, defina o valor que o Amazon Lex V2 usa para determinar se uma intenção é a intenção correta. Você pode ajustar esse valor depois de testar seu bot.
4. Escolha Adicionar.

Adicionar intenções

As intenções são as metas que seus usuários desejam alcançar, como pedir flores ou reservar um hotel. Seu bot deve ter pelo menos uma intenção.

Por padrão, todos os bots contêm uma única intenção embutida, a intenção de fallback. Essa intenção é usada quando o Amazon Lex V2 não reconhece nenhuma outra intenção. Por exemplo, se um usuário disser “Quero pedir flores” para uma intenção de agendamento de hotel, a intenção de fallback será acionada.

Adicionar uma intenção

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot ao qual você deseja adicionar a intenção e, em Adicionar idiomas, escolha Exibir idiomas.
3. Escolha o idioma ao qual adicionar a intenção e, em seguida, escolha Intenções.
4. Escolha Adicionar intenção, dê um nome à sua intenção e escolha Adicionar.
5. No editor de intenção, adicione os detalhes da sua intenção.

    - Fluxo de conversa: use o diagrama de fluxo de conversa para ver como pode ser um diálogo com seu bot. Você pode escolher diferentes seções da conversa para ir para essa seção do editor de intenções.
    - Detalhes da intenção: dê um nome e uma descrição à intenção para ajudar a identificar o propósito da intenção. Você também pode ver o identificador exclusivo que o Amazon Lex V2 atribuiu à intenção.
    - Contextos: defina os contextos de entrada e saída para a intenção. Um contexto é uma variável de estado associada a uma intenção. Um contexto de saída é definido quando uma intenção é atendida. Uma intenção com um contexto de entrada só pode ser reconhecida se o contexto estiver ativo. Uma intenção sem contextos de entrada sempre pode ser reconhecida.
    - Exemplos de expressões: você deve fornecer 10 ou mais frases que espera que seus usuários usem para iniciar uma intenção. O Amazon Lex V2 generaliza essas frases para reconhecer que o usuário deseja iniciar a intenção.
    - Resposta inicial: a mensagem inicial enviada ao usuário após a invocação da intenção. É possível fornecer respostas, inicializar valores e definir a próxima etapa que o Amazon Lex V2 executará para responder ao usuário no início da intenção.

- Slots: defina os slots, ou parâmetros, necessários para cumprir a intenção. Cada slot tem um tipo que define os valores que podem ser inseridos no slot. Você pode escolher entre os tipos de slot personalizados ou escolher um tipo de slot embutido.
- Confirmação: essas solicitações e respostas são usadas para confirmar ou recusar o cumprimento da intenção. O prompt de confirmação solicita que o usuário revise os valores dos slots. Por exemplo, “Reservei um quarto de hotel para sexta-feira. Está certo?” A resposta de recusa é enviada ao usuário quando ele recusa a confirmação. Você pode fornecer respostas, definir valores e definir a próxima etapa que o Amazon Lex V2 executará, correspondendo a uma resposta de confirmação ou recusa do usuário.
- Atendimento: resposta enviada ao usuário durante o processo de atendimento. Você pode definir atualizações do andamento do atendimento no seu início e periodicamente enquanto estiver em andamento. Por exemplo, “Estou alterando sua senha, isso pode levar alguns minutos” e “Ainda estou trabalhando em sua solicitação”. As atualizações de atendimento só podem ser usadas com conversas em transmissão. Você também pode definir uma mensagem de sucesso pós-atendimento, uma mensagem de falha e uma mensagem de tempo limite. Você pode enviar mensagens de pós-atendimentos tanto para conversas em transmissão como regulares. Por exemplo, se o atendimento for bem-sucedido, você poderá enviar “Eu alterei sua senha”. Se o atendimento não for bem-sucedido, você poderá enviar uma resposta com mais informações, como “Não consegui alterar sua senha, entre em contato com o suporte técnico para obter ajuda”. Se o atendimento demorar mais do que o período de tempo limite configurado, você poderá enviar uma mensagem informando ao usuário, como “Nossos servidores estão muito ocupados no momento. Tente sua solicitação novamente.” É possível fornecer respostas, configurar valores e definir a próxima etapa que o Amazon Lex V2 executará para responder ao usuário.
- Respostas de encerramento: resposta enviada ao usuário após a intenção ser cumprida e todas as outras mensagens serem reproduzidas. Por exemplo, um agradecimento por reservar um quarto de hotel. Ou pode fazer com que o usuário inicie uma intenção diferente, como: “Agradecemos por reservar um quarto, você gostaria de alugar um carro?” É possível fornecer respostas e configurar as próximas ações de atendimento após cumprir a intenção e responder com a resposta final.
- Hooks de código: indique se você está usando uma função AWS Lambda para inicializar a intenção e validar a entrada do usuário. Você especifica a função do Lambda no alias usada para executar o bot.

6. Escolha Salvar intenção para salvar a configuração da intenção.

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Configurar solicitações em uma ordem específica

Você pode configurar o bot para reproduzir mensagens em uma ordem predefinida marcando a caixa Reproduzir mensagens em ordem. Caso contrário, o bot reproduz a mensagem e as variações em ordem aleatória.

Os prompts ordenados permitem que a mensagem e as variações de um grupo de mensagens sejam reproduzidas em ordem entre as novas tentativas. Você pode usar a reformulação alternativa de uma mensagem quando uma resposta inválida para a solicitação é fornecida pelo usuário ou para confirmação da intenção. Até duas variações da mensagem original podem ser definidas em cada slot. É possível escolher se deseja reproduzir as mensagens em ordem ou aleatoriamente.

O prompt solicitado oferece suporte a todos os quatro tipos de mensagens: texto, resposta personalizada de carga útil, SSML e grupo de cartões. As respostas são ordenadas dentro do mesmo grupo de mensagens. Grupos de mensagens diferentes são independentes.

Tópicos

Enunciados de amostra

Você cria exemplos de enunciados que são variações de frases que se espera que os usuários usem para iniciar uma intenção. Por exemplo, para uma intenção **`BookFlight`**, você pode incluir frases como as seguintes:

1. Quero reservar um voo
2. me ajude a pegar um voo.
3. passagens aéreas, por favor!
4. voo de {*`DepartureCity`*} para {*`DestinationCity`*}

Você deve fornecer 10 ou mais enunciados de amostra. Forneça amostras que representem uma ampla variedade de estruturas de frases e palavras que os usuários possam pronunciar. Considere também frases incompletas, como nos exemplos 3 e 4 acima. *`Você também pode usar espaços definidos para a intenção em um enunciado de amostra colocando colchetes ao redor do nome do slot, como em {DepartureCity} no exemplo 4.`* Se incluir nomes de slots em um exemplo de enunciado, o Amazon Lex V2 preenche os espaços da intenção com os valores que o usuário fornece no enunciado.

Uma variedade de exemplos de declarações ajuda o Amazon Lex V2 a generalizar para reconhecer efetivamente que o usuário deseja iniciar a intenção.

Você pode adicionar exemplos de expressões no editor de intenções, no Visual Conversation Builder ou com as operações da API CreateIntent ou UpdateIntent. Também é possível gerar exemplos de enunciados automaticamente aproveitando os recursos de IA generativa do Amazon Bedrock. Para obter mais informações, consulte Geração de enunciados.

Usar o editor de intenções ou o Visual Conversation Builder

1. No editor de intenções, navegue até a seção Enunciados de amostra. No Visual Conversation Builder, encontre a seção Enunciados de amostra no bloco Iniciar.
2. Na caixa com o texto transparente **`I want to book a flight`**, digite um enunciado de amostra. Selecione Adicionar enunciado para adicionar o enunciado.
3. Visualize os exemplos de enunciados adicionados no modo Visualização ou Texto sem formatação. Em Texto sem formatação, cada linha é um enunciado separado. No Modo de visualização, passe o mouse sobre um enunciado para revelar as seguintes opções:

- Selecione a caixa de texto para editar o enunciado.
- Selecione o botão x à direita da caixa de texto para excluir o enunciado.
- Arraste o botão à esquerda da caixa de texto para alterar a ordem dos exemplos de enunciados.

4. Use a barra de pesquisa na parte superior para pesquisar seus exemplos de enunciados e o menu suspenso ao lado dela para classificar pela ordem em que você adicionou os enunciados ou em ordem alfabética.

Usar uma operação de API

1. Crie uma nova intenção com a operação CreateIntent ou atualize uma existente com a operação UpdateIntent.
2. A solicitação da API inclui um campo `sampleUtterances`, que mapeia para uma matriz de objetos SampleUtterance.
3. Para cada enunciado de amostra que você deseja adicionar, anexe um objeto `SampleUtterance` à matriz. Adicione o exemplo de expressão como o valor do campo `utterance`.
4. Para editar e excluir exemplos de enunciados, envie uma solicitação `UpdateIntent`. A lista de enunciados fornecida no campo `sampleUtterances` substitui os enunciados existentes.

> **Important**
>
> Qualquer campo que você deixar em branco na solicitação `UpdateIntent` fará com que as configurações existentes na intenção sejam excluídas. Use a operação DescribeIntent para retornar a configuração do bot e copiar todas as configurações que não deseja que sejam excluídas na solicitação `UpdateIntent`.

Estrutura da intenção

Os tópicos a seguir descrevem as diferentes etapas que um bot executa para cumprir uma intenção e como configurar cada uma dessas etapas:

Tópicos

- Resposta inicial

- Slots
- Confirmação
- Atendimento
- Resposta de encerramento

Resposta inicial

A resposta inicial é enviada ao usuário depois que o Amazon Lex V2 determina a intenção e antes de começar a extrair valores de slot. Você pode usar essa resposta para informar o usuário sobre a intenção que foi reconhecida e preparar para as informações coletadas para cumprir a intenção.

Por exemplo, se a intenção é agendar uma consulta de serviço para um carro, a resposta inicial pode ser:

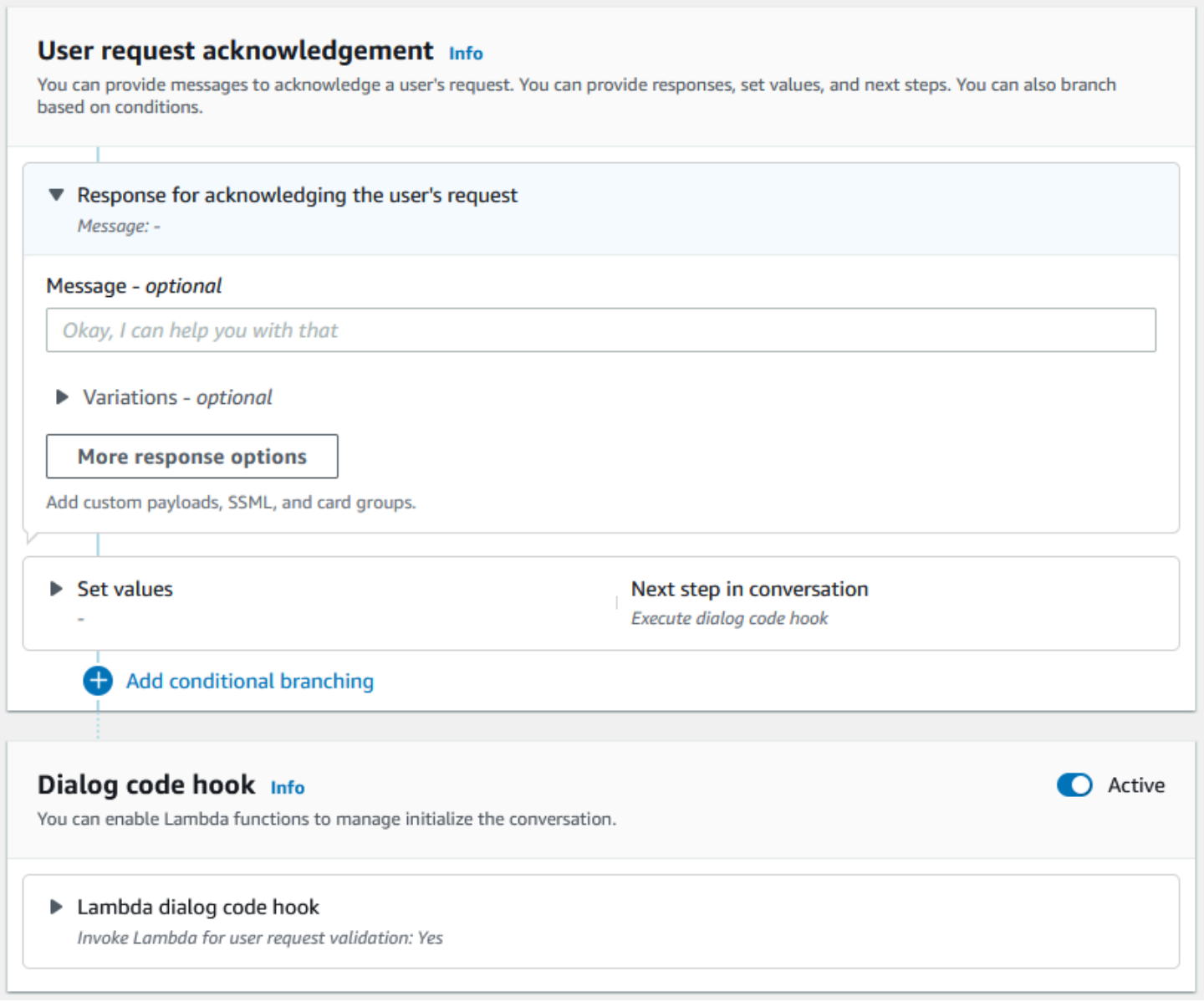
> Eu posso te ajudar a agendar uma consulta. Você precisará fornecer a marca, o modelo e o ano do seu carro.

Uma mensagem de resposta inicial não é obrigatória. Se não fornecida, o Amazon Lex V2 continuará seguindo a próxima etapa da resposta inicial.

Você pode configurar as seguintes opções na resposta inicial:

- Configurar a próxima etapa: você pode fornecer a próxima etapa da conversa, como pular para uma ação de diálogo específica, obter um espaço específico ou pular para uma intenção diferente. Para mais informações, consulte Configurar as próximas etapas na conversa.
- Definir valores: você pode definir valores para slots e atributos de sessão. Para obter mais informações, consulte Definir valores durante a conversa.
- Adicionar ramificação condicional: você pode aplicar condições depois de reproduzir a resposta inicial. Quando uma condição é avaliada como verdadeira, as ações que você define são tomadas. Para mais informações, consulte Adicionar condições às conversas ramificadas.
- Executar hook de código de diálogo: você pode definir um hook de código Lambda para inicializar dados e executar a lógica de negócios. Para mais informações, consulte Invocar hook de código de diálogo. Se a opção de executar a função do Lambda estiver habilitada para a intenção, o hook do código de diálogo será executado por padrão. Você pode desativar o hook do código de diálogo ativando o botão Ativo.

Na ausência de uma condição ou de uma próxima etapa explícita, o Amazon Lex V2 passa para o próximo slot em ordem de prioridade.



> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Slots

Os slots são valores fornecidos pelo usuário para cumprir a intenção. Existem dois tipos de slots:

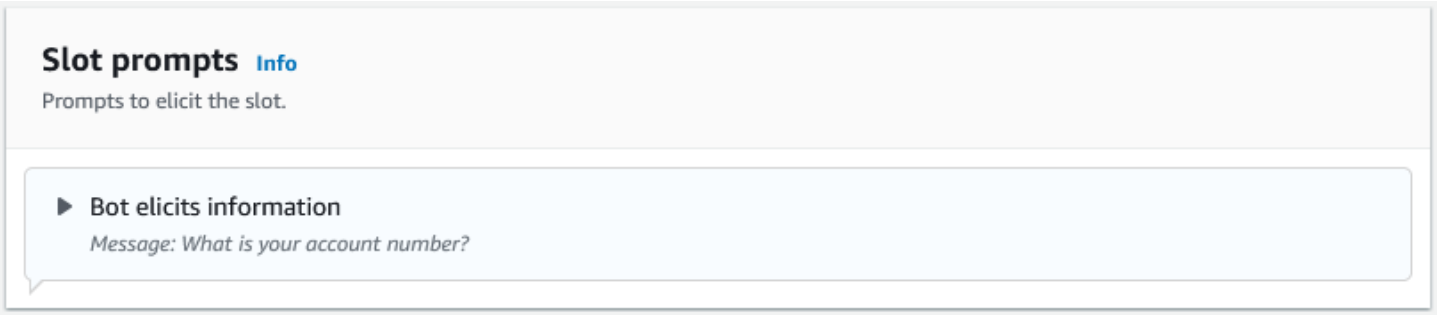
- Tipo de slot embutido: você pode usar tipos de slots integrados para capturar valores padrão, como número, nome e cidade. Para obter os tipos de slot embutidos com suporte, consulte Tipos de slot integrados.
- Tipo de slot personalizado: você pode usar tipos de slots personalizados para capturar valores personalizados específicos da intenção. Por exemplo, você pode usar um tipo de slot personalizado para capturar o tipo de conta como “Cheque” ou “Poupança”. Para mais informações, consulte Tipo de slot personalizado.

Para definir um slot em uma intenção, você precisa configurar o seguinte:

- Informações do slot: esse campo contém um nome e uma descrição opcional para o slot. Por exemplo, você pode fornecer o nome do slot como “AccountNumber” para capturar os números da conta. Se o espaço for necessário como parte do fluxo de conversação para cumprir a intenção, ele deverá ser marcado como necessário.
- Tipo de slot: um tipo de slot define a lista de valores que um slot pode aceitar. Você pode criar um tipo de slot personalizado ou usar um tipo de slot predefinido.
- Prompt de slot: um prompt de slot é uma pergunta feita ao usuário para coletar informações. Você pode configurar o número de novas tentativas usadas para coletar informações e a variação do prompt usado para cada nova tentativa. Também é possível ativar a invocação de uma função do Lambda após cada nova tentativa para processar a entrada capturada e tentar resolver para uma entrada válida.
- Esperar e continuar (opcional): ao ativar esse comportamento, os usuários podem dizer frases como “espere um segundo” para fazer com que o bot espere que eles encontrem as informações e as forneçam. Isso é ativado somente para streaming de conversas. Para mais informações, consulte Permitir que o bot espere que o usuário forneça mais informações.
- Respostas de captura de slots: você pode configurar uma resposta de sucesso e uma resposta de falha com base no resultado da captura do valor do slot a partir da entrada do usuário.
- Ramificação condicional: você pode aplicar condições depois de reproduzir a resposta inicial. Quando uma condição é avaliada como verdadeira, as ações que você define são tomadas. Para mais informações, consulte Adicionar condições às conversas ramificadas.

- Hook de código de diálogo: você também pode usar um hook de código Lambda para validar os valores do slot e executar a lógica de negócios. Para mais informações, consulte Invocar hook de código de diálogo.
- Tipo de entrada do usuário: você pode configurar o tipo de entrada para que o bot possa aceitar uma modalidade específica. Por padrão, as modalidades de áudio e DTMF são aceitas. É possível configurá-lo seletivamente para somente áudio ou somente DTMF.
- Tempos limite e durações de entrada de áudio: você pode configurar tempos limite de áudio, incluindo tempo limite de voz e tempo limite de silêncio. Além disso, a duração máxima do áudio também pode ser definida.
- Tempo limite de entrada, caracteres e comprimentos do DTMF: você pode definir o tempo limite do DTMF junto com o caractere de exclusão e o caractere final. Além disso, a duração máxima do DTMF também pode ser definida.
- Tamanho do texto: você pode definir o tamanho máximo para a modalidade de texto.

Depois que o prompt do slot é reproduzido, o usuário fornece o valor do slot como entrada. Se o Amazon Lex V2 não entender o valor de um slot fornecido pelo usuário, ele tentará extrair o slot novamente até entender um valor ou até exceder o número máximo de novas tentativas que você configurou para o slot. Usando as configurações avançadas de repetição, você pode definir os tempos limite, restringir o tipo de entrada e ativar ou desativar a interrupção da solicitação inicial e das novas tentativas. Depois de cada tentativa de capturar a entrada, o Amazon Lex V2 pode chamar a função do Lambda configurada para o bot com um rótulo de invocação fornecido para novas tentativas. Você pode usar a função do Lambda, por exemplo, para aplicar sua lógica de negócios para tentar resolvê-la com um valor válido. Essa função do Lambda pode ser ativada nas Opções avançadas para solicitações de slots.



Você pode definir as respostas que o bot deve enviar ao usuário quando o valor do slot for inserido ou se o número máximo de novas tentativas for excedido. Por exemplo, para um bot para agendar o serviço de um carro, você pode enviar uma mensagem ao usuário quando o número de identificação do veículo (VIN) for inserido:

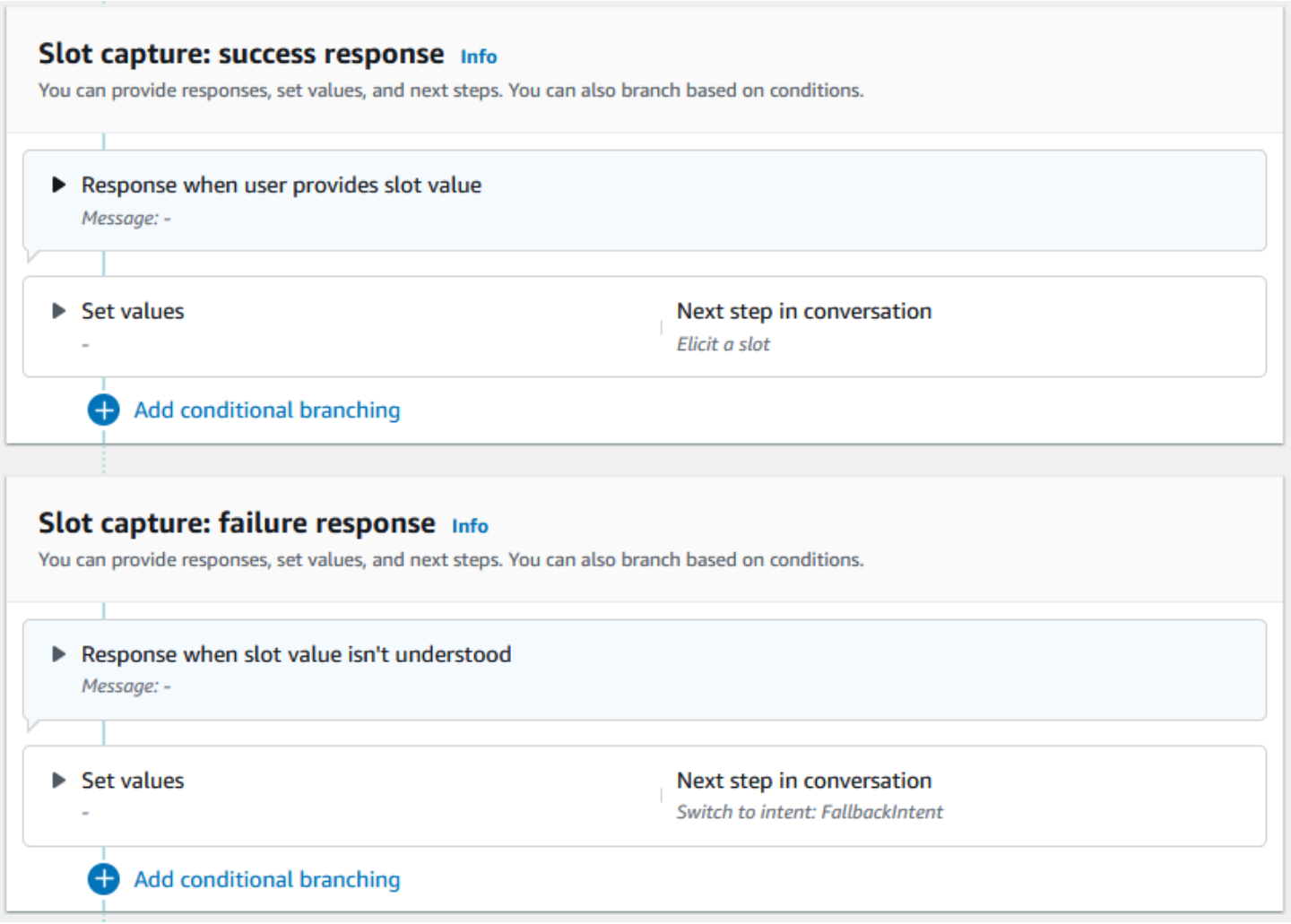
Agradecemos por fornecer o número VIN do seu carro. Agora vou agendar uma consulta.

Você pode criar duas respostas:

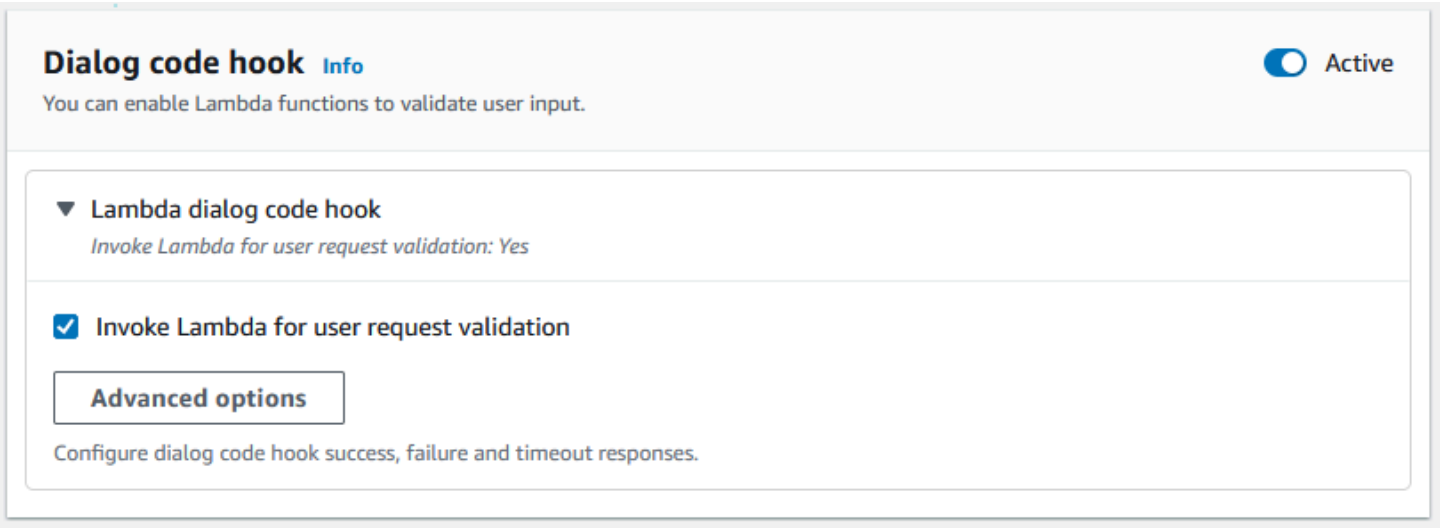
- Resposta de sucesso: enviada quando o Amazon Lex V2 compreende o valor de um slot.
- Resposta de falha: enviada quando o Amazon Lex V2 não consegue entender o valor de um slot do usuário após o número máximo de novas tentativas.

É possível definir valores, configurar as próximas etapas e aplicar condições que correspondem a cada resposta para criar o fluxo de conversação.

Na ausência de uma condição ou de uma próxima etapa explícita, o Amazon Lex V2 passa para o próximo slot em ordem de prioridade.



Você pode usar uma função do Lambda para validar um valor de slot que um usuário inseriu e determinar qual deve ser a próxima ação. Por exemplo, a função de validação pode ser usada para garantir que o valor inserido esteja no intervalo correto ou que esteja formatado corretamente. Para ativar a função do Lambda, escolha a caixa de seleção Invocar função do Lambda e o botão Ativo na seção Hook do código de diálogo. É possível especificar um rótulo de invocação para o hook do código de diálogo. Esse rótulo de invocação pode ser usado na função do Lambda para escrever a lógica de negócios correspondente à elicitação do slot.



Os slots que não são necessários para a intenção não fazem parte do fluxo principal da conversa. No entanto, se um enunciado do usuário contiver um valor que seu bot identifique como correspondente a um slot opcional, ele poderá preencher o slot com esse valor. Por exemplo, se você configurar um bot de inteligência de negócios para ter um slot `City` opcional e a expressão do usuário **`What is the sales for April in San Diego?`**, o bot preencherá o slot opcional com **`San Diego`**. A lógica de negócios pode ser configurada para usar o valor opcional do slot, se presente.

Os slots não necessários para a intenção não podem ser obtidos usando as próximas etapas. Essas etapas podem ser preenchidas somente durante a elicitação da intenção (como no exemplo anterior) ou podem ser obtidas definindo o estado da caixa de diálogo na função do Lambda. Se o slot for obtido usando a função do Lambda, você deverá usar a função do Lambda para decidir a próxima etapa na conversa após a conclusão da elicitação do slot. Para ativar o suporte para a próxima etapa da construção do bot, você deve marcar o slot conforme necessário para a intenção.

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o

caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

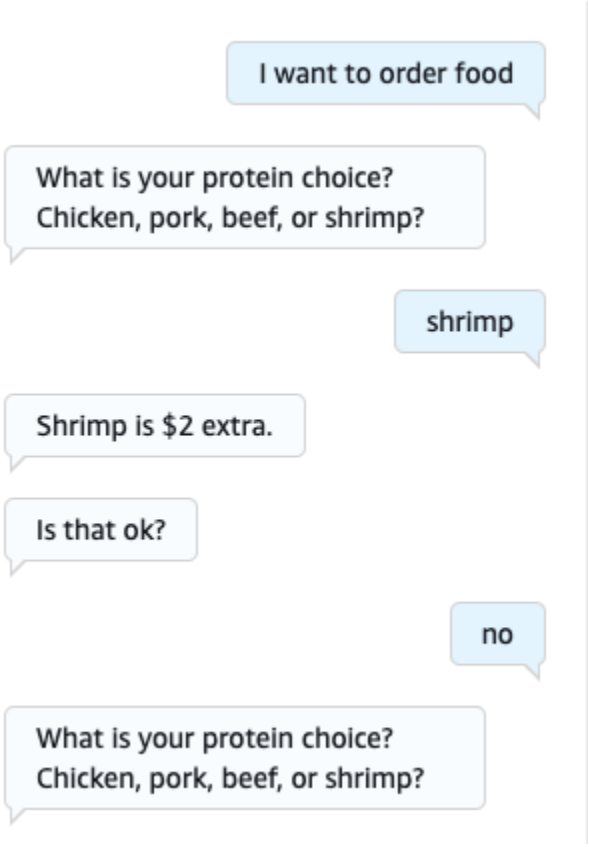
Os tópicos a seguir descrevem como configurar um bot para reativar um valor de slot que já foi preenchido e como criar um slot que consiste em vários valores:

Tópicos

- Escolher slots novamente
- Uso de vários valores em um slot

Escolher slots novamente

Você pode configurar o bot para escolher novamente um slot que já foi preenchido definindo o valor desse slot como **`null`** e definindo a próxima etapa na conversa para retornar à obtenção desse slot. Por exemplo, talvez você queira escolher novamente um slot depois que seu cliente recusar a confirmação da obtenção do slot com base em informações adicionais, como na conversa a seguir:



Você pode configurar um loop a partir da resposta de confirmação para escolher novamente o slot com o editor de intenção ou com o Usando o Visual Conversation Builder.

> **Note**
>
> Você pode voltar para escolher um slot em qualquer ponto da conversa, desde que defina o valor desse slot como **`null`** com antecedência.

Reprodução do exemplo acima com o editor de intenção

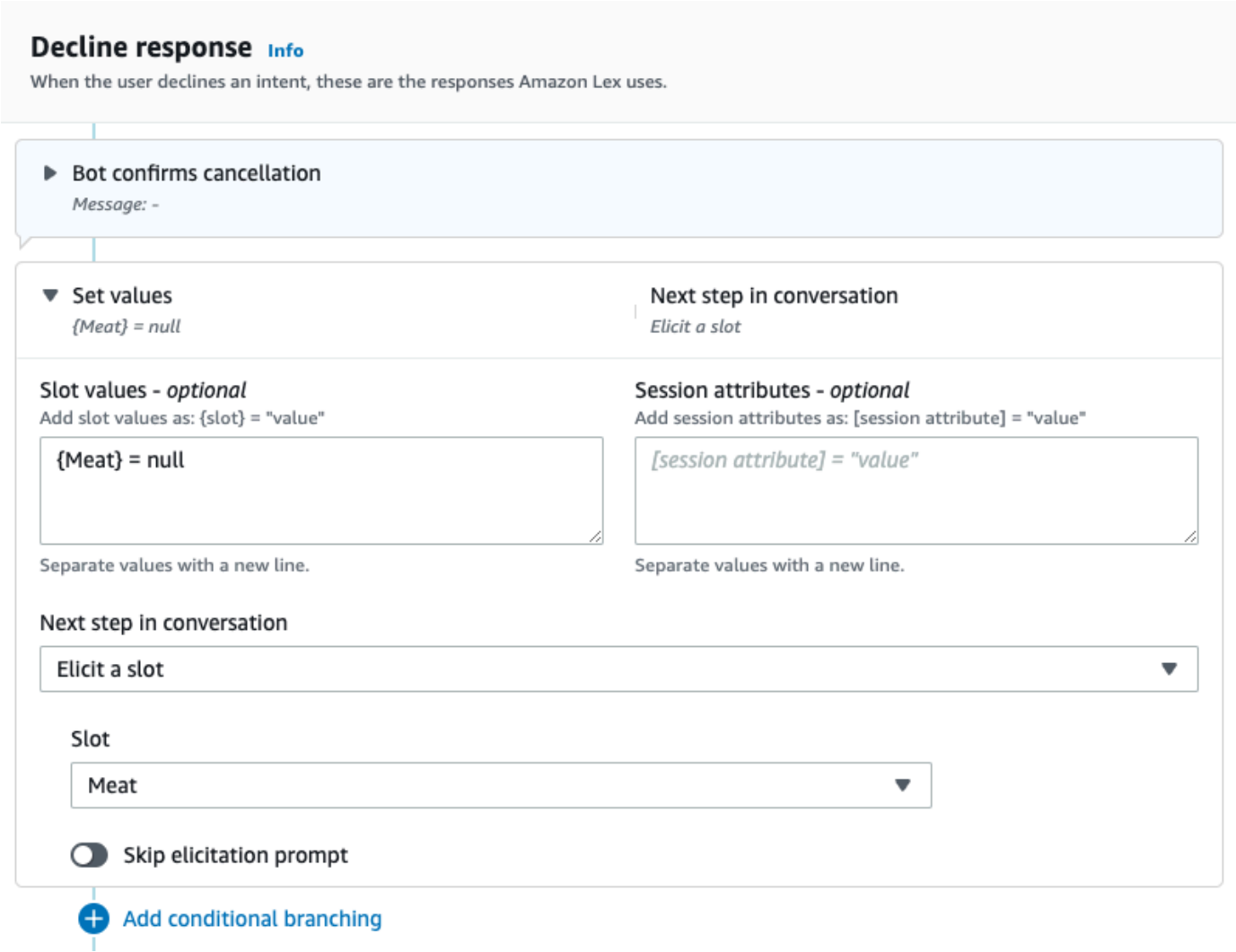
1. Na seção Confirmação do editor de intenção, selecione a seta para a direita ao lado de Solicitações para confirmar a intenção para expandir a seção.
2. Selecione Opções avançadas na parte inferior.
3. Na seção Recusar resposta, selecione a seta para a direita ao lado de Definir valores para expandir a seção. Preencha esta seção com as seguintes etapas, como na imagem abaixo:

a. Defina o valor do slot que deseja escolher novamente como **`null`**. Neste exemplo, queremos escolher novamente o slot `Meat`, então inserimos **`{Meat} = null`** na seção Valores do slot.

b. No menu suspenso, em Próxima etapa da conversa, selecione Escolher um slot.

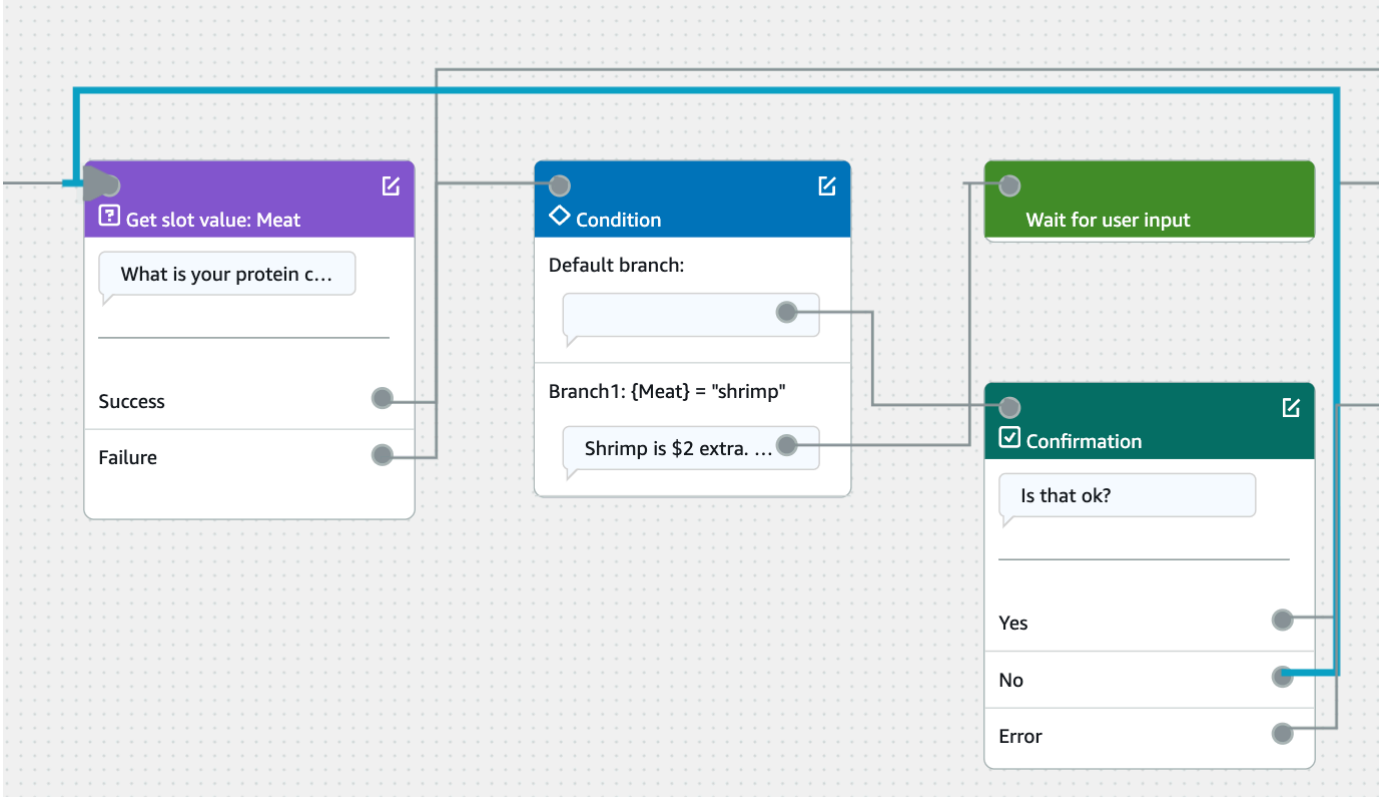
c. Uma seção Slot aparecerá. No menu suspenso abaixo, selecione o slot que você deseja escolher novamente.

d. Selecione Opções de atualização para confirmar suas alterações.

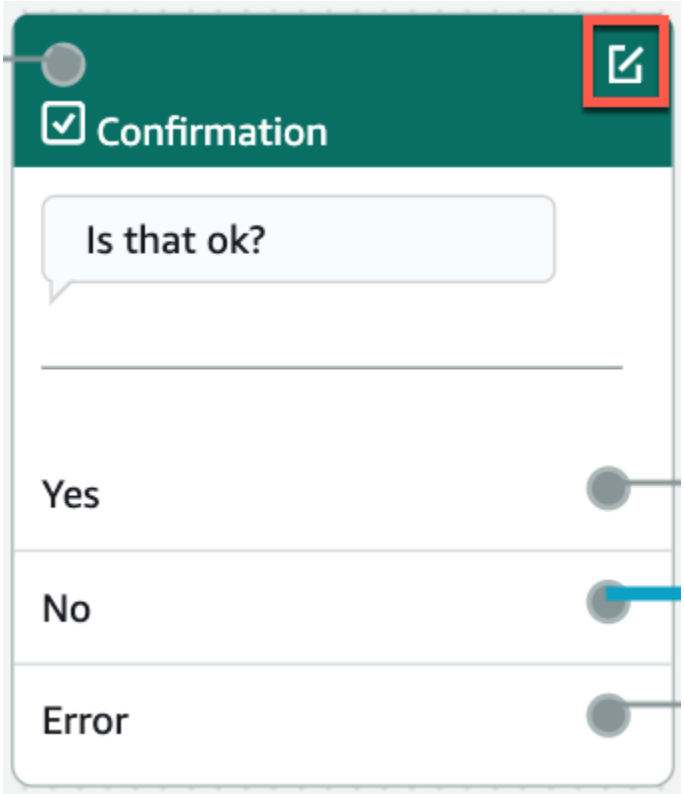


Reprodução do exemplo acima com o Visual Conversation Builder

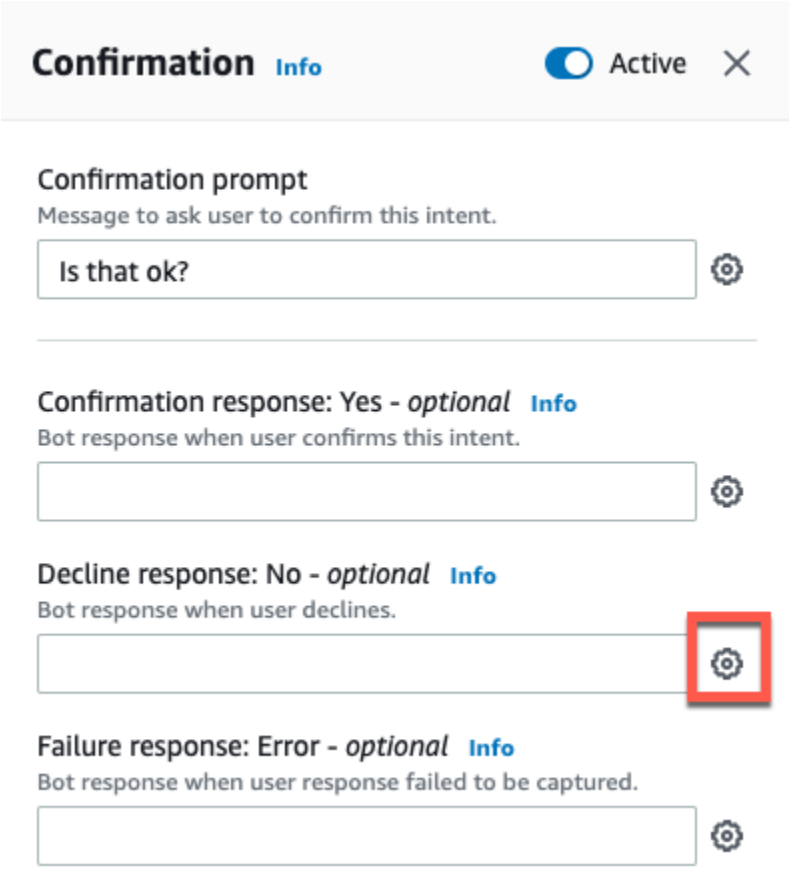
1. Crie uma conexão da porta Não do bloco Confirmação com a porta de entrada do bloco Obter valor do slot: Carne.



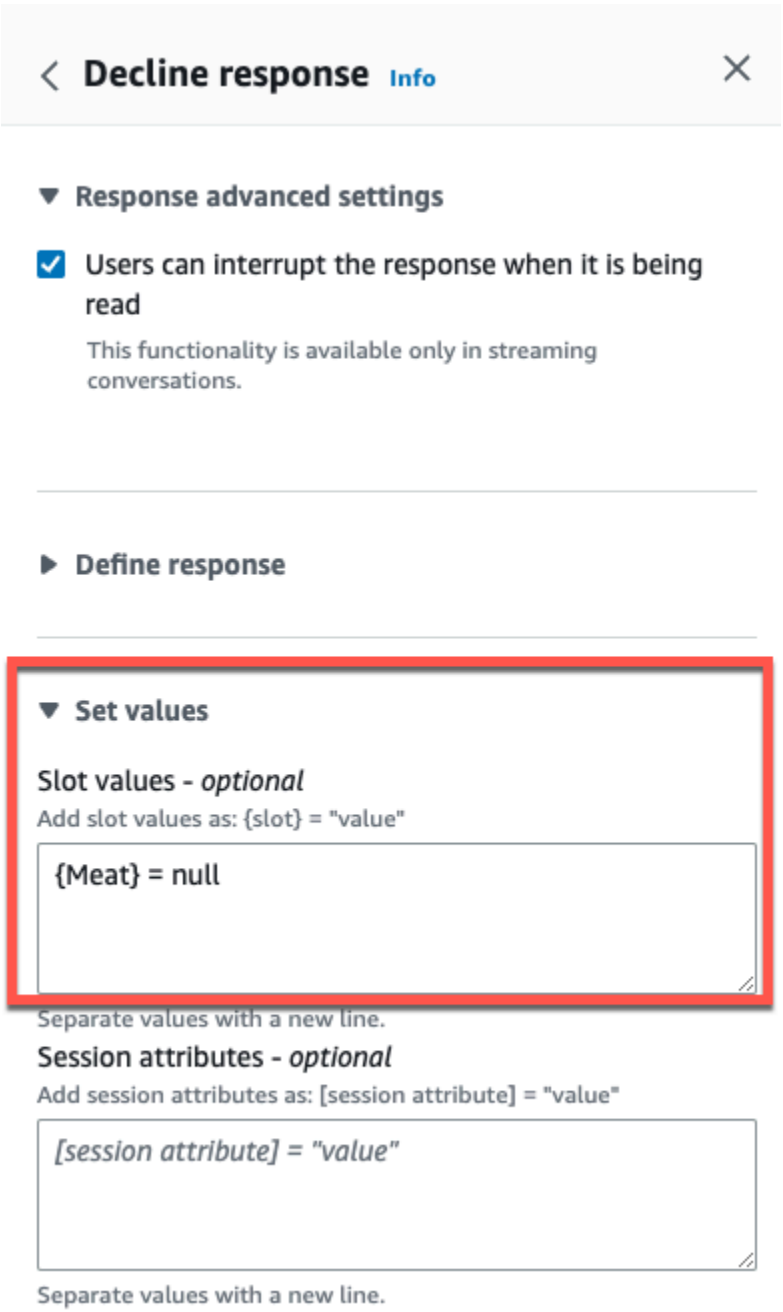
2. Selecione o ícone Editar no canto superior direito do bloco de Confirmação.



3. Selecione o ícone de engrenagem ao lado da resposta do bot na seção Recusar resposta.



4. Na seção Definir valores, adicione “{Meat} = null” na caixa Valores do slot.



5. Selecione Salvar intenção.

Uso de vários valores em um slot

> **Note**
>
> Vários slots de valor são compatíveis somente no idioma inglês (EUA).

Para algumas intenções, talvez você queira capturar vários valores para um único slot. Por exemplo, um bot de pedido de pizza pode ter uma intenção com o seguinte enunciado:

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

A intenção espera que o slot `{toppings}` contenha uma lista dos recheios que o cliente deseja em sua pizza, por exemplo, “calabresa e abacaxi”.

Para configurar um slot para capturar vários valores, defina o campo `allowMultipleValues` no slot como verdadeiro. Você pode definir o campo usando o console ou com a operação CreateSlot ou UpdateSlot.

Só é possível marcar slots com tipos de slots personalizados como slots de vários valores.

Para um slot de vários valores, o Amazon Lex V2 retorna uma lista de valores de slot na resposta à operação RecognizeText ou RecognizeUtterance Veja a seguir as informações do slot retornadas para o enunciado “Quero uma pizza com calabresa e abacaxi” do bot 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"
                    ]
                }
            }
        ]
    }
}
```

Os slots com vários valores sempre retornam uma lista de valores. Quando o enunciado contém apenas um valor, a lista de valores retornados contém somente uma resposta.

O Amazon Lex V2 reconhece vários valores separados por espaços, vírgulas (,) e a conjunção “e”. Os slots de vários valores funcionam com entrada de texto e voz.

Você pode usar slots de vários valores em prompts. Por exemplo, você pode definir a solicitação de confirmação de uma intenção como:

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

Quando o Amazon Lex V2 envia a solicitação ao usuário, ele envia “Você gostaria que eu pedisse sua pizza de pepperoni e abacaxi?”

Os slots de vários valores oferecem suporte a valores padrão únicos. Se vários valores padrão forem fornecidos, o Amazon Lex V2 preencherá o slot somente com o primeiro valor disponível. Para mais informações, consulte Usar valores de slot padrão.

Você pode usar a ofuscação de slots para mascarar os valores de um slot com vários valores nos logs de conversação. Ao ofuscar valores de slot, o valor de cada um dos valores de slot é substituído pelo nome do slot. Para mais informações, consulte Ocultar valores de slot nos logs de conversa.

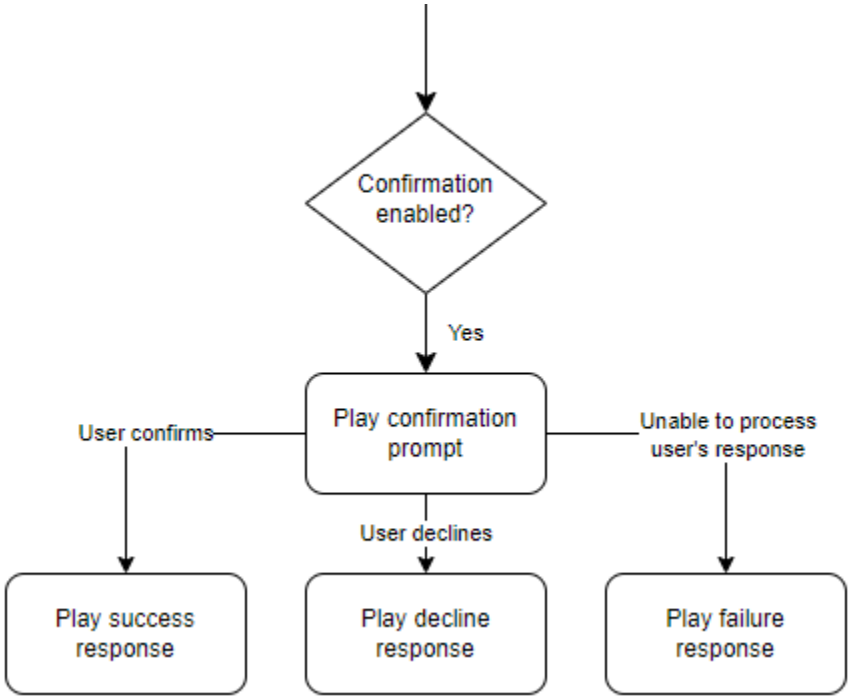
Confirmação

Depois que a conversa com o usuário for concluída e os valores de slot da intenção forem preenchidos, você poderá configurar uma solicitação de confirmação para perguntar ao usuário se os valores de slot estão corretos. Por exemplo, um bot que agenda compromissos de serviço para carros pode perguntar ao usuário o seguinte:

Tenho serviço para o seu Honda Civic 2017 agendado para 25 de março às 15h. Está certo?

Você pode definir 3 tipos de respostas para a solicitação de confirmação:

- Resposta de confirmação: essa resposta é enviada ao usuário quando ele confirma a intenção. Por exemplo, depois que o usuário responde “sim” ao prompt “você deseja fazer o pedido?”
- Resposta de recusa: essa resposta é enviada ao usuário quando ele nega a intenção. Por exemplo, depois que o usuário responde “não” ao prompt “você deseja fazer o pedido?”
- Resposta de falha: essa resposta é enviada ao usuário quando a solicitação de confirmação não pode ser processada. Por exemplo, se a resposta do usuário não for entendida ou não puder ser resolvida em sim ou não.



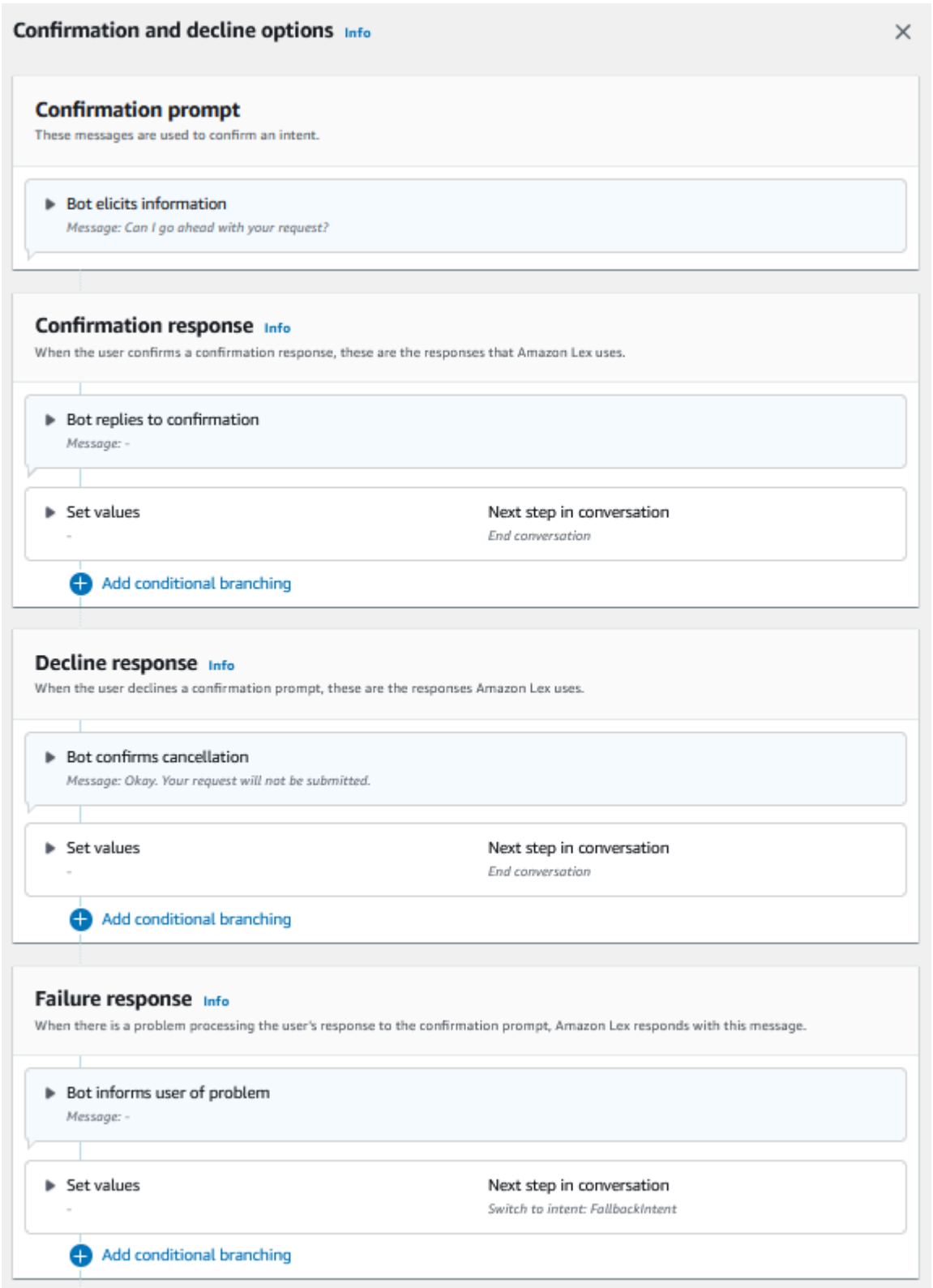
Se uma solicitação de confirmação não for especificada, o Amazon Lex V2 passa para a etapa de atendimento ou para a resposta de encerramento.

É possível definir valores, configurar as próximas etapas e aplicar condições correspondentes a cada resposta para criar o fluxo de conversação. Na ausência de uma condição ou de uma próxima etapa explícita, o Amazon Lex V2 passa para o passo de atendimento.

Você também pode ativar o hook de código de diálogo para validar as informações capturadas na intenção antes de enviá-las para atendimento. Para usar um hook de código, habilite o hook de código de diálogo nas opções avançadas do prompt de confirmação. Além disso, configure a

próxima etapa do estado anterior para executar o hook do código de diálogo. Para mais informações, consulte Invocar hook de código de diálogo.

> **Note**
>
> Se você usar um hook de código para acionar a etapa de confirmação em runtime, deverá marcar a etapa de confirmação como Ativa no momento da criação.


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**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Usar uma função do Lambda para validar uma intenção.

Você pode definir um hook de código Lambda para validar a intenção antes de enviá-la para atendimento. Para usar um hook de código, habilite o hook de código de diálogo nas opções avançadas do prompt de confirmação.

Ao usar um hook de código, as ações que o Amazon Lex V2 executa podem ser definidas após a execução do hook de código. É possível criar três tipos de regras:

- Resposta de sucesso: enviada ao usuário quando o hook de código é concluído com êxito.
- Resposta de falha: enviada ao usuário quando o hook de código não é executado com êxito ou quando ele retorna `Failure` na resposta.
- Resposta de tempo limite: enviada ao usuário quando o hook de código não é concluído no período de tempo limite configurado.

Atendimento

Depois que todos os valores de slot são fornecidos pelo usuário para a intenção, o Amazon Lex V2 atende à solicitação do usuário. Você pode configurar as seguintes opções para o atendimento.

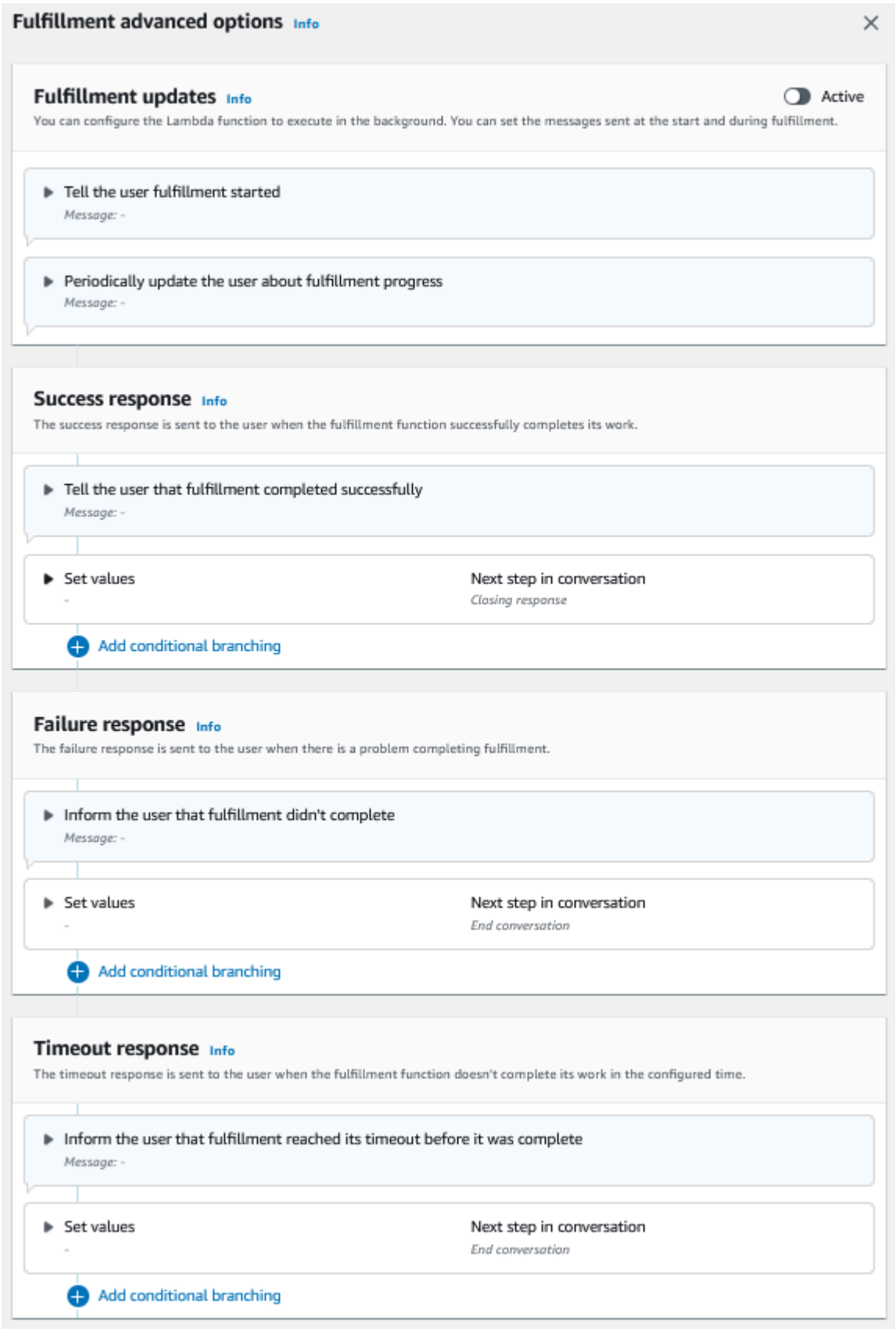
- Hook de código de atendimento: você pode usar essa opção para controlar a invocação do Lambda de atendimento. Se a opção for desativada, o atendimento será bem-sucedido sem invocar a função do Lambda.
- Atualizações de atendimento: você pode habilitar atualizações de atendimento para funções do Lambda que levam mais do que alguns segundos para serem concluídas, para que o usuário saiba que o processo está em andamento. Para mais informações, consulte Configurar atualizações do progresso do atendimento. Essa funcionalidade só está disponível para streaming de conversas.

- Respostas de atendimento: você pode configurar uma resposta de sucesso, uma resposta de falha e uma resposta de tempo limite. A resposta apropriada é retornada ao usuário com base no status da invocação do Lambda de atendimento.

Há três respostas possíveis de atendimento:

- Resposta de sucesso: uma mensagem enviada quando o Lambda de atendimento é concluído com êxito.
- Resposta de falha: uma mensagem enviada se o atendimento falhar ou se o Lambda não puder ser concluído por algum motivo.
- Resposta de tempo limite: uma mensagem enviada se a função do Lambda de atendimento não terminar dentro do tempo limite configurado.

É possível definir valores, configurar as próximas etapas e aplicar condições correspondentes a cada resposta para criar o fluxo de conversação. Na ausência de uma condição ou de uma próxima etapa explícita, o Amazon Lex V2 passa para a resposta de encerramento.


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**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

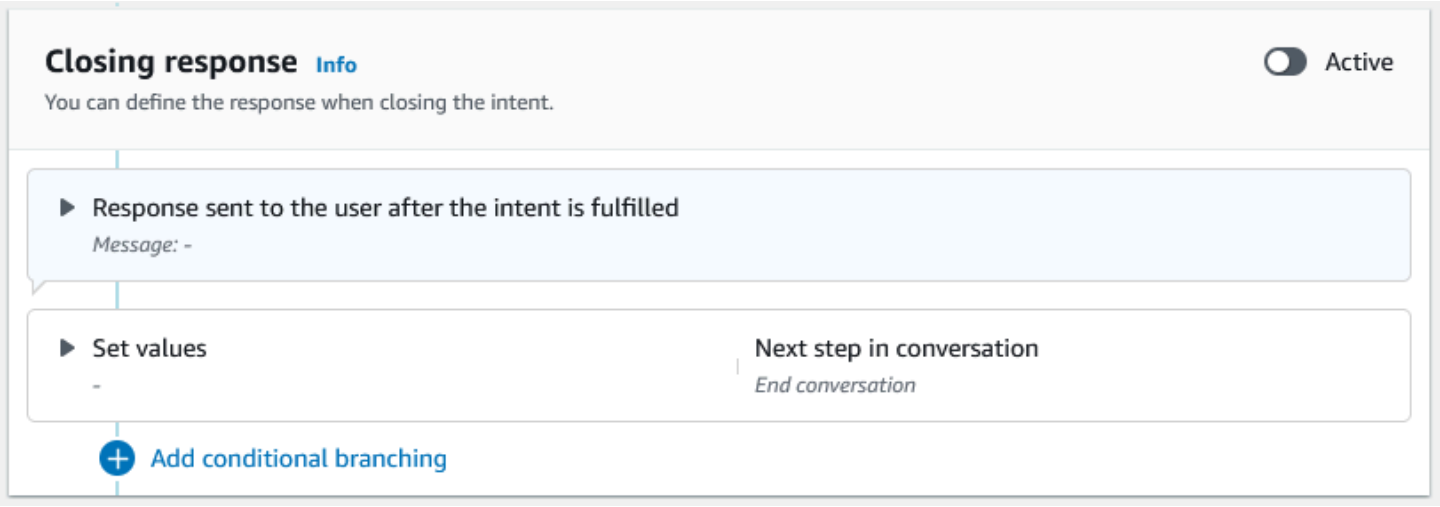
Resposta de encerramento

A resposta de encerramento é enviada ao usuário após a intenção ser atendida. A resposta de encerramento pode ser usada para encerrar a conversa ou para informar ao usuário que ele pode continuar com outra intenção. Por exemplo, em um bot de reserva de viagens, é possível definir a resposta final para reservar um quarto de hotel com a seguinte intenção:

```
Tudo bem, reservei seu quarto de hotel. Posso ajudar você com mais alguma coisa?
```

É possível definir valores, configurar as próximas etapas e aplicar condições após a resposta de encerramento para criar o caminho de conversação. Na ausência de uma condição ou de uma próxima etapa explícita, o Amazon Lex V2 encerra a conversação.

Se uma resposta final não for fornecida ou se nenhuma das condições for considerada verdadeira, o Amazon Lex V2 encerrará a conversa com seu bot.

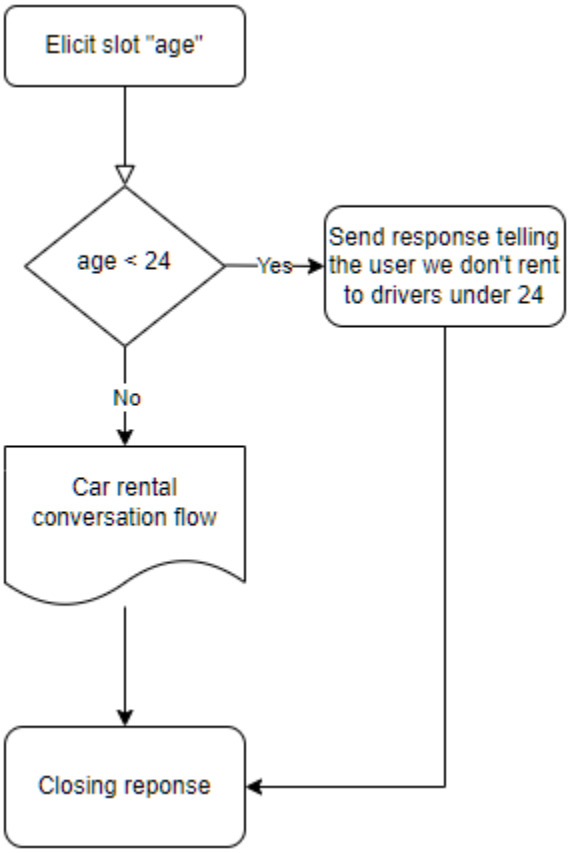


> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Criar caminhos de conversação

Normalmente, o Amazon Lex V2 gerencia o fluxo de conversas com seus usuários. Para bots simples, o fluxo padrão pode ser suficiente para criar uma boa experiência para seus usuários. No entanto, para bots mais complexos, talvez você queira assumir o controle da conversa e direcionar o fluxo para caminhos mais complexos.

Por exemplo, em um bot que reserva aluguel de carros, talvez você não alugue para motoristas mais jovens. Nesse caso, você pode criar uma condição para verificar se um motorista tem menos de uma certa idade e, em caso afirmativo, pular para a resposta de encerramento.



Para criar essas interações, você pode configurar a próxima etapa em cada ponto da conversa, avaliar condições, definir valores e invocar hooks de código.

A ramificação condicional ajuda a criar caminhos para seus usuários por meio de interações complexas. É possível usar uma ramificação condicional a qualquer momento em que passar o controle da conversa para o seu bot. Por exemplo, você pode criar uma condição antes que o bot extraia o primeiro valor do slot, pode criar uma condição entre obter cada valor do slot ou pode criar uma condição antes que o bot encerre a conversa. Para obter uma lista dos locais em que você pode adicionar condições, consulte Adicionar intenções.

Quando você cria um bot, o Amazon Lex V2 cria um caminho padrão para a conversa com base na ordem de prioridade dos slots. Para personalizar o caminho da conversa, você pode modificar a próxima etapa em qualquer ponto da conversa. Para ter mais informações, consulte Configurar as próximas etapas na conversa.

Para criar caminhos alternativos com base nas condições, você pode usar uma ramificação condicional em qualquer ponto da conversa. Por exemplo, é possível criar uma condição antes que o bot obtenha o primeiro valor do slot. Você pode criar uma condição entre obter o valor de cada slot

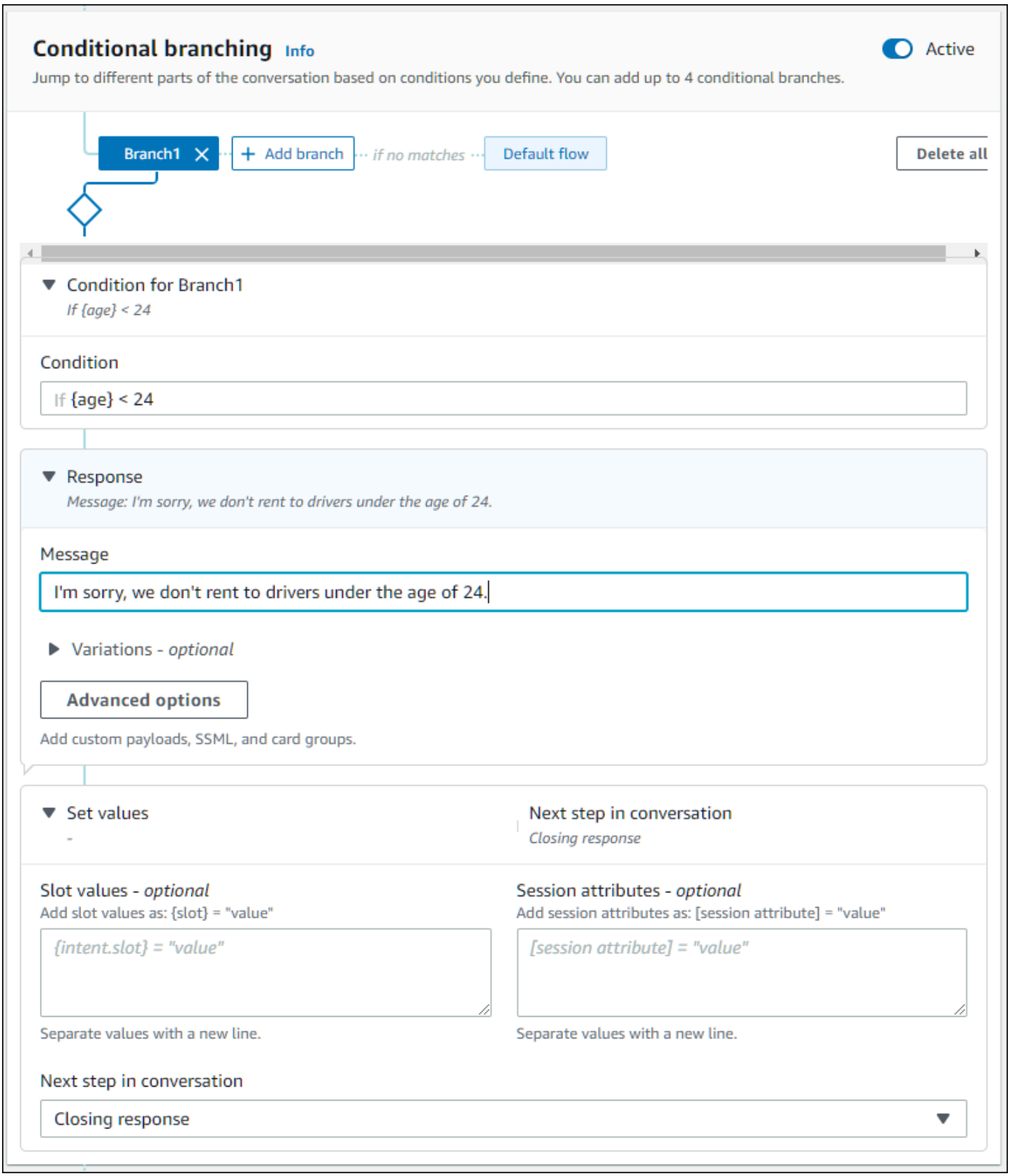
ou criar uma condição antes que o bot encerre a conversa. Para obter uma lista dos locais em que permitem adicionar condições, consulte Adicionar condições às conversas ramificadas.

Você pode definir condições com base nos valores dos slots, nos atributos da sessão, no modo de entrada e na transcrição da entrada ou em uma resposta da Amazon Kendra.

É possível configurar os valores de slots e dos atributos da sessão em qualquer momento da conversa. Para ter mais informações, consulte Definir valores durante a conversa.

Você também pode definir a próxima ação como hook de código de diálogo para executar uma função do Lambda. Para ter mais informações, consulte Invocar hook de código de diálogo.

A imagem a seguir mostra a criação de um caminho para um slot no console. Neste exemplo, o Amazon Lex V2 exibirá o slot “age”. Se o valor do slot for menor que 24, o Amazon Lex V2 salta para a resposta de fechamento, caso contrário, o Amazon Lex seguirá o caminho padrão.


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**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para ter mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Configurar as próximas etapas na conversa

Você pode configurar uma próxima etapa em cada estágio da conversa para criar conversas. Normalmente, o Amazon Lex V2 configura automaticamente as próximas etapas padrão para cada estágio da conversa de acordo com a seguinte ordem.

Resposta inicial → Elicitação de slots → Confirmação (se ativa) → Cumprimento (se ativa) → Resposta de encerramento (se ativa) → Encerrar conversa

Você pode modificar as próximas etapas padrão e criar a conversa com base na experiência esperada do usuário. As próximas etapas a seguir podem ser configuradas em cada estágio da conversa:

Ir para

- Resposta inicial — A conversa é reiniciada a partir do início da intenção. Você pode optar por ignorar a resposta inicial ao configurar a próxima etapa.
- Elicitar um espaço — Você pode extrair qualquer espaço na intenção.
- Avalie as condições — Você pode avaliar as condições e ramificar a conversa em qualquer etapa da conversa.
- Invocar hook de código de diálogo — Você pode invocar a lógica de negócios em qualquer etapa.
- Confirme a intenção — O usuário será solicitado a confirmar a intenção.
- Cumprir a intenção — O cumprimento da intenção começará como a próxima etapa.
- Resposta de encerramento — A resposta de encerramento será devolvida ao usuário.

Mudar para

- Intenção — Você pode fazer a transição para uma intenção diferente e continuar a conversa com essa intenção. Também pode pular a resposta inicial da intenção ao fazer a transição.
- Intent: slot específico — Você pode obter diretamente um slot específico em uma intenção diferente se já tiver capturado alguns valores de slot na intenção atual.

Aguarde a entrada do usuário — O bot espera que o usuário forneça informações para reconhecer qualquer nova intenção. Você pode configurar prompts como “Há mais alguma coisa em que eu possa ajudá-lo?” antes de definir a próxima etapa. O bot estará em estado de diálogo `ElicitIntent`.

Encerrar conversa — A conversa com o bot está encerrada.

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para ter mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Definir valores durante a conversa

O Amazon Lex V2 oferece a capacidade de definir valores de slots e valores de atributos de sessão em cada etapa da conversa. Em seguida, você pode usar esses valores durante a conversa para avaliar as condições ou usá-los durante o cumprimento da intenção.

É possível definir valores de slot para a intenção atual. Se a próxima etapa da conversa for invocar outra intenção, você poderá definir valores de slot da nova intenção.

Se o slot atribuído não for preenchido ou se o caminho JSON não puder ser analisado, o atributo será definido como `null`.

Use a sintaxe a seguir ao usar valores de slot e atributos de sessão:

- Valores do slot — coloque o nome do slot entre chaves (“{ }”). Para valores de slot na intenção atual, é só usar o nome do slot. Por exemplo, `{slot}`. Se estiver definindo um valor na próxima

intenção, você deverá usar o nome da intenção e o nome do slot para identificar o slot. Por exemplo, `{intent.slot}`.

Exemplos:

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

O valor de um slot pode ser qualquer um dos seguintes:

- uma string constante
- um caminho JSON que se refere ao bloco de transcrições na resposta do Amazon Lex (para inglês-EUA e inglês-REINO UNIDO)
- um atributo da sessão

Exemplos:

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

> **Note**
>
> Os valores do slot também podem ser definidos como `null`. Se precisar extrair novamente um valor de slot que tenha sido preenchido, defina o valor como `null` antes de solicitar ao cliente o valor do slot novamente. Se o slot atribuído não for preenchido ou se o caminho JSON não puder ser analisado, o atributo será definido como `null`.

- Atributos da sessão — coloque o nome do atributo entre colchetes (“[ ]”). Por exemplo, `[sessionAttribute]`.

  Exemplos:

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

  O valor do atributo de sessão pode ser qualquer um dos seguintes:

- uma string constante
- um caminho JSON que se refere ao bloco de transcrições na resposta do Amazon Lex (para inglês-EUA e inglês-REINO UNIDO)
- uma referência de valor de slot

> **Note**
>
> Se o slot atribuído não for preenchido ou se o caminho JSON não puder ser analisado, o atributo será definido como `null`.

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para ter mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Adicionar condições às conversas ramificadas

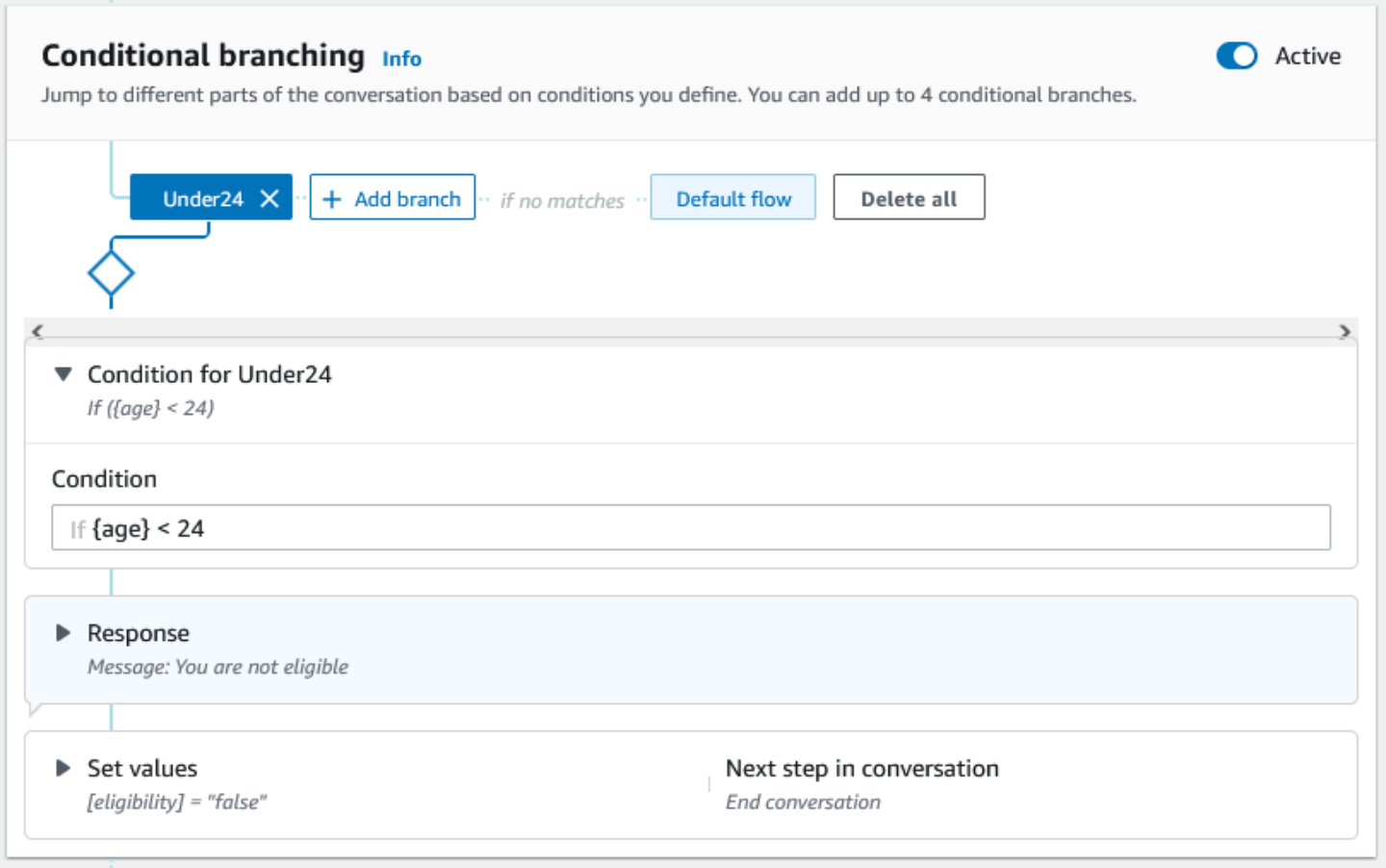
Você pode usar a ramificação condicional para controlar o caminho que seu cliente percorre na conversa com seu bot. As condições podem ser definidas com base nos valores dos slots, nos atributos da sessão, no modo de entrada e na transcrição da entrada ou em uma resposta da Amazon Kendra.

É possível definir até quatro ramificações. Cada ramo tem uma condição que deve ser satisfeita para que o Amazon Lex V2 o siga. Se nenhuma das ramificações tiver sua condição satisfeita, uma padrão será seguida.

Ao definir uma ramificação, você define a ação que o Amazon Lex V2 deve realizar se as condições correspondentes a essa ramificação forem avaliadas como verdadeiras. Você pode definir usando qualquer um destes métodos:

- Uma resposta enviada ao usuário.
- Valores de slots a serem aplicados aos slots.

- Valores de atributo de sessão para a sessão atual.
- A próxima etapa da conversa. Para ter mais informações, consulte Criar caminhos de conversação.



Cada ramo condicional tem uma expressão booleana que deve ser satisfeita para que o Amazon Lex V2 o siga. Existem operadores booleanos, funções e operadores quantificadores e de comparação que podem ser usados para suas condições. Por exemplo, a condição a seguir retornará verdadeira se o slot {age} for menor que 24.

```
{age} < 24
```

A condição a seguir retornará verdadeira se o slot de vários valores {toppings} contiver a palavra “abacaxi”.

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

Você pode combinar vários operadores de comparação com um operador booleano para condições mais complexas. Por exemplo, a condição a seguir retorna verdadeira se o valor do slot {make} for “Honda” e o valor do slot {model} for “Civic”. Use parênteses para definir a ordem de avaliação.

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

Os tópicos a seguir fornecem detalhes sobre os operadores e funções de ramificação condicional.

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para ter mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Tópicos

Operadores de comparação

O Amazon Lex V2 oferece suporte aos seguintes operadores de comparação de condições:

- Igual (=)
- Não é igual (>)
- Menor que (<)
- Menor ou igual a (<=)
- Maior que (>)
- Maior ou igual a (>=)

Ao usar um operador de comparação, ele usa as seguintes regras.

- O lado esquerdo deve ser uma referência. Por exemplo, para referenciar um valor de slot, você usa `{slotName}`. Para referenciar um valor de atributo de sessão, você usa `[attribute]`. Para o modo de entrada e a transcrição de entrada, você usa `$.inputMode` e `$.inputTranscript`.
- O lado direito deve ser constante e do mesmo tipo do lado esquerdo.
- Qualquer expressão que faça referência a um atributo que não tenha sido definido é tratada como inválida e não é avaliada.
- Ao comparar um slot de vários valores, o valor usado é uma lista separada por vírgula de todos os valores interpretados.

As comparações são baseadas no tipo de slot da referência. Eles são resolvidos da seguinte forma:

- Strings: strings são comparados com base em sua representação ASCII. A comparação diferencia maiúsculas de minúsculas.
- Números: slots baseados em números são convertidos da representação da string em um número e depois comparados.
- Data/hora: os slots baseados na hora são comparados com base na série temporal. A data ou hora anterior é considerada menor. Para durações, períodos mais curtos são considerados menores.

Operadores booleanos

O Amazon Lex V2 oferece suporte a operadores booleanos para combinar operadores de comparação. Ele permite a criação declarações semelhantes às seguintes:

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

Você também pode usar as seguintes operações:

- E (&&)
- OU (||)
- NÃO (!)

Operadores quantificadores

Os operadores quantificadores avaliam os elementos de uma sequência e determinam se um ou mais elementos satisfazem a condição.

- CONTÉM: determina se o valor especificado está contido em um slot de vários valores e retorna verdadeiro se estiver. Por exemplo, `{toppings} CONTAINS "pineapple"` retorna verdadeiro se o usuário pediu abacaxi em sua pizza.

Funções

As funções devem ser prefixadas com a string `fn.`. O argumento para a função é uma referência a um slot, atributo de sessão ou atributo de solicitação. O Amazon Lex V2 fornece duas funções para obter informações dos valores de slots, sessionAttribute ou requestAttribute.

- fn.COUNT(): conta o número de valores em um slot de vários valores.

  Por exemplo, se o slot `{toppings}` contiver o valor “calabresa, abacaxi”:

  ```
  fn.COUNT({toppings}) = 2
  ```
- Fn.IS_SET(): o valor é verdadeiro se um slot, atributo de sessão ou atributo de solicitação estiver definido na sessão atual.

  Com base no exemplo anterior:

  ```
  fn.IS_SET({toppings})
  ```
- fn.length () — valor é o tamanho do valor do atributo da sessão, valor do slot ou atributo do slot definido na sessão atual. Essa função não suporta slots de vários valores ou slots compostos.

  Exemplo:

  Se o slot `{credit-card-number}` contiver o valor “123456781234":

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

Expressões condicionais

Aqui estão alguns exemplos de expressões condicionais. OBSERVAÇÃO: `$.` representa o ponto de entrada para da resposta JSON do Amazon Lex. O valor a seguir `$.` será analisado na resposta do Amazon Lex para recuperar o valor. Expressões condicionais usando o bloco de referência de

caminho JSON para transcrições na resposta do Amazon Lex só serão suportadas nas mesmas localidades que oferecem suporte às pontuações de transcrição do ASR.

Tipo de valor	Caso de uso	Expressões condicionais
Slot personalizado	O valor do slot `pizzaSize` é igual a grande	`{pizzaSize} = "large"`
Slot personalizado	`pizzaSize` é igual a grande ou médio	`{pizzaSize} = "large"` OU `{pizzaSize} = "medium"`
Slot personalizado	Expressões com `( )` e `AND/OR`	`{pizzaType} = "pepperoni"` OU `{pizzaSize} = "medium"` OU `{pizzaSize} = "small"`
Slot personalizado (slot de vários valores)	Verifique se uma das coberturas é cebola	`{toppings} CONTAINS "Onion"`
Slot personalizado (slot de vários valores)	O número de coberturas é superior a 3	`fn.COUNT({topping}) > 2`
`AMAZON.AlphaNumeric`	`bookingID` é ABC123	`{bookingID} = "ABC123"`
`AMAZON.Number`	o valor do slot de idade é maior que 30	`{age} > 30`
`AMAZON.Number`	o valor do slot de idade é igual a 10	`{age} = 10`
`AMAZON.Date`	O valor do slot `dateOfBirth` anterior a 1990	`{dateOfBirth} < "1990-10-01"`
`AMAZON.State`	O valor do slot `destinationState` é igual a Washington	`{destinationState} = "washington"`

Tipo de valor	Caso de uso	Expressões condicionais
`AMAZON.Country`	O valor do slot `destinationCountry` não é Estados Unidos	`{destinationCountry} != "united states"`
`AMAZON.FirstName`	O valor do slot `firstName` é João	`{firstName} = "John"`
`AMAZON.PhoneNumber`	O valor do slot `phoneNumber` é 716767891932	`{phoneNumer} = 716767891932`
`AMAZON.Percentage`	Verifique se o valor percentual do slot é maior ou igual a 78	`{percentage} >= 78`
`AMAZON.EmailAddress`	O valor do slot `emailAddress` é userA@hmail.com	`{emailAddress} = "userA@hmail.com"`
`AMAZON.LastName`	O valor do slot `lastName` é Fulano	`{lastName} = "Doe"`
`AMAZON.City`	O valor do slot da cidade é igual a Seattle	`{city} = "Seattle"`
`AMAZON.Time`	O horário é depois das 20h	`{time} > "20:00"`
`AMAZON.StreetName`	O valor do slot `streetName` Boren Avenue	`{streetName} = "boren avenue"`
`AMAZON.Duration`	O valor do slot `travelDuration` é inferior a 2 horas	`{travelDuration} < P2H`
Modo de entrada	O modo de entrada é fala	`$.inputMode = "Speech"`
Transcrição de entrada	A transcrição de entrada é igual a “Quero uma pizza grande”	`$.inputTranscript = "I want a large pizza"`

Tipo de valor	Caso de uso	Expressões condicionais
Atributos da sessão	verifique o atributo customer_subscription_type	`[customer_subcription_type] = "yearly"`
Atributos de solicitação	verifique o sinalizador retry_enabled	`((retry_enabled)) = "TRUE"`
Resposta de Kendra	A resposta de Kendra contém perguntas frequentes	`fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1) ))`
Expressão condicional com transcrições	Expressões condicionais usando o caminho JSON de transcrições	`$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5`
Definir atributos de sessão	Defina os atributos da sessão usando transcrições, caminho JSON e valores de slot.	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`
Definir valores do slot	Defina os atributos da sessão usando transcrições, caminho JSON e valores de slot.	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName` refere-se ao nome de um slot no bot Amazon Lex. Se o slot não for resolvido (nulo) ou se o slot não existir, as atribuições serão ignoradas em runtime.

`sessionAttribute` refere-se ao nome do atributo da sessão que é definido pelo cliente no momento da criação.

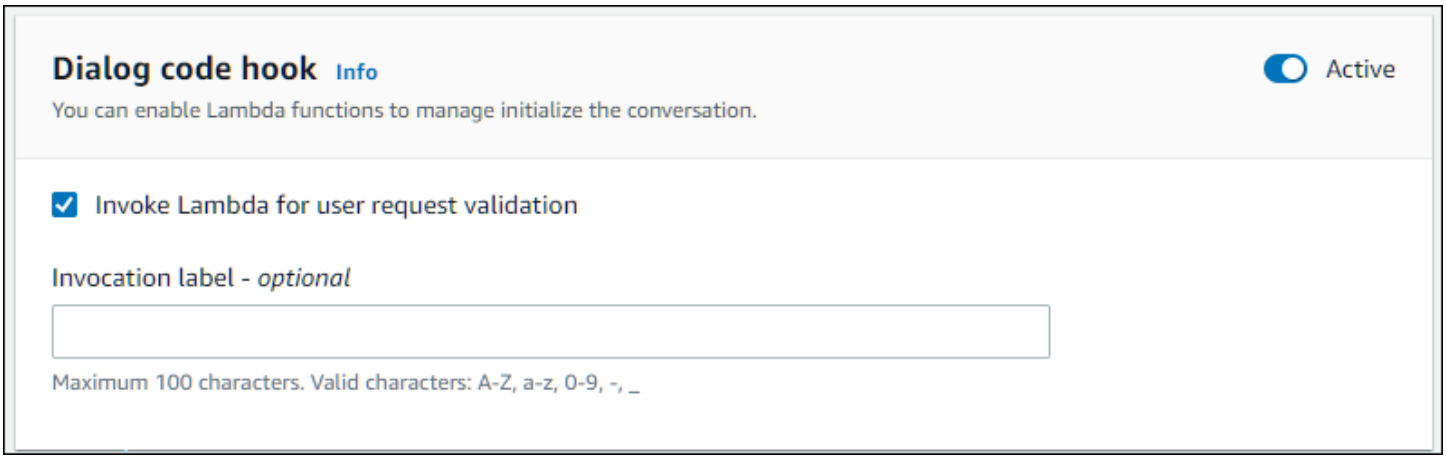
Invocar hook de código de diálogo

Em cada etapa da conversa, quando o Amazon Lex envia uma mensagem ao usuário, você pode usar uma função do Lambda como a próxima etapa da conversa. Você pode usar a função para implementar a lógica de negócios com base no estado atual da conversa.

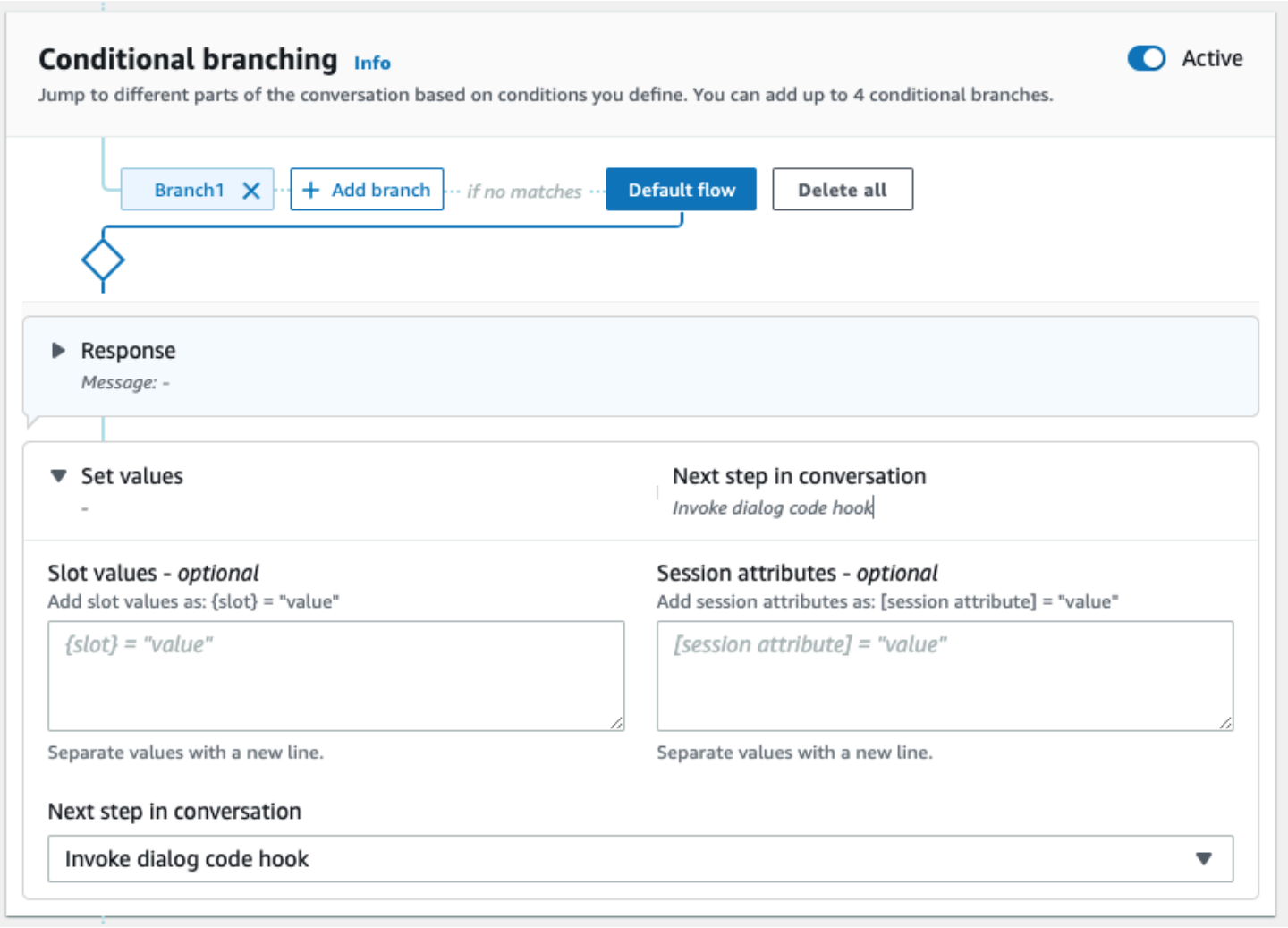
A função do Lambda que é executada está associada ao alias do bot que você está usando. Para invocar a função do Lambda em todos os hooks de código de diálogo em sua intenção, você deve selecionar Usar uma função Lambda para inicializar e validar a intenção. Mais informações sobre como escolher uma função do Lambda, ver Criar e anexar uma função do Lambda a um alias de bot.

Há duas etapas para usar uma função do Lambda. Primeiro, você deve ativar o hook do código de diálogo em qualquer ponto da conversa. Segundo, você deve definir a próxima etapa da conversa para usar o hook de código de diálogo.

A imagem a seguir mostra o hook do código de diálogo ativado.

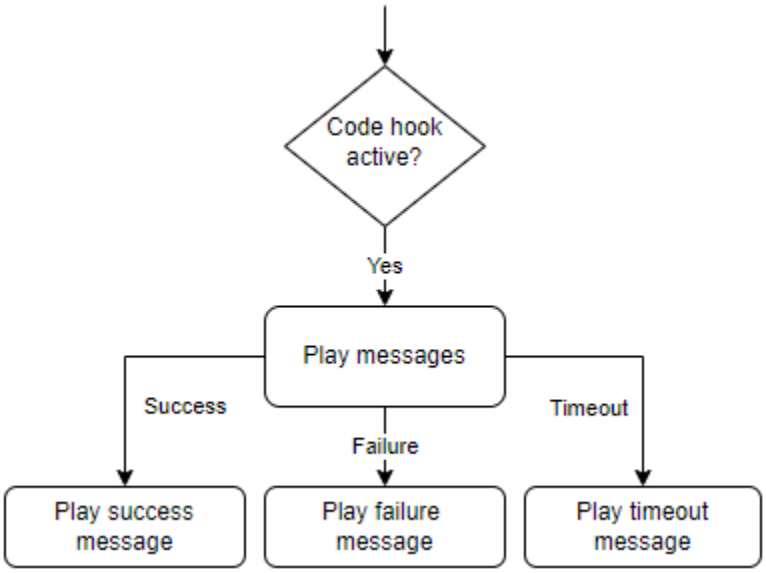


Em seguida, defina o hook de código como a próxima ação para a etapa da conversa. Você pode fazer isso configurando a próxima etapa da conversa para Invocar o hook de código de diálogo. A imagem a seguir mostra uma ramificação condicional em que invocar o hook do código de diálogo é a próxima etapa do caminho padrão da conversa.



Quando os hooks de código estão ativos, você pode definir três respostas para retornar ao usuário:

- Sucesso: enviado quando a função do Lambda foi concluída com sucesso.
- Falha: enviado se houve um problema com a execução da função do Lambda ou se a função do Lambda retornou um valor `intent.state` de `Failed`.
- Tempo limite: enviado se a função do Lambda não for concluída no período de tempo limite configurado.



Escolha o hook de código de diálogo do Lambda e, em seguida, escolha Opções avançadas para ver as três opções de respostas que correspondem à invocação da função do Lambda. É possível definir valores, configurar as próximas etapas e aplicar condições correspondentes a cada resposta para criar o fluxo de conversação. Na ausência de uma condição ou de uma próxima etapa explícita, o Amazon Lex V2 decide a próxima etapa com base no estado atual da conversa.

Na página de opções avançadas, você também pode optar por ativar ou desativar a invocação da função do Lambda. Quando a função é ativada, o hook do código de diálogo é invocado com a invocação do Lambda, seguida pela mensagem de sucesso, falha ou tempo limite com base nos resultados da invocação do Lambda. Quando a função está desativada, o Amazon Lex V2 não executa a função do Lambda e age como se o hook do código de diálogo tivesse sido bem-sucedido.

Você também pode definir um rótulo de invocação que é enviado para a função do Lambda quando ela é invocada por essa mensagem. Isso pode ajudar a identificar a seção da sua função do Lambda a ser executada.

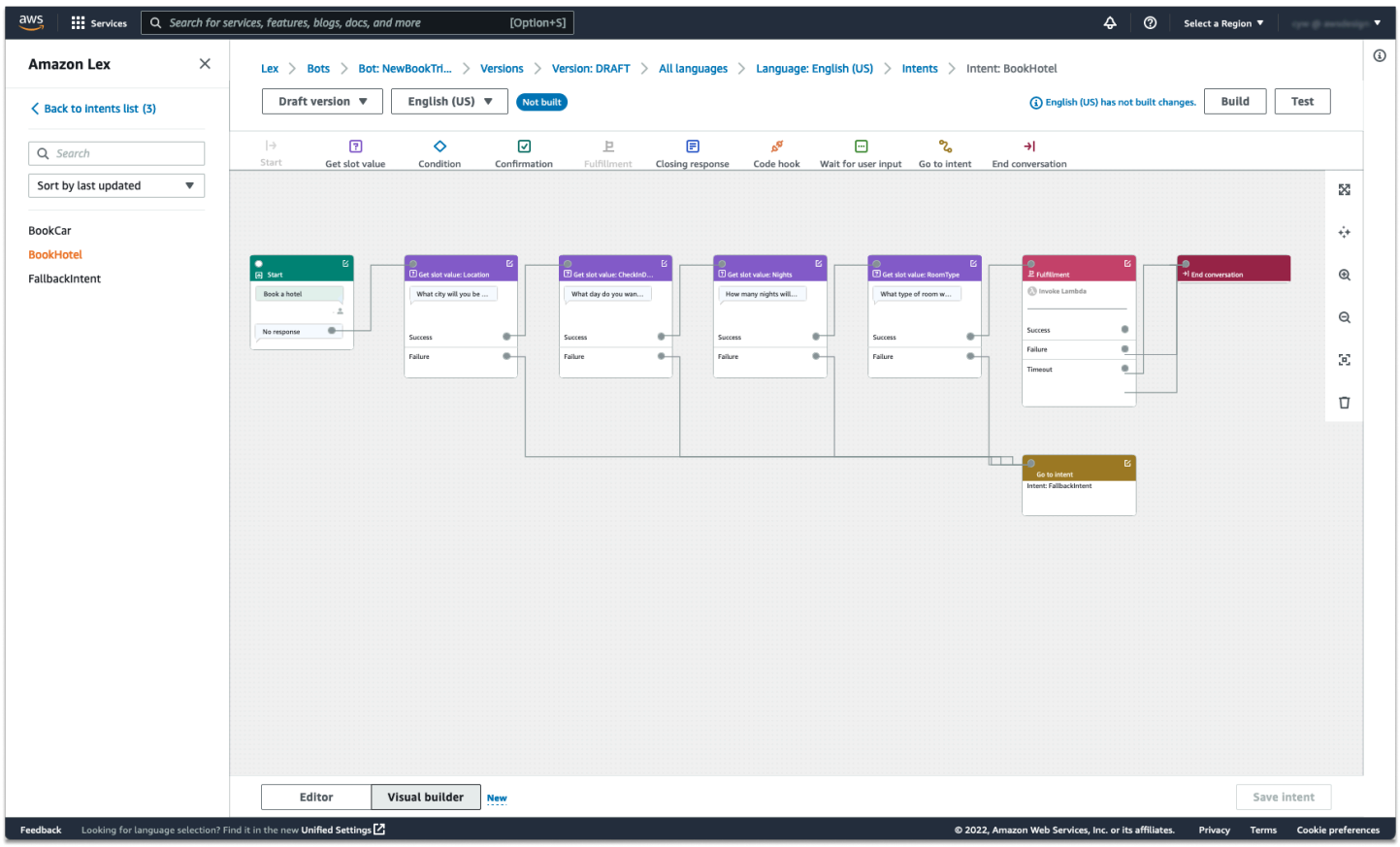
> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para ter mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Usando o Visual Conversation Builder

O Visual Conversation Builder é um criador de conversas de arrastar e soltar para projetar e visualizar facilmente caminhos de conversação usando intenções em um ambiente visual rico.

Acessar o Visual Conversation Builder

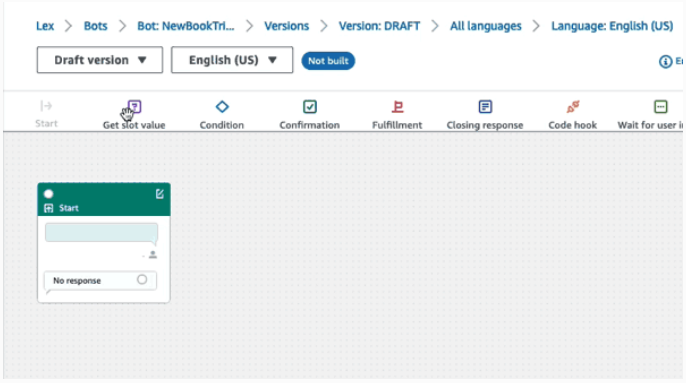
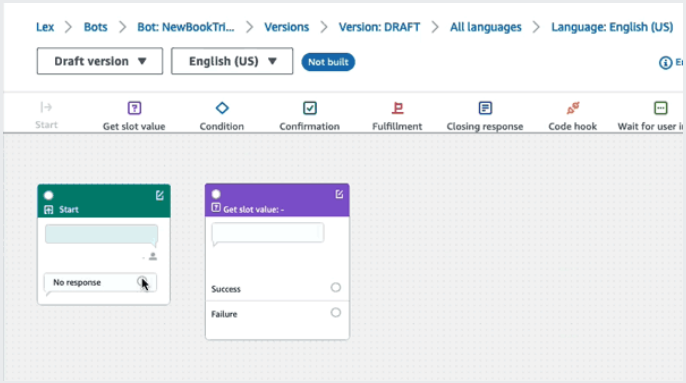
1. No console do Amazon Lex V2, selecione um bot e selecione Intenções no painel de navegação à esquerda.
2. Acesse o editor de intenções de uma das seguintes formas:
   - Selecione Adicionar intenção no canto superior direito da seção Intenções e, em seguida, escolha adicionar uma intenção vazia ou uma intenção incorporada.
   - Escolha o nome de uma intenção na seção Intenções.
3. No editor de intenção, selecione Construtor visual no painel na parte inferior da tela para acessar o Visual Conversation Builder.
4. Para retornar à interface do editor de intenção do menu, selecione Editor.

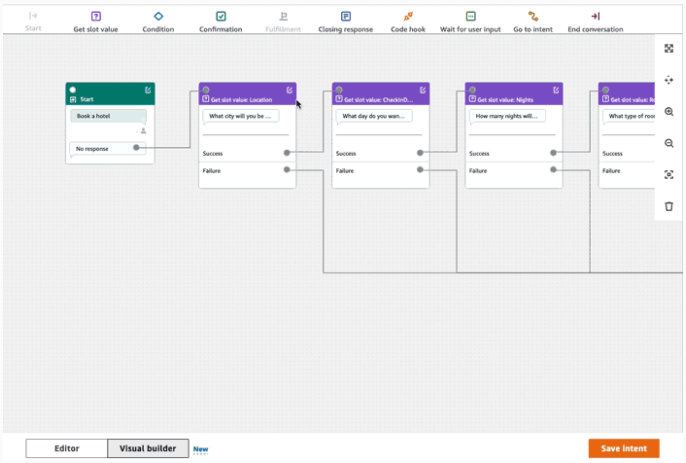
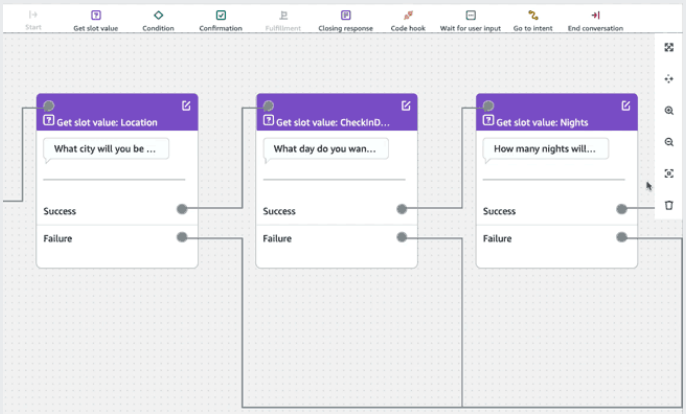
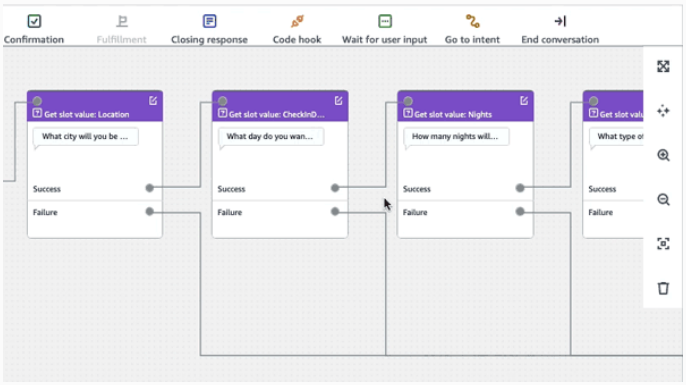


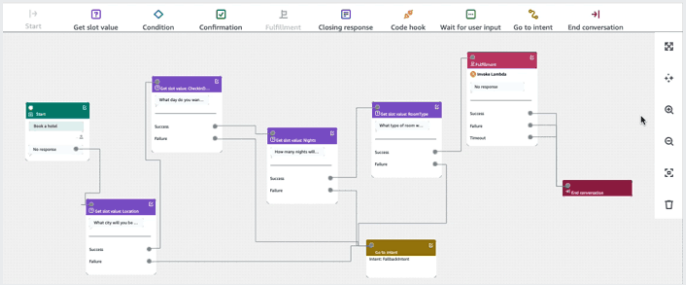
O Visual Conversation Builder oferece uma interface de usuário mais intuitiva com a capacidade de visualizar e modificar o fluxo da conversa. Ao arrastar e soltar os blocos, você pode estender um fluxo existente ou reordenar as etapas da conversa. Você pode desenvolver um fluxo de conversação com ramificações complexas sem escrever nenhum código Lambda.

Essa mudança ajuda a dissociar o design do fluxo de conversação de outras lógicas de negócios no Lambda. O Visual Conversation Builder pode ser usado em conjunto com o editor de intenções existente para criar fluxos de conversação. No entanto, é recomendável usar a visualização do editor visual para fluxos de conversação mais complexos.

Ao salvar uma intenção, o Amazon Lex V2 pode conectar automaticamente as intenções quando determina que há conexões perdidas, o Amazon Lex V2 sugere uma conexão ou você pode selecionar sua própria conexão para o bloco.

Ação	Exemplo
Adicionar um bloco ao espaço de trabalho	
Fazer uma conexão entre blocos	

Ação	Exemplo
Abrir o painel de configuração em um bloco	
Ampliar para caber	
Excluir um bloco do fluxo de conversa	

Ação	Exemplo
Limpeza automática do espaço de trabalho	

Terminologia:

Bloco: a unidade básica de construção de um fluxo de conversação. Cada bloco tem uma funcionalidade específica para lidar com diferentes casos de uso de uma conversa.

Porta: cada bloco contém portas, que podem ser usadas para conectar um bloco a outro. Os blocos podem conter portas de entrada e portas de saída. Cada porta de saída representa uma variação funcional específica de um bloco (como erros, tempos limite ou sucesso).

Borda: uma borda é uma conexão entre a porta de saída de um bloco e a porta de entrada de outro bloco. É parte de uma ramificação em um fluxo de conversação.

Fluxo de conversação: um conjunto de blocos conectados por bordas que descreve as interações em nível de intenção com um cliente.

Blocos

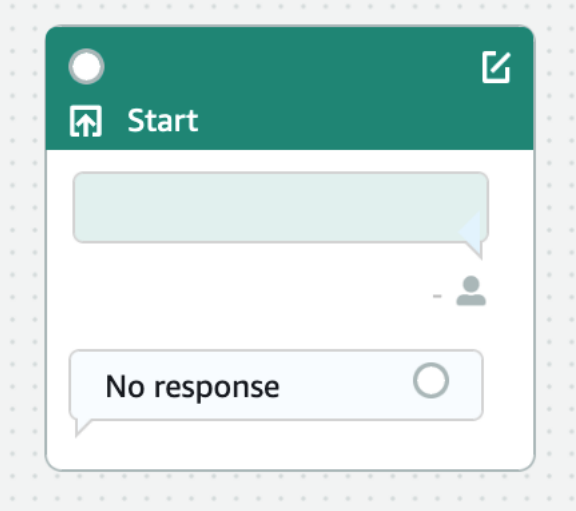
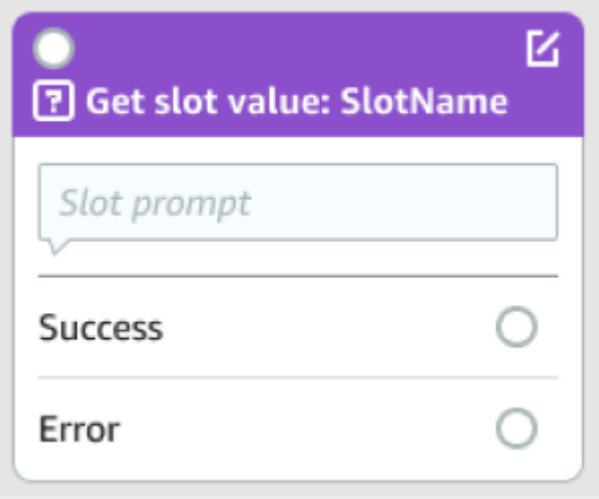
Os blocos são os alicerces de um design de fluxo de conversação. Representam diferentes estados dentro da intenção, que vão desde o início da intenção até a entrada do usuário até o fechamento.

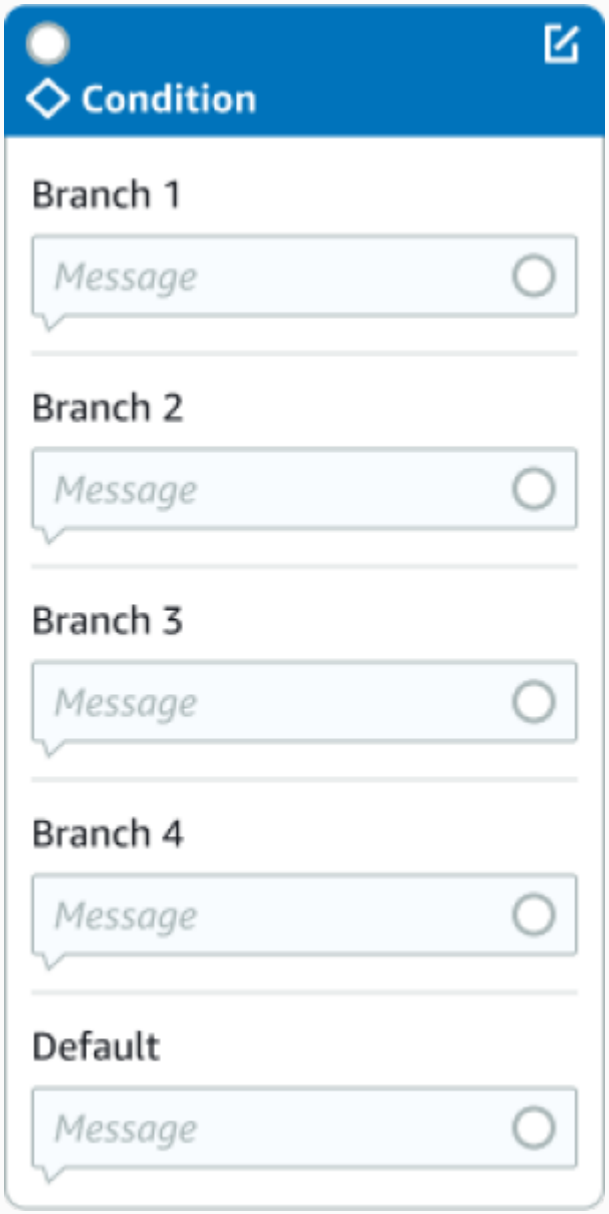
Cada bloco tem um ponto de entrada e um ou vários pontos de saída com base no tipo de bloco. Cada ponto de saída pode ser configurado com uma mensagem correspondente à medida que a conversa prossegue pelos pontos de saída. Para blocos com vários pontos de saída, eles estão relacionados ao status correspondente ao nó. Para um nó condicional, os pontos de saída representam as diferentes condições.

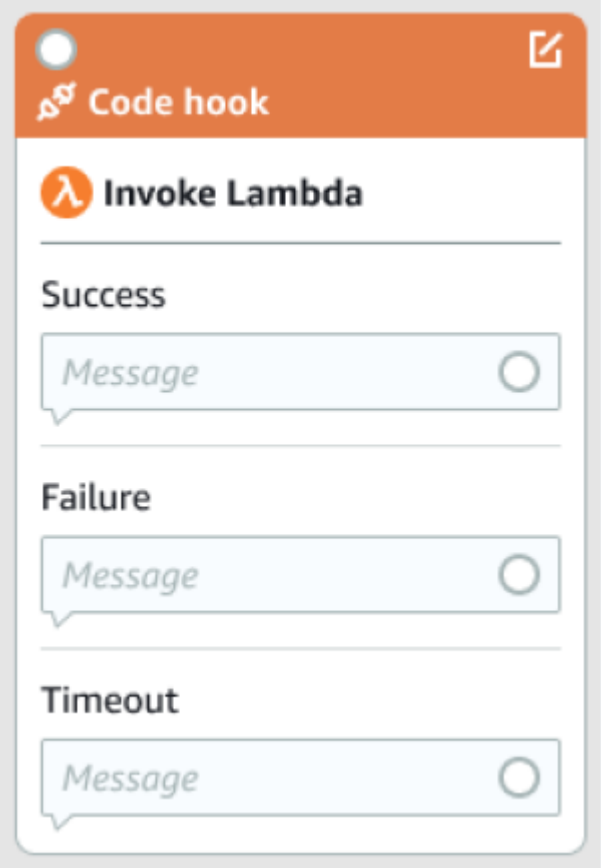
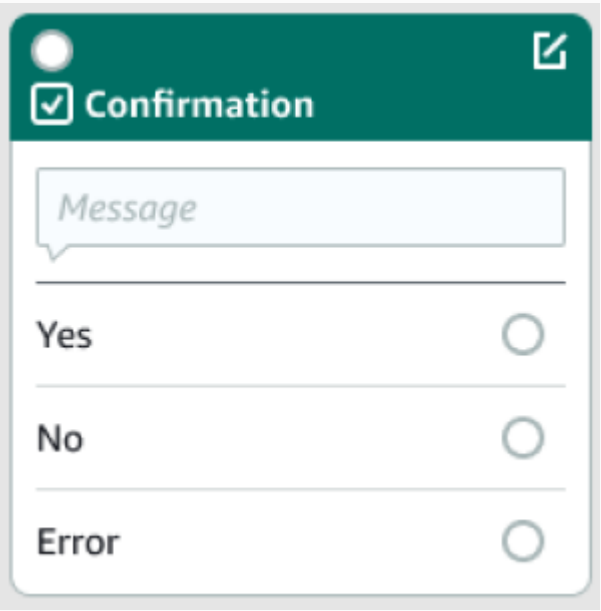
Cada bloco tem um painel de configuração, que abre ao clicar no ícone Editar no canto superior direito do bloco. O painel de configuração contém campos detalhados que podem ser configurados para corresponder a cada bloco.

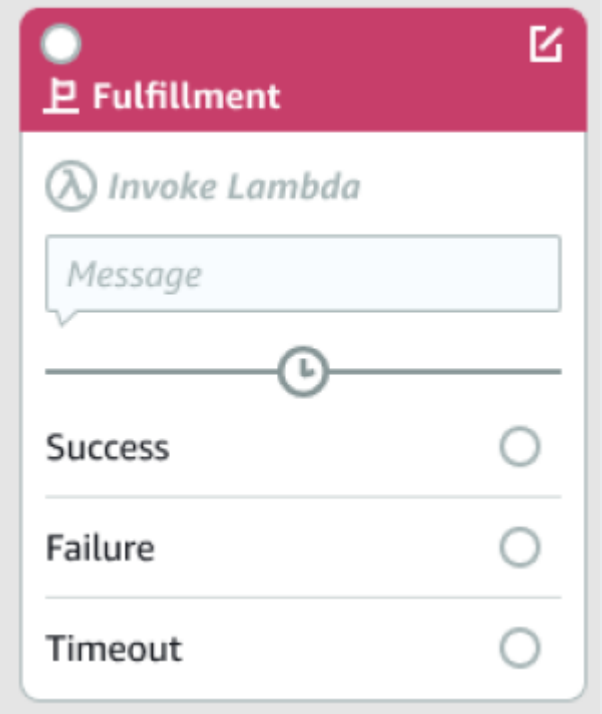
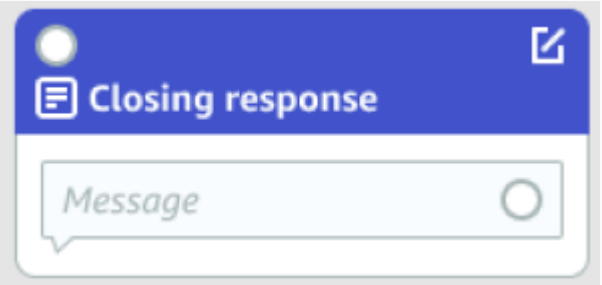
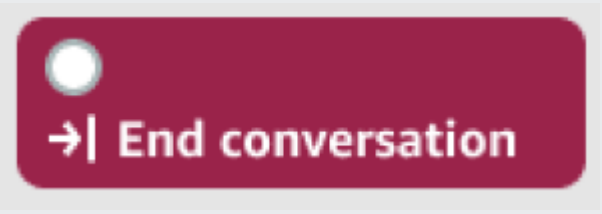
Os prompts e mensagens do bot podem ser configurados diretamente no nó arrastando um novo bloco ou podem ser modificados no painel direito, junto com outros atributos do bloco.

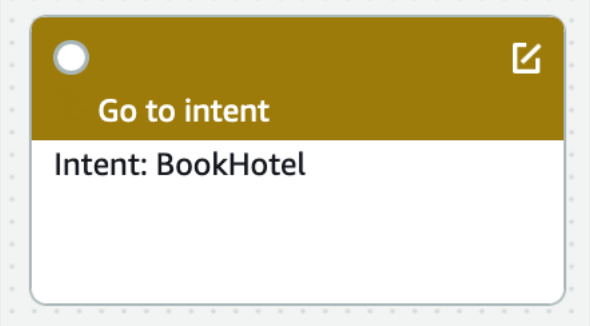
Tipos de bloco: aqui estão os tipos de blocos que você pode usar com o Visual Conversation Builder.

Tipo de bloco	Bloco
Início: a raiz ou o primeiro bloco do fluxo da conversa. Esse bloco também pode ser configurado de forma que o bot possa enviar uma resposta inicial (mensagem de que a intenção foi reconhecida). Para mais informaçõ es, consulte Resposta inicial.	
Obter valor do slot: esse bloco tenta extrair valor para um único slot. Esse bloco tem uma configuração para aguardar a resposta do cliente ao prompt de elicitação do slot. Para mais informações, consulte Slots.	

Tipo de bloco	Bloco
Condição: este bloco contém condicionais. Ele contém até 4 ramificações personalizadas (com condições) e uma ramificação padrão. Para mais informações, consulte Adicionar condições às conversas ramificadas.	

Tipo de bloco	Bloco
Hook de código de diálogo: esse bloco manipula a invocação da função do Lambda de diálogo. Esse bloco contém respostas de bots com base no sucesso, falha ou tempo limite da função do Lambda de diálogo. Para mais informações, consulte Invocar hook de código de diálogo.	
Confirmação: esse bloco consulta o cliente antes do atendimento da intenção. Ele contém respostas de bots com base no cliente dizendo sim ou não ao prompt de confirmação. Para mais informações, consulte Confirmação.	

Tipo de bloco	Bloco
Atendimento: esse bloco trata do atendimento da intenção, geralmente após a elicitação dos slots. Ele pode ser configurado para invocar funções do Lambda, bem como responder com mensagens, se o atendimento for bem-sucedido ou falhar. Para mais informações, consulte Atendimento.	
Resposta de encerramento: esse bloco permite que o bot responda com uma mensagem antes de encerrar a conversa. Para mais informações, consulte Resposta de encerramento.	
Encerrar conversa: esse bloco indica o fim do fluxo da conversa.	
Aguarde a entrada do usuário: esse bloco pode ser usado para capturar a entrada do cliente e mudar para outra intenção com base na expressão.	

Tipo de bloco	Bloco
Ir para a intenção: esse bloco pode ser usado para acessar uma nova intenção ou para obter diretamente um espaço específico dessa intenção.	

Tipos de portas

Todos os blocos contêm uma porta de entrada, que é usada para conectar seus blocos principais. A conversa só pode fluir para a porta de entrada de um determinado bloco a partir da porta de saída do bloco pai. No entanto, os blocos podem conter zero, uma ou várias portas de saída. Os blocos sem nenhuma porta de saída significam o fim do fluxo de conversação na intenção atual (`GoToIntent`, `EndConversation`, `WaitForUserInput`).

Regras de design de intenção:

- Todos os fluxos em uma intenção começam com o bloco inicial.
- As mensagens correspondentes a cada ponto de saída são opcionais.
- Você pode configurar os blocos para definir valores correspondentes a cada ponto de saída no painel de configuração.
- Somente um único bloco de início, confirmação, atendimento e fechamento pode existir em um único fluxo dentro de uma intenção. Podem existir várias condições, hook de código de diálogo, obtenção de valores de slot, término de conversa, transferência e espera por blocos de entrada do usuário.
- Um bloco condicional não pode ter uma conexão direta com um bloco condicional. O mesmo se aplica ao hook de código de diálogo.
- Fluxos circulares são permitidos em três blocos, mas um conector de entrada para Iniciar intenção não é permitido.
- Um slot opcional não tem um conector de entrada ou uma conexão de saída e é usado principalmente para capturar quaisquer dados presentes durante a elicitação da intenção. Todos os outros slots que fazem parte do caminho da conversa devem ser obrigatórios.

Blocos

- O bloco inicial deve ter uma borda de saída.
- Cada bloco obter valor do slot deve ter uma borda de saída da porta de sucesso, se o slot for necessário.
- Cada bloco de condição deve ter uma borda de saída de cada ramificação se o bloco estiver ativo.
- Um bloco condicional não pode ter mais de um pai.
- Um bloco de condição ativo deve ter uma borda de entrada.
- Cada bloco hook de código ativo deve ter uma borda de saída de cada porta: com êxito, com falha e tempo limite.
- Um bloco hook de código ativo deve ter uma borda de entrada.
- Um bloco confirmação ativo deve ter uma borda de entrada.
- Um bloco atendimento ativo deve ter uma borda de entrada.
- Um bloco encerramento ativo deve ter uma borda de entrada.
- Um bloco condicional deve ter pelo menos uma ramificação não padrão.
- Um bloco ir para intenção deve ter uma intenção especificada.

Bordas:

- Um bloco condicional não pode estar conectado a outro bloco condicional.
- Um bloco hook de código não pode ser conectado a outro bloco hook de código.
- Um bloco condicional só pode ser conectado a zero ou a um bloco hook de código.
- A conexão (hook de código -> condição -> hook de código) não é válida.
- Um bloco atendimento não pode ter um bloco hook de código como filho.
- Um bloco condicional, que é filho do bloco de atendimento, não pode ter um bloco hook de código como filho.
- Um bloco encerramento não pode ter um bloco hook de código como filho.
- Um bloco condicional que é filho do bloco de encerramento não pode ter um bloco hook de código como filho.
- Um bloco inicial, de confirmação ou obter valor do slot não pode ter mais do que um bloco hook de código em sua cadeia de dependência.

> **Note**
>
> Em 17 de agosto de 2022, o Amazon Lex V2 lançou uma mudança na forma como as conversas são gerenciadas com o usuário. Essa alteração oferece mais controle sobre o caminho que o usuário percorre na conversa. Para mais informações, consulte Compreender o gerenciamento do fluxo de conversas. Os bots criados antes de 17 de agosto de 2022 não são compatíveis com mensagens de hook de código de diálogo, definição de valores, configuração das próximas etapas e adição de condições.

Intenções integradas

Para ações comuns, você pode usar a biblioteca de intenções integradas padrão. Para criar uma intenção de uma intenção integrada, escolha uma intenção no console e forneça um novo nome. A nova intenção tem a configuração de intenção básica, como os exemplos de enunciados.

Na implementação atual, você não pode:

- Adicionar ou remover exemplos de enunciados da intenção básica
- Configurar slots para intenções integradas

Para adicionar uma intenção integrada a um bot

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot ao qual adicionar a intenção integrada.
3. No menu à esquerda, escolha o idioma e, em seguida, escolha Intenções.
4. Escolha Adicionar intenção e, em seguida, escolha Usar intenção incorporada.
5. Em Intenção incorporada, escolha a intenção a ser usada.
6. Dê um nome à intenção e escolha Adicionar.
7. Use o editor de intenção para configurar a intenção conforme necessário para o seu bot.

Tópicos

- AMAZON.CancelIntent
- AMAZON.FallbackIntent

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

AMAZON.CancelIntent

Responde a palavras e frases que indicam que o usuário deseja cancelar a interação atual. Seu aplicativo pode usar essa intenção para remover valores de tipo de slot e outros atributos antes de encerrar a interação com o usuário.

Enunciados comuns:

- cancelar
- não se preocupe
- esqueça

AMAZON.FallbackIntent

Quando a entrada de um usuário em uma intenção não é o que um bot espera, você pode configurar o Amazon Lex V2 para invocar uma intenção de fallback. Por exemplo, se a entrada do usuário "Eu gostaria de pedir doce" não corresponder a uma intenção em seu bot `OrderFlowers`, o Amazon Lex V2 invocará a intenção de fallback para lidar com a resposta.

O tipo de `AMAZON.FallbackIntent` intenção incorporado é adicionado ao seu bot automaticamente quando você cria um bot usando o console ou quando adiciona uma localidade a um bot usando a CreateBotLocaleoperação.

A invocação de uma intenção de fallback usa duas etapas. Na primeira etapa, a intenção de fallback é correspondida com base na entrada do usuário. Quando a intenção de fallback é correspondida, a maneira como o bot se comporta depende do número de novas tentativas configuradas para um prompt.

O Amazon Lex V2 corresponde à intenção de fallback nestas situações:

- A entrada do usuário para uma intenção não corresponde à entrada esperada pelo bot
- A entrada de áudio é ruído ou a entrada de texto não é reconhecida como palavras.
- A entrada do usuário é ambígua, e o Amazon Lex V2 não consegue determinar qual intenção invocar.

A intenção de fallback é invocada quando:

- Uma intenção não reconhece a entrada do usuário como um valor de slot após o número configurado de tentativas.
- Uma intenção não reconhece a entrada do usuário como uma resposta a um prompt de confirmação após o número configurado de tentativas.

Você não pode adicionar o seguinte a uma intenção de fallback:

- Enunciados
- Slots
- Um prompt de confirmação

Usar uma função do Lambda com uma intenção de fallback

Quando uma intenção de fallback é invocada, a resposta depende da configuração do parâmetro `fulfillmentCodeHook` para a operação CreateIntent. O bot realiza uma das seguintes ações:

- Retorna as informações de intenção para o aplicativo cliente.
- Chama a função do Lambda de validação e preenchimento dos aliases. Ele chama a função com as variáveis de sessão que são definidas para a sessão.

Para obter mais informações sobre como definir a resposta quando uma intenção de fallback é invocada, consulte o parâmetro `fulfillmentCodeHook` da operação CreateIntent.

Se você usar a função do Lambda com sua intenção de fallback, poderá usar essa função para chamar outra intenção ou executar alguma forma de comunicação com o usuário, como coletar um número de retorno de chamada ou abrir uma sessão com um representante de atendimento ao cliente.

Uma intenção de fallback pode ser invocada várias vezes na mesma sessão. Por exemplo, imagine que a função do Lambda usa a ação de diálogo `ElicitIntent` para solicitar ao usuário uma intenção diferente. Se o Amazon Lex V2 não conseguir inferir a intenção do usuário após o número configurado de tentativas, ele invocará a intenção de fallback novamente. Ele também invoca a intenção de fallback quando o usuário não responde com um valor de slot válido após o número configurado de tentativas.

É possível configurar uma função do Lambda para controlar o número de vezes que a intenção de fallback é chamada usando uma variável de sessão. Sua função do Lambda poderá executar uma ação diferente se for chamada mais vezes do que o limite definido na função do Lambda. Para obter mais informações sobre variáveis de sessão, consulte Definição dos atributos da sessão.

AMAZON.HelpIntent

Responde a palavras ou frases que indicam que o usuário precisa de ajuda ao interagir com seu bot. Quando essa intenção é invocada, você pode configurar o aplicativo ou a função do Lambda para fornecer informações sobre os recursos do seu bot, fazer perguntas de acompanhamento sobre áreas de ajuda ou entregar a interação a um atendente humano.

Enunciados comuns:

- ajuda
- ajude-me
- você pode me ajudar

AMAZON.KendraSearchIntent

Para pesquisar documentos indexados com o Amazon Kendra, use a intenção `AMAZON.KendraSearchIntent`. Quando o Amazon Lex V2 não consegue determinar a próxima ação em uma conversa com o usuário, ele aciona a intenção de pesquisa.

O `AMAZON.KendraSearchIntent` está disponível somente em Inglês (EUA) (inglês-EUA) e nas regiões Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon) e Europa (Irlanda).

O Amazon Kendra é machine-learning-based um serviço de pesquisa que indexa documentos em linguagem natural, como documentos PDF ou arquivos do Microsoft Word. Ele pode pesquisar documentos indexados e retornar os seguintes tipos de respostas a uma pergunta:

- Uma resposta

- Uma entrada de uma pergunta frequente que pode responder à pergunta
- Um documento relacionado à pergunta

Para ver um exemplo de uso de `AMAZON.KendraSearchIntent`, consulte Exemplo: criar um bot de perguntas frequentes para um índice do Amazon Kendra..

Se você configurar uma intenção `AMAZON.KendraSearchIntent` para o bot, o Amazon Lex V2 chamará a intenção sempre que não conseguir determinar o enunciado do usuário para uma intenção. Se não houver resposta do Amazon Kendra, a conversa continuará conforme configurado no bot.

> **Note**
>
> No momento, o Amazon Lex V2 não oferece suporte ao `AMAZON.KendraSearchIntent` durante a elicitação de slots. Se o Amazon Lex V2 não puder determinar a expressão do usuário para um slot, ele chama o `AMAZON.FallbackIntent`.

Quando `AMAZON.KendraSearchIntent` é usado com o `AMAZON.FallbackIntent` no mesmo bot, o Amazon Lex V2 usa as intenções da seguinte forma:

1. O Amazon Lex V2 chama `AMAZON.KendraSearchIntent`. A intenção chama a operação `Query` do Amazon Kendra.
2. Se o Amazon Kendra retornar uma resposta, o Amazon Lex V2 exibirá o resultado para o usuário.
3. Se não houver resposta do Amazon Kendra, o Amazon Lex V2 avisará novamente o usuário. A próxima ação depende da resposta do usuário.
   - Se a resposta do usuário contiver um enunciado reconhecido pelo Amazon Lex V2, como preencher um valor de slot ou confirmar uma intenção, a conversa com o usuário prosseguirá conforme configurado para o bot.
   - Se a resposta do usuário não contiver um enunciado reconhecido pelo Amazon Lex V2, o Amazon Lex V2 fará outra chamada para a operação `Query`.
4. Se não houver resposta após o número configurado de novas tentativas, o Amazon Lex V2 chamará `AMAZON.FallbackIntent` e encerrará a conversa com o usuário.

Há três maneiras de usar `AMAZON.KendraSearchIntent` para fazer uma solicitação ao Amazon Kendra:

- Deixe que a intenção da pesquisa faça a solicitação por você. O Amazon Lex V2 chama o Amazon Kendra com o enunciado do usuário como a string de pesquisa. Ao criar a intenção, você pode definir uma string de filtro de consulta que limite o número de respostas retornadas pelo Amazon Kendra. O Amazon Lex V2 usa o filtro na solicitação de consulta.
- Adicione outros parâmetros de consulta à solicitação para restringir os resultados da pesquisa usando a função do Lambda. Adicione um campo `kendraQueryFilterString` que contém parâmetros de consulta do Amazon Kendra à ação de diálogo `delegate`. Quando parâmetros de consulta são adicionados à solicitação com a função do Lambda, eles têm precedência sobre o filtro de consulta definido quando a intenção foi criada.
- Crie uma consulta usando a função do Lambda. É possível criar uma solicitação de consulta completa do Amazon Kendra enviada pelo Amazon Lex V2. Especifique a consulta no campo `kendraQueryRequestPayload` na ação de diálogo `delegate`. O campo `kendraQueryRequestPayload` tem precedência sobre o campo `kendraQueryFilterString`.

Para especificar o parâmetro `queryFilterString` ao criar um bot ou especificar o campo `kendraQueryFilterString` ao chamar a ação `delegate` em uma função do Lambda do diálogo, especifique uma string usada como filtro de atributo para a consulta do Amazon Kendra. Se a string não for um filtro de atributo válido, você receberá uma exceção `InvalidBotConfigException` em runtime. Para obter mais informações sobre filtros de atributo, consulte Usar atributos de documento para filtrar consultas no Guia do desenvolvedor do Amazon Kendra.

Para ter controle sobre a consulta enviada pelo Amazon Lex V2 para o Amazon Kendra, é possível especificar uma consulta no campo `kendraQueryRequestPayload` na função do Lambda. Se a consulta não for válida, o Amazon Lex V2 retornará uma exceção `InvalidLambdaResponseException`. Para obter mais informações, consulte a Operação de consulta no Guia do desenvolvedor do Amazon Kendra.

Para obter um exemplo de como usar a `AMAZON.KendraSearchIntent`, consulte Exemplo: criar um bot de perguntas frequentes para um índice do Amazon Kendra..

Política do IAM para pesquisa Amazon Kendra

Para usar a `AMAZON.KendraSearchIntent` intenção, você deve usar uma função que forneça políticas AWS Identity and Access Management (IAM) que permitam que o Amazon Lex V2 assuma uma função de tempo de execução que tenha permissão para chamar a intenção do Amazon Kendra. `Query` As configurações do IAM que você usa dependem de você criá-las `AMAZON.KendraSearchIntent` usando o console Amazon Lex V2 ou usando um SDK da AWS ou

o AWS Command Line Interface ()AWS CLI. Quando o console é usado, é possível escolher entre adicionar permissão para chamar o Amazon Kendra para a função vinculada ao serviço do Amazon Lex V2 ou usar uma função especificamente para chamar a operação `Query` do Amazon Kendra. Ao usar o AWS CLI ou um SDK para criar a intenção, você deve usar uma função específica para chamar a `Query` operação.

Anexar permissões

É possível usar o console para anexar permissões para acessar a operação `Query` do Amazon Kendra à função vinculada ao serviço padrão do Amazon Lex V2. Quando você anexa permissões à função vinculada ao serviço, não precisa criar e gerenciar uma função de runtime especificamente para se conectar ao índice do Amazon Kendra.

O usuário, a função ou o grupo usado para acessar o console do Amazon Lex V2 deve ter permissões para gerenciar políticas de função. Anexe a política do IAM à função de acesso do console. Quando essas permissões são concedidas, a função tem permissões para alterar a política de função vinculada ao serviço existente.

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

Especificar uma função

Você pode usar o console AWS CLI, o ou a API para especificar uma função de tempo de execução a ser usada ao chamar a operação Amazon `Query` Kendra.

O usuário, a função ou o grupo utilizado para especificar a função de runtime deve ter a permissão `iam:PassRole`. A política a seguir define a permissão. É possível usar as chaves de contexto de condição `iam:AssociatedResourceArn` e `iam:PassedToService` para limitar ainda mais o escopo das permissões. Para obter mais informações, consulte IAM e AWS STS Condition Context Keys no Guia AWS Identity and Access Management do usuário.

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

A função de runtime que o Amazon Lex V2 precisa usar para chamar o Amazon Kendra deve ter as permissões `kendra:Query`. Quando você usa um perfil do IAM existente para permissão para chamar a operação `Query` do Amazon Kendra, a função deve ter a política a seguir anexada.

É possível usar o console do IAM, a API do IAM ou a AWS CLI para criar uma política e anexá-la a uma função. Essas instruções usam a CLI da AWS para criar a função e as políticas.

> **Note**
>
> O código a seguir é formatado para Linux e MacOS. Para Windows, substitua o caractere de continuação de linha do Linux (\) pelo circunflexo (^).

Como adicionar permissão de operação de consulta a uma função

1. Crie um documento chamado **`KendraQueryPolicy.json`** no diretório atual, adicione a ele o código a seguir e salve-o.

   ```
   {
   ```

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

2. No AWS CLI, execute o comando a seguir para criar a política do IAM para executar a operação Amazon `Query` Kendra.

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

3. Anexe a política ao perfil do IAM utilizado para chamar a operação `Query`.

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

É possível optar por atualizar a função vinculada ao serviço do Amazon Lex V2 ou usar uma função que você criou ao criar o `AMAZON.KendraSearchIntent` para o bot. O procedimento a seguir mostra como escolher o perfil do IAM a ser usado.

Como especificar a função de runtime para AMAZON.KendraSearchIntent

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot ao qual você deseja adicionar a `AMAZON.KendraSearchIntent`.
3. Escolha o sinal de adição (+) ao lado de Intenções.
4. Em Adicionar intenção, escolha Pesquisar intenções existentes.
5. Em Intenções de pesquisa, insira **`AMAZON.KendraSearchIntent`** e escolha Adicionar.

6. Em Copiar intenção interna, insira um nome para a intenção, como **`KendraSearchIntent`**, e escolha Adicionar.
7. Abra a seção Consulta do Amazon Kendra.
8. Para Perfil do IAM, escolha uma das seguintes opções:

   - Para atualizar a função vinculada ao serviço do Amazon Lex V2 para permitir que o bot consulte índices do Amazon Kendra, escolha Adicionar permissões do Amazon Kendra.
   - Para usar uma função que tenha permissão para chamar a operação `Query` do Amazon Kendra, escolha Usar uma função existente.

Usar atributos de solicitação e sessão como filtros

Para filtrar a resposta do Amazon Kendra a itens relacionados à conversa atual, use atributos de sessão e solicitação como filtros adicionando o parâmetro `queryFilterString` ao criar o bot. Especifique um espaço reservado para o atributo ao criar a intenção e, depois, o Amazon Lex V2 um valor antes de chamar o Amazon Kendra. Para obter mais informações sobre atributos de solicitação, consulte Definição de atributos de solicitação. Para obter mais informações sobre atributos de sessão, consulte Definição dos atributos da sessão.

Veja a seguir um exemplo de parâmetro `queryFilterString` que usa uma string para filtrar a consulta do Amazon Kendra.

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

Veja a seguir um exemplo de um parâmetro `queryFilterString` que usa um atributo de sessão chamado `"SourceURI"` para filtrar a consulta do Amazon Kendra.

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

Veja a seguir um exemplo de parâmetro `queryFilterString` que usa um atributo de solicitação chamado `"DepartmentName"` para filtrar a consulta do Amazon Kendra.

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

Os filtros `AMAZON.KendraSearchInteng` usam o mesmo formato dos filtros de pesquisa do Amazon Kendra. Para obter mais informações, consulte Usar atributos de documento para filtrar os resultados da pesquisa no Guia do desenvolvedor do Amazon Kendra.

A string do filtro de consulta usada com o `AMAZON.KendraSearchIntent` deve usar letras minúsculas para a primeira letra de cada filtro. Por exemplo, veja a seguir um filtro de consulta válido para o `AMAZON.KendraSearchIntent`.

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

Usar a resposta da pesquisa

O Amazon Kendra retorna a resposta a uma pesquisa na instrução `IntentClosingSetting` da intenção. A intenção deve ter uma instrução `closingResponse`, a menos que uma função do Lambda produza uma mensagem de resposta de conclusão.

O Amazon Kendra tem cinco tipos de respostas.

- As duas respostas a seguir exigem que uma pergunta frequente seja configurada para seu índice Amazon Kendra. Para obter mais detalhes, consulte Adicionar perguntas e respostas diretamente a um índice.
  - `x-amz-lex:kendra-search-response-question_answer-question-<N>`: a pergunta de uma pergunta frequente correspondente à pesquisa.
  - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`: a resposta de uma pergunta frequente correspondente à pesquisa.

- As três respostas a seguir exigem que uma fonte de dados seja configurada para seu índice Amazon Kendra. Para obter mais detalhes, consulte Criação de uma fonte de dados.
  - `x-amz-lex:kendra-search-response-document-<N>`: um trecho de um documento no índice relacionado ao texto do enunciado.
  - `x-amz-lex:kendra-search-response-document-link-<N>`: o URL de um documento no índice relacionado ao texto do enunciado.
  - `x-amz-lex:kendra-search-response-answer-<N>`: um trecho de um documento no índice que responde à pergunta.

As respostas são retornadas em atributos `request`. Pode haver até cinco respostas para cada atributo, numeradas de 1 a 5. Para obter mais informações sobre respostas, consulte Tipos de resposta no Guia do desenvolvedor do Amazon Kendra.

A instrução `closingResponse` deve ter um ou mais grupos de mensagens. Cada grupo de mensagens contém uma ou mais mensagens. Cada mensagem pode conter uma ou mais variáveis de espaço reservado que são substituídas por atributos de solicitação na resposta do Amazon Kendra. Deve haver pelo menos uma mensagem no grupo de mensagens em que todas as variáveis na mensagem são substituídas por valores de atributo de solicitação na resposta de runtime, ou deve haver uma mensagem no grupo sem variáveis de espaço reservado. Os atributos de solicitação são ativados com parênteses duplos ("(("))"). As mensagens do grupo de mensagens a seguir correspondem a qualquer resposta do Amazon Kendra:

- “Encontrei uma pergunta de FAQ para você: ((x-amz-lex: kendra-search-response-question _resposta-pergunta-1)), e a resposta é ((x-amz-lex: _resposta-resposta-1))” kendra-search-response-question
- “Encontrei um trecho de um documento útil: ((x-amz-lex: kendra-search-response-document -1))”
- “Acho que a resposta às suas perguntas é ((x-amz-lex: kendra-search-response-answer -1))”

Usar uma função do Lambda para gerenciar a solicitação e a resposta

A intenção `AMAZON.KendraSearchIntent` pode usar o hook de código de diálogo e o hook de código de atendimento para gerenciar a solicitação ao Amazon Kendra e a resposta. Utilize a função do Lambda do hook de código de diálogo quando quiser modificar a consulta enviada ao Amazon Kendra e a função do Lambda de hook de código de atendimento quando quiser modificar a resposta.

Criar uma consulta com o hook de código de diálogo

É possível usar o hook de código de diálogo para criar uma consulta para enviar ao Amazon Kendra. O uso do hook de código de diálogo é opcional. Se você não especificar um hook de código de diálogo, o Amazon Lex V2 criará uma consulta a partir do enunciado do usuário e usará a `queryFilterString` fornecida quando você configurou a intenção, se você a tiver fornecido.

É possível usar dois campos na resposta do hook de código de diálogo para modificar a solicitação ao Amazon Kendra:

- `kendraQueryFilterString`: use essa string para especificar filtros de atributo para a solicitação do Amazon Kendra. É possível filtrar a consulta usando qualquer um dos campos de índice definidos no índice. Para obter a estrutura da string de filtro, consulte Usar atributos de documento para filtrar consultas no Guia do desenvolvedor do Amazon Kendra. Se a string de filtro especificada não for válida, você receberá uma exceção `InvalidLambdaResponseException`. A string `kendraQueryFilterString` substitui qualquer string de consulta especificada na `queryFilterString` configurada para a intenção.
- `kendraQueryRequestPayload` : use essa string para especificar uma consulta do Amazon Kendra. Sua consulta pode usar qualquer um dos atributos do Amazon Kendra. Se você não especificar uma consulta válida, receberá uma exceção `InvalidLambdaResponseException`. Para obter mais informações, confira Consulta no Guia do desenvolvedor do Amazon Kendra.

Depois de criar o filtro ou a sequência de caracteres de consulta, você envia a resposta para o Amazon Lex V2 com o `dialogAction` campo da resposta definido como `delegate`. O Amazon Lex V2 envia a consulta para a Amazon Kendra e, em seguida, retorna a resposta da consulta ao hook do código de atendimento.

Usar o hook de código de atendimento para a resposta

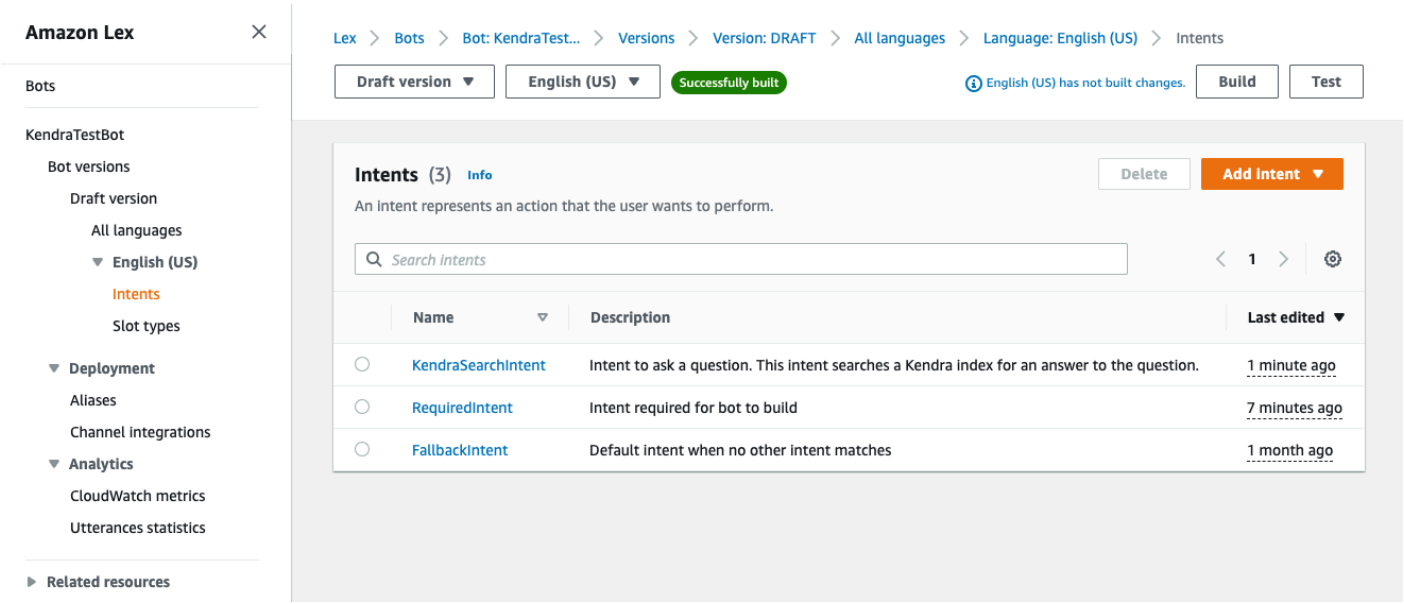
Depois que o Amazon Lex V2 envia uma consulta ao Amazon Kendra, a resposta da consulta é retornada para a função de cumprimento `AMAZON.KendraSearchIntent` da função do Lambda. O evento de entrada para o hook de código contém a resposta completa do Amazon Kendra. Os dados da consulta estão na mesma estrutura que os retornados pela operação `Query` do Amazon Kendra. Para obter mais informações, consulte Sintaxe de resposta de consulta no Guia do desenvolvedor do Amazon Kendra.

O hook de código de atendimento é opcional. Se não houver nenhum ou se o hook de código não retornar uma mensagem na resposta, o Amazon Lex V2 usará a instrução `closingResponse` para respostas.

Exemplo: criar um bot de perguntas frequentes para um índice do Amazon Kendra.

Este exemplo cria um bot do Amazon Lex V2 que usa um índice do Amazon Kendra para fornecer respostas às perguntas dos usuários. O bot de perguntas frequentes gerencia a caixa de diálogo para o usuário. Ele usa a intenção `AMAZON.KendraSearchIntent` para consultar o índice e apresentar a resposta ao usuário. Aqui está um resumo de como você criará seu bot de perguntas frequentes usando um índice da Amazon Kendra:

1. Criar um bot com o qual seus clientes vão interagir para obter respostas do bot.
2. Criar uma intenção personalizada. Como `AMAZON.KendraSearchIntent` e `AMAZON.FallbackIntent` são intenções secundárias, seu bot exige pelo menos uma outra intenção que contenha pelo menos um enunciado. Essa intenção permite que o bot crie, mas não seja usado de outra forma. Portanto, seu bot de perguntas frequentes conterá pelo menos três intenções, como na imagem abaixo:



3. Adicionar a intenção `AMAZON.KendraSearchIntent` ao bot e configurá-lo para trabalhar com o índice do Amazon Kendra.
4. Teste o bot fazendo uma consulta e verificando se os resultados do seu índice Amazon Kendra são documentos que respondem à consulta.

Pré-requisitos

Antes de usar este exemplo, é necessário criar um índice do Amazon Kendra. Para obter mais informações, consulte Começar a usar um console do Amazon Kendra no Guia do desenvolvedor do

Amazon Kendra. Para este exemplo, escolha o conjunto de dados de amostra (Documentação da AWS de amostra) como sua fonte de dados.

Como criar um bot de perguntas frequentes:

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. No painel de navegação, escolha Bots.
3. Escolha Criar bot.
    a. Para Método de criação, escolha Criar um bot em branco.
    b. Na seção Configuração do bot, dê ao bot um nome que indique sua função, como **`KendraTestBot`**, e uma descrição opcional. O nome deve ser exclusivo em sua conta.
    c. A seção Permissões do IAM, a opção Criar um perfil com permissões básicas do Amazon Lex. Isso criará um perfil do AWS Identity and Access Management (IAM) com as permissões que o Amazon Lex V2 precisa para executar seu bot.
    d. Na seção Lei de Proteção à Privacidade Online para Crianças (COPPA), escolha Não.
    e. Nas seções Tempo limite da sessão ociosa e Configurações avançadas, deixe as configurações padrão e escolha Avançar.
    f. Agora você está na seção Adicionar idioma ao bot. No menu, em Interação por voz, selecione Nenhuma. Este é apenas um aplicativo baseado em texto. Mantenha as configurações padrão para os campos restantes.
    g. Escolha Concluído. O Amazon Lex V2 cria seu bot e uma intenção padrão chamada NewIntent, e leva você à página para configurar essa intenção.

Para criar um bot com sucesso, você deve criar pelo menos uma intenção separada de `AMAZON.FallbackIntent` e de `AMAZON.KendraSearchIntent`. Essa intenção é necessária para criar o bot do Amazon Lex V2, mas não é usada para a resposta de perguntas frequentes. Essa intenção deve conter pelo menos um exemplo de enunciado e ele não deve se aplicar a nenhuma das perguntas feitas pelo cliente.

Como criar a intenção necessária:

1. Na seção Detalhes da intenção, dê um nome à intenção, como **`RequiredIntent`**.
2. Na seção Exemplos de enunciados, digite um enunciado na caixa ao lado de Adicionar enunciado, como **`Required utterance`**. Em seguida, escolha Adicionar enunciado.

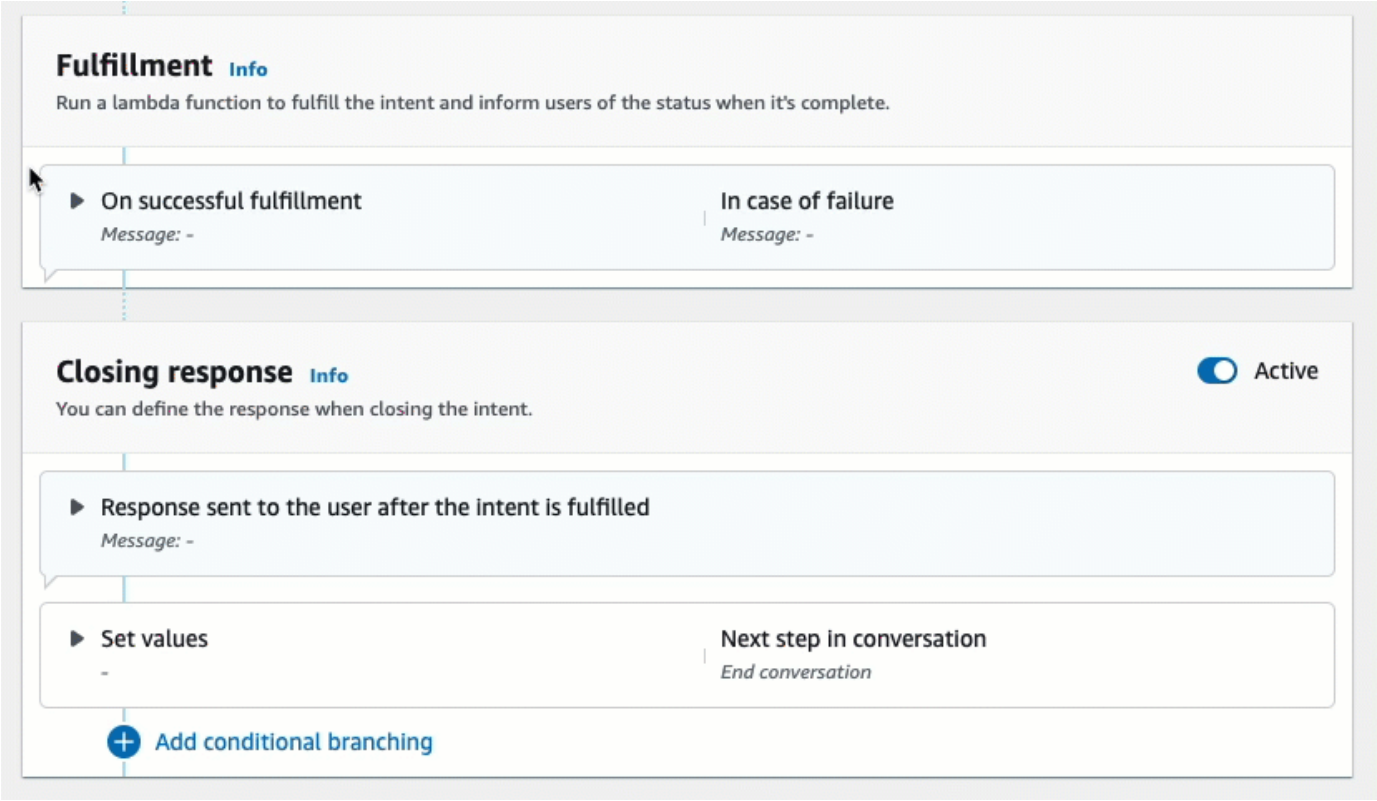
3. Selecione Salvar intenção.

Crie a intenção de pesquisar um índice do Amazon Kendra e a mensagem de resposta que deve ser retornada.

Para criar uma AMAZON. KendraSearchIntent mensagem de intenção e resposta:

1. Selecione Voltar à lista de intenções no painel de navegação para retornar à página Intenções do seu bot. Escolha Adicionar intenção e selecione Usar intenção incorporada no menu suspenso.
2. Na caixa que aparece, selecione o menu em Intenção incorporada. Insira **`AMAZON.KendraSearchIntent`** na barra de busca e então escolha-a na lista.
3. Dê à entrada um nome, como **`KendraSearchIntent`**.
4. No menu suspenso Índice do Amazon Kendra, escolha o índice que a intenção deve pesquisar. O índice que você criou na seção Pré-requisitos deve estar disponível.
5. Selecione Adicionar.
6. No editor de intenções, role para baixo até a seção Processamento, selecione a seta para a direita para expandir a seção e adicione a seguinte mensagem na caixa em Em processamento bem-sucedido:

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



Para obter mais informações sobre a resposta de pesquisa do Amazon Kendra, consulte Usando a resposta de pesquisa.

7. Escolha Salvar intenção e selecione Criar para criar o bot. Quando o bot estiver pronto, o banner na parte superior da tela ficará verde e exibirá uma mensagem de sucesso.

Por fim, use a janela de teste do console para testar as respostas do bot.

Para testar o bot de perguntas frequentes:

1. Depois que o bot for construído com sucesso, selecione Testar.
2. Insira **`What is Amazon Kendra?`** na janela de teste do console. Verifique se o bot responde com um link.
3. Para obter mais informações sobre configuraçãoAMAZON.KendraSearchIntent, consulte AMAZON.KendraSearchIntente. KendraConfiguration

AMAZON.PauseIntent

Responde a palavras e frases que permitem ao usuário pausar uma interação com um bot para que ele possa retornar a ela mais tarde. Sua função ou aplicativo Lambda precisa salvar dados de intenção em variáveis de sessão, ou você precisa usar a GetSession operação para recuperar dados de intenção ao retomar a intenção atual.

Enunciados comuns:

- pausar
- pausar esse item

AMAZON.QnAIntent

> **Note**
>
> Antes de aproveitar os recursos de IA generativa, você deve atender aos seguintes pré-requisitos:
>
> 1. Navegue até o console do Amazon Bedrock e inscreva-se para acessar o modelo Anthropic Claude que você pretende usar (para obter mais informações, consulte Acesso ao modelo). Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock.
> 2. Ative os recursos de IA generativa para a localidade do seu bot. Para isso, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa.

Responde às perguntas do cliente usando um FM do Amazon Bedrock para pesquisar e resumir as respostas das perguntas frequentes. Essa intenção será ativada quando um enunciado não for classificado em nenhuma das outras intenções presentes no bot. Observe que essa intenção não será ativada para enunciados perdidos ao inferir um valor de slot. Uma vez reconhecido, a `AMAZON.QnAIntent`, usa o modelo especificado do Amazon Bedrock para pesquisar a base de conhecimento configurada e responder à pergunta do cliente.

Se a resposta do FM for insatisfatória ou se houver falha na chamada para o FM, o Amazon Lex V2 então invocará a `AMAZON.FallbackIntent`.

> **Warning**
>
> Você não pode usar a `AMAZON.QnAIntent` e a `AMAZON.KendraSearchIntent` na mesma localidade do bot.

As opções de repositório de conhecimento a seguir estão disponíveis. Você já deve ter criado o repositório de conhecimento e indexado os documentos nele.

- OpenSearch Domínio do serviço — contém documentos indexados. Para criar um domínio, siga as etapas em Criação e gerenciamento de domínios do Amazon OpenSearch Service.
- Índice Amazon Kendra: contém documentos indexados de perguntas frequentes. Para criar um índice do Amazon Kendra, siga as etapas em Criar um índice.
- Base de conhecimento Amazon Bedrock: contém fontes de dados indexadas. Para configurar uma base de conhecimento, siga as etapas em Criar uma base de conhecimento.

Se você selecionar essa intenção, configure os campos a seguir e selecione Adicionar para adicionar a intenção.

- Modelo do Bedrock: escolha o fornecedor e o modelo de base a ser usado para essa intenção. Atualmente, o Anthropic Claude V2 e o Anthropic Claude Instant são suportados.
- Repositório de conhecimento: escolha a fonte da qual você deseja que o modelo extraia informações para responder às perguntas dos clientes. As fontes a seguir estão disponíveis.
  - OpenSearch— Configure os seguintes campos.
    - Endpoint do domínio: forneça o endpoint do domínio que você criou para o domínio ou o que foi fornecido após a criação do domínio.
    - Nome do índice: forneça o índice a ser pesquisado. Para obter mais informações, consulte Indexação de dados no Amazon OpenSearch Service.
    - Escolha como você deseja retornar a resposta ao cliente.
      - Resposta exata: quando essa opção está habilitada, o valor no campo Resposta é usado como está para a resposta do bot. O modelo de base configurado do Amazon Bedrock é usado para selecionar o conteúdo exato da resposta no estado em que se encontra, sem qualquer síntese ou resumo do conteúdo. Especifique o nome dos campos de pergunta e resposta que foram configurados no OpenSearch banco de dados.

    - Incluir campos: retorna uma resposta gerada pelo modelo usando os campos que você especificar. Especifique o nome de até cinco campos que foram configurados no OpenSearch banco de dados. Use um ponto e vírgula (;) para separar os campos.
- Amazon Kendra: configure os campos a seguir.
    - Índice do Amazon Kendra: selecione o índice do Amazon Kendra que você deseja que o bot pesquise.
    - Filtro do Amazon Kendra: para criar um filtro, marque essa caixa de seleção. Para obter mais informações sobre o filtro de pesquisa do Amazon Kendra, consulte Usar atributos de documento para filtrar os resultados da pesquisa.
    - Resposta exata: para permitir que seu bot retorne a resposta exata retornada pelo Amazon Kendra, marque essa caixa de seleção. Caso contrário, o modelo do Amazon Bedrock selecionado gerará uma resposta com base nos resultados.

      > **Note**
      >
      > Para usar esse recurso, você deve primeiro adicionar perguntas de perguntas frequentes ao seu índice seguindo as etapas em Adicionar perguntas frequentes (FAQs) a um índice.
- Base de conhecimento do Amazon Bedrock: se escolher essa opção, especifique a ID da base de conhecimento. Você pode encontrar o ID verificando a página de detalhes da base de conhecimento no console ou enviando uma GetKnowledgeBasesolicitação.

As respostas da QNAintent serão armazenadas nos atributos da solicitação, conforme mostrado abaixo:

- `x-amz-lex:qnA-search-response`: a resposta da QnAIntent à pergunta ou enunciado.
- `x-amz-lex:qnA-search-response-source`: aponta para o documento ou lista de documentos usados para gerar a resposta.

AMAZON.RepeatIntent

Responde a palavras e frases que permitem ao usuário repetir a mensagem anterior. Seu aplicativo precisa usar uma função do Lambda para salvar as informações da intenção anterior nas variáveis da sessão, ou você precisa usar a GetSession operação para obter as informações da intenção anterior.

Enunciados comuns:

- repetir
- dizer isso de novo
- repetir isso

AMAZON.ResumeIntent

Responde a palavras e frases que permitem ao usuário retomar uma intenção anteriormente pausada. Sua função do Lambda ou aplicativo deve gerenciar as informações necessárias para retomar a intenção anterior.

Enunciados comuns:

- retomar
- continuar
- prosseguir

AMAZON.StartOverIntent

Responde a palavras e frases que permitem ao usuário parar de processar a intenção atual e recomeçar do início. Você pode usar sua função do Lambda ou a operação `PutSession` para obter novamente o valor do primeiro slot.

Enunciados comuns:

- começar de novo
- reiniciar
- começar novamente

AMAZON.StopIntent

Responde a palavras e frases que indicam que o usuário deseja parar de processar a intenção atual e encerrar a interação com um bot. Seu aplicativo ou função do Lambda deve limpar todos os atributos e valores de tipo de slot existentes e, em seguida, encerrar a interação.

Enunciados comuns:

- parar
- desligar
- cale-se

Adicionar tipos de slot

Os tipos de slot definem os valores que os usuários podem fornecer para as variáveis de intenção. Defina os tipos de slots para cada idioma para que os valores sejam específicos desse idioma. Por exemplo, para um tipo de slot que lista cores de tinta, você pode incluir o valor "red" em inglês, "rouge" em francês e "rojo" em espanhol.

Este tópico descreve como criar tipos de slots personalizados que fornecem valores para os slots da intenção. Você também pode usar tipos de slots integrados para valores padrão. Por exemplo, é possível usar o tipo de slot integrado `AMAZON.Country` para obter uma lista de países do mundo.

Como criar um tipo de slot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot ao qual você deseja adicionar o idioma, escolha Estrutura de conversação e, em seguida, Todos os idiomas.
3. Escolha o idioma ao qual adicionar o tipo de slot e selecione Tipos de slot.
4. Escolha Adicionar tipo de slot, dê um nome ao tipo de slot e escolha Adicionar.
5. No editor de tipo de slot, adicione os detalhes do tipo de slot.

    - Resolução do valor do slot – determina como os valores do slot são resolvidos. Se você escolher Expandir valores, o Amazon Lex V2 usará os valores como valores representativos para treinamento. Se você usar Restringir aos valores do slot, os valores permitidos para o slot serão restritos aos que você fornecer.
    - Valores do tipo de slot – Os valores do slot. Se você escolher Restringir aos valores do slot, poderá adicionar sinônimos para o valor. Por exemplo, para o valor “futebol americano”, você pode adicionar o sinônimo “futebol”. Se o usuário digitar “futebol” em uma conversa com o bot, o valor real do slot será “futebol americano”.
    - Use valores de slot como vocabulário personalizado – ative essa opção para ajudar a melhorar o reconhecimento de valores de slots e sinônimos em conversas de áudio. Não ative

essa opção quando os valores do slot forem termos comuns, como “sim”, “não”, “um”, “dois”, “três” etc.

6. Selecione Salvar tipo de slot.

O Amazon Lex V2 oferece os seguintes tipos de slots:

Tópicos

Tipos de slot integrados

O Amazon Lex oferece suporte a tipos de slots integrados que definem como os dados no slot são reconhecidos e tratados. É possível criar slots desses tipos em suas intenções. Isso elimina a necessidade de criar valores de enumeração para dados de slot comumente usados, como data, hora e local. Os tipos de slot integrados não têm versões.

Tipo de slot	Descrição breve	Localidades compatíveis	
AMAZÔNIA. AlphaNumeric	Reconhece palavras compostas de letras e números.	Todas as localidades, exceto coreano (ko-KR)	
AMAZON.City	Reconhece palavras que representam uma cidade.	Todas as localidades	
AMAZON.Confirmation	Reconhece palavras que significam “Sim”, “Não”, “Talvez” e “Não sei” e as converte em um formato padrão	Inglês (inglês-EUA e inglês-REINO UNIDO, inglês-AUSTRÁLIA, inglês-ÍNDIA, inglês-ÁFRICA DO SUL)	

Tipo de slot	Descrição breve	Localidades compatíveis	
	(Sim/Não/Talvez/Não sei).		
AMAZON.Country	Reconhece palavras que representam um país.	Todas as localidades	
AMAZON.Date	Reconhece palavras que representam uma data e as converte em um formato padrão.	Todas as localidades	
AMAZON.Duration	Reconhece palavras que representam duração e as converte em um formato padrão.	Todas as localidades	
AMAZÔNIA. EmailAddress	Converte palavras que representam um endereço de e-mail em um endereço de e-mail padrão.	Todas as localidades	
AMAZÔNIA. FirstName	Reconhece palavras que representam um nome.	Todas as localidades	
AMAZÔNIA. LastName	Reconhece palavras que representam um sobrenome.	Todas as localidades	
AMAZON.Number	Reconhece palavras numéricas e as converte em dígitos.	Todas as localidades	

Tipo de slot	Descrição breve	Localidades compatíveis	
AMAZON.Percentage	Converte palavras que representam uma porcentagem em um número e um sinal de porcentagem (%).	Todas as localidades	
AMAZÔNIA. PhoneNumber	Converte palavras que representam um número de telefone em uma string numérica.	Todas as localidades	
AMAZON.State	Reconhece palavras que representam um estado.	Todas as localidades	
AMAZÔNIA. StreetName	Reconhece palavras que representam o nome de uma rua.	Todas as localidades	
AMAZON.Time	Converte palavras que indicam horários em um formato de hora.	Todas as localidades	
AMAZON.UK PostalCode	Reconhece palavras que representam um código postal do Reino Unido e as converte em um formato padrão.	Somente inglês (britânico) (en-GB)	

Tipo de slot	Descrição breve	Localidades compatíveis	
AMAZÔNIA. FreeFormInput	Reconhece cadeias de caracteres que consistem em qualquer palavra ou caractere.	Todas as localidades	

AMAZÔNIA. AlphaNumeric

Reconhece strings compostas de letras e números, como **APQ123**.

Esse tipo de slot não está disponível na localidade coreana (ko-KR).

Você pode usar o tipo de slot `AMAZON.AlphaNumeric` para strings que contenham:

- Caracteres alfabéticos, como **ABC**
- Caracteres numéricos, como **123**
- Uma combinação de caracteres alfanuméricos, como **ABC123**

O tipo de slot `AMAZON.AlphaNumeric` suporta entradas usando estilos de ortografia. Você pode usar os spell-by-word estilos spell-by-letter e para ajudar seus clientes a inserir cartas. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

Você pode adicionar uma expressão regular ao tipo de slot `AMAZON.AlphaNumeric` para validar os valores inseridos para o slot. Por exemplo, é possível usar uma expressão regular para validar:

- Códigos postais canadenses
- Números de carteira de motorista
- Números de identificação de veículo

Use uma expressão regular padrão. O Amazon Lex V2 suporta os seguintes caracteres na expressão regular:

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

O Amazon Lex V2 também oferece suporte a caracteres Unicode em expressões regulares. O formato é `\u`*`Unicode`*. Use quatro dígitos para representar caracteres Unicode. Por exemplo, `[\u0041-\u005A]` é equivale a [A-Z].

Os seguintes operadores de expressão regular não são aceitos:

- Repetidores infinitos: *, + ou {x,} sem limite superior.
- Curinga (.)

O comprimento máximo da expressão regular é de 300 caracteres. O comprimento máximo de uma string armazenada em um tipo de slot `AMAZON.AlphaNumeric` que usa uma expressão regular é de 30 caracteres.

A seguir estão alguns exemplos de expressões regulares.

- Strings alfanuméricas, como **APQ123** ou **APQ1**: `[A-Z]{3}[0-9]{1,3}` ou um `[A-DP-T]{3}[1-5]{1,3}` mais restrito
- Formato internacional de correio prioritário do Serviço Postal dos EUA, como **CP123456789US**: `CP[0-9]{9}US`
- Números de roteamento bancário, como **123456789**: `[0-9]{9}`

Para definir a expressão regular para um tipo de slot, use o console ou a operação CreateSlotType. A expressão regular é validada quando você salva o tipo de slot. Se a expressão não for válida, o Amazon Lex V2 retornará uma mensagem de erro.

Quando você usa uma expressão regular em um tipo de slot, o Amazon Lex V2 verifica a entrada em slots desse tipo em relação à expressão regular. Se a entrada corresponder à expressão, o valor será aceito para o slot. Se a entrada não corresponder, o Amazon Lex V2 solicitará que o usuário repita a entrada.

AMAZON.City

Fornece uma lista de cidades locais e mundiais. O tipo de slot reconhece variações comuns de nomes de cidades. O Amazon Lex V2 não faz a conversão de uma variação para um nome oficial.

Exemplos:

- Nova York
- Reykjavik

- Tóquio
- Versalhes

AMAZON.Confirmation

Esse tipo de slot reconhece frases e palavras de entrada que correspondem a frases e palavras “Sim”, “Não”, “Talvez” e “Não sei” para o Amazon Lex V2 e as converte em um dos quatro valores. Ele pode ser usado para capturar a confirmação ou o reconhecimento do usuário. Com base no valor final resolvido, você pode criar condições para criar vários caminhos de conversação.

Por exemplo: .

se {confirmation} = “Sim”, atende a intenção

caso contrário, extraia outro slot

Exemplos:

- Sim: é, isso, ok, claro, eu tenho, posso concordar...
- Não: não, negativo, esquece, vou recusar, de jeito nenhum...
- Talvez: é possível, talvez, às vezes, pode ser, isso pode estar certo...
- Não sei: não sei, desconhecido, não faço ideia, não tenho certeza, quem sabe...

A partir de 17 de agosto de 2023, se houver um tipo de slot personalizado chamado “Confirmação”, o nome deverá ser alterado para evitar conflitos com a Confirmação de slot integrada. Na navegação do lado esquerdo do console Lex, acesse o tipo de slot (para um tipo de slot personalizado existente chamado Confirmação) e atualize o nome do tipo de slot. O nome do novo tipo de slot não deve ser “Confirmação”, que é uma palavra-chave reservada para o tipo de slot de confirmação incorporado.

AMAZON.Country

Os nomes dos países de todo o mundo. Exemplos:

- Austrália
- Alemanha
- Japão
- Estados Unidos
- Uruguai

AMAZON.Date

Converte palavras que representam datas em um formato de data.

A data é fornecida de acordo com sua intenção no formato de data ISO-8601. A data em que sua intenção é recebida no slot pode variar dependendo da frase específica proferida pelo usuário.

- Enunciados mapeados para uma data específica, como "hoje", "agora" ou "vinte e cinco de novembro", convertem em uma data completa: `2020-11-25` O padrão é datas iguais ou posteriores à data atual.
- Enunciados mapeados para uma semana futura, como "semana que vem", são convertidos na data do primeiro dia da semana. No formato ISO-8601, a semana começa na segunda-feira e termina no domingo. Por exemplo, se hoje for 25/11/2020, a "próxima semana" será convertida em `2020-11-29`. As datas mapeadas para a semana atual ou anterior são convertidas para o primeiro dia da semana. Por exemplo, se hoje for 25/11/2020, a "última semana" será convertida em `2020-11-16`.
- Expressões mapeadas para um mês futuro, mas não para um dia específico, como "próximo mês", são convertidas para o último dia do mês. Por exemplo, se hoje for 25/11/2020, "próximo mês" será convertido em `2020-12-31`. Para datas mapeadas para o mês atual ou anterior são convertidas para o primeiro dia do mês. Por exemplo, se hoje for 25/11/2020, "esse mês" mapeia para `2020-11-01`.
- Expressões mapeadas para um ano futuro, mas não para um mês ou dia específico, como "ano que vem", são convertidas no último dia do ano seguinte. Por exemplo, se hoje for 25/11/2020, o "próximo ano" será convertido em `2021-12-31`. Para datas mapeadas para o ano atual ou anterior são convertidas para o primeiro dia do ano. Por exemplo, se hoje for 25/11/2020, o "último ano" será convertido em `2019-01-01`.

AMAZON.Duration

Converte palavras que indicam durações em uma duração numérica.

A duração é resolvida em um formato baseado no formato de duração ISO-8601, `PnYnMnWnDTnHnMnS`. O `P` indica que essa é uma duração, `n` é um valor numérico e a letra maiúscula após `n` é o elemento específico de data ou hora. Por exemplo, `P3D` significa 3 dias. O `T` é usado para indicar que os valores restantes representam elementos de tempo em vez de elementos de data.

Exemplos:

- "dez minutos": `PT10M`
- "cinco horas": `PT5H`
- "três dias": `P3D`
- "quarenta e cinco segundos": `PT45S`
- "oito semanas": `P8W`
- "sete anos": `P7Y`
- "cinco horas e dez minutos": `PT5H10M`
- "dois anos três horas dez minutos": `P2YT3H10M`

AMAZÔNIA. EmailAddress

Reconhece palavras que representam um endereço de e-mail fornecido, como nomedeusuário@domínio. Os endereços podem incluir os seguintes caracteres especiais em um nome de usuário: sublinhado (_), hífen (-), ponto (.) e o sinal de mais (+).

O tipo de slot `AMAZON.EmailAddress` suporta entradas usando estilos de ortografia. Você pode usar os spell-by-word estilos spell-by-letter e para ajudar seus clientes a inserir endereços de e-mail. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

AMAZÔNIA. FirstName

Nomes próprios comumente usados. Esse tipo de slot reconhece nomes formais, apelidos informais e nomes que consistem em mais de uma palavra. O nome enviado para sua intenção é o valor enviado pelo usuário. O Amazon Lex V2 não converte do apelido para o nome formal.

Para nomes que soam parecidos, mas com grafia diferente, o Amazon Lex V2 envia sua intenção em um único formulário comum.

O tipo de slot `AMAZON.FirstName` suporta entradas usando estilos de ortografia. Você pode usar os spell-by-word estilos spell-by-letter e para ajudar seus clientes a inserir nomes. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

Exemplos:

- Emily
- John

- Sophie
- Anil Kumar

AMAZÔNIA. FirstName também retorna uma lista de nomes estreitamente relacionados com base no valor original. Você pode usar a lista de valores resolvidos para se recuperar de erros de digitação, confirmar o nome com o usuário ou realizar uma pesquisa no banco de dados para encontrar nomes válidos em seu diretório de usuários.

Por exemplo, a entrada “John” pode resultar no retorno de nomes relacionados adicionais, como “John J” e “John-Paul”.

Veja a seguir o formato da resposta para o tipo de slot integrado `AMAZON.FirstName`:

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

AMAZÔNIA. LastName

Sobrenomes comumente usados. Para nomes com sons parecidos e escritos de forma diferente, o Amazon Lex V2 envia sua intenção em um único formulário comum.

O tipo de slot `AMAZON.LastName` suporta entradas usando estilos de ortografia. Você pode usar os spell-by-word estilos spell-by-letter e para ajudar seus clientes a inserir nomes. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

Exemplos:

- Brosky
- Dasher
- Evers
- Parres

- Welt

AMAZÔNIA. LastName também retorna uma lista de nomes estreitamente relacionados com base no valor original. Você pode usar a lista de valores resolvidos para se recuperar de erros de digitação, confirmar o nome com o usuário ou realizar uma pesquisa no banco de dados para encontrar nomes válidos em seu diretório de usuários.

Por exemplo, a entrada “Smith” pode resultar no retorno de nomes relacionados adicionais, como “Smyth” e “Smithe”.

Veja a seguir o formato da resposta para o tipo de slot integrado `AMAZON.LastName`:

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

AMAZON.Number

Converte palavras ou números que expressam um número em dígitos, incluindo números decimais. A tabela a seguir mostra como o tipo de slot `AMAZON.Number` captura palavras numéricas.

Entrada	Resposta
cento e vinte e três ponto quatro cinco	123.45
cento e vinte e três ponto quatro cinco	123.45
ponto quatro dois	0.42
ponto quarenta e dois	0.42
232.998	232.998

Entrada	Resposta
50	50
-15	-15
menos 15	-15

AMAZON.Percentage

Converte palavras e símbolos que representam uma porcentagem em um valor numérico com um sinal de porcentagem (%).

Se o usuário inserir um número sem um sinal de porcentagem ou as palavras "por cento", o valor do slot será definido para o número. A tabela a seguir mostra como o tipo de slot `AMAZON.Percentage` captura porcentagens.

Entrada	Resposta
50%	50%
0,4 por cento	0.4%
23,5%	23,5%
vinte e cinco por cento	25%

AMAZÔNIA. PhoneNumber

Converte os números ou as palavras que representam um número de telefone em um formato de string sem a pontuação da seguinte forma.

Tipo	Descrição	Entrada	Resultado
Número internacional com sinal de mais (+) no começo	Número de 11 dígitos com sinal de mais no começo.	+61 7 4445 1061	`+61744431061`
		1 (509) 555-1212	`+15095551212`

Tipo	Descrição	Entrada	Resultado
Número internacional sem sinal de mais (+) no começo	Número de 11 dígitos sem sinal de mais no começo	1 (509) 555-1212 61 7 4445 1061	15095551212 61744451061
Número nacional	Número de 10 dígitos sem código internacional	(03) 5115 4444 (509) 555-1212	0351154444 5095551212
Número local	número de telefone sem código internacional ou código de área	555-1212	5551212

AMAZON.State

Os nomes das regiões geográficas e políticas dos países.

Exemplos:

- Baviera
- Prefeitura de Fukushima
- Noroeste do Pacífico
- Queensland
- Gales

AMAZÔNIA. StreetName

Os nomes das ruas dentro de um endereço típico. Isso inclui apenas o nome da rua, não o número da casa.

Exemplos:

- Avenida Canberra
- Rua da frente

- Estrada do Mercado

AMAZON.Time

Converte palavras que representam tempos em valores temporais. `AMAZON.Time` pode resolver horários exatos, valores ambíguos e intervalos de tempo. O valor do slot pode ser resolvido nos seguintes intervalos de tempo:

- AM
- PM
- MO (manhã)
- AF (tarde)
- EV (noite)
- NI (noite)

Quando o usuário insere um horário ambíguo, o Amazon Lex V2 usa o atributo `slots` de um evento para passar resoluções para horários ambíguos à função do Lambda. Por exemplo, se seu bot solicitar um horário de entrega, o usuário poderá responder: "10 horas". Esse horário é ambíguo. Pode ser 10h ou 22h. Nesse caso, o valor no campo `interpretedValue` é `null`, e o campo `resolvedValues` contém as duas possíveis resoluções de horário. O Amazon Lex V2 insere o seguinte na função do Lambda:

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

Quando o usuário responde com um horário não ambíguo, o Amazon Lex V2 envia o horário para a função do Lambda no campo `interpretedValue` do atributo `slots` do evento do Lambda. Por exemplo, se o usuário responder à solicitação para um horário de entrega com "10:00 AM", o Amazon Lex V2 inserirá o seguinte na função do Lambda:

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

Se o usuário responder à solicitação para um horário de entrega com "de manhã", o Amazon Lex V2 inserirá o seguinte na função do Lambda:

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

Para obter mais informações sobre os dados enviados do Amazon Lex V2 para uma função do Lambda, consulte Interpretar o formato do evento de entrada.

AMAZON.UK PostalCode

Converte palavras que representam um código postal do Reino Unido em um formato padrão para códigos postais no Reino Unido. O tipo de slot `AMAZON.UKPostalCode` valida e resolve o código postal em um conjunto de formatos padronizados, mas não verifica se o código postal é válido. Seu aplicativo deve validar o código postal.

Esse tipo de slot `AMAZON.UKPostalCode` está disponível somente na localidade com o idioma em Inglês (REINO UNIDO) (inglês-REINO UNIDO).

O tipo de slot `AMAZON.UKPostalCode` suporta entradas usando estilos de ortografia. Você pode usar os spell-by-word estilos spell-by-letter e para ajudar seus clientes a inserir cartas. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

O tipo de slot reconhece somente os formatos de código postal válidos listados abaixo, usados no Reino Unido. Os formatos válidos são (“A” representa uma letra e “9" representa um dígito):

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

Para entrada de texto, o usuário pode inserir qualquer combinação de letras maiúsculas e minúsculas. O usuário pode usar ou omitir o espaço no código postal. O valor resolvido sempre incluirá o espaço no local adequado para o código postal.

Para entrada falada, o usuário pode falar os caracteres individuais ou usar pronúncias de letras duplas, como “duplo A” ou “duplo 9". Também é possível usar pronúncias de dois dígitos, como “noventa e nove” para “99".

> **Note**
>
> Nem todos os códigos postais do Reino Unido são reconhecidos. Somente os formatos listados acima são suportados.

AMAZÔNIA. FreeFormInput

`AMAZON.FreeFormInput` pode ser usado para capturar entradas de formato livre do usuário final. Reconhece cadeias de caracteres que consistem palavras ou caracteres. O valor resolvido é todo o enunciado de entrada.

Exemplo:

Bot: forneça feedback sobre sua experiência de chamada.

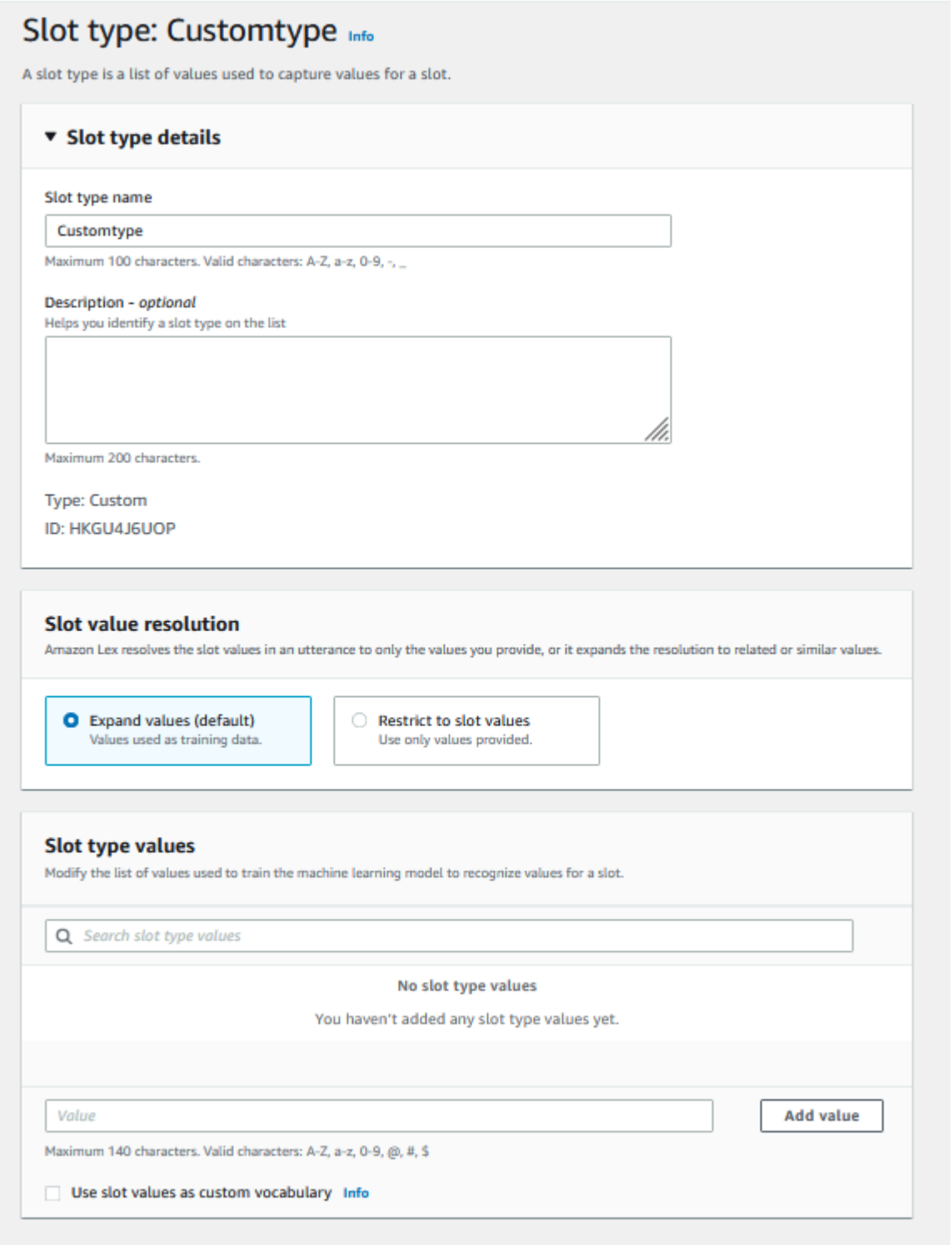
Usuário: Recebi as respostas para todas as minhas perguntas e consegui concluir a transação.

Nota:

- `AMAZON.FreeFormInput` pode ser usado para capturar entradas como-é de formato livre do usuário final.
- `AMAZON.FreeFormInput` não pode ser usado em amostras intencionais de enunciados.
- `AMAZON.FreeFormInput` não pode ter amostras de enunciados em um slot.
- `AMAZON.FreeFormInput` só é reconhecido quando elicitado.
- `AMAZON.FreeFormInput` não suporta esperar e continuar.
- `AMAZON.FreeFormInput` atualmente não é compatível com o canal Amazon Connect Chat.
- Quando um slot `AMAZON.FreeFormInput` é acionado, `FallbackIntent` não será acionado.
- Quando um slot `AMAZON.FreeFormInput` é acionado, não haverá mudança de intenção.

Tipo de slot personalizado

Para cada intenção, você pode especificar parâmetros que indicam as informações necessárias para atender a solicitação do usuário. Esses parâmetros ou slots têm um tipo. Um tipo de slot é uma lista de valores que o Amazon Lex V2 usa para treinar o modelo de machine learning para reconhecer os valores de um slot. Por exemplo, você pode definir um tipo de slot chamado `Genres` com valores como “comédia”, “aventura”, “documentário” etc. Você pode definir sinônimos para um valor de tipo de slot. Por exemplo, você pode definir os sinônimos "engraçado" e "humor" para o valor "comédia".



É possível configurar o tipo de slot para expandir os valores do slot. Os valores do slot serão usados como dados de treinamento e o modelo definirá o slot com o valor fornecido pelo usuário se ele for semelhante aos valores e sinônimos do slot. Esse é o comportamento padrão. O Amazon Lex V2 mantém uma lista de possíveis resoluções para um slot. Cada entrada na lista fornece um valor

resolvido que o Amazon Lex V2 reconhece como possibilidades adicionais para o slot. Um valor resolvido é o melhor esforço para corresponder ao valor do slot. A lista contém até cinco valores.

Como alternativa, você pode configurar o tipo de slot para restringir a resolução aos valores do slot. Nesse caso, o modelo definirá um valor de slot inserido pelo usuário como um valor de slot existente somente se for igual ao valor desse slot ou se for um sinônimo. Por exemplo, se o usuário inserir "engraçado", isso será definido como o valor do slot "comédia".

Quando o valor inserido pelo usuário é um sinônimo de um valor do tipo de slot, o modelo retorna esse valor do tipo de slot como a primeira entrada na lista de `resolvedValues`. Por exemplo, se o usuário digitar “engraçado”, o modelo preencherá o campo `originalValue` com o valor “engraçado” e a primeira entrada no campo resolvedValues com “comédia”. Você pode configurar o `valueSelectionStrategy` quando cria ou atualiza um tipo de slot com a operação CreateSlotType para que o valor do slot seja preenchido com o primeiro valor na lista de resolução.

Os tipos de slots personalizados oferecem suporte a entradas que usam estilos de soletração. Você pode usar os spell-by-word estilos spell-by-letter e para ajudar seus clientes a inserir cartas. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

Se você estiver usando uma função do Lambda, o evento de entrada para a função incluirá uma lista de resoluções chamada `resolvedValues`. O exemplo a seguir mostra a seção de slots da entrada para uma função do Lambda:

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

Para cada tipo de slot, você pode definir um máximo de 10.000 valores e sinônimos. Cada bot pode ter um número total de 50.000 valores e sinônimos de tipos de slots. Por exemplo, você pode ter cinco tipos de slot, cada um com 5.000 valores e 5.000 sinônimos ou pode ter 10 tipos de slot, cada um com 2.500 valores e 2.500 sinônimos.

Um tipo de slot personalizado não deve ter o mesmo nome dos tipos de slot integrados. Por exemplo, um tipo de slot personalizado não deve ser nomeado com as palavras-chave reservadas de Data, Número ou Confirmação. Essas palavras-chave são reservadas para tipos de slots integrados. Para obter uma lista de tipos de slots integrados, consulte Tipos de slot integrados.

Tipo de slot de gramática

Com o tipo de slot de gramática, você pode criar sua própria gramática no formato XML de acordo com a especificação SRGS para coletar informações em uma conversa. O Amazon Lex V2 reconhece expressões que correspondem às regras especificadas na gramática. Você também pode fornecer regras de interpretação semântica usando tags ECMAScript nos arquivos de gramática. O Amazon Lex retorna propriedades definidas nas tags como valores resolvidos quando ocorre uma correspondência.

Só é possível criar tipos de slots de gramática nos idiomas inglês (Austrália), inglês (Reino Unido) e inglês (EUA).

Há duas partes em um tipo de slot de gramática. A primeira é a própria gramática escrita usando o formato de especificação SRGS. A gramática interpreta o enunciado do usuário. Se o enunciado for aceito pela gramática, ele será correspondido, caso contrário, será rejeitado. Se um enunciado for correspondido, ele será encaminhado para o script, se houver.

A segunda faz parte de um tipo de slot de gramática e é um script opcional escrito em ECMAScript que transforma a entrada nos valores resolvidos retornados pelo tipo de slot. Por exemplo, você pode usar um script para converter números falados em dígitos. As instruções ECMAScript estão incluídas no elemento **<tag>**.

O exemplo a seguir está no formato XML de acordo com a especificação SRGS que mostra uma gramática válida aceita pelo Amazon Lex V2. Ele define um tipo de slot de gramática que aceita números de cartão e determina se eles são para contas regulares ou premium. Para obter mais informações sobre a sintaxe aceitável, consulte os tópicos Definição de gramática e Formato do script.

```
<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>
```

A gramática acima aceita apenas dois tipos de números de cartão: 7241 ou 7001. Opcionalmente, ambos podem ser prefixados com “número do cartão”. Também contém tags ECMAScript que podem ser usadas para interpretação semântica. Com a interpretação semântica, o enunciado “cartão número sete, dois, quatro, um” retornaria o seguinte objeto:

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

Esse objeto é retornado como uma string serializada em JSON no objeto `resolvedValues` retornado pelas operações RecognizeText, RecognizeUtterance e StartConversation

Adicionar um tipo de slot de gramática

Como adicionar um tipo de slot de gramática

1. Faça upload da definição XML do tipo de slot para um bucket do S3. Anote o nome do bucket e o caminho para o arquivo.

> **Note**
>
> O tamanho máximo do arquivo é 100 kB.

2. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
3. No menu à esquerda, selecione Bots e escolha o bot ao qual adicionar o tipo de slot de gramática.
4. Selecione Exibir idiomas e escolha o idioma ao qual adicionar o tipo de slot de gramática.
5. Escolha Exibir tipos de slots.
6. Escolha Adicionar tipo de slot e selecione Adicionar tipo de slot de gramática.
7. Dê um nome ao tipo de slot e selecione Adicionar.
8. Escolha o bucket do S3 que contém o arquivo de definição e insira o caminho para o arquivo. Selecione Salvar tipo de slot.

Definição de gramática

Este tópico mostra as partes da especificação SRGS com as quais o Amazon Lex V2 é compatível. Todas as regras são definidas na especificação SRGS. Para mais informações, consulte a recomendação do W3C da Speech recognition grammar specification version 1.0.

Tópicos

- Declarações de cabeçalho
- Elementos XML compatíveis
- Tokens
- Referência de regras
- Sequências e encapsulamento
- Repetições
- Linguagem
- Tags
- Pesos

Este documento inclui material copiado e derivado da Speech Recognition Grammar Specification Version 1.0 do W3C (disponível em https://www.w3.org/TR/speech-grammar/). Veja abaixo as informações de citação:

Copyright © 2004 W3C® (MIT, ERCIM, Keio, Todos os direitos reservados. Aplicam-se as regras de responsabilidade, marca registrada, uso de documentos e licenciamento de software do W3C.

O documento de especificação do SRGS, uma Recomendação do W3C, está disponível no W3C sob a licença a seguir.

Texto da licença

Licença

Ao usar e/ou copiar este documento ou o documento do W3C ao qual esta declaração está vinculada, você (o licenciado) concorda que leu, entendeu e cumprirá os seguintes termos e condições:

A permissão para copiar e distribuir o conteúdo deste documento ou do documento W3C ao qual esta declaração está vinculada, em qualquer meio, para qualquer finalidade e sem taxa ou royalties, é concedida, desde que você inclua o seguinte em TODAS as cópias do documento, ou partes dele, que você usar:

- Um link ou URL para o documento original do W3C.
- O aviso de direitos autorais preexistentes do autor original ou, se não existir, um aviso (preferencialmente em hipertexto, mas uma representação textual é permitida) no formato: “Copyright © [$date-of-document] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang). http://www.w3.org/Consortium/Legal/2015/doc-license”
- Se existir, o STATUS do documento do W3C.

Quando o espaço permitir, a inclusão do texto completo deste AVISO deve ser fornecida. Solicitamos que a atribuição de autoria seja fornecida em qualquer software, documento ou outros itens ou produtos que você criar de acordo com a implementação do conteúdo deste documento ou de qualquer parte dele.

Nenhum direito de criar modificações ou derivados de documentos do W3C é concedido de acordo com esta licença, exceto da seguinte forma: Para facilitar a implementação das especificações técnicas estabelecidas neste documento, qualquer pessoa pode preparar e distribuir trabalhos

derivados e partes deste documento em software, em materiais de apoio que acompanham o software e na documentação do software, DESDE que todos esses trabalhos incluam o aviso abaixo. NO ENTANTO, a publicação de trabalhos derivados deste documento para uso como especificação técnica é expressamente proibida.

Além disso, os “Componentes de código” — IDL da Web em seções claramente marcadas como IDL da Web; e a marcação definida pelo W3C (HTML, CSS etc.) e o código da linguagem de programação de computador claramente marcados como exemplos de código — estão licenciados sob a Licença de Software do W3C.

O aviso é:

"Copyright © 2015 W3C® (MIT, ERCIM, Keio, Beihang). Este software ou documento inclui material copiado ou derivado de [título e URI do documento do W3C].”

Isenção de responsabilidade

ESTE DOCUMENTO É FORNECIDO “COMO ESTÁ” E OS DETENTORES DE DIREITOS AUTORAIS NÃO FAZEM REPRESENTAÇÕES OU GARANTIAS, EXPRESSAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO SE LIMITANDO A, GARANTIAS DE COMERCIALIZAÇÃO, ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA, NÃO VIOLAÇÃO OU TÍTULO; QUE O CONTEÚDO DO DOCUMENTO É ADEQUADO PARA QUALQUER FINALIDADE; NEM QUE A IMPLEMENTAÇÃO DE TAIS CONTEÚDOS NÃO INFRINGIRÁ QUAISQUER PATENTES, DIREITOS AUTORAIS, MARCAS REGISTRADAS OU OUTROS DIREITOS DE TERCEIROS.

OS DETENTORES DE DIREITOS AUTORAIS NÃO SERÃO RESPONSÁVEIS POR QUAISQUER DANOS DIRETOS, INDIRETOS, ESPECIAIS OU CONSEQUENTES DECORRENTES DE QUALQUER USO DO DOCUMENTO OU DA EXECUÇÃO OU IMPLEMENTAÇÃO DE SEU CONTEÚDO.

O nome e as marcas comerciais dos detentores de direitos autorais NÃO podem ser usados em publicidade ou publicidade relacionada a este documento ou seu conteúdo sem permissão prévia específica por escrito. A titularidade dos direitos autorais deste documento permanecerá sempre com os detentores dos direitos autorais.

Declarações de cabeçalho

A tabela a seguir mostra as declarações de cabeçalho compatíveis com o tipo de slot de gramática. Para obter mais informações, consulte Declarações de cabeçalho de gramática na recomendação do W3C Speech Recognition Grammar Specification Version 1.

Declaração	Requisitos de especificação	Formulário XML	Suporte ao Amazon Lex	Especificação
Versão gramatical	Obrigatório	4.3: `version` atributo no elemento `grammar`	Obrigatório	SRGS
Namespace XML	Obrigatório (somente XML)	4.3: atributo `xmlns` no elemento `grammar`	Obrigatório	SRGS
Tipo de documento	Obrigatório (somente XML)	4.3: DOCTYPE XML	Recomendado	SRGS
Codificação de caracteres	Recomendado	4.4: atributo `encoding` na declaração XML	Recomendado	SRGS
Linguagem	Necessário no modo de voz Ignorado no modo DTMF	4.5: atributo `xml:lang` no elemento `grammar`	Necessário no modo de voz Ignorado no modo DTMF	SRGS
Modo	Opcional	4.6: atributo `mode` no elemento `grammar`	Opcional	SRGS
Regra raiz	Opcional	4.7: atributo `root` no elemento `grammar`	Obrigatório	SRGS
Formato de tag	Opcional	4.8: atributo `tag-format`	String literal e ECMAScript são compatíveis	SRGS, SISR

Declaração	Requisitos de especificação	Formulário XML	Suporte ao Amazon Lex	Especificação
		no elemento `grammar`		
URI de base	Opcional	4.9: atributo `xml:base` no elemento `grammar`	Opcional	SRGS
Léxico de pronúncia	Opcional, vários permitidos	4.10: elemento `lexicon`	Sem suporte	SRGS, PLS
Metadados	Opcional, vários permitidos	4.11.1: elemento `meta`	Obrigatório	SRGS
Metadados XML	Opcional, somente XML	4.11.2: elemento `metadata`	Opcional	SRGS
Tag	Opcional, vários permitidos	4.12: elemento `tag`	Tags globais não compatíveis	SRGS

Exemplo

```
<?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">
```

Elementos XML compatíveis

O Amazon Lex V2 oferece suporte aos seguintes elementos XML para gramáticas personalizadas:

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

Tokens

A tabela a seguir mostra as especificações de token compatíveis com o tipo de slot de gramática. Para obter mais informações, consulte Tokens na recomendação do W3C Speech recognition grammar specification version 1.

Tipo de token	Exemplo	Compatível?
Token único sem aspas	olá	Sim
Token único sem aspas: não alfabético	2	Sim
Token único sem aspas, sem espaço em branco	"olá"	Sim, coloque aspas duplas quando ele contém apenas um único token
Dois tokens delimitados por espaço em branco	boa viagem	Sim
Quatro tokens delimitados por espaço em branco	isso é um teste	Sim
Token único com aspas, incluindo espaço em branco	"São Francisco	Não

Tipo de token	Exemplo	Compatível?
Token XML único na tag <token>	<token>São Francisco</token>	Não (o mesmo que um token único com aspas com espaço em branco)

Observações

- Token único com aspas, incluindo espaço em branco – A especificação exige que as palavras entre aspas duplas sejam tratadas como um único token. O Amazon Lex V2 os trata como tokens delimitados por espaços em branco.
- Token XML único em <token> – A especificação requer palavras delimitadas por <token> para representar um token. O Amazon Lex V2 os trata como tokens delimitados por espaços em branco.
- O Amazon Lex V2 gera um erro de validação quando qualquer uso é encontrado na gramática.

Exemplo

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

Referência de regras

A tabela a seguir resume as várias formas de referência de regras que são possíveis em documentos de gramática. Para obter mais informações, consulte Referência de regras na recomendação do W3C Speech recognition grammar specification version 1.

Tipo de referência	Formulário XML	Compatível
2.2.1 Referência de regra local explícita	`<ruleref uri="#rulename"/>`	Sim

Tipo de referência	Formulário XML	Compatível
2.2.2 Referência explícita a uma regra nomeada de uma gramática identificada por um URI	`<ruleref uri="grammarURI#rulename"/>`	Não
2.2.2 Referência implícita à regra raiz de uma gramática identificada por um URI	`<ruleref uri="grammarURI"/>`	Não
2.2.2 Referência explícita a uma regra nomeada de uma gramática identificada por um URI com um tipo de mídia	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	Não
2.2.2 Referência implícita à regra raiz de uma gramática identificada por um URI com um tipo de mídia	`<ruleref uri="grammarURI" type="media-type"/>`	Não

2.2.3 Definições de regras especiais	`<ruleref special="NULL"/>`
	`<ruleref special="VOID"/>`
	`<ruleref special="GARBAGE"/>`
		Não

Observações

1. O URI de gramática é um URI externo. Por exemplo, `http://grammar.example.com/world-cities.grxml`.
2. O tipo de mídia pode ser:
   - `application/srgs+xml`
   - `text/plain`

Exemplo

```
<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>
```

Sequências e encapsulamento

O exemplo a seguir mostra as sequências compatíveis. Para obter mais informações, consulte Sequências e encapsulamento na recomendação do W3C Speech recognition grammar specification version 1.

Exemplo

```
<!-- 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>
```

Repetições

A tabela a seguir mostra as expansões repetidas de regras compatíveis. Para obter mais informações, consulte Repetições na recomendação do W3C Speech recognition grammar specification version 1.

Formulário XML Exemplo	Comportamento	Compatível?
repeat="n" repeat="6"	A expressão contida é repetida exatamente “n” vezes. “n” deve ser “0" ou um inteiro positivo.	Sim
repeat="m-n" repeat="4-6"	A expansão contida é repetida entre “m” e “n” vezes (inclusive). “m” e “n” devem ser “0" ou um número inteiro positivo, e “m” deve ser menor ou igual a “n”.	Sim
repeat="m-" repeat="3-"	A expansão contida é repetida “m” vezes ou mais (inclusive). “m” deve ser “0" ou um número inteiro positivo. Por exemplo, “3-” declara que a expansão contida pode ocorrer três, quatro, cinco ou mais vezes.	Sim
repeat="0-1"	A expansão contida é opcional.	Sim
<item repeat="2-4" repeat-prob="0.8">		Não

Linguagem

A discussão a seguir se aplica aos identificadores de idioma aplicados às gramáticas. Para obter mais informações, consulte Idioma na recomendação do W3C Speech recognition grammar specification version 1.

Por padrão, uma gramática é um documento de um único idioma com um identificador de idioma fornecido na declaração do idioma no cabeçalho da gramática. Todos os tokens dessa gramática, a menos que seja declarado de outra forma, serão tratados de acordo com o idioma da gramática. Declarações de idioma em nível gramatical não são compatíveis.

No seguinte exemplo:

1. A declaração de cabeçalho da gramática para o idioma “inglês-EUA” é compatível com o Amazon Lex V2.
2. O anexo de idioma em nível de item (destacado em *vermelho*) não é compatível. O Amazon Lex V2 gera um erro de validação se um anexo de idioma for diferente da declaração do cabeçalho.

```
<?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>
```

Tags

A discussão a seguir se aplica às tags definidas para gramáticas. Para obter mais informações, consulte Tags na recomendação do W3C Speech recognition grammar specification version 1.

Com base na especificação SRGS, as tags podem ser definidas das seguintes maneiras:

1. Como parte de uma declaração de cabeçalho, conforme descrito em Declarações de cabeçalho.
2. Como parte de uma definição **<rule>**.

Os seguintes formatos de tag são compatíveis:

- `semantics/1.0` (SISR, ECMAScript)
- `semantics/1.0-literals` (literais de string SISR)

Os seguintes formatos de tag não são compatíveis:

- `swi-semantics/1.0` (Proprietário da Nuance)

Exemplo

```
<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>
```

Pesos

Você pode adicionar o atributo de peso a um elemento. O peso é um valor positivo de ponto flutuante que representa o grau em que a frase no item é priorizada durante o reconhecimento de fala. Para obter mais informações, consulte Pesos na recomendação do W3C, Speech recognition grammar specification version 1.

Os pesos devem ser maiores que 0 e menores ou iguais a 10 e podem ter somente uma casa decimal. Se o peso for maior que 0 e menor que 1, a frase será priorizada negativamente. Se o peso for maior que 1 e menor ou igual a 10, a frase será priorizada positivamente. Um peso de 1 é equivalente a não atribuir peso algum, não havendo priorização para a frase.

Atribuir pesos apropriados aos itens para melhorar o desempenho do reconhecimento de fala é uma tarefa difícil. Veja aqui algumas dicas que você pode seguir para atribuir pesos:

- Comece com uma gramática sem a atribuição de pesos aos itens.
- Determine quais padrões na fala são frequentemente identificados de maneira errada.
- Aplique valores diferentes para pesos até notar uma melhora no desempenho do reconhecimento de fala, sem apresentar regressões.

Exemplo 1

Por exemplo, se você tem uma gramática para aeroportos e observa que Nova York é frequentemente identificada de maneira errada como Newark, é possível priorizar positivamente Nova York atribuindo a ela um peso de 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>
```

Exemplo 2

Por exemplo, você tem uma gramática para o código de reserva da companhia aérea que começa com uma letra do alfabeto inglês seguida por três dígitos. Provavelmente, o código de reserva começa com B ou D, mas você observa que B é frequentemente identificado como P de maneira errada, e D como T. É possível priorizar 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 do script

O Amazon Lex V2 oferece suporte aos atributos do ECMAScript a seguir para definir gramáticas.

O Amazon Lex V2 oferece suporte aos seguintes atributos do ECMAScript ao especificar tags na gramática. `tag-format` deve ser enviado para `semantics/1.0` quando as tags ECMAScript forem usadas na gramática. Para obter mais informações, consulte a Especificação de linguagem 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">
```

Tópicos

- Declaração de variável
- Expressões
- Instrução If
- Instrução switch
- Declarações de função
- Instrução de iteração
- Instrução de bloco
- Comentários
- Instruções sem suporte

Este documento contém material do padrão ECMAScript (disponível em https://www.ecma-international.org/publications-and-standards/standards/ecma-262/). O documento de especificação da linguagem ECMAScript está disponível na Ecma International sob a licença a seguir.

Texto da licença

© 2020 Ecma International

Este documento pode ser copiado, publicado e distribuído a terceiros, e determinados trabalhos derivados dele podem ser preparados, copiados, publicados e distribuídos, no todo ou em parte, desde que o aviso de direitos autorais acima e esta Licença e Isenção de Responsabilidade de Direitos Autorais estejam incluídos em todas essas cópias e trabalhos derivados. Os únicos trabalhos derivados permitidos sob esta Licença de Direitos Autorais e Isenção de Responsabilidade são:

(i) trabalhos que incorporem todo ou parte deste documento com a finalidade de fornecer comentários ou explicações (como uma versão comentada do documento);

(ii) trabalhos que incorporem todo ou parte deste documento com o objetivo de incorporar atributos que forneçam acessibilidade;

(iii) traduções deste documento para outros idiomas além do inglês e em diferentes formatos; e

(iv) trabalhos que fazem uso desta especificação em produtos em conformidade com o padrão, implementando (por exemplo, copiando e colando total ou parcialmente) a funcionalidade nela contida.

No entanto, o conteúdo deste documento em si não pode ser modificado de forma alguma, inclusive removendo o aviso de direitos autorais ou as referências à Ecma International, exceto conforme necessário para traduzi-lo para outros idiomas além do inglês ou em um formato diferente.

A versão oficial de um documento da Ecma International é a versão em inglês no site da Ecma International. Em caso de discrepâncias entre uma versão traduzida e a versão oficial, a versão oficial prevalecerá.

As permissões limitadas concedidas acima são perpétuas e não serão revogadas pela Ecma International ou seus sucessores ou cessionários. Este documento e as informações aqui contidas são fornecidos “NO ESTADO EM QUE SE ENCONTRAM” E A ECMA INTERNATIONAL SE ISENTA DE TODAS AS GARANTIAS, EXPRESSAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO SE LIMITANDO A, QUALQUER GARANTIA DE QUE O USO DAS INFORMAÇÕES AQUI CONTIDAS

NÃO INFRINGIRÁ NENHUM DIREITO DE PROPRIEDADE OU QUALQUER GARANTIA IMPLÍCITA DE COMERCIALIZAÇÃO OU ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA.”

Declaração de variável

Uma declaração de variável define uma ou mais variáveis.

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

Expressões

Tipo de expressão	Sintaxe	Exemplo	Compatível?
Literal de expressão regular	Literal de string contendo caracteres especiais regex válidos	`"^\d\.$"`	Não
Função	`function functionName(parameters) { functionBody}`	`var x = function calc() { return 10; }`	Não
Excluir	`delete expression`	`delete obj.property;`	Não
Nulo	`void expression`	`void (2 == '2');`	Não
Typeof	`typeof expression`	`typeof 42;`	Não
Índice de membros	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	Sim

Tipo de expressão	Sintaxe	Exemplo	Compatível?
Ponto de membro	`expression . identifier`	`out.value`	sim
Argumentos	`expression (arguments)`	`new Date('1994-10-11')`	Sim
Pós-incremento	`expression++`	`var x=10; x++;`	Sim
Pós-decremento	`expression--`	`var x=10; x--;`	Sim
Pré-incremento	`++expression`	`var x=10; ++x;`	Sim
Pré-incremento	`--expression`	`var x=10; --x;`	Sim
Mais unário/Menos unário	`+expression / -expression`	`+x / -x;`	Sim
Negação de bits	`~ expression`	`const a = 5; consol e.log( ~a );`	Sim
Negação de lógica	`! expression`	`!(a > 0 || b > 0)`	Sim
Multiplicativo	`expression ('*' | '/' | '%') expression`	`(x + y) * (a / b)`	Sim
Aditivo	`expression ('+' | '-') expression`	`(a + b) - (a - (a + b))`	Sim

Tipo de expressão	Sintaxe	Exemplo	Compatível?
Mudança de bits	`expression ('<<' | '>>' | '>>>') expression`	`(a >> b) >>> c`	Sim
Relativo	`expression ('<' | '>' | '<=' | '>=') expression`	`if (a > b) { ... }`	Sim
Em	`expression in expression`	`fruits[0] in otherFruits;`	Sim
Igualdade	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	Sim
Bit e/xor/ou	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	Sim
Lógico e/ou	`expression ('&&' | '||') expression`	`if (a && (b ||c)) { ...}`	Sim
Ternários	`expression ? expression : expression`	`a > b ? obj.prop : 0`	Sim
Atribuição	`expression = expression`	`out.value = "string";`	Sim

Tipo de expressão	Sintaxe	Exemplo	Compatível?
Operador de atribuição	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	Sim
Operador de atribuição bitwise	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	Sim
Identificador	`identifierSequence` onde identifierSequence é uma sequência de caracteres válidos	`fruits=[10, 20, 30];`	Sim
Nulo literal	`null`	`x = null;`	Sim
Booleano literal	`true | false`	`x = true;`	Sim
String literal	`'string' / "string"`	`a = 'hello', b = "world";`	Sim
Literal decimal	`integer [.] digits [exponent]`	`111.11 e+12`	Sim
Literal hexadecimal	`0 (x | X)[0-9a-fA-F]`	`0x123ABC`	Sim
Literal octal	`0 [0-7]`	`"051"`	Sim

Tipo de expressão	Sintaxe	Exemplo	Compatível?
Matriz literal	`[ expressio n, ... ]`	`v = [a, b, c];`	Sim
Objeto literal	`{property: value, ...}`	`out = {value: 1, flag: false};`	Sim
Entre parênteses	`( expressions )`	`x + (x + y)`	Sim

Instrução If

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

Nota: no exemplo anterior, `expressions` e `statements` devem ser uma das instruções compatíveis neste documento.

Instrução switch

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

Nota: no exemplo anterior, `expressions` e `statements` devem ser uma das instruções compatíveis neste documento.

Declarações de função

```
function functionIdentifier([parameterList, ...]) {
```

```
     <function body>
}
```

Instrução de iteração

As instruções de iteração podem ser qualquer uma das seguintes:

```
// 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
}
```

Instrução de bloco

```
{
   statements
}

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

Nota: no exemplo anterior, a instrução `statements` fornecida no bloco deve ser uma das compatíveis neste documento.

Comentários

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

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

Instruções sem suporte

O Amazon Lex V2 não é compatível com os atributos do ECMAScript a seguir.

Tópicos

- Instrução vazia
- Instrução de continuação
- Instrução de interrupção
- Instrução de retorno
- Instrução de lançamento
- Instrução de tentativa
- Instrução do depurador
- Instruções rotuladas
- Instrução de classe

Instrução vazia

A instrução vazia é usada para não fornecer nenhuma instrução. Veja a seguir a sintaxe de uma instrução vazia:

```
;
```

Instrução de continuação

A instrução de continuação sem um rótulo é compatível com a Instrução de iteração. A instrução de continuação com um rótulo não é compatível.

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

Instrução de interrupção

A instrução de interrupção sem um rótulo é compatível com a Instrução de iteração. A instrução de interrupção com um rótulo não é compatível.

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

Instrução de retorno

```
return expression;
```

Instrução de lançamento

A instrução de lançamento é usada para lançar uma exceção definida pelo usuário.

```
throw expression;
```

Instrução de tentativa

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

Instrução do depurador

A instrução do depurador é usada para invocar a funcionalidade de depuração fornecida pelo ambiente.

```
debugger;
```

Instruções rotuladas

A instrução rotulada pode ser usada com instruções `break` ou `continue`.

```
label:
   statements


// Example
let str = '';

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

console.log(str);
```

Instrução de classe

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

Gramáticas do setor

As gramáticas do setor são um conjunto de arquivos XML para usar com o tipo de slot de gramática. Você pode usá-las para oferecer rapidamente uma experiência consistente ao usuário final ao migrar fluxos de trabalho de resposta de voz interativa para o Amazon Lex V2. Você pode selecionar

entre uma variedade de gramáticas pré-criadas em três domínios: serviços financeiros, seguros e telecomunicações. Há também um conjunto genérico de gramáticas que você pode usar como ponto de partida para suas próprias gramáticas.

As gramáticas contêm as regras para coletar as informações e as tags ECMAScript para interpretação semântica.

Gramáticas para serviços financeiros (download)

As seguintes gramáticas são compatíveis com serviços financeiros: números de conta e roteamento, números de cartão de crédito e empréstimo, pontuação de crédito, datas de abertura e fechamento da conta e número da previdência social.

Número da conta

```
<?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>
```

Número de roteamento

```
<?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>
```

Números de cartão de crédito

```
<?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 do empréstimo

```
<?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>
```

Pontuação de crédito

```
<?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 de abertura da conta

```
<?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 de pagamento automático

```
<?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 de validade do cartão de crédito

```
<?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 do extrato

```
<?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 da transação

```
<?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>
```

Valor da transferência

```
<?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>
```

Número da previdência social

```
<?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>
```

Gramáticas para seguros (download)

As seguintes gramáticas são compatíveis com o domínio de seguros: números de sinistros e apólices, números de carteira de motorista e placa, datas de vencimento, datas de início e datas de renovação, valores de sinistros e apólices.

ID do sinistro

```
<?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 da apólice

```
<?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>
```

Número da carteira de motorista

```
<?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>
```

Número da placa

```
<?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 de validade do cartão de crédito

```
<?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 de expiração da apólice, dia/mês/ano

```
<?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 de renovação da apólice, mês/ano

```
<?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 de início da apólice

```
<?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>
```

Valor do sinistro

```
<?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>
```

Valor do prêmio

```
<?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>
```

Quantidade da apólice

```
<?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>
```

Gramáticas para telecomunicações (download)

As seguintes gramáticas são compatíveis com telecomunicações: número de telefone, número de série, número SIM, CEP dos EUA, data de validade do cartão de crédito, início do plano, datas de renovação e expiração, data de início do serviço, quantidade de equipamentos e valor da fatura.

Número de telefone

```
<?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>
```

Número de série

```
<?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>
```

Número do 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>
```

CEP dos EUA

```
<?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 de validade do cartão de crédito

```
<?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 de expiração do plano, dia/mês/ano

```
<?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 de renovação do plano, mês/ano

```
<?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 de início do plano, mês/dia

```
<?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 de início do serviço, mês/dia

```
<?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>
```

Quantidade de equipamentos

```
<?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>
```

Valor da fatura

```
<?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>
```

Gramáticas genéricas (download)

Fornecemos as seguintes gramáticas genéricas: alfanumérica, moeda, data (dd/mm/aa), números, saudação, hesitação e atendente.

Alfanumérico

```
<?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>
```

Moeda

```
<?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, dd/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, dd/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>
```

Números, dígitos

```
<?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>
```

Números ordinais

```
<?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>
```

Atendente

```
<?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>
```

Saudação

```
<?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>
```

Hesitação

```
<?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>
```

Tipos de slot composto

Um slot composto é uma combinação de dois ou mais slots que capturam várias informações em uma única entrada do usuário. Por exemplo, você pode configurar o bot para obter a localização solicitando a “cidade e estado ou CEP”. Por outro lado, quando a conversa é configurada para usar tipos de slots separados, ela resulta em uma experiência conversacional rígida (“O que é a cidade?” seguido por “Qual é o CEP?”). Com um slot composto, você pode capturar todas as informações por meio de um único slot. Um slot composto é uma combinação de slots chamados subslots, como cidade, estado e CEP.

Você pode usar uma combinação dos tipos de slots Amazon Lex disponíveis (integrados) e seus próprios slots (slots personalizados). É possível criar expressões lógicas para capturar informações dentro dos subslots necessários. Por exemplo: cidade e estado ou CEP.

O tipo de slot composto só está disponível no idioma inglês-EUA.

Criar um tipo de slot composto

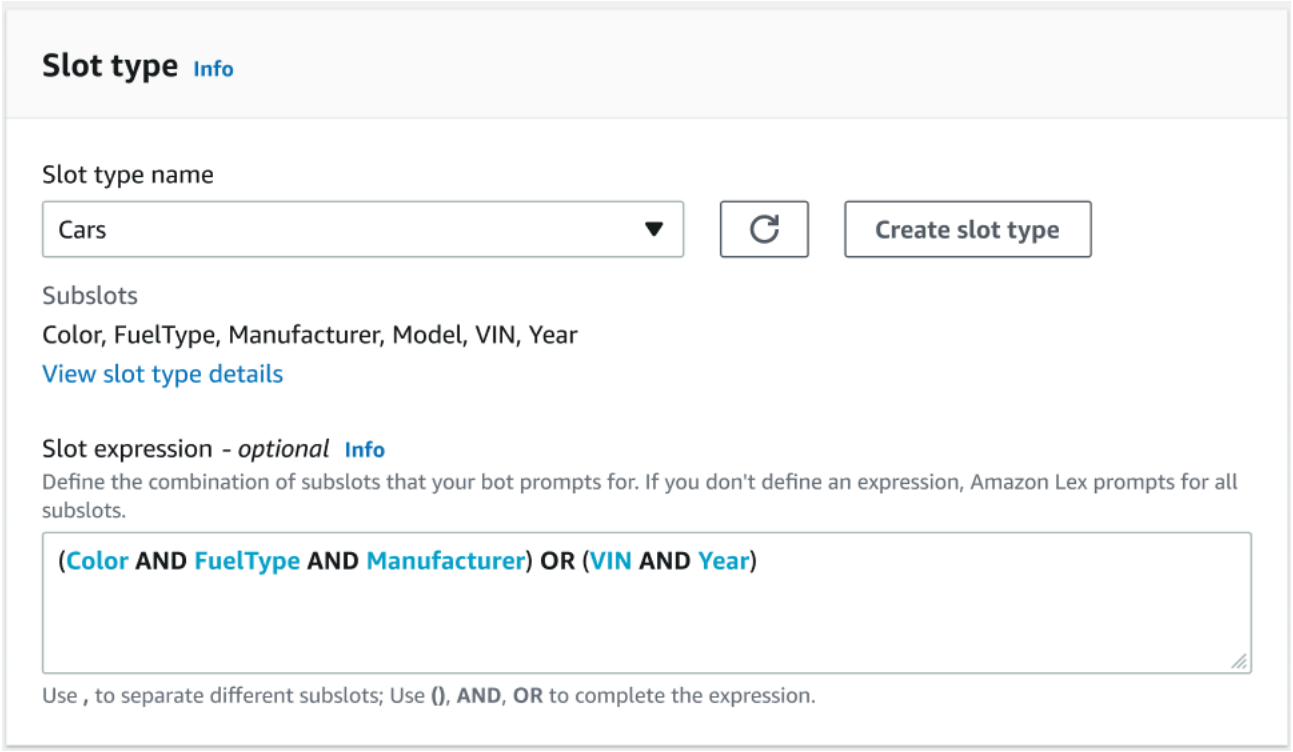
Para usar subslots em um slot composto, você deve primeiro configurar o tipo de slot composto. Para fazer isso, use as etapas do console de adicionar um tipo de slot ou a operação da API. Depois de escolher o nome e uma descrição para o tipo de slot composto, forneça informações sobre os subslots. Para obter mais informações sobre como adicionar um tipo de slot, consulte Adicionar tipos de slot

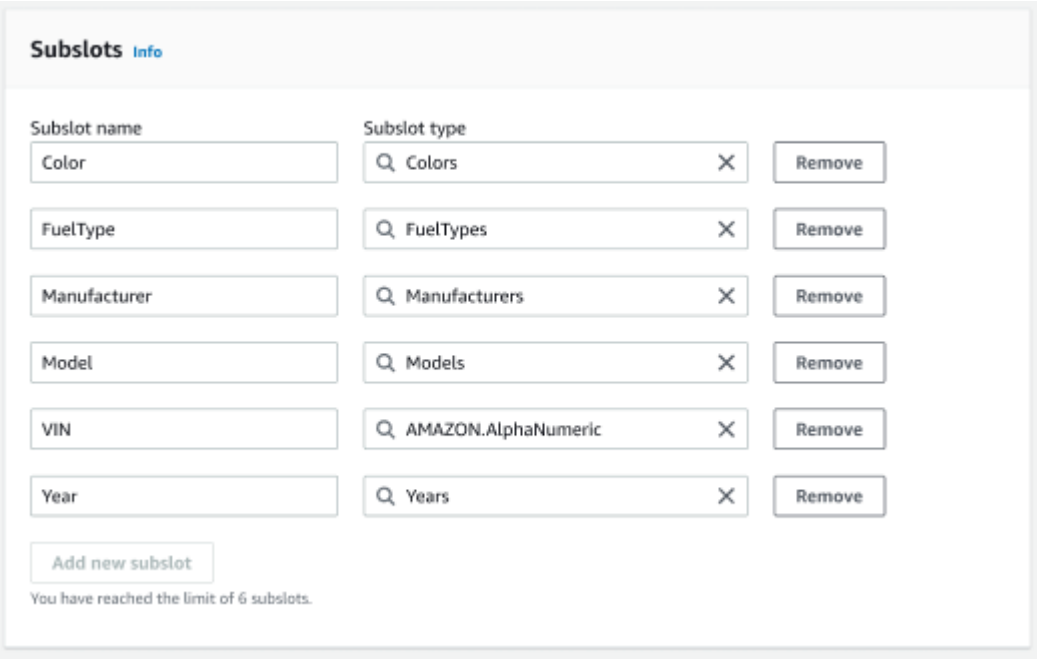
Subslots

Um tipo de slot composto requer a configuração dos slots subjacentes, chamados de subslots. Se deseja obter várias informações de um cliente em uma solicitação, configure uma combinação de subslots. Por exemplo: cidade e estado ou CEP. Você pode adicionar até 6 subslots a um slot composto.

Slots de tipos de slots singulares podem ser usados para adicionar subslots ao tipo de slot composto. No entanto, você não pode usar um tipo de slot composto como um tipo de slot para um subslot.

As imagens a seguir ilustram um slot composto “Carro”, que é uma combinação de subslots: Cor, Tipo de combustível, Fabricante, Modelo, VIN e Ano.





Construtor de expressões

Para impulsionar o preenchimento de um slot composto, você pode, opcionalmente, usar o construtor de expressões. Com o construtor de expressões é possível criar uma expressão lógica de slot para capturar os valores de subslot necessários na ordem desejada. Como parte da expressão booleana, você pode usar operadores como E e OU. Com base na expressão projetada, quando os subslots necessários são preenchidos, o slot composto é considerado preenchido.

Usar um tipo de slot composto

Para algumas intenções talvez você queira capturar slots diferentes como parte de um único slot. Por exemplo, um bot de agendamento de manutenção de automóveis pode ter uma intenção com o seguinte enunciado:

```
My car is a {car}
```

A intenção espera que o slot composto {car} contenha uma lista dos slots, incluindo detalhes do carro. Por exemplo, “2021 White Toyota Camry”.

O slot composto difere de um slot de vários valores. O slot composto é composto por vários slots, cada um com seu próprio valor. Por outro lado, um slot de vários valores é um slot singular que pode conter uma lista de valores. Para obter mais informações sobre slots de vários valores, consulte Uso de vários valores em um slot

Para um slot composto, o Amazon Lex retorna um valor para cada subslot na resposta à operação `RecognizeText` ou `RecognizeUtterance`. A seguir estão as informações do slot retornadas para o enunciado: “Quero agendar um serviço para meu “2021 White Toyota Camry” a partir do bot CarService.

```
"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"
            }
        }
    },
    ...
}
```

Um espaço composto pode ser obtido no primeiro turno ou no enésimo turno de uma conversa. Com base nos valores de entrada fornecidos, o slot composto pode extrair os demais subslots necessários.

Os slots compostos sempre retornam um valor para cada subslot. Quando o enunciado não contém um valor reconhecível para um determinado subslot, não há resposta retornada para esse subslot específico.

Os slots compostos funcionam com entrada de texto e voz.

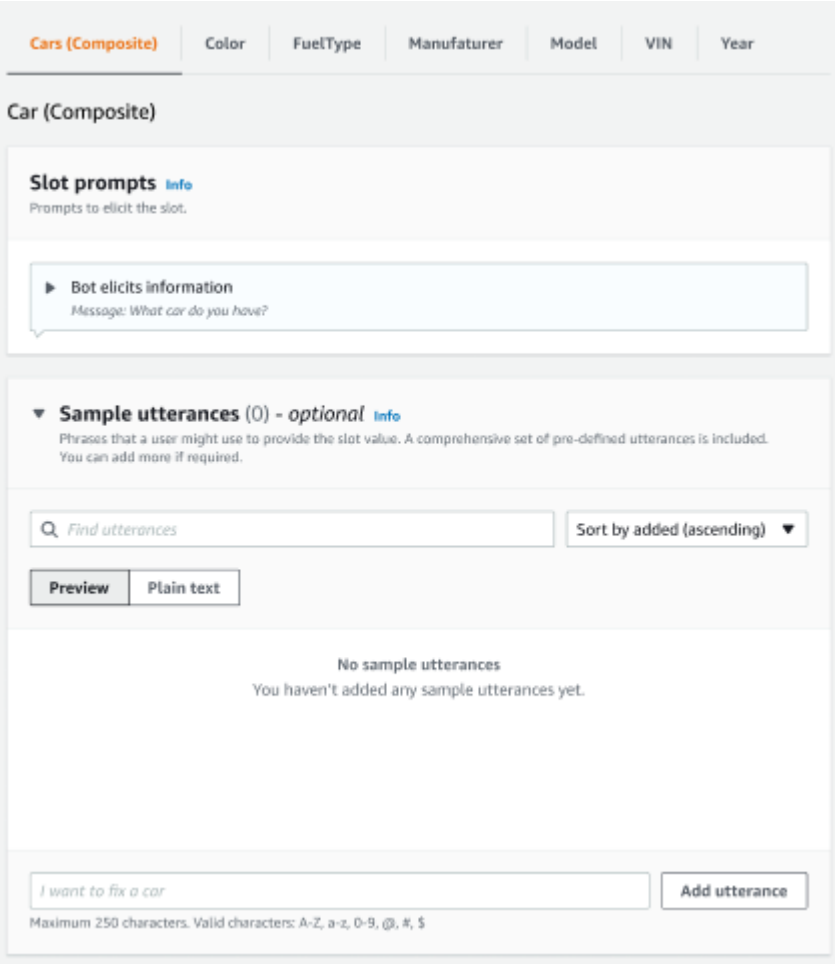
Ao adicionar um slot a um intent, um slot composto só está disponível como um tipo de slot personalizado.

Você pode usar slots compostos em prompts. Por exemplo, é possível definir a solicitação de confirmação de uma intenção.

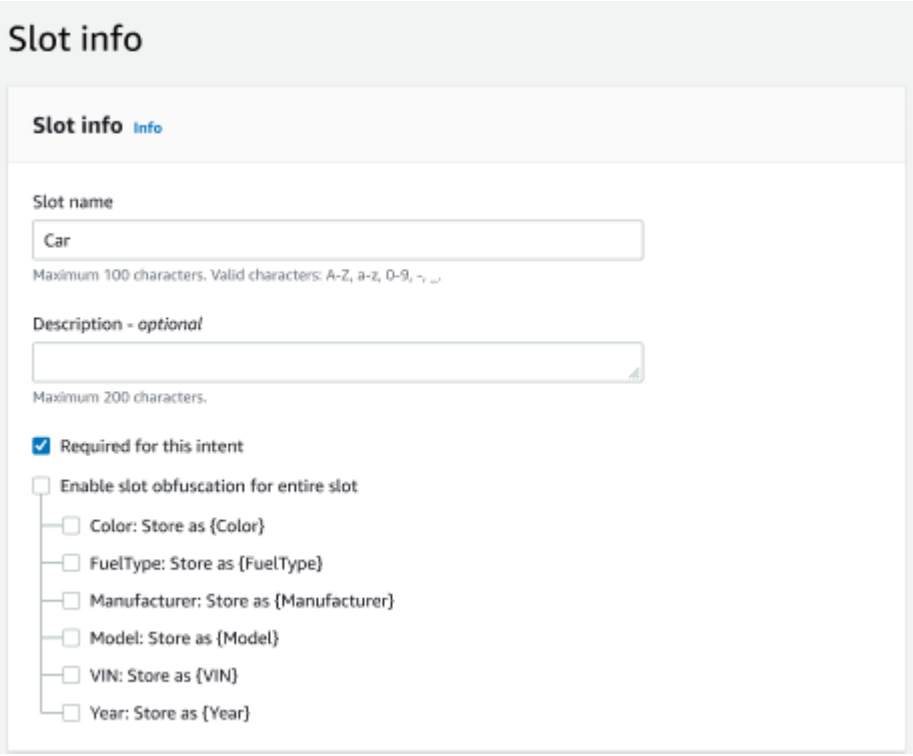
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

Quando o Amazon Lex envia a solicitação ao usuário, ele envia “Você gostaria de agendar o serviço para seu Toyota Camry branco 2021?”

Cada subslot é configurado como um slot. Você pode adicionar solicitações de slot para extrair o subslot e amostras de enunciados. Você pode ativar esperar e continuar para um subslot, bem como para os valores padrão. Para obter mais informações, consulte Usar valores de slot padrão.



A ofuscação de slots pode ser usada para mascarar todo o espaço composto nos logs de conversas. Observe que a ofuscação do slot é aplicada no nível do slot composto e, quando ativada, os valores dos subslots pertencentes a um slot composto são ofuscados. Ao ofuscar valores de slot, o valor de cada um dos valores de slot é substituído pelo nome do slot. Para mais informações, consulte Ocultar valores de slot nos logs de conversa.



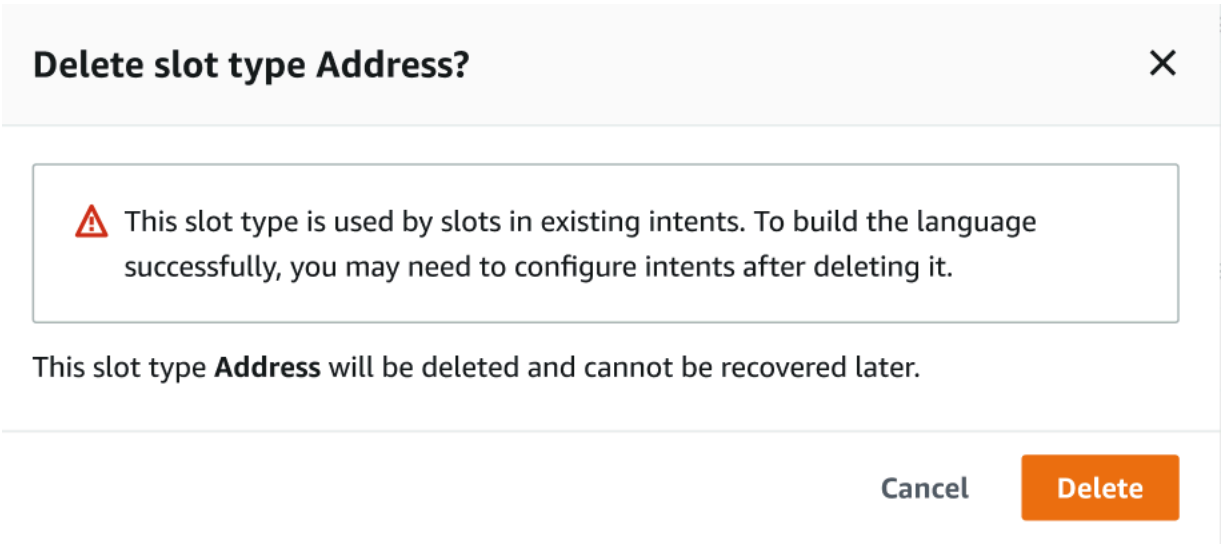
Editar um tipo de slot composto

Você pode editar um subslot de dentro da configuração do slot composto para modificar o nome e o tipo do subslot. No entanto, quando um slot composto estiver sendo usado por uma intenção, você precisará editar as intenções antes de modificar o subslot.

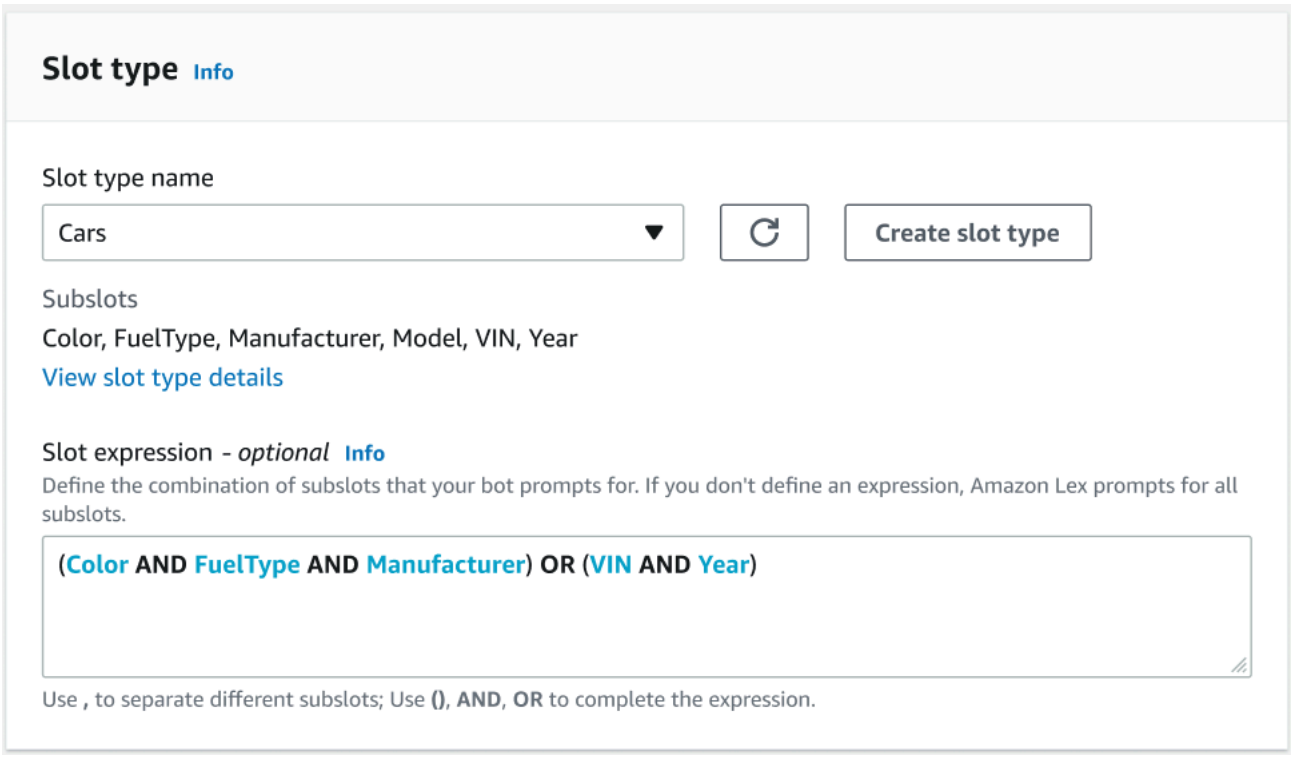
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

Excluir um tipo de slot composto

Você pode excluir um subslot de dentro da configuração do slot composto. Observe que quando um subslot está sendo usado dentro de uma intenção, os subslots ainda são removidos dessa intenção.



A expressão de slot no construtor de expressões fornece um alerta para informar sobre os subslots excluídos.



Testar um bot usando o console

O console Amazon Lex V2 contém uma janela de teste que você pode usar para testar a interação com o bot. Use a janela de teste para ter uma conversa de teste com o bot e ver as respostas que o aplicativo recebe do bot.

Há dois tipos de testes que você pode realizar com o bot. O primeiro, o teste expresso, permite que você teste o bot com as frases exatas que usou para criá-lo. Por exemplo, se você adicionou a expressão “Quero colher flores” à sua intenção, será possível testar o bot usando essa frase exata.

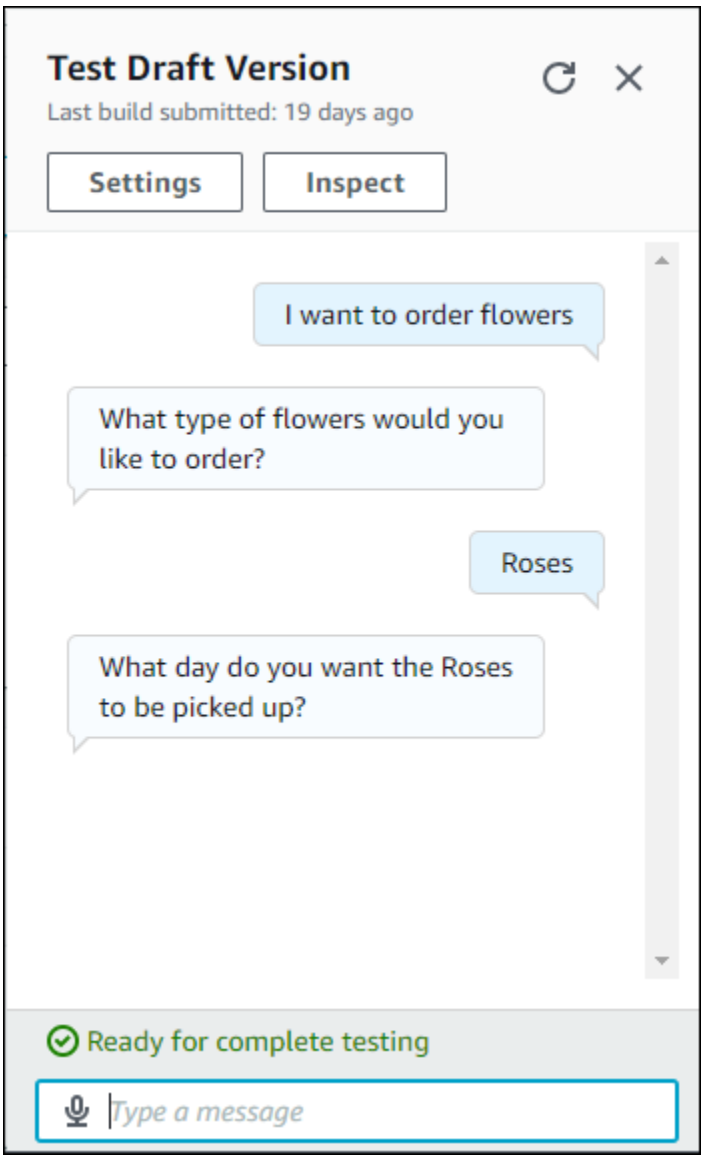
O segundo tipo, teste completo, permite que você teste o bot usando frases relacionadas aos enunciados configurados. Por exemplo, você pode usar a frase “Posso pedir flores” para iniciar uma conversa com o bot.

Teste o bot usando um alias e um idioma específicos. Se você estiver testando a versão de desenvolvimento do bot, use o alias `TestBotAlias` para testar.

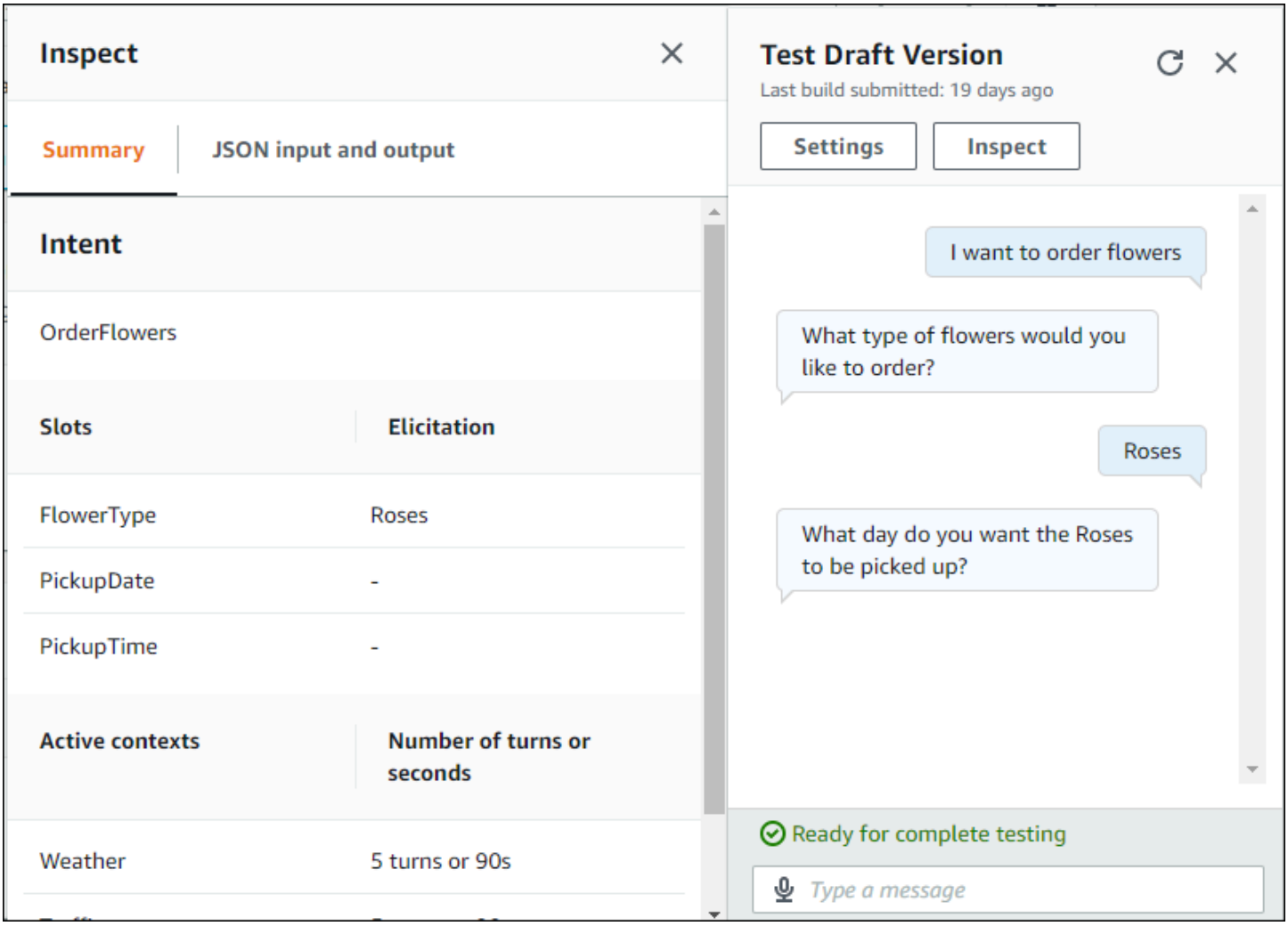
Como abrir a janela de teste

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot para testar na lista de bots.
3. No menu à esquerda, selecione Aliases.
4. Na lista de aliases, escolha o alias a ser testado.
5. Em Idiomas, escolha o botão de rádio do idioma a ser testado e selecione Testar.

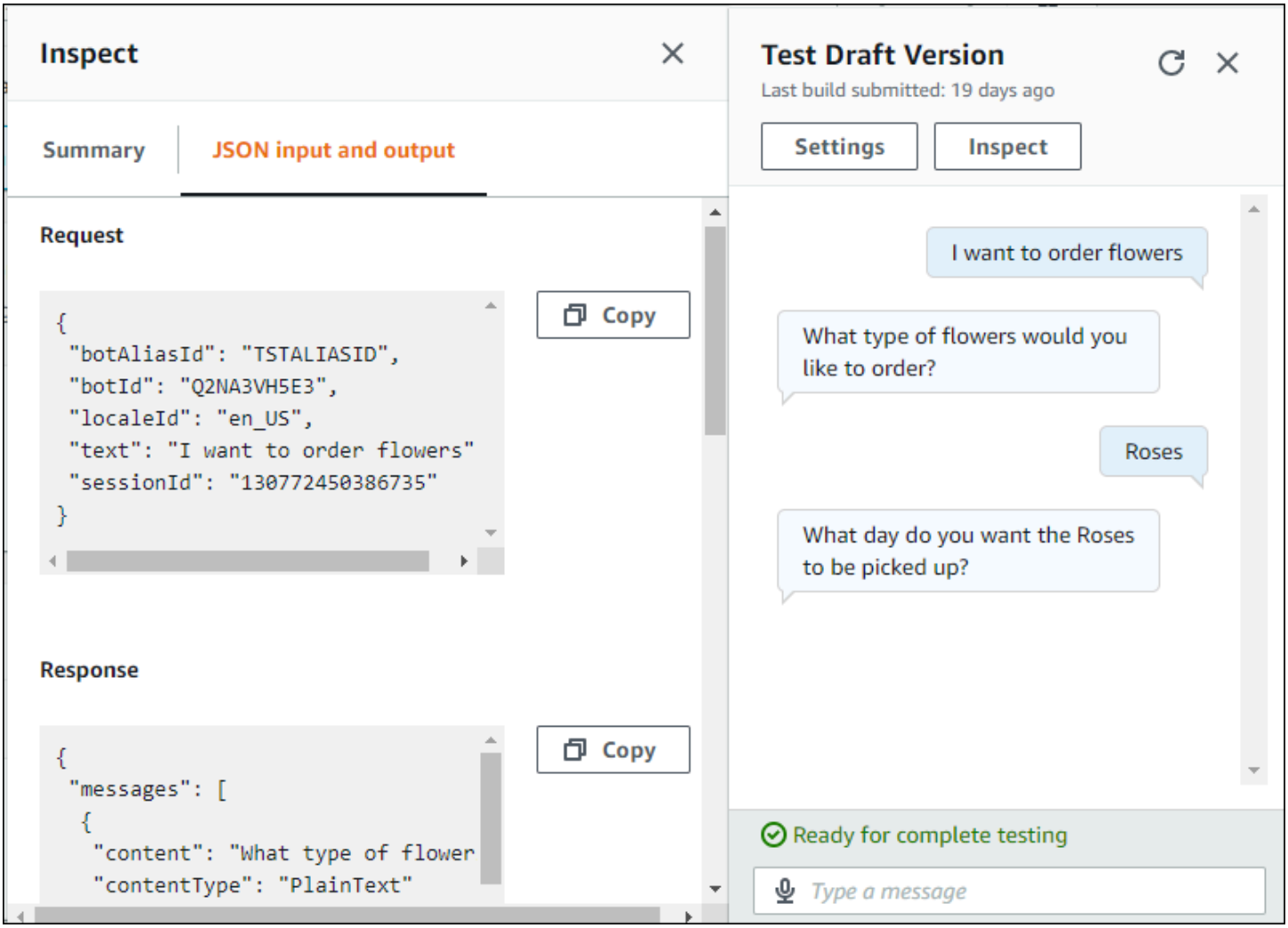
Depois de escolher Testar, a janela de teste será aberta no console. Você pode usar a janela de teste para interagir com o bot, conforme mostrado no gráfico a seguir.



Além da conversa, você também pode escolher Inspecionar na janela de teste para ver as respostas retornadas pelo bot. A primeira visualização mostra um resumo das informações retornadas do bot para a janela de teste.



Você também pode usar a janela de inspeção de teste para ver as estruturas JSON enviadas entre o bot e a janela de teste. É possível ver a solicitação na janela de teste e a resposta do Amazon Lex V2.


Inspect
Summary
JSON input and output
Request
Copy
{
"botAliasId": "TSTALIASID",
"botId": "Q2NA3VH5E3",
"localeId": "en_US",
"text": "I want to order flowers"
"sessionId": "130772450386735"
}
Response
Copy
{
"messages": [
{
"content": "What type of flower
"contentType": "PlainText"
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

Otimizar a criação de bots e o desempenho com IA generativa

> **Note**
>
> Esses recursos usam IA generativa. Ao usar o serviço, lembre-se de que ele pode fornecer respostas imprecisas ou inadequadas. Para obter mais informações, consulte a Política de IA responsável da AWS.
> Desenvolvido pelo Amazon Bedrock: AWS implementa a detecção automática de abusos. Como os recursos de IA generativa do Amazon LEX V2 são baseados no Amazon Bedrock, os usuários herdam os controles implementados no Amazon Bedrock para garantir a segurança e o uso responsável da IA.

Aproveite os recursos de IA generativa do Amazon Bedrock para automatizar e acelerar o processo de criação de bots do Amazon Lex V2. Você pode executar os seguintes processos com a ajuda do Amazon Bedrock:

- Criar bots e preenchê-los com intenções e tipos de slots relevantes de forma eficiente usando a descrição em linguagem natural.
- Gerar automaticamente exemplos de expressões para as intenções do seu bot.
- Melhorar o desempenho da resolução de slots de seus bots.
- Criar uma intenção para ajudar a responder às perguntas do seu cliente.

É possível ativar recursos de IA generativa para o Amazon Lex V2 por meio do console ou da API.

> **Note**
>
> Antes de aproveitar os recursos de IA generativa, você deve atender aos seguintes pré-requisitos:
>
> 1. Navegue até o console do Amazon Bedrock e inscreva-se para acessar o modelo Anthropic Claude que você pretende usar (para obter mais informações, consulte Acesso ao modelo). Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock.

2. Ative os recursos de IA generativa para a localidade do seu bot. Para isso, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa.

Using the console

1. Faça login AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Selecione o bot e o local no bot para o qual você deseja ativar os recursos de IA generativa.
3. Na seção Configurações de IA generativa, selecione Configurar.
4. Alterne o botão Habilitado para cada recurso que você quiser ativar. Selecione o modelo e a versão que deseja usar para criar o recurso. Habilitar um recurso pode acarretar custos adicionais. Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock. Para saber mais sobre um recurso, selecione o tópico correspondente na lista abaixo. Selecione Salvar depois de ativar os recursos desejados. Um banner verde de êxito será exibido para confirmar que os recursos foram ativados.

Using the API

1. Para habilitar recursos generativos de IA para um novo bot, use a CreateBotoperação para criar um novo bot.
2. Envie uma CreateBotLocalesolicitação, modificando o `generativeAISettings` objeto conforme necessário. Se você estiver habilitando os recursos de um bot existente, envie uma UpdateBotLocalesolicitação em vez disso.

   a. Para permitir o uso do construtor de bots descritivo, modifique o objeto `descriptiveBotBuilder`. Especifique o modelo de base a ser usado no campo `modelArn` e defina o valor `enabled` como `True`.

   b. Para permitir a melhoria da resolução de slots, modifique o objeto `slotResolutionImprovement`. Especifique o modelo de base a ser usado no campo `modelArn` e defina o valor `enabled` como `True`.

   c. Para habilitar a geração de um exemplo de enunciado, modifique o objeto `sampleUtteranceGeneration`. Especifique o modelo de base a ser usado no campo `modelArn` e defina o valor `enabled` como `True`.

Tópicos

Usar o construtor de bots descritivo

> **Note**
>
> Antes de aproveitar os recursos de IA generativa, você deve atender aos seguintes pré-requisitos:
>
> 1. Navegue até o console do Amazon Bedrock e inscreva-se para acessar o modelo Anthropic Claude que você pretende usar (para obter mais informações, consulte Acesso ao modelo). Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock.
> 2. Ative os recursos de IA generativa para a localidade do seu bot. Para isso, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa.

O construtor de bots descritivo permite que você aproveite o acesso do Amazon Bedrock a grandes modelos de linguagem para melhorar a eficiência do processo de criação de bots. Você fornece um prompt usando linguagem natural que inclui a finalidade do bot e as ações que ele deve realizar. O Amazon Lex V2 aproveita os recursos do Amazon Bedrock para gerar intenções e tipos de slots relevantes para seu bot com base na sua descrição. Depois de escolher as intenções e os tipos de slots que deseja manter, você pode então iterar no bot para modificá-lo de acordo com seu caso de uso específico. O construtor de bots descritivo economiza seu tempo, permitindo que você evite a necessidade de criar intenções e tipos de slots manualmente para o bot.

O construtor de bots descritivo está disponível nas localidades em inglês (veja as localidades que começam com `en_` na tabela em Idiomas e locais aceitos pelo Amazon Lex V2).

Antes de criar o bot, faça o seguinte:

1. Verifique se sua função tem as permissões corretas revisando as etapas em Permissões necessárias para criar um bot com descrição em linguagem natural.

2. Decida a descrição a ser usada. É possível consultar Exemplos de descrições de bot para obter exemplos de descrições de bots.

Crie um bot usando linguagem natural para descrever o que o bot deve ser capaz de fazer. O Amazon Lex V2 invoca os modelos do Amazon Bedrock para gerar intenções e tipos de slots adequados ao caso de uso do seu bot. Você pode criar o bot com o console ou a API.

Console

Criar um bot usando o construtor de bots descritivo

1. Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Na página Bots, selecione Criar bot.
3. Para o Método de criação, escolha Construtor de bots descritivo – GenAI.
4. Dê ao seu bot um nome e uma descrição opcional, configure as permissões do IAM e escolha se o bot está sujeito à COPPA ou não. Depois, selecione Próximo.
5. Selecione um idioma para criar o bot, uma voz para o bot e um limite de confiança para a classificação da intenção (para obter mais informações, consulte Usar pontuações de confiança de intenção).
6. Em Criador de bots descritivo – GenAI, forneça uma descrição para o bot que deseja criar. A descrição deve ser detalhada e precisa para ajudar a gerar intenções adequadas e suficientes para o bot. Inclua uma lista de ações para melhorar o processo de criação da intenção.
7. Selecione um fornecedor de modelo e um modelo em Selecionar modelo.
8. Para criar o bot em outra localidade, escolha Adicionar outro idioma. Quando terminar de adicionar idiomas, selecione Concluído. O Amazon Lex V2 cria o bot e o construtor de bots descritivo gera intenções e slots para ele. Quando a localidade é gerada, o banner muda de azul para verde. Selecione Revisar para ver as intenções geradas e os tipos de slots.

> **Note**
>
> Atualmente, o construtor de bots descritivo está disponível somente em localidades em inglês. No entanto, você pode copiar um bot para um local diferente do inglês depois de criá-lo.

Revise as intenções geradas e os tipos de slots e adicione-os ao bot

1. Se houver intenções e tipos de slots suficientes aplicáveis ao caso de uso do bot, você poderá revisar as intenções geradas.

   a. Revise as Intenções geradas.

      i. Escolha uma caixa de seleção ao lado de uma intenção para removê-la da lista de intenções a serem adicionadas ao bot.

      ii. Escolha um nome de intenção para ver os Exemplos de enunciados e os Slots gerados para a intenção.

      iii. Por padrão, todos os enunciados e slots estão selecionados. Escolha uma caixa de seleção para remover esse item da intenção. Selecione Adicionar à seleção para manter os itens marcados na intenção.

   b. Revise os Tipos de slots gerados.

      i. Marque uma caixa de seleção ao lado de um tipo de slot para removê-lo da lista de intenções a serem adicionadas ao bot.

      ii. É possível adicionar valores a um tipo de slot depois de adicioná-lo ao bot

2. Quando tiver escolhido as intenções e os tipos de slots que deseja, selecione Adicionar intenções e tipos de slots na parte superior da página para adicionar as intenções e os tipos de slots ao bot.

3. Após a adição dos recursos, um banner verde de êxito será exibido. Acesse Intenções e Tipos de slot para editar os gerados e adicionar mais valores.

4. Se as Intenções geradas e os Tipos de slots gerados forem, em sua maioria, inaplicáveis ao bot que você deseja criar, execute as etapas a seguir.

   a. Selecione Nova geração na seção Detalhes do construtor de bots descritivo.

   b. Reescreva o prompt e selecione Gerar novamente para gerar novas intenções e tipos de slots. Os resultados serão diferentes se você usar um modelo diferente.

      > **Important**
      >
      > Não há garantia de que as mesmas intenções e slots serão gerados. Você será cobrado toda vez que gerar novamente as intenções e os tipos de slots.

API

Criar o bot usando descrição em linguagem natural

Quando você usa o construtor de bots descritivo por meio da API, ele cria uma definição de bot em um arquivo .zip em um bucket do Amazon S3. Baixe esse arquivo e importe a definição do bot para o Amazon Lex V2 para criar o bot.

1. Envie uma solicitação do CreateBot para criar um bot. Depois, envie uma solicitação createBotLocale para criar uma localidade para o bot.
2. Envie uma solicitação StartBotResourceGeneration, especificando a ID, a versão e a localidade do bot. Use `DRAFT` para a versão bot. Forneça sua solicitação no campo `generationInputPrompt`. A descrição deve ser detalhada e precisa para ajudar a gerar intenções adequadas e suficientes para o bot. Inclua uma lista de ações para melhorar o processo de criação da intenção.
3. Anote o valor do `generationId` na resposta.
4. Envie uma solicitação DescribeBotResourceGeneration usando a `generationId` que você recebeu na resposta `StartBotResourceGeneration`. Inclua a ID, a versão e a localidade do bot.
5. Se a `generationStatus` na resposta `DescribeBotResourceGeneration` for `Complete`, o campo `generatedBotLocaleUrl` também será preenchido. Use esse URI do Amazon S3 para baixar a definição do bot seguindo as etapas em Baixar um objeto.

Verificar a definição de bot gerada e importá-la

1. Use esse URI do `generationStatus` na resposta `DescribeBotResourceGeneration` para baixar a definição do bot seguindo as etapas em Baixar um objeto.
2. É possível modificar diretamente o conteúdo gerado para o caso de uso específico do seu bot editando o arquivo. Também é possível enviar outra solicitação `StartBotResourceGeneration` para gerar novamente intenções e slots.

> **Important**
>
> Não há garantia de que as mesmas intenções e slots serão gerados. Você será cobrado toda vez que gerar novamente as intenções e os tipos de slots.

3. Para importar a definição do bot, siga as etapas em Importação.

4. Após a importação, é possível modificar as intenções e os slots gerados usando as operações UpdateIntent, UpdateSlot e UpdateSlotType.

Para listar metadados sobre todos os itens gerados para uma localidade de bot, use a operação ListBotResourceGenerations. Use qualquer um dos valores `generationId` retornados em uma solicitação `DescribeBotResourcGeneration` para recuperar o URI do Amazon S3 para uma definição de bot gerada.

Tópicos

- Exemplos de descrições de bot
- Permissões necessárias para criar um bot com descrição em linguagem natural

Exemplos de descrições de bot

Industry	Exemplos de prompt
Serviços financeiros	Somos um serviço financeiro de cartões que ajuda os usuários a realizar tarefas quando recebem um novo cartão, como ativar o cartão, enviar um e-mail ou um PIN, verificar um novo cartão (usando um código postal). Também os ajudamos nas tarefas associadas ao cartão existente, como perguntar sobre os benefícios do cartão de crédito, informar perda de cartão, solicitar um novo cartão, redefinir o PIN do cartão ou pagar a fatura do cartão.
Serviços de alimentação	Quero um bot para ajudar os clientes a fazer pedidos de comidas (usando ID do item, quantidade, tamanho), verificar o status do pedido e cancelar um pedido. Use a ID do pedido para indexar pedidos.
Companhia aérea	Somos um domínio de companhia aérea que ajuda os usuários a reservar passagens

Industry	Exemplos de prompt
	aéreas, verificar detalhes de uma reserva, obter o recibo de um voo reservado, consultar o status do voo, reagendar voos reservado s, obter detalhes do voo e cancelar voos reservados. Também é possível gerar intenções adicionais se elas oferecerem suporte às funções na descrição do domínio.
Seguros	Objetivo: Somos uma seguradora que vende apólices de seguro para automóveis, residênci as e pensões. Quero um bot que possa verificar o status da solicitação, registrar uma solicitação, fazer pagamentos de apólices e cancelar uma apólice. Usamos policy_id e os quatro últimos dígitos do SSN para identific ação e validação da conta. Espero que o bot tenha pelo menos as seguintes intenções e slots: autenticação – policy_id, tipo de last4SSNpolicy: automóvel, residencial, status da annuitypolicy: saldo do cheque, data de vencimento do cheque, cheque coveragemake um pagamento: pagamento único, parcelas, valor
Gerenciamento de veículos	Estamos criando um bot Towed Cars Lookup que ajuda os motoristas de uma cidade que tiveram o carro guinchado a descobrir onde está o carro. Esse bot deve perguntar o endereço ou local de onde o automóvel foi guinchado e detalhes sobre o veículo, como placa e a marca, o modelo e o ano de fabricação do carro. O bot deve responder com a localização de onde o carro guinchado está estacionado e o horário de funcionamento.

Industry	Exemplos de prompt
Viajar	Sou agente de viagens e quero um bot para ajudar meus clientes a reservar uma viagem para a Disney. A Disney tem vários parques em todo o mundo e também tem hotéis, restaurantes e entretenimento especial que podem ser reservados. Os usuários do bot devem poder modificar ou cancelar a reserva. As reservas devem incluir, no mínimo, o parque, as datas e o hotel. Refeições ou entretenimento podem ser incluídos opcionalmente, e podem ser adicionados ou alterados posteriormente.

Permissões necessárias para criar um bot com descrição em linguagem natural

- Para acessar esse recurso no console do Amazon Lex V2, certifique-se de que sua função de console tenha a permissão `bedrock:ListFoundationModels`.
- O perfil do IAM associado ao bot deve ter a permissão `bedrock:InvokeModel`. Quando você habilita o recurso com o console do Amazon Lex, a política é adicionada automaticamente à função do bot, desde que o bot esteja usando uma função vinculada ao serviço gerada pelo Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
      }
  ]
}
```

Geração de enunciados

> **Note**
>
> Antes de aproveitar os recursos de IA generativa, você deve atender aos seguintes pré-requisitos:
>
> 1. Navegue até o console do Amazon Bedrock e inscreva-se para acessar o modelo Anthropic Claude que você pretende usar (para obter mais informações, consulte Acesso ao modelo). Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock.
> 2. Ative os recursos de IA generativa para a localidade do seu bot. Para isso, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa.

Use a geração de enunciados para automatizar a criação de exemplos de enunciados de acordo com sua intenção. Em vez de inserir manualmente exemplos de enunciados, o Amazon Lex V2 gera exemplos de enunciados para você com base no nome da intenção, na descrição e nos exemplos de enunciados existentes, para que você possa reduzir o tempo e o esforço gastos para descobrir e escrever seus próprios exemplos de enunciados. Depois que o Amazon Lex V2 gerar os enunciados, você poderá editá-los e excluí-los. Use essa ferramenta para agilizar a criação de exemplos de enunciados para o processo de reconhecimento da intenção.

Para permitir a geração de enunciados, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa para ativar os recursos de IA generativa.

É possível gerar enunciados com o console ou a API.

Console

1. Navegue até a seção Exemplos de enunciados de qualquer intenção em seu bot (no Visual Conversation Builder, ela está no bloco Iniciar).
2. Selecione o botão Gerar enunciados para gerar cinco exemplos de enunciados. Se sua intenção tiver mais de 25 exemplos de enunciados, o botão Gerar enunciados será desativado.

3. Os enunciados gerados são exibidos com um banner verde que diferencia os enunciados gerados dos enunciados existentes.
4. Passe o mouse sobre um enunciado para exibir as opções para editar, excluir e classificar os enunciados gerados.

API

1. Envie uma solicitação GenerateBotElement, preenchendo a intenção e a ID do bot, a versão e a localidade para os quais você deseja gerar exemplos de enunciados.
2. A resposta retorna uma lista de objetos SampleUtterance, cada um contendo um enunciado gerado.
3. Para adicionar os enunciados à intenção, envie uma solicitação UpdateIntent e adicione os enunciados ao campo `sampleUtterances`.

Tópicos

- Permissões para a geração de enunciados

Permissões para a geração de enunciados

Para acessar esse recurso no console Amazon Lex V2, certifique-se de que sua função de console tenha as permissões `bedrock:ListFoundationModels` e `bedrock:InvokeModel`.

Usar a resolução assistida de slots

> **Note**
>
> Antes de aproveitar os recursos de IA generativa, você deve atender aos seguintes pré-requisitos:
>
> 1. Navegue até o console do Amazon Bedrock e inscreva-se para acessar o modelo Anthropic Claude que você pretende usar (para obter mais informações, consulte Acesso ao modelo). Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock.
> 2. Ative os recursos de IA generativa para a localidade do seu bot. Para isso, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa.

É possível melhorar a precisão de alguns slots integrados no fluxo de conversas do seu bot usando a resolução assistida de slots. A resolução assistida de slots usa grandes modelos de linguagem (LLMs) do Amazon Bedrock para melhorar o reconhecimento de alguns slots integrados, o que resulta em uma melhor interpretação das respostas do cliente durante a inferência de slots. Para enunciados que não puderam ser resolvidos normalmente, o Amazon Lex tentará resolvê-los pela segunda vez usando o Amazon Bedrock.

A resolução assistida de slots permite que você use o poder dos modelos de base (FMs) do Amazon Bedrock para melhorar a precisão dos seguintes slots integrados:

- `AMAZON.Alphanumeric` sem compatibilidade com regex
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

Você pode habilitar a resolução assistida de slots para qualquer intenção que use os slots integrados listados acima. A resolução assistida de slots não se aplica a slots personalizados ou aos slots integrados da Amazon não listados acima.

Você pode coletar dados sobre as melhorias de precisão depois de habilitar a resolução assistida de slots em seu bot do Amazon Lex usando logs e métricas de conversas.

- Logs de conversas: as interpretações terão a `interpretationSource` como o `Bedrock`, caso o Amazon Bedrock tenha sido usado para resolver o slot.
- Métricas do CloudWatch: as métricas serão publicadas nas dimensões listadas na métrica do CloudWatch. Para saber mais, consulte Monitorar o Amazon Lex com o Amazon CloudWatch.

Para usar o construtor de bots descritivo, certifique-se de que seu perfil do IAM tenha as permissões adequadas seguindo as etapas em Permissões para a resolução assistida de slots.

Tópicos

- Exemplos de resolução assistida de slots

- Habilitar a resolução assistida de slots na tela de configuração da IA generativa
- Habilitar a resolução assistida de slots nas configurações do slot
- Permissões para a resolução assistida de slots

Exemplos de resolução assistida de slots

Veja abaixo alguns exemplos em que a resolução assistida de slots é capaz de transformar de forma inteligente os enunciados do usuário em um valor.

AMAZON.Number

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
Viajar	AMAZON.Number	numberOfNightsStayed	Quantas noites você ficou durante a viagem?	Uma semana inteira, sete noites.	7
Setor bancário	AMAZON.Number	numberOfPeopleOnTheAccount	Quantas pessoas estão na conta?	Eu e minha esposa.	2
Viajar	AMAZON.Number	numberOfStops	Quantas paradas?	Uma no Japão. Uma em LA.	2

AMAZON.AlphaNumeric

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
Aluguel de carros	AMAZON.Alphanumeric	transactionId	Qual é a ID da transação ?	Acho que era alfa whisky echo oito três	AWE8349RJ

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
				quatro nove romeo juliet.	
Viajar	AMAZON.Alphanumeric	confirmationCode	Qual é o número de confirmação da sua reserva?	O número de confirmação é BLT2UE.	BLT2UE

AMAZON.Date

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido	currentDate
Aluguel de carros	AMAZON.Date	dueDate	Quando o contrato de locação expira?	O contrato termina no dia 1º do próximo mês.	01-12-2023	09-11-2023
Viajar	AMAZON.Date	returnDate	Quando você vai voltar?	Mais tarde, hoje, por volta das 19.	09-11-2023	09-11-2023

AMAZON.PhoneNumber

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
Seguros	AMAZON.PhoneNumber	policyHolder	Qual é o número de	O número de telefone do segurado é	1234567890

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
			telefone do segurado?	123-456-7890.	
Varejo	AMAZON.PhoneNumber	phoneLookup	Qual é seu número de telefone para que eu possa encontrar a conta?	Acho que está em 413-570-9617, deixe-me verificar.	4135709617

AMAZON.Country

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
Viajar	AMAZON.Country	nativeCountry	Qual é o seu país de origem?	Sou indiano.	Índia
Setor bancário	AMAZON.Country	countryItinerary	Para quais países você viajará com seu cartão de débito?	Vou viajar para Nova Delhi.	Índia

AMAZON.City

Vertical	Tipo de slot	Intenção	Pergunta	Resposta	Valor resolvido
Seguros	AMAZON.City	policyHolderCity	Em que cidade o	Moro em Springfield.	Springfield

Vertical	Tipo de slot	Intenção	Pergunta	Resposta	Valor resolvido
			segurado reside?		
Viajar	AMAZON.City	destinationCity	Para qual cidade você vai viajar?	Vou viajar para Tóquio.	Tóquio

AMAZON.Confirmation

Vertical	slotType	slotName	slotPrompt	enunciado	Valor resolvido
Seguros	AMAZON.Confirmation	policyExpired	A apólice de seguro expirou?	Sim, infelizmente expirou.	Sim
Setor bancário	AMAZON.Confirmation	hasInvestments	Você tem algum investimento?	Ainda não investi em nada.	Não

Habilitar a resolução assistida de slots na tela de configuração da IA generativa

É possível habilitar a resolução assistida de slots integrados compatíveis navegando até a tela IA generativa.

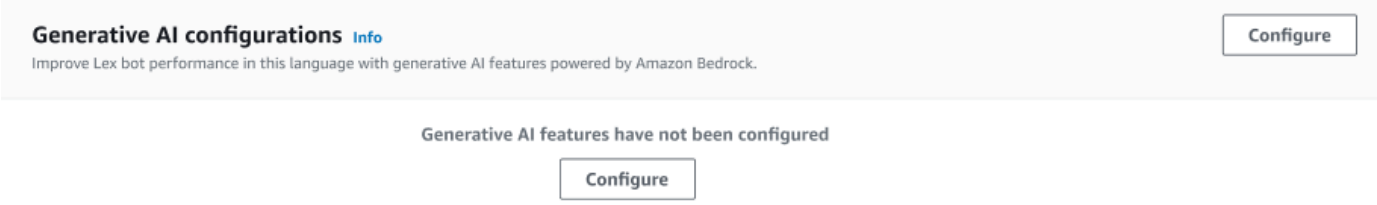
Se o slot for um slot integrado compatível, você terá a opção de ativar a resolução assistida de slots no nível do slot.

1. Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. No painel de navegação, em Bots, selecione o bot que você deseja usar para a resolução assistida de slots.

3. Selecione o idioma inglês (EUA) para o bot que você deseja habilitar.
4. Acesse a seção Configuração de IA generativa na tela.
5. Selecione Acessar o Amazon Bedrock para se inscrever e habilitar o recurso, caso ele não tenha sido habilitado.

> **Note**
>
> Se você não tiver acesso aos modelos de base do Amazon Bedrock, consulte Acessar o Amazon Bedrock. Clique em Acessar o Amazon Bedrock para acessar a página do Amazon Bedrock, onde você poderá se inscrever para acessar os modelos de base. No momento, a resolução assistida de slots é compatível com Claude V2 e Claude Instant V1. Sugerimos usar Claude V2 para obter melhores resultados.

6. Se já tiver acesso aos modelos de base do Bedrock, você deverá ver um botão Configurar. Clique nele para acessar a página de configuração de IA generativa para ativar os recursos de IA generativa no Lex.



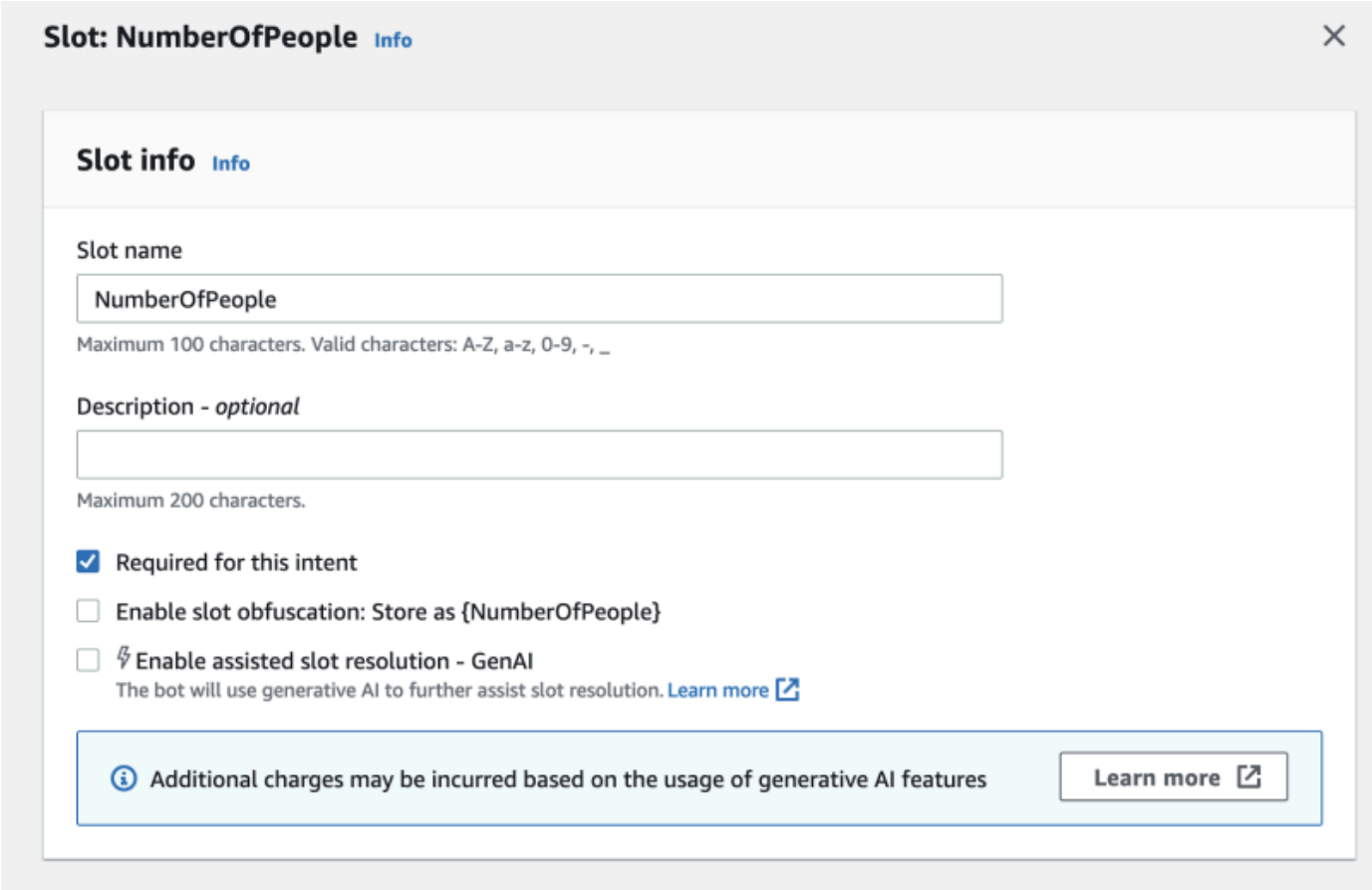
7. No canto superior direito da caixa, mova o controle deslizante para a direita para escolher a configuração Ativado.
8. Escolha o botão Habilitar para ativar a resolução assistida dos slots selecionados.
9. É possível desabilitar a resolução assistida de slots selecionando os slots na lista e selecionando o botão Desabilitar.

Habilitar a resolução assistida de slots nas configurações do slot

É possível habilitar a resolução assistida de slots integrados compatíveis navegando até o nível do slot para cada intenção que tenha slots. Os slots devem ser um dos slots integrados compatíveis listados acima para que haja a opção de ativar a resolução assistida de slots. Se o slot não tiver a opção de ativar a resolução assistida de slots, a opção ficará esmaecida.

> **Note**
>
> Primeiro, você deve ativar o recurso de resolução assistida de slots no painel de IA generativa para ativar o recurso para slots individuais.

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. No painel de navegação, em Bots, selecione o bot que deseja usar para a resolução assistida de slots.
3. Em Todos os idiomas, selecione Inglês (EUA) para expandir a lista.
4. No painel esquerdo, escolha Intenções para visualizar uma lista de intenções no bot que você selecionou.
5. Na tela Intenções, escolha a intenção que contém os slots que você deseja modificar.
6. Selecione o nome da intenção para visualizar os slots dessa intenção.
7. Selecione o botão Opções avançadas na seção Slots.
8. Marque a caixa de seleção Habilitar resolução assistida de slots para habilitar o recurso.



9. Escolha o botão Atualizar slot no canto inferior direito da tela. Isso ativará a resolução assistida dos slots que você escolheu.

É possível habilitar a resolução assistida de slots integrados compatíveis fazendo chamadas de API.

- Siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa para habilitar a resolução assistida de slots para a localidade do seu bot.
- Envie uma solicitação UpdateSlot, especificando o slot para o qual você deseja habilitar a resolução assistida de slots. No campo `slotResolutionSetting`, defina o valor `slotResolutionStrategy` como `EnhancedFallback`. Para criar um slot com a resolução assistida de slots habilitada, em vez disso, envie uma solicitação CreateSlot.

Permissões para a resolução assistida de slots

- Para acessar esse recurso no console do Amazon Lex V2, certifique-se de que sua função de console tenha a permissão `bedrock:ListFoundationModels`.

- O perfil do IAM associado ao bot deve ter a permissão `bedrock:InvokeModel`. Quando você habilita o recurso com o console do Amazon Lex, a política é adicionada automaticamente à função do bot, desde que o bot esteja usando uma função vinculada ao serviço gerada pelo Amazon Lex.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:Region::foundation-model/modelId"
          ]
      }
  ]
}
```

AMAZON.QnAIntent

> **Note**
>
> Antes de aproveitar os recursos de IA generativa, você deve atender aos seguintes pré-requisitos:
>
> 1. Navegue até o console do Amazon Bedrock e inscreva-se para acessar o modelo Anthropic Claude que você pretende usar (para obter mais informações, consulte Acesso ao modelo). Para obter informações sobre preços para usar o Amazon Bedrock, consulte Definição de preços do Amazon Bedrock.
> 2. Ative os recursos de IA generativa para a localidade do seu bot. Para isso, siga as etapas em Otimizar a criação de bots e o desempenho com IA generativa.

Você pode aproveitar os FMs do Amazon Bedrock para ajudar a responder às perguntas dos clientes em uma conversa com o bot. O Amazon Lex V2 oferece uma `AMAZON.QnAIntent` integrada

que pode ser adicionada ao seu bot. Essa intenção aproveita os recursos de IA generativa do Amazon Bedrock ao reconhecer as perguntas dos clientes e pesquisar uma resposta nos seguintes repositórios de conhecimento (por exemplo, **`Can you provide me details on the baggage limits for my international flight?`**). Esse recurso reduz a necessidade de configurar perguntas e respostas usando o diálogo baseado em tarefas dentro das intenções do Amazon Lex V2. Essa intenção também reconhece perguntas complementares (por exemplo, **`What about domestic flight?`**) com base no histórico da conversa e fornece a resposta adequada.

Certifique-se de que seu perfil do IAM tenha as permissões adequadas para acessar a `AMAZON.QnAIntent` seguindo as etapas em Permissões para a AMAZON.QnAIntent.

Para aproveitar a `AMAZON.QnAIntent`, é necessário ter configurado um dos repositórios de conhecimento a seguir.

- Banco OpenSearch de dados do Amazon Service — Para obter mais informações, consulte Criação e gerenciamento de domínios do Amazon OpenSearch Service.
- Índice do Amazon Kendra: para obter mais informações, consulte Criar um índice.
- Base de conhecimento do Amazon Bedrock: para obter mais informações, consulte Criar uma base de conhecimento.

É possível instalar a `AMAZON.QnAIntent` de duas maneiras:

Configurar usando as configurações de IA generativa

1. No console do Amazon Lex V2, selecione Bots no painel de navegação esquerdo e escolha o bot ao qual deseja adicionar a intenção na seção Bots.
2. No painel de navegação esquerdo, selecione o idioma ao qual deseja adicionar a intenção.
3. Na seção Configurações de IA generativa, selecione Configurar.
4. Na seção Configurações de QnA, selecione Criar intenção de QnA.

Configurar adicionando uma intenção integrada ao seu bot

1. No console do Amazon Lex V2, selecione Bots no painel de navegação esquerdo e escolha o bot ao qual deseja adicionar a intenção na seção Bots.
2. No painel de navegação esquerdo, selecione Intenções no idioma ao qual deseja adicionar a intenção.
3. Selecione Adicionar intenção e escolha Usar intenção integrada no menu suspenso.

4. Para obter mais detalhes sobre a configuração da `AMAZON.QnAIntent`, consulte AMAZON.QnAIntent.

> **Note**
>
> A `AMAZON.QnAIntent` será ativada quando um enunciado não for classificado em nenhuma das outras intenções presentes no bot. Essa intenção será ativada quando um enunciado não for classificado em nenhuma das outras intenções presentes no bot. Observe que essa intenção não será ativada para enunciados perdidos ao inferir um valor de slot. Uma vez reconhecido, a `AMAZON.QnAIntent` usa o modelo especificado do Amazon Bedrock para pesquisar a base de conhecimento configurada e responder à pergunta do cliente.

Tópicos

- Permissões para a AMAZON.QnAIntent

Permissões para a AMAZON.QnAIntent

Para acessar esse recurso no console Amazon Lex V2, certifique-se de que sua função de console tenha permissões `bedrock:ListFoundationModels`.

O perfil do IAM associado ao bot deve ter as permissões necessárias para `AMAZON.QnAIntent` a seguir. A função de bot deve ter permissões para chamadas `bedrock:InvokeModel`. Também é necessário anexar uma instrução para cada datastore que especificar nos seus bots `AMAZON.QnAIntent` (consulte as instruções `Permissions to access Amazon Kendra index`, `Permissions to access OpenSearch Service index` e `Permissions to access knowledge base in Amazon Bedrock` na política abaixo). Quando você habilita o recurso com o console do Amazon Lex, as políticas são adicionadas automaticamente ao bot, desde que o bot esteja usando uma função vinculada ao serviço gerada pelo 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"
            ]
        }
    ]
}
```

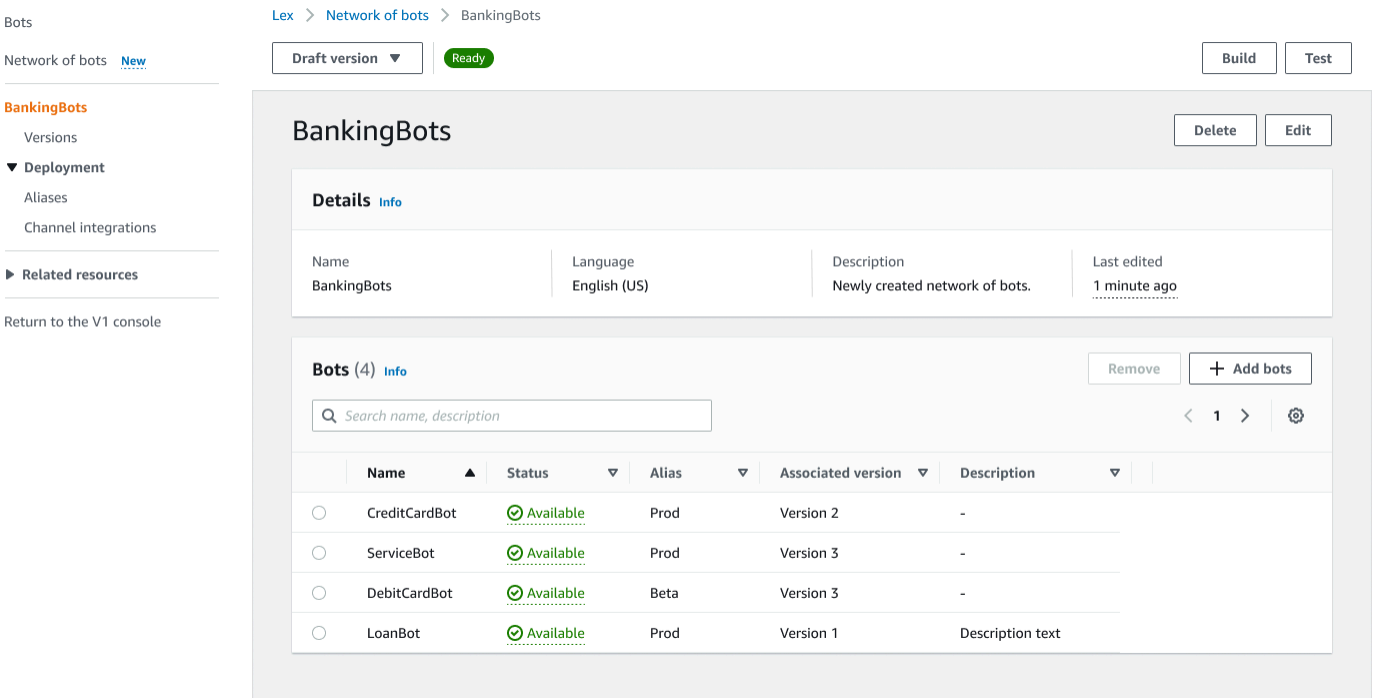
Criar uma rede de bots

Uma rede de bots permite que as empresas ofereçam uma experiência de usuário unificada em vários bots. Com uma rede de bots, as empresas podem adicionar vários bots a uma única rede para permitir o gerenciamento flexível e independente do ciclo de vida dos bots. A rede expõe uma única interface unificada para o usuário final e encaminha a solicitação para o bot apropriado com base na entrada do usuário.

As equipes podem colaborar para criar uma rede de bots para atender a várias necessidades comerciais, mantendo e adicionando bots à rede à medida que bots aprimorados são implantados na produção. Os desenvolvedores podem simplificar e acelerar a implantação e as melhorias integrando vários bots em uma única rede.

A rede de bots está disponível atualmente apenas no idioma en-US.

> **Note**
>
> Atualmente, uma rede de bots está limitada a uma conta. Você não pode adicionar bots de outras contas.



Criar uma rede de bots

Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home. Escolha Rede de bots no menu lateral. Você deve ter construído pelo menos um bot para criar uma rede de bots.

Etapa 1: definir configurações de rede de bots

1. Na seção Detalhes, insira o nome da sua rede e forneça uma descrição opcional.
2. Na seção de permissões do IAM, escolha um perfil do AWS Identity and Access Management (IAM) que forneça permissão ao Amazon Lex V2 para acessar outros serviços da AWS, como o Amazon CloudWatch. Você pode fazer com que o Amazon Lex V2 crie o perfil ou você pode escolher uma função existente com as permissões do CloudWatch. Para obter mais informações, consulte Identity and access management for Amazon Lex V2.
3. Na seção Lei de Proteção à Privacidade Online para Crianças (COPPA), escolha a resposta apropriada. Consulte DataPrivacy para obter mais informações.
4. Na seção Tempo limite da sessão ociosa, escolha por quanto tempo o Amazon Lex V2 mantém uma sessão com um usuário aberto. O Amazon Lex V2 mantém variáveis de sessão durante a sessão para que seu bot possa retomar uma conversa com as mesmas variáveis. Consulte Setting the session timeout para obter mais informações.
5. Na seção Adicionar configurações de idioma, escolha uma voz para seu bot interagir com os usuários. Você pode digitar uma frase na amostra de voz e selecionar Reproduzir para ouvir a voz.
6. Na seção Configurações avançadas, adicione opcionalmente tags que ajudem a identificar o bot. As tags podem ser utilizadas para controlar o acesso e monitorar recursos. Para obter mais informações, consulte Marcação de recursos.
7. Escolha Avançar para criar a rede de bots e passar a adicionar bots.

Etapa 2: Adicionar bots

1. Na seção Bots, selecione + Adicionar bots.
2. Um modal Add bots aparecerá. Escolha um bot para adicionar no menu suspenso Bot e o alias do bot que você deseja usar no menu suspenso Alias.

   O alias deve apontar para uma versão numerada do bot e não para a versão de rascunho. É possível adicionar até 5 bots. Um bot pode ser adicionado a até 25 redes diferentes.

3. Selecione + Adicionar bot para adicionar mais bots à sua rede. Para remover um bot, selecione Remover ao lado do bot que você deseja remover. Quando terminar de adicionar bots, escolha Salvar para fechar o modal.
4. Selecione Salvar para concluir a criação da sua rede.

Gerenciar sua rede de bots

Depois de criar sua rede de bots, você irá para uma página onde poderá gerenciar e construir sua rede. Ou você pode acessar esta página selecionando Rede de bots no menu lateral e escolhendo o nome da rede a ser gerenciada.

1. Para editar as informações da sua rede, selecione Editar acima da seção Detalhes. Para excluir a rede, selecione Excluir acima da seção Detalhes.
2. Na seção Bots, você pode adicionar mais bots selecionando + Adicionar bots. Você também pode adicionar bots se navegar até a página Bots no menu lateral do console do Amazon Lex V2. Ative o botão de rádio ao lado do bot que você deseja adicionar e selecione Adicionar a uma rede de bots no menu suspenso Ações.

   No menu suspenso Rede de bots no modal exibido, escolha a rede à qual você deseja adicionar o bot. Em seguida, escolha o alias do bot que você deseja usar no menu suspenso Alias do bot. Selecione Adicionar para adicionar o bot à rede que você escolheu.
3. Você pode remover bots da sua rede pressionando o botão de rádio ao lado de um bot e escolhendo Remover.
4. Quando terminar de configurar sua rede, selecione Construir no canto superior direito para criar sua rede. Pode levar alguns minutos para que ele seja construído. Se a construção for bem-sucedida, um banner verde de sucesso será exibido na parte superior da página.
5. Depois que a rede for criada, você poderá selecionar Testar no canto superior direito para que uma janela de bate-papo apareça no canto inferior direito. Você pode usar essa janela de chat para conversar com os bots da sua rede e garantir que os fluxos e transições da conversa estejam configurados corretamente.

> **Note**
>
> Se você adicionar, remover ou atualizar bots em sua rede, deverá reconstruí-la.

Versões

Você pode criar versões diferentes da sua rede de bots. Para gerenciar versões, escolha sua rede no menu lateral do console do Amazon Lex V2 e selecione Versões.

1. Selecione Criar versão para criar uma nova versão da sua rede de bots. Você pode adicionar uma descrição opcional. Escolha Criar para criar a versão.
2. Ao alternar o botão de rádio ao lado de uma versão da sua rede de bots, você pode selecionar Associar alias à versão para apontar um alias para essa versão.
3. Para gerenciar uma versão da sua rede, selecione o nome da versão na seção Versões. Na página seguinte, você pode editar detalhes da versão e gerenciar os bots dentro da versão e seu alias associado.

Aliases

Você pode usar aliases para implantar suas redes. Para gerenciar aliases, escolha sua rede no menu lateral do console do Amazon Lex V2 e selecione Aliases.

1. Selecione Criar alias para criar um novo alias.
2. Dê um nome ao alias e uma Descrição opcional na seção Detalhes do alias. Você pode escolher uma versão para associar o alias à seção Associar a uma versão e adicionar tags na seção Tags. Escolha Criar para criar o alias.
3. Para gerenciar um alias para sua rede, selecione o nome do alias na seção Aliases. Na página a seguir, você pode editar detalhes do alias e gerenciar suas tags, integrações de canais e políticas baseadas em recursos. Também é possível visualizar o histórico de sua associação com versões da rede.

Integrações de canais

Para integrar sua rede de bots a uma plataforma de mensagens, escolha sua rede de bots no menu lateral do console do Amazon Lex V2. Em seguida, selecione Integrações de canais.

1. Selecione Adicionar canal para integrar sua rede a um novo canal.
2. Na seção Plataforma, escolha a plataforma na qual você deseja implantar seu bot em Selecionar plataforma. Um perfil do IAM será criado. Escolha uma chave no menu suspenso em Chave KMS para proteger suas informações.

3. No canal de Configuração de integração, insira o Nome e uma Descrição opcional. Escolha um Alias no menu suspenso.
4. Obtenha o SID da sua conta e o token de autenticação da plataforma e preencha os campos SID da conta e Token de autenticação. Consulte Integrating your bots para obter mais informações.
5. Selecione Criar para concluir a integração do canal.

> **Note**
>
> Atualmente, a rede de bots não está disponível na voz ou no chat do Amazon Connect.

Implantação de bots

Depois de criar e testar seu bot, ele estará pronto para ser implantado e interagir com seus clientes. Nesta seção, aprenda a criar versões do seu bot depois de fazer uma atualização. Use aliases para apontar diferentes versões do seu bot quando elas estiverem prontas para implantação. Saiba como integrar seus bots a plataformas de mensagens, aplicativos móveis e sites.

Tópicos

Versionamento e aliases

O Amazon Lex V2 consegue criar versões e aliases de bots e de redes de bots para que você possa controlar a implementação que os aplicativos dos seus clientes usam. Uma versão funciona como um instantâneo numerado do seu trabalho. Você pode apontar um alias para a versão do seu bot que você deseja que esteja disponível para seus clientes. Entre a criação de uma versão e outra, você pode continuar atualizando a versão de `Draft` do seu bot sem afetar a experiência do usuário.

Versões

O Amazon Lex V2 consegue criar versões de bots para que você possa controlar a implementação que os aplicativos dos seus clientes usam. Uma versão é um snapshot numerado do seu trabalho que você pode criar para uso em diferentes partes de seu fluxo de trabalho, como desenvolvimento, implementação beta e produção.

A versão de rascunho

Quando você cria um bot de Amazon Lex V2, há somente uma versão, a `Draft`.

`Draft` é a cópia de trabalho do seu bot. Você pode atualizar apenas a versão `Draft`. Até você criar sua primeira versão, `Draft` será a única versão do bot que você tem.

A versão de `Draft` do seu bot fica associada ao `TestBotAlias`. A `TestBotAlias` só deve ser usada para testes manuais. O Amazon Lex V2 limita o número de solicitações de runtime que você pode fazer para o alias `TestBotAlias` do bot.

Criar uma versão

Quando você versiona um bot do Amazon Lex V2, você cria um snapshot numerado do bot para que possa usar o bot da forma como existia quando a versão foi criada. Após criar uma versão numérica, ela permanecerá a mesma enquanto você continuar trabalhando na versão de rascunho do seu aplicativo.

Quando cria uma versão, você pode escolher os locais a serem incluídos nela. Não é necessário escolher todos os locais em um bot. Além disso, ao criar uma versão, você pode escolher um local de uma versão antiga. Por exemplo, se você tiver três versões de um bot, poderá escolher uma localidade da versão de `Draft` e outra da segunda versão ao criar a quarta versão.

Se você excluir uma localidade da versão de `Draft`, ela não será excluída de uma versão numerada.

Se uma versão do bot não for usada por seis meses, o Amazon Lex V2 a marcará como inativa. Quando uma versão estiver inativa, você não poderá usar operações de runtime com o bot. Para ativar o bot, reconstrua todos os idiomas associados à versão.

Como atualizar um bot do Amazon Lex V2

Você pode atualizar apenas a versão de `Draft` de um bot do Amazon Lex V2. Não é possível alterar as versões. Você pode criar uma outra versão a qualquer momento após atualizar um recurso no console ou com a operação CreateBotVersion.

Como excluir um bot ou uma versão do Amazon Lex V2

O Amazon Lex V2 é compatível com a exclusão de um bot ou versão usando o console ou uma das operações de API:

- DeleteBot
- DeleteBotVersion

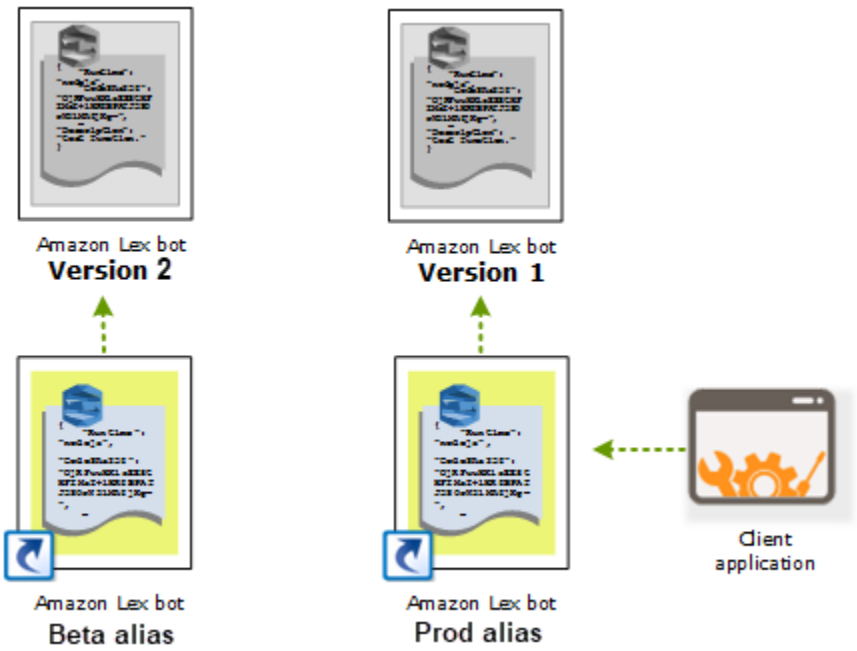
Aliases

Os bots do Amazon Lex V2 também trabalham com aliases. Um alias é um ponteiro para uma versão específica de um bot. Com um alias, você pode atualizar com facilidade a versão que seus aplicativos cliente estão usando. Por exemplo, você pode apontar um alias para a versão 1 do seu bot. Quando estiver pronto para atualizar o bot, você cria a versão 2 e altera o alias para apontar para a nova versão. Como suas aplicações usam o alias ao invés de uma versão específica, todos os seus clientes obtêm a nova funcionalidade sem a necessidade de atualizações.

Um alias é um indicador para uma versão específica de um bot do Amazon Lex V2. Use um alias para permitir que os aplicativos de clientes usem uma versão específica do bot sem exigir que o aplicativo acompanhe qual é a versão.

Quando você cria um bot, o Amazon Lex V2 cria um alias chamado `TestBotAlias`, que você pode usar para testar seu bot. O alias `TestBotAlias` fica sempre associado à versão de `Draft` do bot. Use apenas o alias `TestBotAlias` nos testes, e lembre-se de que o Amazon Lex V2 limita o número de solicitações de runtime que você pode fazer ao alias.

O exemplo a seguir mostra duas versões de um bot do Amazon Lex V2: versão 1 e versão 2. Cada uma dessas versões de bot tem um alias associado, BETA e PROD, respectivamente. As aplicações dos clientes usam o alias PROD para acessar o bot.



Ao criar uma segunda versão do bot, você pode atualizar o alias para apontar para a nova versão do bot usando o console ou a operação UpdateBotAlias. Quando você altera o alias, todas as

aplicações de seus clientes usam a nova versão. Se houver um problema com a nova versão, você poderá reverter para a versão anterior, simplesmente alterando o alias para apontar para essa versão.

Ao configurar seus aplicativos clientes para chamar as APIs Amazon Lex Runtime V2 para permitir que os clientes interajam com seu bot, você usa o alias que aponta a versão que você deseja que seus clientes usem.

> **Note**
>
> Embora você possa testar a versão `Draft` de um bot no console, recomendamos que, ao integrar um bot ao aplicativo do cliente, crie primeiro uma versão e um alias que aponte para essa versão. Use o alias na aplicação de seus clientes pelos motivos explicados nesta seção. Quando você atualiza um alias, o Amazon Lex V2 usa a versão atual para todas as sessões em andamento. Novas sessões usam a nova versão.

Usar um aplicativo Java para interagir com um bot do Amazon Lex V2

O AWS SDK for Java 2.0 fornece uma interface que você pode usar nos aplicativos Java para interagir com os bots. Use o SDK para Java para criar aplicativos cliente para os usuários.

O aplicativo a seguir interage com o bot OrderFlowers que você criou em Exercício 1: criar um bot a partir de um exemplo. Ele usa o `LexRuntimeV2Client` do SDK para Java para chamar a operação RecognizeText para conduzir uma conversa com o bot.

A saída da conversa parece com o exemplo a seguir:

```
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.
```

Para as estruturas JSON enviadas entre o aplicativo cliente e o bot Amazon Lex V2, consulte Exercício 2: revisar o fluxo da conversação.

Para executar o aplicativo de amostra, é necessário fornecer as seguintes informações:

- botId – O identificador atribuído ao bot quando ele foi criado. Você pode ver a ID do bot no console Amazon Lex V2 na página Configurações do bot.
- botAliasId – O identificador atribuído ao alias do bot quando você o criou. Você pode ver a ID do alias do bot no console do Amazon Lex V2 na página Aliases. Se não conseguir ver a ID do alias na lista, selecione o ícone de engrenagem no canto superior direito e ative a ID do alias.
- localeId – O identificador da localidade que você usou para o bot. Para obter uma lista de localidades, consulte Idiomas e locais aceitos pelo Amazon Lex V2.
- accessKey e secretKey – As chaves de autenticação da sua conta. Se não tiver um conjunto de chaves, crie-as usando o console do AWS Identity and Access Management.
- sessionId – Um identificador para a sessão com o bot Amazon Lex V2. Nesse caso, o código usa um UUID aleatório.
- region – Se o bot não estiver na região Leste dos EUA (Norte da Virgínia), lembre-se de alterar a região.

Os aplicativos usam uma função chamada `getRecognizeTextRequest` para criar solicitações individuais para o bot. A função cria uma solicitação com os parâmetros necessários para enviar ao 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;
    }
}
```

Resiliência global

> **Note**
>
> Esse recurso está disponível somente para instâncias Amazon Connect e Amazon Lex V2 criadas nas regiões Leste dos EUA (Norte da Virgínia) e Oeste dos EUA (Oregon).
> Para obter acesso a esse recurso, entre em contato com o arquiteto de soluções ou gerente técnico de contas do Amazon Connect.

A resiliência global permite replicar um bot em uma região secundária. A região secundária pode ser ativada com a replicação automática do bot do usuário nas duas regiões. Você terá uma região de backup no caso de uma interrupção regional. Quando a Resiliência Global está ativa, os novos bots criados são replicados em uma segunda AWS região.

Depois de ativar esse recurso, você pode automatizar a replicação dos bots do Amazon Lex V2 e seus recursos, versões e aliases em uma região pareada AWS quase em tempo real. Com esse recurso, você pode monitorar o número da versão do bot original e da réplica para garantir que a réplica do bot permaneça sincronizada com o bot original. Ao habilitar a replicação, você pode ativar a AWS região predeterminada na qual deseja que o bot seja replicado (as regiões são baseadas em pares predeterminados). Todas as atualizações do bot de origem na região de origem são automaticamente atualizadas para o bot replicado na segunda região.

> **Note**
>
> Quando a Resiliência Global for ativada, somente bots, versões e aliases criados após a ativação do recurso serão replicados na região replicada. Bots, versões e aliases criados anteriormente não estarão presentes na região replicada. A segunda região identificada é somente para leitura e em pares predeterminados. As atualizações do bot são restritas à região em que o bot foi criado inicialmente.

Informações adicionais sobre o uso da Resiliência Global:

- Atualmente, a Resiliência Global só funciona com pares predeterminados de regiões.

us-east-1	us-west-2
eu-west-2	eu-central-1

- Você pode criar uma réplica de qualquer bot do Amazon Lex V2. Você deve criar uma nova versão e um novo alias para o bot depois que a Resiliência Global for ativada.
- Os aliases habilitados na Resiliência Global só podem ser associados às versões habilitadas para Resiliência Global.

Limitações:

- O Global Resiliency não replica bots criados com slots que usam LLM, como CFAQ e Utterance Generation.
- A resiliência global não replica uma rede de bots, mas qualquer bot que faça parte da rede de bots ainda pode ser replicado individualmente.

Tópicos

- Permissões para replicar bots e gerenciar réplicas de bots
- Implantando resiliência global

Permissões para replicar bots e gerenciar réplicas de bots

Se uma função do IAM tiver a AmazonLexFullAccesspolítica anexada, ela poderá criar e gerenciar réplicas de bots.

Se você preferir criar uma função com permissões mínimas para a Resiliência Global, use a política a seguir, que contém as seguintes declarações.

- Permissões para acessar a função vinculada ao serviço Amazon Lex V2 para replicação de bots.
- Permissões para permitir que o Amazon Lex V2 crie uma função vinculada a serviços para replicação de bots em seu nome.
- Permissões para chamar as APIs de replicação de bots.

```
{
    "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/*"
            ]
        }
    ]
}
```

Você pode restringir ainda mais as permissões modificando-as da seguinte maneira.

- Substitua * por IDs específicos de bots ou aliases de bots para limitar as permissões a bots ou aliases de bots específicos.
- Use um subconjunto das `lex BotReplica` ações para restringir a função a ações específicas.

Para ver um exemplo, consulte Permita que os usuários criem e visualizem réplicas de bots, mas não as excluam.

Implantando resiliência global

Painel de informações sobre resiliência global

Você pode acessar as seguintes informações no painel Resiliência global:

- Detalhes da fonte — Informações sobre a região de origem do seu bot, tipo de réplica, data de ativação da replicação e última versão criada. Use essas informações para rastrear as iterações do seu bot.
- Detalhes da replicação — Depois de criar sua réplica de bot, você pode rastrear a região replicada, o tipo de réplica, a data de sincronização da última versão e a última versão replicada. Use essas informações para rastrear a sincronização da réplica do seu bot.
- Região de origem — A região em que a resiliência global está ativada. Você pode fazer alterações na região de origem para replicar o bot nas duas regiões.
- Tipo de réplica — Indica se o bot é somente de leitura ou é capaz de ler e gravar com base na região.
- Região de réplica — A região secundária usada para replicar seu bot de origem para resiliência global. Atualmente, a Resiliência Global funciona apenas com pares regionais IAD/PDX e LDN/FRA.
- Data de ativação da replicação — A data e a hora em que a réplica do bot foi ativada.
- Última versão criada — A última versão do bot associada à réplica na região de origem.

Possibilitando a resiliência global

> **Note**
>
> Esse recurso está disponível somente para instâncias Amazon Connect e Amazon Lex V2 criadas nas regiões Leste dos EUA (Norte da Virgínia) e Oeste dos EUA (Oregon).
> Para obter acesso a esse recurso, entre em contato com o arquiteto de soluções ou gerente técnico de contas do Amazon Connect.

Antes de ativar a resiliência global no console do Amazon Lex V2, você deve garantir que o usuário que habilita a replicação de bots tenha permissão para criar funções vinculadas ao serviço (SLR). A Global Resiliency usará essas credenciais do FAS para criar uma SLR na conta habilitada quando for

CreateReplica invocada. Para obter mais informações sobre como configurar a SLR para resiliência global no Amazon Lex V2, consulte a política gerenciada da AWS:. AmazonLexFullAccess

Ative a resiliência global e configure a replicação de bots para uma segunda região:

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot que você deseja replicar na navegação de bots no painel de navegação do lado esquerdo.
3. Escolha Implantação > Resiliência global.
4. Selecione o botão Criar réplica no canto superior direito da janela para criar uma versão de rascunho do seu bot.

   > **Note**
   >
   > Verifique se você não tem nenhum bot na região secundária com o mesmo nome do bot que você deseja replicar. (Seu bot deve ter um nome exclusivo).

5. Vá para Resiliência global e clique em Criar réplica - Essa ação cria uma versão preliminar do seu bot. (você não precisa voltar para a guia Resiliência Global, exceto para revisar o status ou ver detalhes de futuras compilações).

   > **Note**
   >
   > Você também pode criar um bot de aliases para replicação no Global Resiliency acessando Alias e selecionando Create New Alias for Global Resiliency enabled bot. Somente os aliases criados após a ativação da replicação serão replicados.

6. Vá para Alias - Crie um novo alias para o bot habilitado para resiliência global. Somente os aliases criados após a ativação da replicação serão replicados.
7. Vá para Versão - Crie uma nova versão para o bot habilitado para resiliência global. Somente as versões criadas após a ativação da replicação serão replicadas.

> **Note**
>
> Os clientes ainda têm controle total do gerenciamento de suas políticas e tags baseadas em recursos para bots replicados. As funções Lambda e CloudWatch os grupos de registros

precisarão ser implantados nas duas regiões com os mesmos identificadores. Os usuários não precisarão associar a função lambda novamente na região da réplica.

Desativando a resiliência global

Você pode desativar a resiliência global a qualquer momento selecionando o botão Desativar resiliência global. Essa ação impede que seu bot de origem e quaisquer aliases e versões associados a ele sejam replicados em outras regiões.

Usando APIs com resiliência global

Você pode fazer chamadas de API no Global Resiliency usando as seguintes APIs. Informações adicionais sobre as APIs de resiliência global e o Amazon Lex V2 podem ser encontradas no Guia de API do Amazon Lex V2.

- CreateBotReplica

  Ative a resiliência global e crie um bot replicado. Requer `replicaRegion`.

  Para obter mais informações, consulte CreateBotReplicao Guia da API Lex.
- DeleteBotReplica

  Desative a resiliência global e exclua o bot replicado. Requer `replicaRegion` e `botId`.

  Para obter mais informações, consulte DeleteBotReplicao Guia da API Lex.
- ListBotReplicas

  Liste os bots replicados na zona secundária. Requer `botId`.

  Para obter mais informações, consulte ListBotReplicaso Guia da API Lex.
- DescribeBotReplica

  Resumo das informações do bot replicado. Requer `replicaRegion` e `botId`.

  Para obter mais informações, consulte DescribeBotReplicao Guia da API Lex.

Integrar um bot do Amazon Lex V2 a uma plataforma de mensagens

Esta seção explica como integrar bots do Amazon Lex V2 nas plataformas de sistema de mensagens do Facebook, Slack e Twilio. Se você ainda não tem um bot do Amazon Lex V2, crie um. Neste tópico, consideramos que você está usando o bot que criou em Exercício 1: criar um bot a partir de um exemplo. No entanto, você pode usar qualquer bot.

> **Note**
>
> Ao armazenar suas configurações do Facebook, Slack ou Twilio, o Amazon Lex V2 usa um para criptografar informações. AWS KMS key Na primeira vez que você cria um canal para uma dessas plataformas de mensagens, o Amazon Lex V2 cria uma chave padrão gerenciada pelo cliente (`aws/lex`) em sua AWS conta ou você pode selecionar sua própria chave gerenciada pelo cliente. O Amazon Lex V2 só aceita chaves simétricas. Para obter mais informações, consulte o Guia do desenvolvedor do AWS Key Management Service.

Quando uma plataforma de sistema de mensagens envia uma solicitação ao Amazon Lex V2, informações específicas à plataforma são incluídas como um atributo de solicitação para a função do Lambda. Use esse atributo para personalizar o comportamento do seu bot. Para ter mais informações, consulte Definição de atributos de solicitação.

Atributo de solicitação comum

Atributo	Descrição
x-amz-lex:canais:plataforma	Um dos seguintes valores: • `Facebook` • `Slack` • `Twilio`

Integrar um bot Amazon Lex V2 ao Facebook Messenger

Você pode fazer o host do bot Amazon Lex V2 no Facebook Messenger. Ao fazer isso, os usuários do Facebook podem interagir com o bot para atender às intenções.

Antes de começar, você precisa criar uma conta de desenvolvedor do Facebook em https://developers.facebook.com.

É necessário executar as seguintes etapas:

Tópicos

Etapa 1: crie um aplicativo do Facebook

No portal de desenvolvedor do Facebook, crie um aplicativo do Facebook e uma página do Facebook.

Como criar um aplicativo do Facebook

1. Abra https://developers.facebook.com/apps
2. Escolha Criar aplicativo.
3. Na página Criar um aplicativo, selecione Negócios e Avançar.
4. Nos campos Adicionar o nome do aplicativo, E-mail de contato do aplicativo e Conta comercial, faça as escolhas apropriadas para o aplicativo. Selecione Criar aplicativo para continuar.
5. Em Adicionar produtos ao aplicativo, selecione Configurar no bloco do Messenger.
6. Na seção Tokens de acesso, selecione Adicionar ou remover páginas.
7. Escolha uma página para usar com o aplicativo e selecione Avançar.
8. Em O que o aplicativo está autorizado a fazer, mantenha os padrões e selecione Concluído.
9. Na página de confirmação, escolha OK.
10. Na seção Tokens de acesso, selecione Gerar token e, em seguida, copie o token. Insira esse token no console do Amazon Lex V2.
11. No menu, selecione Configurações e Básico.
12. Em Segredo do aplicativo, selecione Mostrar e copie o segredo. Insira esse token no console do Amazon Lex V2.

Próxima etapa

Etapa 2: Integre o Facebook Messenger ao bot Amazon Lex V2

Etapa 2: Integre o Facebook Messenger ao bot Amazon Lex V2

Nesta etapa, você vincula o bot Amazon Lex V2 ao Facebook.

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot Amazon Lex V2 que você criou.
3. No menu à esquerda, selecione Integrações de canais e, em seguida, selecione Adicionar canal.
4. Em Criar canal, faça o seguinte:
    a. Em Plataforma, selecione Facebook.
    b. Em políticas de identidade, selecione a chave AWS KMS para proteger as informações do canal. A chave padrão é fornecida pelo Amazon Lex V2.
    c. Em Configuração de integração, dê ao canal um nome e uma descrição opcional. Selecione o alias que aponta para a versão do bot a ser usada e escolha o idioma compatível com o canal.
    d. Em Configuração adicional, faça o seguinte:
        - Alias – Uma string que identifica o aplicativo que está chamando o Amazon Lex V2. Você pode usar qualquer string. Registre essa string, você a insere no console do desenvolvedor do Facebook.
        - Token de acesso à página – O token de acesso à página que você copiou do console do desenvolvedor do Facebook.
        - Chave secreta do aplicativo – A chave secreta que você copiou do console do desenvolvedor do Facebook.
    e. Selecione Criar
    f. O Amazon Lex V2 mostra a lista de canais do bot. Na lista, escolha o canal que você acabou de criar.
    g. Em URL de retorno de chamada, registre a URL de retorno de chamada. Insira essa URL no console do desenvolvedor do Facebook.

Próxima etapa

Etapa 3: Complete a integração do Facebook

Etapa 3: Complete a integração do Facebook

Nesta etapa, use o console do desenvolvedor do Facebook para concluir a integração com o Amazon Lex V2.

Como concluir a integração com o Facebook Messenger

1. Abra https://developers.facebook.com/apps
2. Na lista de aplicativos, selecione o aplicativo que você está integrando com o Facebook Messenger.
3. No menu à esquerda, selecione Messenger e Configurações.
4. Na seção Webhooks:
    a. Selecione Adicionar URL de retorno de chamada.
    b. Em Editar URL de retorno de chamada, digite o seguinte:
        - URL de retorno de chamada – Insira a URL de retorno de chamada que você registrou no console do Amazon Lex V2.
        - Verificar token – Insira o alias que você inseriu no console do Amazon Lex V2.
    c. Escolha Verificar e salvar.
    d. Selecione Adicionar assinaturas em Webhooks ao lado da página.
    e. Na janela que aparece, escolha `messages` e clique em Salvar.

Próxima etapa

Etapa 4: Teste a integração

Etapa 4: Teste a integração

Agora, você pode começar uma conversa no Facebook Messenger com o bot do Amazon Lex V2.

Como testar a integração entre o Facebook Messenger e um bot Amazon Lex V2

1. Abra a página do Facebook que você associou ao bot na etapa 1.

2. Na janela do Messenger, use os mesmos enunciados de teste fornecidos em Exercício 1: criar um bot a partir de um exemplo.

Integrar um bot de Amazon Lex V2 com o Slack

Este tópico fornece instruções para integrar um bot Amazon Lex V2 com o aplicativo de sistema de mensagens Slack. Execute as seguintes etapas:

Tópicos

Etapa 1: Cadastre-se no Slack e crie uma equipe do Slack

Cadastre-se em uma conta do Slack e crie uma equipe do Slack. Para obter instruções, consulte Uso do Slack. Na próxima seção, você criará uma aplicação do Slack que qualquer equipe do Slack pode instalar.

Próxima etapa

Etapa 2: Crie um aplicativo do Slack

Etapa 2: Crie um aplicativo do Slack

Nesta seção, faça o seguinte:

1. Crie um aplicativo do Slack no console da API do Slack.
2. Configure o aplicativo para adicionar mensagens interativas ao bot.

No final desta seção, você obterá as credenciais do aplicativo (ID do cliente, segredo do cliente e token de verificação). Na próxima etapa, você usará essas informações para integrar o bot no console do Amazon Lex V2.

Como criar um aplicativo do Slack

1. Faça login no console da API do Slack em https://api.slack.com API .
2. Crie um aplicativo.

   Depois de que você cria o aplicativo com êxito, o Slack exibe a página Informações básicas do aplicativo.
3. Configure os atributos do aplicativo da seguinte forma:

   - No menu à esquerda, escolha Interatividade e atalhos.
     - Selecione o seletor para ativar os componentes interativos.
     - Na caixa URL da solicitação especifique qualquer URL válida. Por exemplo, você pode usar o **`https://slack.com`**.

       > **Note**
       >
       > Por enquanto, insira qualquer URL válida para obter o token de verificação necessário na próxima etapa. Você atualizará essa URL depois de adicionar a associação de canal do bot no console do Amazon Lex.

     - Escolha Salvar alterações.
4. No menu esquerdo, em Configurações, escolha Informações básicas. Registre as seguintes credenciais do aplicativo:

   - ID do cliente
   - Segredo do cliente
   - Token de verificação

Próxima etapa

Etapa 3: Integre o aplicativo do Slack com o bot Amazon Lex V2

Etapa 3: Integre o aplicativo do Slack com o bot Amazon Lex V2

Nesta seção, integre o aplicativo do Slack que você criou com o bot Amazon Lex V2 criado com integrações de canais.

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot Amazon Lex V2 que você criou.
3. No menu à esquerda, selecione Integrações de canais e, em seguida, selecione Adicionar canal.
4. Em Criar canal, faça o seguinte:

   a. Em Plataforma, escolha Slack.

   b. Em políticas de identidade, selecione a chave AWS KMS para proteger as informações do canal. A chave padrão é fornecida pelo Amazon Lex V2.

   c. Em Configuração de integração, dê ao canal um nome e uma descrição opcional. Selecione o alias que aponta para a versão do bot a ser usada e escolha o idioma compatível com o canal.

   > **Note**
   >
   > Se o bot estiver disponível em vários idiomas, será necessário criar um canal diferente e um aplicativo diferente para cada idioma.

   d. Em Configuração adicional, faça o seguinte:

   - ID do cliente – insira a ID do cliente do Slack.
   - Segredo do cliente – insira o segredo do cliente no Slack.
   - Token de verificação – insira o token de verificação do Slack.
   - URL da página de êxito – a URL da página que o Slack deve abrir quando o usuário é autenticado. Normalmente, você deixa este campo em branco.

5. Escolha Criar para criar o canal.
6. O Amazon Lex V2 mostra a lista de canais do bot. Na lista, escolha o canal que você acabou de criar.
7. Em URL de retorno de chamada, registre o endpoint e o endpoint OAuth.

Próxima etapa

Etapa 4: Complete a integração do Slack

Etapa 4: Complete a integração do Slack

Nesta seção, use o console da API do Slack para completar a integração com o aplicativo do Slack.

1. Faça login no console da API do Slack em https://api.slack.com API. Selecione o aplicativo que você criou em Etapa 2: Crie um aplicativo do Slack.
2. Atualize o atributo OAuth e permissões da seguinte forma:
   a. No menu esquerdo, selecione OAuth e permissões.
   b. Na seção URLs de redirecionamento, adicione o endpoint do OAuth fornecido pelo Amazon Lex na etapa anterior. Selecione Adicionar e Salvar URLs.
   c. Na seção Escopos do token do bot, adicione duas permissões com o botão Adicionar um escopo do OAuth. Filtre a lista com o seguinte texto:
      - **`chat:write`**
      - **`team:read`**
3. Atualize o atributo Interatividade e atalhos atualizando o valor URL da solicitação para o endpoint que o Amazon Lex forneceu na etapa anterior. Insira o endpoint que você salvou na etapa 3 e selecione Salvar alterações.
4. Assine o atributo Assinaturas de Eventos da seguinte forma:
   - Ative eventos escolhendo a opção Ativar.
   - Defina o valor URL da solicitação para o endpoint que o Amazon Lex forneceu na etapa anterior.
   - Na seção Assinar eventos do bot, selecione Adicionar evento de usuário do bot e adicione o evento do bot **`message.im`** para habilitar mensagens diretas entre o usuário final e o bot do Slack.
   - Salve as alterações.
5. Ative o envio de mensagens na guia de mensagens da seguinte forma:
   - No menu à esquerda, selecione Página inicial do aplicativo.
   - Na seção Mostrar guias, escolha Permitir que os usuários enviem comandos e mensagens com barras na guia de mensagens.
6. Escolha Gerenciar distribuição em Configurações. Escolha Adicionar ao Slack para instalar o aplicativo. Se você estiver autenticado em vários espaços de trabalho, primeiro escolha o

espaço de trabalho correto no canto superior direito da lista suspensa. Em seguida, selecione Permitir para autorizar o bot a responder às mensagens.

> **Note**
>
> Se você fizer alguma alteração nas configurações do aplicativo do Slack posteriormente, será necessário refazer essa subetapa.

Próxima etapa

Etapa 5: Teste a integração

Etapa 5: Teste a integração

Agora, use uma janela do navegador para testar a integração do Slack com o bot do Amazon Lex V2.

Como testar o aplicativo do Slack

1. Inicie o Slack. No menu esquerdo, na seção Mensagens diretas, selecione o bot. Se você não vir o bot, escolha o ícone de mais (+) ao lado de Mensagens diretas para procurá-lo.
2. Participe de um bate-papo com o aplicativo do Slack. O bot responde a mensagens.

   Se você criou o bot usando Exercício 1: criar um bot a partir de um exemplo, poderá usar os exemplos de conversas nesse exercício.

Integração de um bot do Amazon Lex V2 com o Twilio SMS

Este tópico fornece instruções para integrar um bot do Amazon Lex V2 com o serviço de mensagens simples (SMS) do Twilio. Execute as seguintes etapas:

Tópicos

Etapa 1: Crie uma conta de SMS do Twilio

Cadastre-se para criar uma conta do Twilio e registre as seguintes informações de conta:

- SID DA CONTA
- TOKEN DE AUTENTICAÇÃO

Para obter instruções de cadastro, consulte https://www.twilio.com/console.

Próxima etapa

Etapa 2: Integre o endpoint de serviço de mensagens do Twilio com o bot do Amazon Lex V2

Etapa 2: Integre o endpoint de serviço de mensagens do Twilio com o bot do Amazon Lex V2

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot Amazon Lex V2 que você criou.
3. No menu à esquerda, selecione Integrações de canais e, em seguida, selecione Adicionar canal.
4. Em Criar canal, faça o seguinte:
    a. Em Plataforma, escolha Twilio.
    b. Em políticas de identidade, selecione a chave AWS KMS para proteger as informações do canal. A chave padrão é fornecida pelo Amazon Lex V2.
    c. Em Configuração de integração, dê ao canal um nome e uma descrição opcional. Selecione o alias que aponta para a versão do bot a ser usada e escolha o idioma compatível com o canal.
    d. Em Configuração adicional, insira o SID da conta e o token de autenticação no painel do Twilio.
5. Selecione Criar.
6. Na lista de canais, escolha o canal que você acabou de criar.
7. Copie a URL de retorno de chamada.

Próxima etapa

Etapa 3: Conclua a integração com o Twilio

Etapa 3: Conclua a integração com o Twilio

Use o console do Twilio para concluir a integração do bot do Amazon Lex V2 com o SMS do Twilio.

1. Abra o console do Twilio em https://www.twilio.com/console.
2. No menu à esquerda, escolha Todos os produtos e serviços e selecione Número de telefone.
3. Se você tiver um número de telefone, selecione-o. Se você não tiver um número de telefone, selecione Comprar um número para obter um.
4. Na seção Mensagens, em CHEGOU UMA MENSAGEM, insira a URL de retorno de chamada do console do Amazon Lex V2.
5. Selecione Salvar.

Próxima etapa

Etapa 4: Teste a integração

Etapa 4: Teste a integração

Use seu celular para testar a integração entre o SMS do Twilio e o bot. Usando seu celular, envie mensagens para o número do Twilio.

Se você criou o bot usando Exercício 1: criar um bot a partir de um exemplo, poderá usar os exemplos de conversas nesse exercício.

Integrar um bot Amazon Lex V2 a uma central de atendimento

Você pode integrar bots do Amazon Lex V2 às centrais de atendimento para permitir casos de uso de autoatendimento usando a API de streaming Amazon Lex V2. Use esses bots como agentes de resposta de voz interativa (IVR) na telefonia ou como um chatbot baseado em texto integrado à sua central de atendimento. Para mais informações sobre as APIs de streaming, consulte Transmissão para um bot do Amazon Lex V2.

Com as APIs de streaming, você pode ativar os seguintes atributos:

- Interrupções (“barge-in”): os chamadores interrompem o bot e respondem a uma pergunta antes que a solicitação seja concluída. Para mais informações, consulte Permitir que seu bot seja interrompido pelo usuário.

- Esperar e continuar: os chamadores instruem o bot a esperar se precisarem de tempo para trazer mais informações durante uma chamada, como número de cartão de crédito ou ID de reserva. Para ter mais informações, consulte Permitir que o bot espere que o usuário forneça mais informações.
- Suporte DTMF: os chamadores fornecem informações por voz ou DTMF de forma intercambiável.
- Suporte a SSML: você pode configurar os prompts do bot do Amazon Lex V2 usando tags SSML para ter mais controle sobre geração de fala a partir de texto. Para mais informações, consulte Como gerar fala a partir de documentos SSML no Guia do desenvolvedor do Amazon Polly.
- Tempos limite configuráveis: você pode configurar quanto tempo esperar até que os clientes terminem de falar antes que o Amazon Lex V2 colete as entradas de fala, como uma resposta a uma pergunta de sim ou não ou, ou uma data ou número de cartão de crédito informado. Para ter mais informações, consulte Configurar tempos limite para capturar a entrada do usuário.
- Atualizações do progresso do atendimento: você pode configurar o bot para responder com várias mensagens com base no status do atendimento durante a execução da lógica de negócios para o atendimento do intent. Você pode configurar o bot para responder com mensagens quando o atendimento começar e terminar, além de fornecer atualizações periódicas para as funções do Lambda de longa duração. Para ter mais informações, consulte Configurar atualizações do progresso do atendimento.

SDK do Amazon Chime

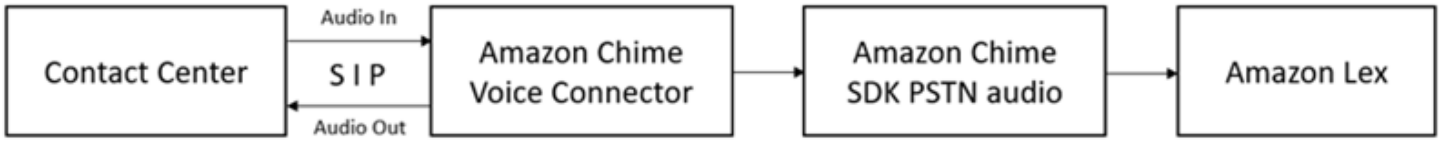
Use o SDK do Amazon Chime para adicionar recursos de áudio, vídeo, compartilhamento de tela e mensagens em tempo real aos seus aplicativos web ou móveis. O SDK do Amazon Chime fornece serviço de áudio de rede telefônica pública comutada (PSTN) para que você possa criar aplicativos de telefonia personalizados com uma função do AWS Lambda.

O áudio de PSTN do Amazon Chime é integrado ao Amazon Lex V2. Você pode usar essa integração para acessar os bots do Amazon Lex V2 como sistemas interativos de resposta de voz (IVR) em centrais de atendimento para interações de áudio. Use isso para integrar o Amazon Lex V2 usando serviços de áudio de PSTN nos cenários a seguir.

Integrações de central de atendimento — Você pode usar o Amazon Chime Voice Connector e o serviço de áudio de PSTN do SDK do Amazon Chime para acessar os bots do Amazon Lex V2. Use-os em qualquer aplicativo de central de atendimento que use o protocolo de iniciação de sessão (SIP) para comunicações de voz. Essa integração adiciona experiências de conversação por voz em linguagem natural à central de atendimento existente on-premises ou na nuvem com suporte SIP.

Para obter uma lista das plataformas de central de atendimento compatíveis, consulte os recursos do Amazon Chime Voice Connector.

O diagrama a seguir mostra a integração entre uma central de atendimento que usa o SIP e o Amazon Lex V2.



Suporte direto à telefonia — Você pode criar soluções de IVR personalizadas para acessar diretamente os bots do Amazon Lex V2 usando um número de telefone provisionado no SDK do Amazon Chime.

Para obter mais informações, consulte um dos tópicos a seguir no Guia do SDK do Amazon Chime.

- Integração SIP usando um Amazon Chime Voice Connector
- Como usar o serviço de áudio PSTN do SDK do Amazon Chime
- Integração do áudio PSTN do Amazon Chime com o Amazon Lex V2

Quando o SDK do Amazon Chime envia uma solicitação para o Amazon Lex V2, ele inclui informações específicas da plataforma para a função do Lambda e seus logs de conversação. Use essas informações para determinar o aplicativo de central de atendimento que está enviando tráfego para o bot.

Atributo de solicitação comum	Valor
x-amz-lex:channels:platform	`Amazon Chime SDK PSTN Audio`

Amazon Connect

O Amazon Connect é uma central de atendimento em nuvem omnicanal. Você pode configurar um centro de contatos em algumas etapas, adicionar agentes de qualquer lugar e começar a interagir com seus clientes. Para mais informações, consulte Começar a usar o Amazon Connect no Guia do administrador do Amazon Connect.

Você pode criar experiências personalizadas para seus clientes usando comunicações omnichannel. Por exemplo, você pode oferecer contato por chat e voz com base na preferência do cliente e nos

tempos de espera estimados. Além disso, os agentes conseguem processar todos os clientes de uma única interface. Por exemplo, eles podem conversar com clientes e criar ou responder às tarefas à medida que são encaminhadas para eles.

Você pode usar o Amazon Connect para interações de áudio com seus clientes ou o Amazon Connect Chat para interações somente de texto.

Para mais informações, consulte os seguintes tópicos no Guia do administrador do Amazon Connect.

- O que é o Amazon Connect
- Adicionar um bot do Amazon Lex V2
- Amazon Connect obtém o bloco de contato de entrada do cliente

Quando uma central de atendimento envia uma solicitação ao Amazon Lex V2, ela inclui informações específicas sobre a plataforma como um atributo de solicitação para a função do Lambda e os logs de conversação. Use essas informações para determinar qual aplicativo de central de atendimento está enviando tráfego para seu bot.

Atributo de solicitação comum

Atributo	Valor
x-amz-lex:canais:plataforma	Um dos seguintes valores: • `Connect` • `Connect Chat`

Genesys Cloud

O Genesys Cloud é um pacote de serviços na nuvem para comunicação corporativa, colaboração e gerenciamento de centrais de atendimento. O Genesys Cloud é construído sobre AWS e usa um ambiente de nuvem distribuído que fornece acesso seguro às organizações em todo o trabalho.

Para mais informações, consulte as seguintes páginas no site da Genesys Cloud.

- Sobre a central de atendimento da Genesys Cloud
- Sobre a integração do Amazon Lex V2

Quando uma central de atendimento envia uma solicitação ao Amazon Lex V2, ela inclui informações específicas sobre a plataforma como um atributo de solicitação para a função do Lambda e os logs de conversação. Use essas informações para determinar qual aplicativo de central de atendimento está enviando tráfego para seu bot.

Atributo de solicitação comum

Atributo	Valor
x-amz-lex:canais:plataforma	• `Genesys Cloud`

Saiba mais

- Potencialize sua central de atendimento com o Amazon Lex e o Genesys Cloud

Gerenciar conversas

Depois de criar um bot, você integra sua aplicação cliente às operações de runtime do Amazon Lex V2 para manter conversas com seu bot.

Quando um usuário inicia uma conversa com seu bot, o Amazon Lex V2 cria uma sessão. Uma sessão encapsula as informações trocadas entre sua aplicação e o bot. Para mais informações, consulte Gerenciamento de sessões com a API Amazon Lex V2.

Uma conversa típica invoca um fluxo de ida e volta entre o usuário e um bot. Por exemplo:

```
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.
```

Ao usar a operação RecognizeText ou RecognizeUtterance, você deve gerenciar a conversa em sua aplicação cliente. Quando você usa a operação StartConversation, o Amazon Lex V2 gerencia a conversa para você.

Para gerenciar a conversa, você deve enviar os enunciados do usuário ao bot até que a conversa chegue a um fim lógico. A conversa atual é capturada no estado da sessão. O estado da sessão é atualizado após cada enunciado do usuário. O estado da sessão contém o estado atual da conversa e é retornado pelo bot em uma resposta a cada enunciado do usuário.

Uma conversa pode estar em qualquer um dos seguintes estados:

- ElicitIntent: indica que o bot ainda não determinou a intenção do usuário.
- ElicitSlot: indica que o bot detectou a intenção do usuário e está reunindo as informações necessárias para cumprir a intenção.
- ConfirmIntent: indica que o bot está esperando que o usuário confirme que as informações coletadas estão corretas.

- Fechado: indica que a intenção do usuário está completa e que a conversa com o bot chegou a um fim lógico.

Um usuário pode especificar uma nova intenção após a conclusão da primeira intenção. Para mais informações, consulte Gerenciar contexto da conversa.

Uma intenção pode ter os seguintes estados:

- InProgress: indica que o bot está coletando as informações necessárias para concluir a intenção. Isso ocorre em conjunto com o estado `ElicitSlot` da conversa.
- Waiting: indica que o usuário solicitou que o bot esperasse quando o bot solicitou informações sobre um slot específico.
- Fulfilled: indica que a lógica de negócios em uma função do Lambda associada à intenção foi executada com sucesso.
- ReadyForFulfillment: indica que o bot reuniu todas as informações necessárias para cumprir a intenção e que a aplicação cliente pode executar a lógica comercial de atendimento.
- Failed: indica falha em uma intenção.

Veja os tópicos a seguir para aprender a usar as APIs do Amazon Lex V2 para gerenciar o contexto de conversas e as sessões entre seu bot e os usuários.

Tópicos

- Gerenciar contexto da conversa
- Gerenciamento de sessões com a API Amazon Lex V2

Gerenciar contexto da conversa

Contexto de conversa são as informações que um usuário, sua aplicação ou uma função do Lambda fornece a um bot do Amazon Lex para atender a uma intenção. O contexto da conversa inclui dados de slot que o usuário fornece, atributos de solicitação definidos pela aplicação cliente e atributos de sessão que a aplicação cliente e as funções do Lambda criam.

Tópicos

- Definir o contexto da intenção
- Usar valores de slot padrão

- Definição dos atributos da sessão
- Definição de atributos de solicitação
- Definição do tempo limite da sessão
- Compartilhamento de informações entre intenções
- Configuração de atributos complexos

Definir o contexto da intenção

Você pode fazer com que o Amazon Lex acione intenções com base no contexto. Um contexto é uma variável de estado que pode ser associada a uma intenção quando você define um bot. Você configura os contextos de uma intenção ao criar a intenção usando o console ou usando a operação CreateIntent. Você só pode usar o contexto na localidade em inglês (EUA) (en-US).

Existem dois tipos de relacionamentos para contextos: contextos de saída e contextos de entrada. Um contexto de saída se torna ativo quando uma intenção associada é cumprida. Um contexto de saída é retornado à sua aplicação na resposta da operação RecognizeText ou RecognizeUtterance e é definido para a sessão atual. Depois que um contexto é ativado, ele permanece ativo pelo número de turnos ou limite de tempo configurado quando o contexto foi definido.

Um contexto de entrada especifica as condições sob as quais uma intenção pode ser reconhecida. Uma intenção só pode ser reconhecida durante uma conversa quando todos os contextos de entrada estão ativos. Uma intenção sem contextos de entrada é sempre elegível para reconhecimento.

O Amazon Lex gerencia automaticamente o ciclo de vida dos contextos que são ativados ao cumprir as intenções com contextos de saída. Você também pode definir contextos ativos em uma chamada para a operação `RecognizeText` ou `RecognizeUtterance`.

Também é possível definir o contexto de uma conversa usando a função do Lambda para a intenção. O contexto de saída do Amazon Lex é enviado para o evento de entrada da função do Lambda. A função do Lambda pode enviar contextos em sua resposta. Para mais informações, consulte Habilitando a lógica personalizada com as funções do AWS Lambda.

Por exemplo, suponha que você tenha a intenção de reservar um carro alugado configurado para retornar um contexto de saída chamado “book_car_filled”. Quando a intenção é cumprida, o Amazon Lex define a variável de contexto de saída “book_car_filled”. Como “book_car_filled” é um contexto ativo, uma intenção com o contexto “book_car_filled” definido como um contexto de entrada agora é considerada para reconhecimento, desde que o enunciado do usuário seja reconhecido como uma

tentativa de obter essa intenção. Você pode usar isso para propósitos que só façam sentido depois de reservar um carro, como enviar um recibo por e-mail ou modificar uma reserva.

Contexto de saída

O Amazon Lex ativa os contextos de saída de uma intenção quando a intenção é cumprida. Você pode usar o contexto de saída para controlar as intenções elegíveis para acompanhar a intenção atual.

Cada contexto tem uma lista de parâmetros que são mantidos na sessão. Os parâmetros são os valores do slot para a intenção cumprida. Você pode usar esses parâmetros para preencher previamente os valores dos slots para outras finalidades. Para obter mais informações, consulte Usar valores de slot padrão.

Você configura o contexto de saída ao criar uma intenção com o console ou com a operação CreateIntent. Você pode configurar uma intenção com mais de um contexto de saída. Quando a intenção é cumprida, todos os contextos de saída são ativados e retornados na resposta RecognizeText ou RecognizeUtterance.

Ao definir um contexto de saída, você também define sua vida útil, a duração ou o número de turnos no qual o contexto é incluído nas respostas do Amazon Lex. Um turno é uma solicitação de sua aplicação para o Amazon Lex. Depois que o número de turnos ou o tempo expirar, o contexto não estará mais ativo.

Sua aplicação pode usar o contexto de saída, conforme necessário. Por exemplo, sua aplicação pode usar o contexto de saída para:

- Alterar o comportamento da aplicação com base no contexto. Por exemplo, uma aplicação de viagens pode ter uma ação diferente para o contexto “book_car_filled” que para “rental_hotel_filled”.
- Retorne o contexto de saída para o Amazon Lex como contexto de entrada para o próximo enunciado. Se o Amazon Lex reconhecer o enunciado como uma tentativa de extrair uma intenção, ele usa o contexto para limitar as intenções que podem ser retornadas àquelas com o contexto especificado.

Contexto de entrada

Você define um contexto de entrada para limitar os pontos da conversa em que a intenção é reconhecida. Intenções sem um contexto de entrada são sempre elegíveis para serem reconhecidas.

Você define os contextos de entrada aos quais uma intenção responde usando o console ou a operação `CreateIntent`. Uma intenção pode ter mais de um contexto de entrada.

Para uma intenção com mais de um contexto de entrada, todos os contextos devem estar ativos para acionar a intenção. Você pode definir um contexto de entrada ao chamar a operação RecognizeText, RecognizeUtterance ou PutSession.

Você pode configurar os slots em uma intenção para obter valores padrão do contexto ativo atual. Os valores padrão são usados quando o Amazon Lex reconhece uma nova intenção, mas não recebe um valor de slot. Você especifica o nome do contexto e o nome do slot no formulário `#context-name.parameter-name` ao definir o slot. Para mais informações, consulte Usar valores de slot padrão.

Usar valores de slot padrão

Ao usar um valor padrão, você especifica uma fonte para um valor de slot a ser preenchido para novas intenções quando nenhum slot é fornecido pela entrada do usuário. Essa fonte pode ser uma caixa de diálogo anterior, atributos de solicitação ou sessão, ou um valor fixo que você define no momento da criação.

É possível usar o seguinte como fonte para seus valores padrão.

- Caixa de diálogo anterior (contextos): #context-name.parameter-name
- Atributos da sessão: [attribute-name]
- Atributos da solicitação: <attribute-name>
- Valor fixo: qualquer valor que não corresponda ao anterior

Ao usar a operação CreateIntent para adicionar slots a uma intenção, você pode adicionar uma lista de valores padrão. Os valores padrão são usados na ordem em que estão listados. Por exemplo, suponha que você tenha uma intenção com um slot com a seguinte definição:

```
"slots": [
    {
        "botId": "string",
        "defaultValueSpec": {
            "defaultValueList": [
                {
                    "defaultValue": "#book-car-fulfilled.startDate"
                },
```

```
                {
                    "defaultValue": "[reservationStartDate]"
                }
            ]
        },
        Other slot configuration settings
    }
]
```

Quando a intenção é reconhecida, o slot chamado “reservation-start-date” tem seu valor definido como um dos seguintes:

1. Se o contexto “book-car-filled” estiver ativo, o valor do parâmetro “startDate” será usado como valor padrão.
2. Se o contexto “book-car-filled” não estiver ativo ou se o parâmetro “startDate” não estiver definido, o valor do atributo de sessão “reservationStartDate” será usado como valor padrão.
3. Se nenhum dos dois primeiros valores padrão for usado, o slot não terá um valor padrão e o Amazon Lex obterá um valor como de costume.

Se um valor padrão for usado para o slot, o slot não será obtido, mesmo que seja necessário.

Definição dos atributos da sessão

Atributos da sessão contêm informações específicas da aplicação que são passadas entre o bot e a aplicação cliente durante uma sessão. O Amazon Lex passa atributos da sessão para todas as funções do Lambda configuradas para um bot. Se uma função do Lambda adicionar ou atualizar atributos da sessão, o Amazon Lex passará as novas informações de volta para a aplicação cliente.

Use atributos de sessão em suas funções do Lambda para inicializar um bot e personalizar solicitações e cartões de resposta. Por exemplo:

- Inicialização: em um bot de pedido de pizza, a aplicação cliente passa o local do usuário como um atributo de sessão na primeira chamada à operação RecognizeText ou RecognizeUtterance. Por exemplo, `"Location": "111 Maple Street"`. A função do Lambda usa essas informações para encontrar a pizzaria mais próxima para fazer o pedido.
- Personalizar solicitações: configura solicitações e cartões de resposta para fazer referência a atributos de sessão. Por exemplo, "Olá, [FirstName], quais coberturas você quer?" Se você passar o nome do usuário como um atributo de sessão (`{"FirstName": "Vivian"}`), o Amazon Lex

substituirá o nome pelo espaço reservado. Em seguida, ele envia uma solicitação personalizada para o usuário: "Oi, Viviane, quais coberturas você quer?"

Os atributos da sessão permanecem durante a vigência da sessão. Eles são criptografados e armazenados pelo Amazon Lex até o final da sessão. O cliente pode criar atributos de sessão em uma solicitação chamando a operação RecognizeText ou RecognizeUtterance com o campo `sessionAttributes` definido para um valor. Uma função do Lambda pode criar um atributo de sessão em uma resposta. Depois que o cliente ou uma função do Lambda cria um atributo de sessão, o valor do atributo armazenado é usado sempre que a aplicação cliente não incluir o campo `sessionAttribute` em uma solicitação para o Amazon Lex.

Por exemplo, suponha que você tenha dois atributos de sessão `{"x": "1", "y": "2"}`. Se o cliente chamar a operação `RecognizeText` ou `RecognizeUtterance` sem especificar o campo `sessionAttributes`, o Amazon Lex chamará a função do Lambda com os atributos de sessão armazenados (`{"x": 1, "y": 2}`). Se a função do Lambda não retornar atributos de sessão, o Amazon Lex retornará os atributos de sessão armazenados à aplicação cliente.

Se a aplicação cliente ou uma função do Lambda passar atributos de sessão, o Amazon Lex atualizará os atributos de sessão armazenados. Passar um valor existente, como `{"x": 2}`, atualiza o valor armazenado. Se você inserir um novo conjunto de atributos de sessão, como `{"z": 3}`, os valores existentes serão removidos e apenas o novo valor será mantido. Quando um mapa vazio, `{}`, é passado, os valores armazenados são apagados.

Para enviar atributos de sessão para o Amazon Lex, crie um mapa de string para string dos atributos. As considerações a seguir mostram como mapear atributos de sessão:

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

Para a operação `RecognizeText`, insira o mapa no corpo da solicitação usando o campo `sessionAttributes` da estrutura `sessionState`, como a seguir.

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

```
}
```

Para a operação `RecognizeUtterance`, codifique o mapa em base64 e o envie como parte do cabeçalho `x-amz-lex-session-state`.

Se você estiver enviando dados binários ou estruturados em um atributo de sessão, deve primeiro transformar os dados em uma string simples. Para mais informações, consulte Configuração de atributos complexos.

Definição de atributos de solicitação

Atributos da solicitação contêm informações específicas de solicitação e aplicam-se apenas à solicitação atual. Uma aplicação cliente envia essas informações ao Amazon Lex. Use atributos de solicitação para passar informações que não precisam ser mantidas durante toda a sessão. Você pode criar seus próprios atributos de solicitação ou usar atributos predefinidos. Para enviar atributos de solicitação, use o cabeçalho `x-amz-lex-request-attributes` em um RecognizeUtterance ou no campo `requestAttributes` em uma solicitação RecognizeText. Como os atributos de solicitação não são persistentes entre as solicitações como os atributos de sessão, eles não são retornados em respostas `RecognizeUtterance` ou `RecognizeText`.

> **Note**
>
> Para enviar informações que são mantidas nas solicitações, use atributos de sessão.

Definição de atributos de solicitação definidos pelo usuário

O atributo de solicitação definido pelo usuário são os dados que você envia para seu bot em cada solicitação. Você enviar as informações no `amz-lex-request-attributes` cabeçalho de uma solicitação `RecognizeUtterance` ou no campo `requestAttributes` de uma solicitação `RecognizeText`.

Para enviar atributos de solicitação ao Amazon Lex, crie um mapa de string para string dos atributos. As considerações a seguir mostram como mapear atributos de solicitação:

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

```
}
```

Para a operação `PostText`, insira o mapa no corpo da solicitação usando o campo `requestAttributes`, como a seguir:

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

Para a operação `PostContent`, codifique o mapa em base64 e o envie como o cabeçalho `x-amz-lex-request-attributes`.

Se você está enviando dados binários ou estruturados em um atributo de solicitação, você deve primeiro transformar os dados em uma string simples. Para mais informações, consulte Configuração de atributos complexos.

Definição do tempo limite da sessão

O Amazon Lex retém informações de contexto, dados de slot e atributos de sessão, até o fim de uma sessão de conversa. Para controlar o tempo de duração de uma sessão em um bot, defina o tempo limite da sessão. Por padrão, a duração da sessão é de 5 minutos, mas você pode especificar qualquer duração entre 0 e 1.440 minutos (24 horas).

Por exemplo, suponha que você crie um bot `ShoeOrdering` que seja compatível com intenções como `OrderShoes` e `GetOrderStatus`. Quando o Amazon Lex detecta que a intenção do usuário é encomendar sapatos, ele pede os dados do slot. Por exemplo, ele pergunta o tamanho, a cor, a marca, etc. Se o usuário fornecer alguns dados do slot, mas não finalizar a compra de sapato, o Amazon Lex memorizará todos os dados do slot e os atributos da sessão inteira. Se o usuário retornar para a sessão antes que ela expire, ele pode fornecer os dados de slot restantes e concluir a compra.

No console do Amazon Lex, você define o tempo limite da sessão ao criar um bot. Com o AWS Command Line Interface (AWS CLI) ou a API, você define o tempo limite ao criar um bot com a operação CreateBot definindo o campo idleSessionTTLInSeconds.

Compartilhamento de informações entre intenções

O Amazon Lex é compatível com o compartilhamento de informações entre intenções. Para compartilhar entre intenções, use contextos de saída ou atributos de sessão.

Para usar contextos de saída, você define um contexto de saída ao criar ou atualizar uma intenção. Quando a intenção é cumprida, as respostas do Amazon Lex V2 contêm o contexto e os valores de slot da intenção como parâmetros de contexto. Você pode usar esses parâmetros como valores padrão em intenções subsequentes ou no código da aplicação ou nas funções do Lambda.

Para usar atributos de sessão, você define os atributos no seu código do Lambda ou da aplicação. Por exemplo, um usuário do bot `ShoeOrdering` começa a pedir sapatos. O bot inicia uma conversa com o usuário, coletando dados de slot como tamanho, cor e marca do sapato. Quando o usuário faz um pedido, a função do Lambda que atende ao pedido define o atributo de sessão `orderNumber`, que contém o número do pedido. Para obter o status do pedido, o usuário usa a intenção `GetOrderStatus`. O bot pode solicitar dados de slot ao usuário, como número do pedido e data do pedido. Quando o bot tem as informações necessárias, ele retorna o status do pedido.

Se você acha que seus usuários podem mudar de intenção durante uma sessão, você pode projetar seu bot para retornar o status do pedido mais recente. Em vez de pedir ao usuário as informações do pedido novamente, você usa o atributo de sessão `orderNumber` para compartilhar informações entre intenções e cumprir a intenção `GetOrderStatus`. O bot faz isso ao retornar o status do último pedido feito pelo usuário.

Configuração de atributos complexos

Os atributos de solicitação e de sessão são mapas de string para string de atributos e valores. Em muitos casos, você pode usar o mapa de string para transferir valores de atributo entre o aplicativo cliente e o bot. Em alguns casos, no entanto, pode ser necessário transferir uma estrutura complexa ou dados binários que não podem ser facilmente convertidos em um mapa de string. Por exemplo, o seguinte objeto JSON representa um arranjo com as três cidades mais populosas dos Estados Unidos:

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

Esse arranjo de dados não funciona bem como mapa de string para string. Nesse caso, você pode transformar um objeto em uma string simples para que você possa enviá-lo ao seu bot com as operações RecognizeText e RecognizeUtterance.

Por exemplo, se você estiver usando o JavaScript, você pode usar a operação `JSON.stringify` para converter um objeto para JSON e a operação `JSON.parse` para converter texto JSON para um objeto JavaScript:

```
// 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);
```

Para enviar atributos de sessão com a operação `RecognizeUtterance`, você deve codificar os atributos em base64 antes de adicioná-los ao cabeçalho de solicitação, como mostrado no seguinte código JavaScript:

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

Você pode enviar dados binários para as operações `RecognizeText` e `RecognizeUtterance` convertendo os dados para uma string codificada em base64 e, em seguida, enviando a string como o valor nos atributos de sessão:

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

```
}
```

Gerenciamento de sessões com a API Amazon Lex V2

Quando um usuário inicia uma conversa com seu bot, o Amazon Lex V2 cria uma sessão. As informações trocadas entre a aplicação e o Amazon Lex V2 compõem o estado da sessão para a conversa. Quando você faz uma solicitação, a sessão é identificada por um identificador especificado por você. Para obter mais informações sobre o identificador de sessão, consulte o campo `sessionId` na operação RecognizeText ou RecognizeUtterance.

É possível modificar o estado da sessão enviado entre o aplicativo e o bot. Por exemplo, você pode criar e modificar os atributos que contêm informações personalizadas sobre a sessão e alterar o fluxo da conversa definindo o contexto de diálogo para interpretar o próximo enunciado.

Há três maneiras de atualizar o estado da sessão.

- Passe as informações da sessão em linha como parte de uma chamada para a operação `RecognizeUtterance` ou `RecognizeText`.
- Use uma função do Lambda com a operação `RecognizeText` ou `RecognizeUtterance` que é chamada a cada turno da conversa. Para mais informações, consulte Habilitando a lógica personalizada com as funções do AWS Lambda. A outra é usar a API de runtime do Amazon Lex V2 na sua aplicação para fazer alterações no estado da sessão.
- Use operações que permitem gerenciar informações da sessão para uma conversa com o bot. As operações são PutSession, GetSession e DeleteSession. Você pode usar essas operações para obter informações sobre o estado da sessão do seu usuário com o bot e ter um controle apurado sobre o estado.

Use a operação `GetSession` quando desejar obter o estado atual da sessão. A operação retorna o estado atual da sessão, incluindo o estado do diálogo com o usuário, todos os atributos de sessão que foram definidos e valores de slot da intenção atual e quaisquer outras intenções que o Amazon Lex V2 identificou como possíveis intenções que correspondem ao enunciado do usuário.

A operação `PutSession` permite manipular diretamente o estado da sessão atual. Você pode definir a sessão, incluindo o tipo de ação de diálogo que o bot executará em seguida e as mensagens que o Amazon Lex V2 enviará ao usuário. Isso oferece a você controle sobre o fluxo da conversa com o bot. Defina o campo `type` da ação de diálogo como `Delegate` para que o Amazon Lex V2 determine a próxima ação do bot.

Você pode usar a operação `PutSession` para criar uma nova sessão com um bot e definir a intenção com a qual ele deve começar. Também é possível usar a operação `PutSession` para mudar de uma intenção para outra. Ao criar uma sessão ou alterar a intenção, você também pode definir o estado da sessão, como valores de slot e atributos de sessão. Quando a nova intenção é concluída, você tem a opção de reiniciar a intenção anterior.

A resposta da operação `PutSession` contém as mesmas informações que a operação `RecognizeUtterance`. Você pode usar essas informações para solicitar a próxima informação ao usuário, da mesma forma que faria com a resposta da operação `RecognizeUtterance`.

Use a operação `DeleteSession` para remover uma sessão existente e começar de novo com uma nova sessão. Por exemplo, ao testar seu bot, você pode usar a operação `DeleteSession` para remover as sessões de teste do seu bot.

As operações de sessão trabalham com as funções do Lambda de atendimento. Por exemplo, se sua função do Lambda retornar `Failed` como o estado de atendimento, você poderá usar a operação `PutSession` para definir o tipo de ação de diálogo como `close` e `fulfillmentState` para `ReadyForFulfillment` para repetir a etapa de atendimento.

Veja a seguir algumas ações que você pode executar com as operações de sessão:

- Fazer com que o bot inicie uma conversa em vez de esperar pelo usuário.
- Alternar entre as intenções durante uma conversa.
- Voltar para uma intenção anterior.
- Iniciar ou reiniciar uma conversa no meio da interação.
- Validar valores de slot e fazer com que o bot solicite novamente valores que não são válidos.

Cada uma delas é descrita com mais detalhes a seguir.

Como iniciar uma nova sessão

Se você desejar que o bot inicie a conversa com o usuário, use a operação `PutSession`.

- Crie uma intenção de boas-vindas sem slots e uma mensagem de conclusão solicitando que o usuário indique uma intenção. Por exemplo, "O que você gostaria de pedir? Você pode dizer "Pedir uma bebida" ou "Pedir uma pizza".
- Chame a operação `PutSession`. Defina o nome da intenção como o nome da intenção de boas-vindas e defina a ação de diálogo como `Delegate`.

- O Amazon Lex responderá com o prompt da sua intenção de boas-vindas para iniciar a conversa com o usuário.

Alternar intenções

Você pode usar a operação `PutSession` para alternar de uma intenção para outra. Também é possível usá-la para voltar para uma intenção anterior. Você pode usar a operação `PutSession` para definir atributos de sessão ou valores de slot para a nova intenção.

- Chame a operação `PutSession`. Defina o nome da intenção como o nome da nova intenção e defina a ação de diálogo como `Delegate`. Você também pode definir quaisquer valores de slot ou atributos de sessão necessários para a nova intenção.
- O Amazon Lex iniciará uma conversa com o usuário que usa a nova intenção.

Como retomar uma intenção anterior

Para retomar uma intenção anterior, use a operação `GetSession` para obter o estado da intenção, executar a interação necessária e, depois, use a operação `PutSession` para definir a intenção como seu estado de diálogo anterior.

- Chame a operação `GetSession`. Armazene o estado da intenção.
- Execute outra interação, como cumprir uma intenção diferente.
- Usando as informações salvas para a intenção anterior, chame a operação `PutSession`. Isso retornará o usuário para a intenção anterior no mesmo local na conversa.

Em alguns casos, pode ser necessário retomar a conversa do usuário com o bot. Por exemplo, digamos que você tenha criado um bot de atendimento ao cliente. Seu aplicativo determina que o usuário precisa conversar com um representante de atendimento ao cliente. Depois de falar com o usuário, o representante poderá direcionar a conversa de volta para o bot com as informações coletadas.

Para retomar uma sessão, use etapas semelhantes a estas:

- Seu aplicativo determina que o usuário precisa falar com um representante de atendimento ao cliente.
- Use a operação `GetSession` para obter o estado de diálogo atual da intenção.

- O representante de atendimento ao cliente se comunica com o usuário e resolve o problema.
- Use a operação `PutSession` para definir o estado de diálogo da intenção. Isso pode incluir definir valores de slot, definir atributos de sessão ou alterar a intenção.
- O bot retoma a conversa com o usuário.

Validação de valores de slot

Você pode validar as respostas ao bot usando seu aplicativo cliente. Se a resposta não for válida, use a operação `PutSession` para obter uma nova resposta do usuário. Por exemplo, suponha que seu bot de pedido de flores só possa vender tulipas, rosas e lírios. Se o usuário pedir cravos, o aplicativo poderá fazer o seguinte:

- Examinar o valor do slot retornado pela resposta `PostText` ou `PostContent`.
- Se o valor do slot não for válido, chame a operação `PutSession`. Seu aplicativo deve limpar o valor do slot, definir o campo `slotToElicit` e definir o valor `dialogAction.type` como `elicitSlot`. Há a opção de você definir os campos `message` e `messageFormat` se quiser alterar a mensagem usada pelo Amazon Lex para obter o valor do slot.

Habilitando a lógica personalizada com as funções do AWS Lambda

Com as funções do AWS Lambda, você pode controlar melhor o comportamento do seu bot Amazon Lex V2 por meio de funções personalizadas que você define.

O Amazon Lex V2 usa uma função do Lambda por alias de bot por linguagem, em vez de uma função do Lambda para cada intenção.

Para integrar uma função do Lambda ao seu bot do Amazon Lex V2, execute as seguintes etapas:

1. Determine de quais campos no evento de entrada você deseja extrair informações para usar em sua função do Lambda.
2. Determine quais campos na resposta você deseja manipular e retornar da sua função do Lambda.
3. Crie uma função no AWS Lambda usando a linguagem de programação de sua escolha e escreva seu script.
4. Certifique-se de que a função retorna uma estrutura que corresponda ao formato da resposta.
5. Implante a função do Lambda.
6. Associe a função do Lambda a um alias de bot do Amazon Lex V2 com o console ou as operações da API.
7. Selecione os estágios de conversação nos quais você deseja invocar sua função do Lambda com o console ou as operações da API.
8. Crie seu bot do Amazon Lex V2 e teste se a função do Lambda funciona, conforme o esperado. Depure sua função com a ajuda do Amazon CloudWatch.

Tópicos

Interpretar o formato do evento de entrada

A primeira etapa na integração de uma função do Lambda em seu bot do Amazon Lex V2 é entender os campos no evento Amazon Lex V2 e determinar as informações desses campos que você deseja usar ao escrever seu script. O objeto JSON a seguir mostra o formato geral de um evento do Amazon Lex V2 passado para uma função do Lambda:

> **Note**
>
> O formato de entrada pode mudar sem uma alteração correspondente para a `messageVersion`. Seu código não deve gerar um erro, se novos campos estiverem presentes.

```
{
    "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 Intenção 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 Intenção for details about the structure
        },
        "prompt": {
            "attempt": string
        }
    },
    "requestAttributes": {
        string: string,
        ...
    },
    "sessionState": {
        // see Estado da sessão for details about the structure
    },
    "transcriptions": [
        {
            "transcription": string,
            "transcriptionConfidence": number,
            "resolvedContext": {
                "intent": string
            },
            "resolvedSlots": {
                slot name: {
                    // see Slots for details about the structure
                },
                ...
            }
        },
        ...
    ]
```

```
}
```

Cada campo no evento de entrada é descrito a seguir:

messageVersion

A versão da mensagem que identifica o formato dos dados de evento que estão indo para a função do Lambda e o formato esperado da resposta de uma função do Lambda.

> **Note**
>
> Você configura esse valor ao definir uma intenção. Na implementação atual, apenas o Amazon Lex V2 oferece suporte à versão 1.0 da mensagem. Portanto, o console assume o valor padrão de 1.0 e não mostra a versão da mensagem.

invocationSource

O gancho de código que o chamou da função do Lambda. Os seguintes valores são possíveis:

`DialogCodeHook`: o Amazon Lex V2 chamou a função do Lambda após a entrada do usuário.

`FulfillmentCodeHook`: o Amazon Lex V2 chamou a função do Lambda depois de preencher todos os slots necessários e a intenção estar pronta para ser cumprida.

inputMode

O modo do enunciado do usuário. Os valores possíveis são:

`DTMF`: o usuário insere o enunciado usando um teclado de tom sensível ao toque (multifrequência de tons duplos).

`Speech`: o usuário falou o enunciado.

`Text`: o usuário digitou o enunciado.

responseContentType

O modo de resposta do bot ao usuário. `text/plain; charset=utf-8` indica que o último enunciado foi escrito, enquanto um valor que começa com `audio` indica que o último enunciado foi falado.

sessionId

O identificador de sessão alfanumérico usado para a conversa.

inputTranscript

Uma transcrição da entrada do usuário.

- Para entrada de texto, esse é o texto que o usuário digitou. Para entrada DTMF, essa é a chave que o usuário insere.
- Para entrada de voz, esse é o texto no qual o Amazon Lex V2 converte o enunciado do usuário para invocar uma intenção ou preencher um slot.

invocationLabel

Um valor que indica a resposta que invocou a função do Lambda. Você pode definir rótulos de invocação para a resposta inicial, os slots e a resposta de confirmação.

bot

Informações sobre o bot que processou a solicitação, consistindo nos seguintes campos:

- id: o identificador atribuído ao bot quando você o criou. Você pode ver o ID do bot no console do Amazon Lex V2 na página Configurações do bot.
- name: o nome que você deu ao bot ao criá-lo.
- localeId: o identificador da localidade que você usou para o seu bot. Para obter uma lista de localidades, consulte Idiomas e locais aceitos pelo Amazon Lex V2.
- version: a versão do bot que processou a solicitação.
- aliasId: o identificador atribuído ao alias do bot quando você o criou. Você pode ver o ID do alias do bot no console do Amazon Lex V2 na página Aliases. Se você não conseguir ver o ID do alias na lista, escolha o ícone de engrenagem no canto superior direito e ative o ID do alias.
- aliasName: o nome que você deu ao alias do bot.

interpretations

Uma lista de informações sobre intenções que o Amazon Lex V2 considera possíveis coincidências com o enunciado do usuário. Cada item é uma estrutura que fornece informações sobre a correspondência do enunciado com uma intenção, com o seguinte formato:

```
{
    "intent": {
        // see Intenção 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
        }
    }
}
```

Os campos dentro da estrutura são os seguintes:

- intent: uma estrutura que contém informações sobre a intenção. Consulte Intenção para obter detalhes sobre a estrutura.
- nluConfidence: uma pontuação que indica o grau de confiança do Amazon Lex V2 de que a intenção corresponde à intenção do usuário.
- sentimentResponse: uma análise do sentimento da resposta, que contém os seguintes campos:
  - sentiment: indica se o sentimento do enunciado é `POSITIVE`, `NEGATIVE`, `NEUTRAL`, ou `MIXED`.
  - sentimentScore: uma estrutura que mapeia cada sentimento em um número que indica o quanto o Amazon Lex V2 está confiante de que o enunciado transmite esse sentimento.
- InterpretationSource: indica se um slot foi resolvido pelo Amazon Lex ou pelo Amazon Bedrock.

proposedNextState

Se a função do Lambda definir o `dialogAction` do `sessionState` para `Delegate`, esse campo aparecerá e mostrará a proposta do Amazon Lex V2 para a próxima etapa da conversa. Caso contrário, o próximo estado dependerá das configurações que você retorna na resposta da sua função do Lambda. Essa estrutura só estará presente se as duas afirmações abaixo forem verdadeiras:

1. O valor de `invocationSource` é `DialogCodeHook`

2. O `type` previsto de `dialogAction` é `ElicitSlot`.

Você pode usar essas informações para adicionar `runtimeHints` no ponto certo da conversa. Consulte Melhorar o reconhecimento dos valores de slots com sugestões de runtime para obter mais informações. `proposedNextState` é uma estrutura que contém os seguintes campos:

A estrutura dos dados de `proposedNextState` é a seguinte:

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Intenção for details about the structure
    },
    "prompt": {
        "attempt": string
    }
}
```

- dialogAction: contém informações sobre a próxima etapa proposta pelo Amazon Lex V2. Os campos da estrutura são os seguintes:
  - slotToElicit: o slot obtido a seguir, conforme proposto pelo Amazon Lex V2. Esse campo aparecerá somente se o valor de `type` for `ElicitSlot`.
  - tipo: a próxima etapa da conversa, conforme proposto pelo Amazon Lex V2. Os seguintes valores são possíveis:

    `Delegate`: o Amazon Lex V2 determina a próxima ação.

    `ElicitIntent`: a próxima ação é extrair uma intenção do usuário.

    `ElicitSlot`: a próxima ação é extrair um valor de slot do usuário.

    `Close`: encerra o processo de cumprimento da intenção e indica que não haverá uma resposta do usuário.

    `ConfirmIntent`: a próxima ação é perguntar ao usuário se os slots estão corretos e se a intenção está pronta para ser cumprida.

- intent: a intenção que o bot determinou que o usuário está tentando cumprir. Consulte Intenção para obter detalhes sobre a estrutura.
- prompt: uma estrutura que contém o campo `attempt`, que é mapeado para um valor que especifica quantas vezes o Amazon Lex V2 solicitou ao usuário o próximo slot. Os valores possíveis são `Initial` para a primeira tentativa e `Retry1`, `Retry2`, `Retry3`, `Retry4` e `Retry5` para as tentativas subsequentes.

requestAttributes

Uma estrutura que contém os atributos de sessão específicos à solicitação que o cliente envia na solicitação. Use atributos de solicitação para passar informações que não precisam ser mantidas durante toda a sessão. Se não houver atributos de solicitação, o valor será nulo. Para obter mais informações, consulte Definição de atributos de solicitação.

sessionState

O estado atual da conversa entre o usuário e seu bot do Amazon Lex V2. Consulte Estado da sessão para obter detalhes sobre a estrutura.

transcriptions

Uma lista de transcrições que o Amazon Lex V2 considera possíveis coincidências com o enunciado do usuário. Para obter mais informações, consulte Como usar pontuações de confiança na transcrição de voz. Cada item é um objeto com o seguinte formato que contém informações sobre uma possível transcrição:

```
{
    "transcription": string,
    "transcriptionConfidence": number,
    "resolvedContext": {
        "intent": string
    },
    "resolvedSlots": {
        slot name: {
            // see Slots for details about the structure
        },
        ...
    }
}
```

Os campos são descritos abaixo:

- transcription: uma transcrição que o Amazon Lex V2 considera uma possível correspondência com o enunciado de áudio do usuário.
- transcriptionConfidence: uma pontuação que indica o grau de confiança do Amazon Lex V2 de que a intenção corresponde à intenção do usuário.
- resolvedContext: uma estrutura que contém o campo `intent` que mapeia a intenção à qual o enunciado pertence.
- resolvedSlots: uma estrutura cujas chaves são os nomes de cada slot que é resolvido pelo enunciado. Cada nome de slot é mapeado para uma estrutura que contém informações sobre esse slot. Consulte Slots para obter detalhes sobre a estrutura.

Preparando o formato de resposta

A segunda etapa na integração de uma função do Lambda em seu bot do Amazon Lex V2 é entender os campos na resposta da função do Lambda e determinar quais parâmetros você deseja manipular. O objeto JSON a seguir mostra o formato geral de uma resposta do Lambda que é retornada ao Amazon Lex V2:

```
{
    "sessionState": {
        // see Estado da sessão 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,
        ...
    }
}
```

Cada campo na resposta é descrito abaixo:

sessionState

O estado da conversa entre o usuário e seu bot do Amazon Lex V2 que você deseja retornar. Consulte Estado da sessão para obter detalhes sobre a estrutura. Este campo é sempre obrigatório.

mensagens

Uma lista de mensagens que o Amazon Lex V2 retorna ao cliente para a próxima rodada da conversa. Se o `contentType` que você fornecer for `PlainText`, `CustomPayload` ou `SSML`, escreva a mensagem que você deseja devolver ao cliente no campo `content`. Se o `contentType` que você fornecer for `ImageResponseCard`, forneça os detalhes do cartão no campo `imageResponseCard`. Se você não fornecer mensagens, o Amazon Lex V2 usa a mensagem apropriada definida quando o bot foi criado.

O campo `messages` é obrigatório se `dialogAction.type` for `ElicitIntent` ou `ConfirmIntent`.

Cada item na lista é uma estrutura no formato a seguir, contendo informações sobre uma mensagem a ser retornada ao usuário. Exemplo:

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
```

```
                "value": string
            },
            ...
        ]
    }
}
```

Uma descrição para cada campo é fornecida abaixo:

- contentType: o tipo de mensagem a ser usada.

  `CustomPayload`: uma string de resposta que você pode personalizar para incluir dados ou metadados para a aplicação.

  `ImageResponseCard`: uma imagem com botões que o cliente pode selecionar. Consulte ImageResponseCard para obter mais informações.

  `PlainText`: uma string de texto simples.

  `SSML`: uma string que inclui Linguagem de marcação de síntese de fala para personalizar a resposta de áudio.
- content: a mensagem a ser enviada ao usuário. Use esse campo se o tipo de mensagem for `PlainText`, `CustomPayload` ou `SSML`.
- imageResponseCard: contém a definição do cartão de resposta a ser mostrado ao usuário. Use esse campo se o tipo de mensagem for `ImageResponseCard`. Mapeia para uma estrutura que contém os seguintes campos:
  - title: o título do cartão de resposta.
  - subtitle: a solicitação para o usuário escolher um botão.
  - imageUrl: um link para uma imagem do cartão.
  - buttons: uma lista de estruturas que contém, informações sobre um botão. Cada estrutura contém um campo `text` com o texto a ser exibido e um campo `value` com o valor a ser enviado ao Amazon Lex V2, se o cliente selecionar esse botão. Você pode incluir até três botões.

requestAttributes

Uma estrutura que contém atributos específicos da solicitação para a resposta ao cliente. Consulte Definição de atributos de solicitação para obter mais informações. Esse campo é opcional.

Campos necessários na resposta

No mínimo, a resposta do Lambda deve incluir um objeto `sessionState`. Dentro disso, forneça um objeto `dialogAction` e especifique o campo `type`. Dependendo do `type` de `dialogAction` que você fornecer, pode haver outros campos obrigatórios para a resposta do Lambda. Esses requisitos são descritos a seguir, juntamente com exemplos mínimos de trabalho:

Delegate

Delegate permite que o Amazon Lex V2 determine a próxima etapa. Nenhum outro campo é necessário.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntent solicita que o cliente expresse uma intenção. Você deve incluir pelo menos uma mensagem no campo `messages` para solicitar a elicitação de uma intenção.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlot solicita que o cliente forneça um valor de slot. Você deve incluir o nome do slot no campo `slotToElicit` do objeto `dialogAction`. Você também deve incluir o `name` do `intent` no objeto `sessionState`.

```
{`
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
    }
}
```

ConfirmIntent

ConfirmIntent confirma os valores dos slots do cliente e se a intenção está pronta para ser cumprida. Você deve incluir o `name` do `intent` no objeto `sessionState` e o `slots` a ser confirmado. Você também deve incluir pelo menos uma mensagem no campo `messages` para solicitar ao usuário a confirmação dos valores dos slots. Sua mensagem deve gerar uma resposta “sim” ou “não”. Se o usuário responder “sim”, o Amazon Lex V2 define `confirmationState` da intenção como `Confirmed`. Se o usuário responder “não”, o Amazon Lex V2 define o `confirmationState` da intenção como `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?"
        }
    ]
}
```

Fechar

Fechar encerra o processo de cumprimento da intenção e indica que nenhuma resposta adicional é esperada do usuário. Você deve incluir o `name` e o `state` do `intent` no objeto `sessionState`. Os estados de intenção compatíveis são `Failed`, `Fulfilled` e `InProgress`.

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Estruturas comuns no evento e na resposta do Lambda

Na resposta do Lambda, há várias estruturas que se repetem. Detalhes sobre essas estruturas comuns são fornecidos nesta seção.

Intenção

```
"intent": {
    "confirmationState": "Confirmed | Denied | None",
    "name": string,
    "slots": {
        // see Slots 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
       }
}
```

O campo `intent` é mapeado para um objeto com os seguintes campos:

confirmationState

Indica se o usuário confirmou os slots da intenção e se a intenção está pronta para ser atendida. Os seguintes valores são possíveis:

`Confirmed`: o usuário confirma que os valores do slot estão corretos.

`Denied`: o usuário indica que os valores do slot estão incorretos.

`None`: o usuário ainda não chegou ao estágio de confirmação.

name

O nome da intenção.

slots

Informações sobre os slots necessários para cumprir a intenção. Consulte Slots para obter detalhes sobre a estrutura.

estado

Indica o estado de cumprimento da intenção. Os seguintes valores são possíveis:

`Failed`: o bot falhou em cumprir a intenção.

`Fulfilled`: o bot concluiu o cumprimento da intenção.

`FulfillmentInProgress`: o bot está no meio do cumprimento da intenção.

`InProgress`: o bot está no meio da obtenção dos valores de slot necessários para cumprir a intenção.

`ReadyForFulfillment`: o bot obteve todos os valores do slot para a intenção e está pronto para cumpri-la.

`Waiting`: o bot está aguardando uma resposta do usuário (limitado a conversas de streaming).

kendraResponse

Contém informações sobre os resultados da consulta de pesquisa do Kendra. Esse campo aparecerá somente se a intenção for `KendraSearchIntent`. Consulte a sintaxe de resposta na chamada de API de Consulta para Kendra para obter mais informações.

Slots

O campo `slots` existe dentro de uma estrutura `intent` e é mapeado para uma estrutura cujas chaves são os nomes dos slots para essa intenção. Se o slot não for um slot de vários valores (consulte Uso de vários valores em um slot para obter mais detalhes), ele será mapeado para uma estrutura com o formato a seguir. Observe que o `shape` é `Scalar`.

```
{
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
```

```
    }
}
```

Se o slot for um slot de vários valores, o objeto para o qual ele é mapeado contém outro campo chamado `values`, que é mapeado para uma lista de estruturas, cada uma contendo informações sobre um slot que compõe o slot de vários valores. O formato de cada objeto na lista corresponde ao do objeto para o qual um slot normal é mapeado. Observe que o `shape` é `List`, mas o `shape` dos slots de componentes abaixo `values` é `Scalar`.

```
{
    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,
                    ...
                ]
            }
        },
```

```
        ...
    ]
}
```

Os campos no objeto do slot estão descritos abaixo:

shape

O formato do slot. Esse valor é `List` válido se houver vários valores no slot (consulte Uso de vários valores em um slot para obter mais detalhes) e, caso contrário, é `Scalar`.

valor

Um objeto contendo informações sobre o valor que o usuário forneceu para um slot e a interpretação do Amazon Lex, no seguinte formato:

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
        ...
    ]
}
```

Os campos são descritos abaixo:

- originalValue: a parte da resposta do usuário à elicitação do slot que o Amazon Lex determina ser relevante para o valor do slot.
- interpretedValue: o valor que o Amazon Lex determina para o slot, de acordo com a entrada do usuário.
- resolvedValues: uma lista de valores que o Amazon Lex determina como possíveis resoluções para a entrada do usuário.

values

Uma lista de objetos contendo informações sobre os slots que compõem o slot de vários valores. O formato de cada objeto corresponde ao de um slot normal, com os campos `shape` e `value` descritos acima. `values` só aparece se o slot consistir em vários valores (consulte Uso de vários valores em um slot para obter mais detalhes). O objeto JSON a seguir mostra dois slots de componentes:

```
"values": [
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    ...
]
```

Estado da sessão

O campo `sessionState` é mapeado para um objeto contendo informações sobre o estado da conversa com o usuário. Os campos reais que aparecem no objeto dependem do tipo de ação da caixa de diálogo. Consulte Campos necessários na resposta para obter os campos obrigatórios em uma resposta do Lambda. O formato do objeto `sessionState` é o seguinte:

```
"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 Intenção for details about the structure
    },
    "originatingRequestId": string
}
```

Os campos são descritos abaixo:

activeContexts

Uma lista de objetos que contém informações sobre um contexto que um usuário está usando em uma sessão. Use contextos para facilitar e controlar o reconhecimento de intenções. Para obter mais

informações sobre contextos, consulte Definir o contexto da intenção. Cada objeto é formatado como segue:

```
{
    "name": string,
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

Os campos são descritos abaixo:

- name: o nome do contexto.
- contextAttributes: um objeto que contém os nomes dos atributos do contexto e os valores para os quais eles são mapeados.
- timeToLive: um objeto que especifica por quanto tempo o contexto permanece ativo. Esse objeto pode conter um ou ambos os seguintes campos:
  - timeToLiveInSeconds: o número de segundos que o contexto permanece ativo.
  - turnsToLive: o número de turnos em que o contexto permanece ativo.

sessionAttributes

Mapa de pares de chaves/valores que representam as informações de contexto específicas da sessão. Para obter mais informações, consulte Definição dos atributos da sessão. O objeto é formatado da seguinte forma:

```
{
    string: string,
    ...
}
```

runtimeHints

Fornece dicas sobre as frases que um cliente provavelmente usará em um slot para melhorar o reconhecimento de áudio. Os valores que você fornece nas dicas aumentam o reconhecimento

de áudio desses valores em relação a palavras com sons semelhantes. O formato do objeto `runtimeHints` é o seguinte:

```
{
    "slotHints": {
        intent name: {
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

O campo `slotHints` é mapeado para um objeto cujos campos são os nomes das intenções no bot. Cada nome de intenção é mapeado para um objeto cujos campos são os nomes dos slots dessa intenção. Cada nome de slot é mapeado para uma estrutura com um único campo `runtimeHintValues`, que é uma lista de objetos. Cada objeto contém um campo `phrase` que é mapeado para uma dica.

dialogAction

Determina a próxima ação a ser tomada pelo Amazon Lex V2. O formato do objeto é o seguinte:

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

Os campos são descritos abaixo:

- slotElicitationStyle: determina como o Amazon Lex V2 interpreta a entrada de áudio do usuário se o `type` da `dialogAction` for `ElicitSlot`. Para obter mais informações, consulte Capturar valores de slots com estilos de soletração. Os seguintes valores são possíveis:

`Default`: o Amazon Lex V2 interpreta a entrada de áudio da maneira padrão para preencher um slot.

`SpellByLetter`: o Amazon Lex V2 escuta a ortografia do valor do slot pelo usuário.

`SpellByWord`: o Amazon Lex V2 escuta a ortografia do valor do slot pelo usuário usando palavras associadas a cada letra (por exemplo, “a como em avião”).

- slotToElicit: define o slot a ser obtido do usuário, se `type` da `dialogAction` for `ElicitSlot`.
- type: define a ação que o bot deve executar. Os seguintes valores são possíveis:

  `Delegate`: permite que o Amazon Lex V2 determine a próxima etapa.

  `ElicitIntent`: solicita que o cliente expresse uma intenção.

  `ConfirmIntent`: confirma os valores dos slots do cliente e se a intenção está pronta para ser cumprida.

  `ElicitSlot`: solicita que o cliente forneça um valor de slot para uma intenção.

  `Close`: encerra o processo de cumprimento da intenção.

intent

Consulte Intenção para obter a estrutura do campo `intent`.

originatingRequestId

Um identificador exclusivo da solicitação. Esse campo é opcional para a resposta do Lambda.

Criar e anexar uma função do Lambda a um alias de bot

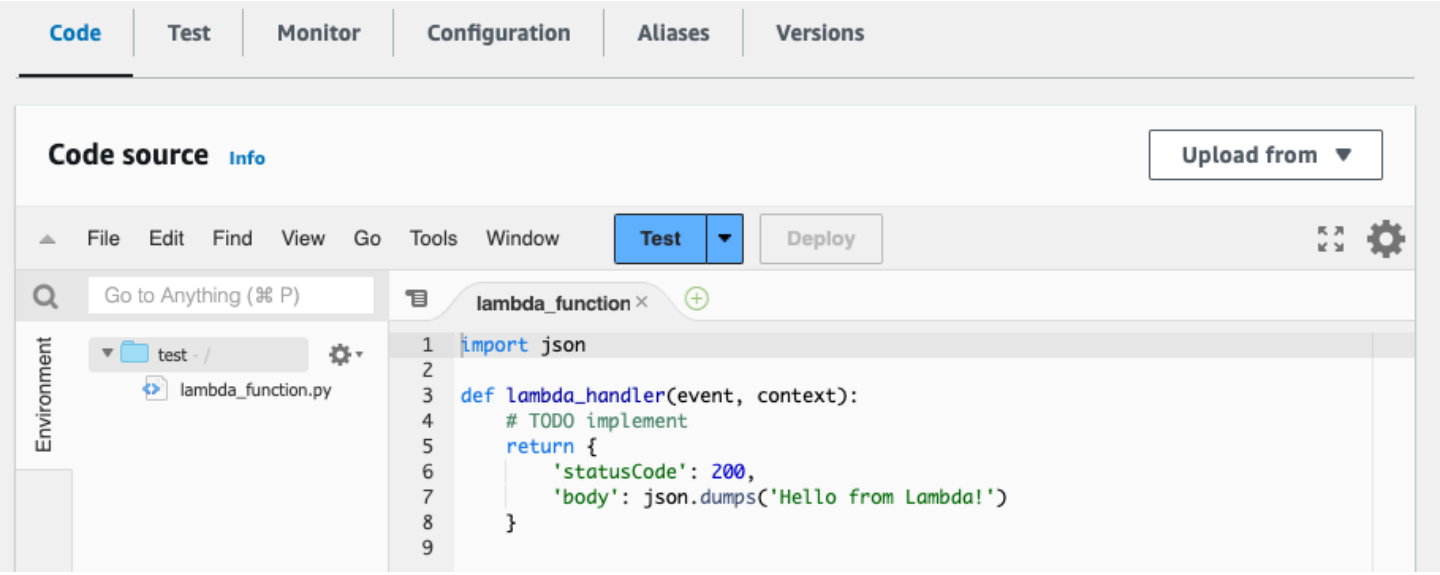
Criar a função do Lambda

Para criar uma função do Lambda para seu bot Amazon Lex V2, acesse o AWS Lambda partir do seu AWS Management Console e crie uma nova função. Você pode consultar o guia do desenvolvedor do AWS Lambda para obter mais detalhes sobre o AWS Lambda.

1. Faça login no AWS Management Console e abra o console AWS Lambda em https://console.aws.amazon.com/lambda/.

2. Escolha Funções na barra lateral esquerda.
3. Selecione Criar função.
4. Você pode selecionar Criar do zero para começar com o mínimo de código, Usar um esquema para selecionar o código de amostra para casos de uso comuns em uma lista, ou Imagem do contêiner para selecionar um contêiner a ser implantado em sua função. Se você selecionar Criar do zero, continue com as seguintes etapas:
   a. Dê à sua função um Nome de função significativo para descrever o que ela faz.
   b. Escolha um idioma no menu suspenso em Runtime para escrever sua função.
   c. Selecione uma Arquitetura de conjunto de instruções para sua função.
   d. Por padrão, o Lambda cria um perfil com permissões básicas. Para usar um perfil existente ou criar um perfil usando modelos de política da AWS, expanda o menu Alterar perfil de execução padrão e selecione uma opção.
   e. Expanda o menu Configurações avançadas para configurar mais opções.
5. Selecione Criar função.

A imagem a seguir mostra o que você vê ao criar uma nova função do zero:



A função do manipulador do Lambda difere dependendo da linguagem que você usa. Ele usa minimamente um objeto JSON do `event` como argumento. Você pode ver os campos no `event` que o Amazon Lex V2 fornece em Interpretar o formato do evento de entrada. Modifique a função do manipulador para finalmente retornar um objeto JSON de `response` que corresponda ao formato descrito em Preparando o formato de resposta.

Depois de terminar de escrever sua função, selecione Implantar para permitir que a função seja usada.

Lembre-se de que você pode associar cada alias de bot a, no máximo, uma função do Lambda. No entanto, você pode definir quantas funções precisar para seu bot no código do Lambda e chamar essas funções na função do manipulador do Lambda. Por exemplo, embora todas as intenções no mesmo alias de bot devam chamar a mesma função do Lambda, você pode criar uma função de roteador que ative uma função separada para cada intenção. Veja a seguir um exemplo de função de roteador que você pode usar ou modificar para sua aplicação:

```
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
```

Adicionar e invocar uma função do Lambda

Para chamar a função do Lambda em seu bot do Amazon Lex V2, você deve primeiro anexar a função a um alias de bot e depois definir os pontos na conversa em que o bot invoca a função. Você pode executar essas etapas com as operações do console ou da API.

Você pode usar as funções do Lambda nos seguintes pontos de uma conversa com um usuário:

- Na resposta inicial após o reconhecimento da intenção. Por exemplo, depois que o usuário diz que quer pedir uma pizza.
- Depois de obter um valor de slot do usuário. Por exemplo, depois que o usuário informa ao bot o tamanho da pizza que deseja pedir.
- Entre cada nova tentativa para obter um slot. Por exemplo, se o cliente não usar um tamanho de pizza reconhecido.
- Ao confirmar uma intenção. Por exemplo, ao confirmar um pedido de pizza.
- Para cumprir uma intenção. Por exemplo, para fazer um pedido de pizza.
- Depois de cumprir a intenção e antes que seu bot encerre a conversa. Por exemplo, para mudar para a intenção de pedir uma bebida.

Tópicos

Usar o console

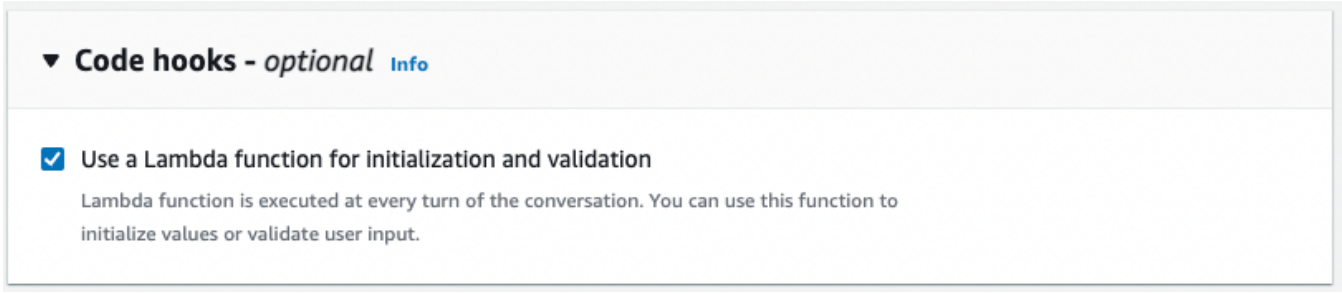
Anexar uma função do Lambda a um alias de bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha Bots no painel lateral esquerdo e, na lista de bots, escolha o nome do bot ao qual você deseja anexar uma função do Lambda.
3. No painel do lado esquerdo, selecione Aliases no menu Implantação.
4. Na lista de aliases, escolha o nome do alias ao qual você deseja anexar uma função do Lambda.
5. No painel Idiomas, selecione o idioma para o qual você deseja que uma função do Lambda use. Selecione Gerenciar idiomas no alias para adicionar um idioma, se ele não estiver no painel.
6. No menu suspenso Fonte, escolha o nome da função do Lambda que você deseja anexar.
7. No menu suspenso Versão da função do Lambda ou do alias, escolha a versão ou o alias da função do Lambda que você deseja usar. Em seguida, selecioneSalvar. A mesma função do Lambda é usada para todos os efeitos em uma linguagem com suporte pelo bot.

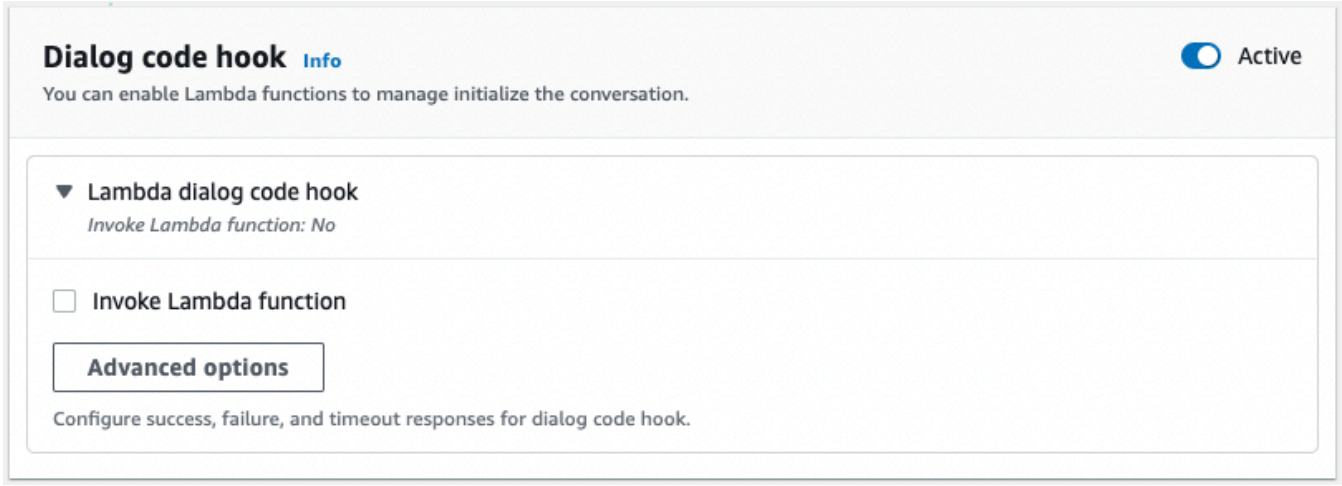
Defina uma intenção de invocar a função do Lambda

1. Depois de selecionar um bot, selecione Intenções no menu do lado esquerdo, abaixo do idioma do bot para o qual você deseja invocar a função do Lambda.
2. Escolha a intenção na qual você deseja invocar a função do Lambda para abrir o editor de intenção.
3. Há duas opções para configurar o gancho de código do Lambda:

    1. Para invocar a função do Lambda após cada etapa da conversa, vá até a seção Ganchos do código na parte inferior do editor de intenção e marque a caixa de seleção Usar uma função do Lambda para inicialização e validação, como na imagem a seguir:

    2. Como alternativa, use a seção Gancho de código de diálogo nos estágios de conversação para invocar a função do Lambda. A seção de Gancho de código de diálogo aparece da seguinte forma:

    Há duas maneiras de controlar como o Amazon Lex V2 chama o gancho de código para obter uma resposta:

    - Alterne o botão Ativo para marcá-lo como ativo ou inativo. Quando um gancho de código está ativo, o Amazon Lex V2 chama o gancho de código. Quando o gancho de código está inativo, o Amazon Lex V2 não executa o gancho de código.

- Expanda a seção de Gancho de código de diálogo do Lambda e marque a caixa de seleção Invocar função do Lambda para marcá-la como habilitada ou desabilitada. Você só pode habilitar ou desabilitar um gancho de código quando ele estiver marcado como ativo. Quando marcado como habilitado, o gancho de código é executado normalmente. Quando estiver desabilitado, o gancho de código não é chamado e o Amazon Lex V2 age como se o gancho de código tivesse sido retornado com sucesso. Para configurar respostas após o gancho do código de diálogo ser bem-sucedido, falhar ou expirar, selecione Opções avançadas

O gancho de código do Lambda pode ser invocado nos seguintes estágios de conversação:

- Para invocar a função como resposta inicial, vá até a seção Resposta inicial, expanda a seta ao lado de Resposta para reconhecer a solicitação do usuário e selecione Opções avançadas. Encontre a seção Gancho de código de diálogo na parte inferior do menu que aparece.
- Para invocar a função após a elicitação do slot, vá até a seção Slots, expanda a seta ao lado da Solicitação de slot relevante e selecione Opções avançadas. Encontre a seção de Gancho de código de diálogo na parte inferior do menu que aparece, logo acima dos Valores padrão.

  Você também pode invocar a função após cada elicitação. Para fazer isso, expanda a opção Bot elicita informações na seção Mensagens de slot, selecione Mais opções de prompt e marque a caixa de seleção ao lado de Invocar gancho de código do Lambda após cada elicitação.
- Para invocar a função para confirmação da intenção, vá até a seção Confirmação, expanda a seta ao lado de Solicitações para confirmar a intenção e selecione Opções avançadas. Encontre a seção Gancho de código de diálogo na parte inferior do menu que aparece.
- Para invocar a função para cumprimento da intenção, vá até a seção Cumprimento. Alterne o botão Ativo para definir o gancho de código como ativo. Expanda a seta ao lado de Sobre o atendimento bem-sucedido e selecione Opções avançadas. Marque a caixa de seleção ao lado de Usar uma função do Lambda para cumprimento na seção Gancho de código do Lambda de atendimento para definir o gancho de código como habilitado.

4. Depois de definir os estágios da conversa nos quais será invocada a função do Lambda, Crie o bot novamente para testar a função.

Usar operações de API

Anexar uma função do Lambda a um alias de bot

Se você estiver criando um novo alias de bot, use a operação CreateBotAlias para anexar uma função do Lambda. Para anexar uma função do Lambda a um alias de bot existente, use a operação UpdateBotAlias. Modifique o campo `botAliasLocaleSettings` para conter as configurações corretas:

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. O campo `botAliasLocaleSettings` é mapeado para um objeto cujas chaves são as localidades nas quais você deseja anexar a função do Lambda. Consulte Idiomas e locais compatíveis para obter uma lista das localidades com suporte e os códigos que são chaves válidas.
2. Para encontrar o `lambdaARN` para uma função do Lambda, abra o console do AWS Lambda em https://console.aws.amazon.com/lambda/home, selecione Funções na barra lateral esquerda e escolha a função a ser associada ao alias do bot. No lado direito da Visão geral da função, encontre o `lambdaARN` em ARN da função. Ele deve conter uma região, o ID da conta e o nome da função.
3. Para permitir que o Amazon Lex V2 invoque a função do Lambda para o alias, defina o campo `enabled` como `true`.

Defina uma intenção de invocar a função do Lambda

Para configurar a invocação da função do Lambda durante uma intenção, use a operação CreateIntent, se estiver criando uma nova intenção ou a operação UpdateIntent, se estiver invocando a função em uma intenção existente. Os campos que controlam a invocação da função do Lambda nas operações de intenção são `dialogCodeHook`, `initialResponseSetting`, `intentConfirmationSetting` e `fulfillmentCodeHook`.

Se você invocar a função durante a elicitação de um slot, use a operação CreateSlot, se estiver criando um novo slot ou a operação UpdateSlot para invocar a função em um slot existente. O campo que controla a invocação da função do Lambda nas operações do slot é o `slotCaptureSetting` do objeto `valueElicitationSetting`.

1. Para configurar o gancho de código de diálogo do Lambda para ser executado após cada turno da conversa, defina o campo `enabled` do seguinte objeto DialogCodeHookSettings no campo `dialogCodeHook` como `true`:

   ```
   "dialogCodeHook": {
       "enabled": boolean
   }
   ```

2. Como alternativa, você pode definir o gancho de código de diálogo do Lambda para ser executado somente em pontos específicos das conversas, modificando o `codeHook` e/ou o campo `elicitationCodeHook` nas estruturas que correspondem aos estágios da conversa nos quais você deseja invocar a função. Para usar o gancho de código de diálogo do Lambda para cumprimento de intenções, use o campo `fulfillmentCodeHook` na operação CreateIntent ou UpdateIntent. As estruturas e os usos desses três tipos de ganchos de código são os seguintes:

codeHook

O campo `codeHook` define as configurações para que o gancho de código seja executado em um determinado estágio da conversa. É um objeto DialogCodeHookInvocationSetting com a seguinte estrutura:

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
```

```
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- Altere o campo `active` para `true` para que o Amazon Lex V2 chame o gancho de código nesse ponto da conversa.
- Altere o campo `enableCodeHookInvocation` para `true` para o Amazon Lex V2 permitir que o gancho de código seja executado normalmente. Se você marcá-lo como `false`, o Amazon Lex V2 age como se o gancho de código tivesse sido retornado com sucesso.
- O `invocationLabel` indica a etapa da caixa de diálogo a partir da qual o gancho de código é invocado.
- Use o campo `postCodeHookSpecification` para especificar as ações e mensagens que ocorrem após o gancho de código ser bem-sucedido, falhar ou atingir o tempo limite.

elicitationCodeHook

O campo `elicitationCodeHook` define as configurações para que o gancho de código seja executado no caso de um slot ou slots precisarem ser reutilizados. Esse cenário pode ocorrer se a elicitação do slot falhar ou a confirmação da intenção for negada. O campo `elicitationCodeHook` é um objeto ElicitationCodeHookInvocationSetting com a seguinte estrutura:

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- Altere o campo `enableCodeHookInvocation` para `true` para o Amazon Lex V2 permitir que o gancho de código seja executado normalmente. Se você marcá-lo como `false`, o Amazon Lex V2 age como se o gancho de código tivesse sido retornado com sucesso.
- O `invocationLabel` indica a etapa da caixa de diálogo a partir da qual o gancho de código é invocado.

fulfillmentCodeHook

O campo `fulfillmentCodeHook` define as configurações do gancho de código a ser executado para cumprir a intenção. Ele é mapeado para o seguinte objeto FulfillmentCodeHookSettings:

```
"fulfillmentCodeHook": {
```

```
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- Altere o campo `active` para `true` para que o Amazon Lex V2 chame o gancho de código nesse ponto da conversa.
- Altere o campo `enabled` para `true` para o Amazon Lex V2 permitir que o gancho de código seja executado normalmente. Se você marcá-lo como `false`, o Amazon Lex V2 age como se o gancho de código tivesse sido retornado com sucesso.
- Use o campo `fulfillmentUpdatesSpecification` para especificar as mensagens que aparecem para atualizar o usuário durante o cumprimento da intenção e o tempo associado a elas.
- Use o campo `postFulfillmentStatusSpecification` para especificar as mensagens e ações que ocorrem após o gancho de código ser bem-sucedido, falhar ou atingir o tempo limite.

Você pode invocar o gancho de código do Lambda nos seguintes pontos de uma conversa definindo os campos `active` e `enableCodeHookInvocation`/`enabled` como `true`:

Durante a resposta inicial

Para invocar a função do Lambda na resposta inicial após o reconhecimento da intenção, use a estrutura `codeHook` no campo `initialResponse` da operação CreateIntent ou UpdateIntent. O campo `initialResponse` é mapeado para o seguinte objeto InitialResponseSetting:

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

Após a elicitação do slot ou durante a nova elicitação do slot

Para invocar a função do Lambda depois de obter um valor de slot, use o campo `slotCaptureSetting` dentro do campo `valueElicitation` da operação CreateSlot ou UpdateSlot. O campo `slotCaptureSetting` é mapeado para o seguinte objeto 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
}
```

- Para invocar a função do Lambda depois que a elicitação do slot for bem-sucedida, use o campo `codeHook`.
- Para invocar a função do Lambda depois que a elicitação de slots falhar e o Amazon Lex V2 tentar novamente a elicitação de slots, use o campo `elicitationCodeHook`.

Após confirmação ou negação da intenção

Para invocar a função do Lambda ao confirmar uma intenção, use o campo `intentConfirmationSetting` da operação CreateIntent ou UpdateIntent. O campo `intentConfirmation` é mapeado para o seguinte objeto 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
}
```

- Para invocar a função do Lambda após o usuário confirmar a intenção e seus slots, use o campo `codeHook`.
- Para invocar a função do Lambda depois que o usuário negar a confirmação da intenção e o Amazon Lex V2 tentar novamente a elicitação de slots, use o campo `elicitationCodeHook`.

Durante o cumprimento da intenção

Para invocar a função do Lambda para cumprir uma intenção, use o campo `fulfillmentCodeHook` na operação CreateIntent ou UpdateIntent. O campo `fulfillmentCodeHook` é mapeado para o seguinte objeto FulfillmentCodeHookSettings:

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. Depois de definir os estágios da conversa nos quais invocar a função do Lambda, use a operação `BuildBotLocale` para reconstruir o bot a fim de testar a função.

Depuração da função do Lambda

O Amazon CloudWatch Logs é uma ferramenta para rastrear chamadas de API e métricas que você pode usar para ajudar a depurar suas funções do Lambda. Quando você testa seu bot no console ou com chamadas de API, os logs do CloudWatch registram cada etapa da conversa. Se você usa uma função de impressão em seu código do Lambda, o CloudWatch também a exibe.

Para ver os logs do CloudWatch para sua função do Lambda

1. Faça login no AWS Management Console e abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
2. No menu Logs na barra lateral esquerda, selecione Grupos de logs.
3. Selecione seu grupo de logs de funções do Lambda, que deve ter o formato `/aws/lambda/`*`function-name`*.
4. A lista de Fluxos de log contém um log para cada sessão com um bot. Escolha um fluxo de logs para visualizá-lo.
5. Na lista de Eventos do log, selecione a seta para a direita ao lado de Data e hora para expandir os detalhes desse evento. Tudo o que você imprimir do seu código Lambda aparecerá como um evento de log. Use essas informações para depurar seu código.
6. Depois de depurar seu código, lembre-se de implantar a função do Lambda e, se estiver usando o console, de recarregar a janela de teste antes de Testar novamente o comportamento do bot.

Personalizar interações do bot

Conheça os seguintes atributos que você pode usar para personalizar as interações dos bots com os usuários expandindo e ajustando o comportamento padrão deles:

Tópicos

Analisar o sentimento dos enunciados dos usuários

É possível usar a análise de sentimento para determinar os sentimentos expressos em um enunciado do usuário. Com as informações de sentimento, é possível gerenciar o fluxo da conversa ou realizar a análise pós-chamada. Por exemplo, se o sentimento do usuário for negativo, você pode criar um fluxo para encaminhar a conversa para um atendente humano.

O Amazon Lex se integra ao Amazon Comprehend para detectar o sentimento do usuário. A resposta do Amazon Comprehend indica se o sentimento geral do texto é positivo, neutro, negativo ou misto. A resposta contém o sentimento mais provável do enunciado do usuário e as pontuações para cada uma das categorias de sentimento. A pontuação representa a probabilidade de o sentimento ter sido detectado corretamente.

Ative a análise de sentimento para um bot usando o console ou a API do Amazon Lex. Ative a análise de sentimentos em um alias para o bot. No console do Amazon Lex:

1. Escolha um alias.
2. Em Detalhes, selecione Editar.
3. Selecione Ativar análise de sentimentos para ativar ou desativar a análise de sentimentos.
4. Selecione Confirmar para salvar as alterações.

Se você estiver usando a API, chame a operação CreateBotAlias com o campo `detectSentiment` definido como `true`.

Quando a análise de sentimento está ativada, a resposta das operações RecognizeText e RecognizeUtterance retorna um campo chamado `sentimentResponse` na estrutura

`interpretations` com outros metadados. O campo `sentimentResponse` tem dois campos, `sentiment` e `sentimentScore`, que contêm o resultado da análise de sentimento. Se você estiver usando uma função do Lambda, o campo `sentimentResponse` será incluído nos dados do evento enviados para a função.

Veja a seguir um exemplo do campo `sentimentResponse` retornado como parte da resposta `RecognizeText` ou `RecognizeUtterance`.

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

O Amazon Lex chama o Amazon Comprehend em seu nome para determinar o sentimento em cada enunciado processado pelo bot. Ao ativar a análise de sentimento, você concorda com os termos e acordos de serviço do Amazon Comprehend. Para obter mais informações sobre a definição de preço do Amazon Comprehend, consulte Definição de preço do Amazon Comprehend.

Para obter mais informações sobre como funciona a análise de sentimento do Amazon Comprehend, consulte Determinar o sentimento no Guia do desenvolvedor do Amazon Comprehend.

Usar pontuações de confiança

Há duas etapas que o Amazon Lex V2 usa para determinar o que um usuário diz. A primeira, o reconhecimento automático de fala (ASR), cria uma transcrição do enunciado do usuário em áudio. A segunda, a compreensão de linguagem natural (NLU), determina o significado do enunciado do usuário para reconhecer a intenção do usuário ou o valor dos slots.

Por padrão, o Amazon Lex V2 retorna o resultado mais provável do ASR e da NLU. Às vezes, pode ser difícil para o Amazon Lex V2 determinar o resultado mais provável. Nesse caso, ele retorna vários resultados possíveis junto com uma pontuação de confiança que indica a probabilidade de o resultado estar correto. Uma pontuação de confiança é uma classificação fornecida pelo Amazon Lex V2 que demonstra a confiança relativa que ele tem no resultado. As pontuações de confiança variam de 0,0 a 1,0.

Você pode usar seu conhecimento de domínio com a pontuação de confiança para ajudar a determinar a interpretação correta do resultado de ASR ou NLU.

A pontuação de confiança do ASR, ou transcrição, é uma classificação do nível de confiança do Amazon Lex V2 na precisão de uma transcrição específica. A pontuação de confiança da NLU, ou intenção, é uma classificação do nível de confiança do Amazon Lex V2 na precisão da intenção especificada pela transcrição principal. Use a pontuação de confiança que melhor se adequa à aplicação.

Tópicos

Usar pontuações de confiança de intenção

Quando um usuário faz um enunciado, o Amazon Lex V2 usa a compreensão de linguagem natural (NLU) para entender a solicitação do usuário e retornar a intenção correta. Por padrão, o Amazon Lex V2 retorna a intenção mais provável definida pelo bot.

Em alguns casos, pode ser difícil para o Amazon Lex V2 determinar a intenção mais provável. Por exemplo, o usuário pode fazer um enunciado ambíguo ou pode haver duas intenções semelhantes. Para ajudar a determinar a intenção correta, você pode combinar seu conhecimento de domínio com as pontuações de confiança da NLU em uma lista de interpretações. Uma pontuação de confiança é uma classificação fornecida pelo Amazon Lex V2 que mostra o nível de confiança na correta identificação da intenção.

Para determinar a diferença entre duas intenções em uma interpretação, você pode comparar suas pontuações de confiança. Por exemplo, se uma intenção tem uma pontuação de confiança de 0,95 e outra tem uma pontuação de 0,65, é provável que a primeira intenção esteja correta. No entanto, se uma intenção tiver uma pontuação de 0,75 e outra tiver uma pontuação de 0,72, haverá ambiguidade entre as duas intenções que você poderá discriminar usando o conhecimento do domínio na aplicação.

Você também pode usar as pontuações de confiança para criar aplicações de teste que determinem se as mudanças nos enunciados de uma intenção fazem diferença no comportamento do bot. Por exemplo, é possível obter as pontuações de confiança das intenções de um bot usando um conjunto de enunciados e, em seguida, atualizar as intenções com novos enunciados. Você pode verificar as pontuações de confiança para ver se houve alguma melhora.

As pontuações de confiança que o Amazon Lex V2 retorna são valores comparativos. Não confie nelas como uma pontuação absoluta. Os valores podem mudar com base em melhorias no Amazon Lex V2.

O Amazon Lex V2 retorna a intenção mais provável e até quatro intenções alternativas com suas pontuações associadas na estrutura `interpretations` em cada resposta. O código JSON a seguir mostra a estrutura `interpretations` na resposta da operação RecognizeText:

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

AMAZON.FallbackIntent

O Amazon Lex V2 retorna `AMAZON.FallbackIntent` como a principal intenção em duas situações:

1. Se as pontuações de confiança de todas as possíveis intenções forem menores que o limite de confiança. Você pode usar o limite padrão ou definir seu próprio limite. Se você tiver o `AMAZON.KendraSearchIntent` configurado, o Amazon Lex V2 também o retornará nessa situação.
2. Se a confiança na interpretação do `AMAZON.FallbackIntent` for maior do que a confiança na interpretação de todas as outras intenções.

Observe que o Amazon Lex V2 não exibe uma pontuação de confiança para `AMAZON.FallbackIntent`.

Definir e alterar o limite de confiança

O limite de confiança deve ser um número entre 0,00 e 1,00. Você pode definir o limite para cada idioma do bot das seguintes maneiras:

Usar o console do Amazon Lex V2

- Para definir o limite ao adicionar um idioma ao bot com Adicionar idioma, você pode inserir o valor desejado no painel Limite de pontuação de confiança.
- Para atualizar o limite, você pode selecionar Editar no painel Detalhes do idioma em um idioma para o bot. Insira o valor desejado no painel Limite da pontuação de confiança.

Usar operações de API

- Para definir o limite, defina o parâmetro `nluIntentConfidenceThreshold` da operação CreateBotLocale.
- Para atualizar o limite de confiança, defina o parâmetro `nluIntentConfidenceThreshold` da operação UpdateBotLocale.

Gerenciamento de sessões

Para alterar a intenção que o Amazon Lex V2 usa em uma conversa com o usuário, você pode usar a resposta da função do Lambda do hook de código de diálogo ou usar as APIs de gerenciamento de sessão na aplicação personalizada.

Usar uma função do Lambda

Quando você usa uma função do Lambda, o Amazon Lex V2 a chama com uma estrutura JSON que contém a entrada para a função. A estrutura JSON contém um campo chamado `currentIntent` que contém a intenção que o Amazon Lex V2 identificou como a intenção mais provável para a expressão do usuário. A estrutura JSON também inclui um campo `alternativeIntents` que contém até quatro intenções adicionais que podem satisfazer a intenção do usuário. Cada intenção inclui um campo chamado `nluIntentConfidenceScore` que contém a pontuação de confiança que o Amazon Lex V2 atribuiu à intenção.

Para usar uma intenção alternativa, especifique-a na ação `ConfirmIntent` ou na caixa de diálogo `ElicitSlot` na função do Lambda.

Para mais informações, consulte Habilitando a lógica personalizada com as funções do AWS Lambda.

Usar a API de gerenciamento de sessões

Para usar uma intenção diferente da intenção atual, use a operação PutSession. Por exemplo, se você decidir que a primeira alternativa é preferível à intenção escolhida pelo Amazon Lex V2, use a operação `PutSession` para alterar as intenções de forma que a próxima intenção com a qual o usuário vai interagir seja aquela que você selecionou.

Para mais informações, consulte Gerenciamento de sessões com a API Amazon Lex V2.

Como usar pontuações de confiança na transcrição de voz

Quando um usuário faz um enunciado por voz, o Amazon Lex V2 usa o reconhecimento automático de fala (ASR) para transcrever a solicitação do usuário antes que ela seja interpretada. Por padrão, o Amazon Lex V2 usa a transcrição mais provável do áudio para interpretação.

Em alguns casos, pode haver mais de uma possível transcrição do áudio. Por exemplo, um usuário pode fazer um enunciado com um som ambíguo, como “Meu nome é John”, que pode ser entendido como “Meu nome é Juan”. Nesse caso, você pode usar técnicas de desambiguação ou combinar seu conhecimento de domínio com a pontuação de confiança da transcrição para ajudar a determinar qual transcrição é a correta em uma lista de transcrições.

O Amazon Lex V2 inclui a transcrição principal e até duas transcrições alternativas para a entrada do usuário na solicitação da função de hook de código Lambda. Cada transcrição contém uma pontuação de confiança que indica a probabilidade de ser a transcrição correta. Cada transcrição também inclui quaisquer valores de slot inferidos a partir da entrada do usuário.

Você pode comparar as pontuações de confiança de duas transcrições para determinar se há ambiguidade entre elas. Por exemplo, se uma transcrição tem uma pontuação de confiança de 0,95 e a outra tem uma pontuação de confiança de 0,65, a primeira transcrição provavelmente está correta e a ambiguidade entre elas é baixa. Se as duas transcrições tiverem pontuações de confiança de 0,75 e 0,72, a ambiguidade entre elas é alta. Talvez você consiga discriminá-las usando seu conhecimento de domínio.

Por exemplo, se os valores de slot inferidos em duas transcrições com uma pontuação de confiança de 0,75 e 0,72 forem “John” e “Juan”, você poderá consultar os usuários no banco de dados para

verificar a existência desses nomes e eliminar uma das transcrições. Se “John” não for um usuário no banco de dados e “Juan” for, você poderá usar o hook de código de diálogo para alterar o valor do slot inferido para “Juan”, o primeiro nome.

As pontuações de confiança que o Amazon Lex V2 retorna são valores comparativos. Não confie nelas como uma pontuação absoluta. Os valores podem mudar com base em melhorias no Amazon Lex V2.

As pontuações de confiança da transcrição de áudio estão disponíveis somente nos idiomas inglês (GB) (en_GB) e inglês (EUA) (en_US). As pontuações de confiança são compatíveis somente com entradas de áudio de 8 kHz. As pontuações de confiança da transcrição não são fornecidas para entrada de áudio da janela de teste no console do Amazon Lex V2 porque ele usa entrada de áudio de 16 kHz.

> **Note**
>
> Antes de usar as pontuações de confiança da transcrição de áudio com um bot existente, é preciso primeiro reconstruir o bot. As versões existentes de um bot não oferecem suporte a pontuações de confiança na transcrição. É necessário criar uma nova versão do bot para usá-las.

Você pode usar pontuações de confiança para vários padrões de design de conversas:

- Se a pontuação de confiança mais alta cair abaixo de um limite devido a um ambiente ruidoso ou à baixa qualidade do sinal, você poderá solicitar que o usuário faça a mesma pergunta para capturar um áudio de melhor qualidade.
- Se várias transcrições tiverem pontuações de confiança semelhantes para valores de slot, como “John” e “Juan”, você poderá comparar os valores com um banco de dados preexistente para eliminar entradas ou solicitar que o usuário selecione um dos dois valores. Por exemplo, “diga 1 para John ou 2 para Juan”.
- Se a lógica de negócios exigir a troca de intenção com base em palavras-chave específicas em uma transcrição alternativa com uma pontuação de confiança próxima à transcrição principal, você poderá alterar a intenção usando a função do Lambda do hook de código de diálogo ou usando operações de gerenciamento de sessão. Para obter mais informações, consulte Gerenciamento de sessões.

O Amazon Lex V2 envia a seguinte estrutura JSON com até três transcrições para a entrada do usuário na função de hook de código 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"
                            ]
                        }
                    }
                ]
            }
```

```
            }
        }
    ]
```

A estrutura JSON contém texto de transcrição, a intenção que foi identificada para o enunciado e valores para quaisquer slots detectados no enunciado. Para a entrada de texto do usuário, as transcrições contêm uma única transcrição com uma pontuação de confiança de 1,0.

O conteúdo das transcrições depende do rumo da conversa e da intenção reconhecida.

Na primeira etapa, a escolha da intenção, o Amazon Lex V2 determina as três principais transcrições. Para a transcrição principal, ele retorna a intenção e quaisquer valores de slot inferidos na transcrição.

Nas etapas subsequentes, na escolha de slots, os resultados dependem da intenção inferida para cada uma das transcrições, conforme a seguir.

- Se a intenção inferida para a transcrição principal for a mesma da etapa anterior e todas as outras transcrições tiverem a mesma intenção, então:
  - Todas as transcrições contêm valores de slot inferidos.

- Se a intenção inferida para a transcrição principal for diferente da etapa anterior e todas as outras transcrições tiverem a mesma intenção, então:
  - A transcrição principal contém os valores de slot inferidos para a nova intenção.
  - As outras transcrições contêm a intenção anterior e valores de slot inferidos para ela.

- Se a intenção inferida para a transcrição principal for diferente da etapa anterior, uma transcrição for igual à intenção anterior e uma transcrição tiver uma intenção diferente, então
  - A transcrição principal contém a nova intenção inferida e quaisquer valores de slot inferidos no enunciado.
  - A transcrição que tem a intenção inferida anterior contém valores de slot inferidos para essa intenção.
  - A transcrição com a intenção diferente não possui um nome de intenção inferida nem valores de slot inferidos.

- Se a intenção inferida para a transcrição principal for diferente da etapa anterior e todas as outras transcrições tiverem intenções diferentes, então:
  - A transcrição principal contém a nova intenção inferida e quaisquer valores de slot inferidos no enunciado.
  - Outras transcrições não contêm intenções inferidas nem valores de slots inferidos.

- Se a intenção inferida para as duas transcrições principais for a mesma e diferente da etapa anterior, e a terceira transcrição tiver uma intenção diferente, então:
  - As duas transcrições principais contêm a nova intenção inferida e quaisquer valores de slot inferidos no enunciado.
  - A terceira transcrição não tem nome de intenção nem valores de slot resolvidos.

Gerenciamento de sessões

Para alterar a intenção que o Amazon Lex V2 usa em uma conversa com o usuário, use a resposta da função do Lambda do hook de código de diálogo. Ou você pode usar as APIs de gerenciamento de sessão no aplicativo personalizado.

Usar uma função do Lambda

Quando você usa uma função do Lambda, o Amazon Lex V2 a chama com uma estrutura JSON que contém a entrada para a função. A estrutura JSON contém um campo chamado `transcriptions` que contém as possíveis transcrições que o Amazon Lex V2 determinou para o enunciado. O campo `transcriptions` contém de uma a três transcrições possíveis, cada uma com uma pontuação de confiança.

Para usar a intenção de uma transcrição alternativa, especifique-a na ação de diálogo `ConfirmIntent` ou `ElicitSlot` na função do Lambda. Para usar um valor de slot de uma transcrição alternativa, defina o valor no campo `intent` na resposta da função do Lambda. Para obter mais informações, consulte Habilitando a lógica personalizada com as funções do AWS Lambda.

Código de exemplo

O exemplo de código a seguir é uma função do Lambda em Python que usa transcrições de áudio para melhorar a experiência de conversação do usuário.

Para usar o código de exemplo, é necessário ter:

- Um bot com um idioma, inglês (GB) (en_GB) ou inglês (EUA) (en_US).
- Uma intenção, OrderBirthStone. Certifique-se de que a opção Usar uma função do Lambda para inicialização e validação esteja selecionada na seção Hooks de código da definição da intenção.
- A intenção deve ter dois slots, “BirthMonth” e “Name”, ambos do tipo `AMAZON.AlphaNumeric`.
- Um alias com a função do Lambda definida. Para obter mais informações, consulte Criar e anexar uma função do Lambda a um alias de 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)
```

Usar a API de gerenciamento de sessões

Para usar uma intenção diferente da intenção atual, use a operação PutSession. Por exemplo, se você decidir que a primeira alternativa é preferível à intenção escolhida pelo Amazon Lex V2, use a operação `PutSession` para alterar as intenções. Dessa forma, a próxima intenção com a qual o usuário interagirá será aquela que você selecionou.

Você também pode usar a operação `PutSession` para alterar o valor do slot na estrutura `intent` para usar um valor de uma transcrição alternativa.

Para obter mais informações, consulte Gerenciamento de sessões com a API Amazon Lex V2.

Personalizar transcrições de fala

Às vezes, o comportamento padrão do bot pode resultar em transcrições de fala imprecisas. Os atributos a seguir estão disponíveis para ajudar o bot a reconhecer palavras ou nomes menos comuns ou facilmente confundidos.

Tópicos

- Melhorar o reconhecimento de fala com um vocabulário personalizado
- Melhorar o reconhecimento dos valores de slots com sugestões de runtime
- Capturar valores de slots com estilos de soletração

Melhorar o reconhecimento de fala com um vocabulário personalizado

Você pode fornecer ao Amazon Lex V2 mais informações sobre como processar conversas de áudio com um bot ao criar um vocabulário personalizado em um idioma específico. Um vocabulário personalizado é uma lista de palavras específicas que você deseja que o Amazon Lex V2 reconheça na entrada de áudio. Geralmente, esses são nomes próprios ou palavras específicas do domínio que o Amazon Lex V2 não reconhece.

Por exemplo, suponha que você tenha um bot de suporte técnico. Você pode adicionar “backup” a um vocabulário personalizado para ajudar o bot a transcrever o áudio corretamente como “backup”, mesmo quando o áudio soar como “pack up”. Um vocabulário personalizado também pode ajudar a

reconhecer palavras raras no áudio, como “solvência”, para serviços financeiros ou nomes próprios, como “Cognito” ou “Monitron”.

Conceitos básicos de vocabulário personalizado

- Um vocabulário personalizado funciona na transcrição da entrada de áudio para um bot. É necessário fornecer exemplos de enunciados para reconhecer uma intenção ou um valor de slot.
- Um vocabulário personalizado é exclusivo para um idioma específico. É necessário configurar vocabulários personalizados de forma independente para cada idioma. Vocabulários personalizados são compatíveis somente com os idiomas inglês (Reino Unido) e inglês (EUA).
- Vocabulários personalizados estão disponíveis com integrações de central de atendimento compatíveis com o Amazon Lex V2. A janela de teste no console do Amazon Lex V2 oferece suporte a vocabulários personalizados para todos os bots do Amazon Lex V2 criados em 31 de julho de 2022 ou após. Se você tiver problemas com vocabulários personalizados na janela de teste, reconstrua o bot e tente novamente.

O Amazon Lex V2 usa vocabulários personalizados para escolher intenções e slots. O mesmo arquivo de vocabulário personalizado é usado para intenções e slots. Você pode desativar seletivamente o recurso de vocabulário personalizado de um slot ao adicionar um tipo de slot.

Escolher uma intenção – Você pode criar um vocabulário personalizado para escolher uma intenção. Essas frases são usadas para transcrição quando o bot está determinando a intenção do usuário. Por exemplo, se você configurou a frase “backup” no vocabulário personalizado, o Amazon Lex V2 transcreverá a entrada do usuário para “você pode fazer backup das minhas fotos, por favor?” — mesmo quando o áudio soar como “você pode pack up minhas fotos, por favor”. É possível especificar o grau de prioridade para cada frase configurando um peso de 0, 1, 2 ou 3. Você também pode especificar uma representação alternativa para a frase na saída final de fala em texto adicionando um campo `displayAs`.

As frases de vocabulário personalizadas usadas para melhorar a transcrição durante a escolha da intenção não afetam as transcrições ao escolher slots. Para obter mais informações sobre a criação de um vocabulário personalizado para escolher intenções, consulte Criar um vocabulário personalizado para escolher intenções e slots.

Escolher slots personalizados – Você pode usar um vocabulário personalizado para melhorar o reconhecimento de slots em conversas de áudio. Para melhorar a capacidade do bot Amazon Lex V2 de reconhecer valores de slot, crie um slot personalizado e adicione os valores de slot ao slot personalizado, depois escolha Usar valores de slot como vocabulário personalizado. Exemplos de

valores de slots incluem nomes de produtos, catálogos ou nomes próprios. Não use palavras ou frases comuns como “sim” e “não” em vocabulários personalizados.

Depois que os valores do slot são adicionados, esses valores são usados para melhorar o reconhecimento do slot quando o bot está aguardando uma entrada para o slot personalizado. Esses valores não são usados para transcrição ao escolher uma intenção. Para mais informações, consulte Adicionar tipos de slot.

Práticas recomendadas para criar um vocabulário personalizado

Escolher uma intenção

- Vocabulários personalizados funcionam melhor quando são usados para direcionar palavras ou frases específicas. Somente adicione palavras a um vocabulário personalizado se elas não forem prontamente reconhecidas pelo Amazon Lex V2.
- Decida quanto peso atribuir a uma palavra com base na frequência com que a palavra não é reconhecida na transcrição e na raridade da palavra na entrada. Palavras difíceis de pronunciar exigem um peso maior.
- Use um conjunto de teste representativo para determinar se um peso é apropriado. Você pode coletar um conjunto de testes de áudio ativando o registro em log de áudio nos logs de conversas.
- Evite usar palavras curtas como “em”, “isso”, “para”, “sim”, “não” em um vocabulário personalizado.

Escolher um slot personalizado

- Adicione os valores ao tipo de slot personalizado que você espera que seja reconhecido. Adicione todos os valores de slot possíveis para o tipo de slot personalizado, independentemente de serem comuns ou raros.
- Ative a opção somente quando o tipo de slot personalizado contiver uma lista de valores de catálogo ou entidades, como nomes de produtos ou fundos mútuos.
- Desative a opção se o tipo de slot for usado para capturar frases genéricas como “sim”, “não”, “não sei”, “talvez” ou palavras genéricas como “um”, “dois”, “três”.
- Limite o número de valores de slots e sinônimos a 500 ou menos para obter o melhor desempenho.

Insira acrônimos ou outras palavras cujas letras devem ser pronunciadas individualmente como letras únicas separadas por um ponto ou um espaço. Não use letras individuais, a menos que elas

façam parte de uma frase, como “J. P. Morgan” ou “A. W. S.”. É possível usar letras maiúsculas ou minúsculas para definir um acrônimo.

Criar um vocabulário personalizado para escolher intenções e slots

Você pode usar o console do Amazon Lex V2 para criar e gerenciar um vocabulário personalizado ou pode usar as operações de API do Amazon Lex V2. Há duas maneiras de criar um vocabulário personalizado por meio do console:

Console

Importe o vocabulário personalizado no console:

1. Abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home
2. Na lista de bots, escolha o bot ao qual você deseja adicionar o vocabulário personalizado.
3. Na página de detalhes do bot, na seção Adicionar idiomas, escolha Exibir idiomas.
4. Na lista de idiomas, escolha o idioma ao qual você deseja adicionar o vocabulário personalizado.

Crie um novo vocabulário personalizado diretamente pelo console:

1. Clique em Criar na seção Vocabulário personalizado da página de detalhes do idioma. Isso abrirá uma janela de edição sem nenhum vocabulário personalizado presente.
2. Adicione entradas para frase, DisplayAs e peso conforme necessário. Você também pode fazer edições embutidas nos itens adicionados atualizando os campos ou excluindo-os da lista.
3. Clique em Salvar. Observação: o novo vocabulário personalizado só é salvo no bot depois que você clica em Salvar.
4. Você pode continuar fazendo edições embutidas nesta página e clicar em Salvar quando terminar.
5. Essa página também permite importar, exportar e excluir um arquivo de vocabulário personalizado do menu suspenso no canto superior direito.

API

Use a API **`ListCustomVocabularyItems`** para visualizar as entradas de vocabulário personalizadas:

1. Use a operação `ListCustomVocabularyItems` para visualizar as entradas de vocabulário personalizadas. O corpo da solicitação será semelhante a este:

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. Observe que `maxResults` e `nextToken` são campos opcionais para o corpo da solicitação.
3. A resposta da operação `ListCustomVocabularyItems` é semelhante a esta:

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

Use a API **`BatchCreateCustomVocabularyItem`** para criar novas entradas de vocabulário personalizadas:

1. Se a localidade do bot ainda não tiver um vocabulário personalizado criado, siga as etapas para usar o StartImport para criar um vocabulário personalizado.
2. Depois que o vocabulário personalizado for criado, use a operação `BatchCreateCustomVocabularyItem` para criar novas entradas de vocabulário personalizado. O corpo da solicitação será semelhante a este:

```
{
```

```
   "customVocabularyItemList": [
        {
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Observe que `weight` e `displayAs` são campos opcionais para o corpo da solicitação.
4. A resposta do `BatchCreateCustomVocabularyItem` será semelhante a esta:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Como essa é uma operação em lote, a solicitação não falhará se um dos itens não for criado. A lista de erros conterá informações sobre o motivo pelo qual a operação falhou nessa entrada específica. A lista de recursos conterá todas as entradas que foram criadas com êxito.
6. Para `BatchCreateCustomVocabularyItem`, é possível encontrar estes tipos de erros:

   - `RESOURCE_DOES_NOT_EXIST`: o vocabulário personalizado não existe. Siga as etapas para criar um vocabulário personalizado antes de chamar essa operação.
   - `DUPLICATE_INPUT`: a lista de entradas contém frases duplicadas.

- `RESOURCE_ALREADY_EXISTS`: a frase fornecida para a entrada já existe no vocabulário personalizado.
- `INTERNAL_SERVER_FAILURE`: houve um erro no back-end ao processar sua solicitação. Isso pode indicar uma interrupção do serviço ou outro problema.

Use a API **`BatchDeleteCustomVocabularyItem`** para excluir entradas de vocabulário personalizado existentes:

1. Se a localidade do bot ainda não tiver um vocabulário personalizado criado, siga as etapas em Usar o StartImport para criar um vocabulário personalizado.
2. Depois que o vocabulário personalizado for criado, use a operação `BatchDeleteCustomVocabularyItem` para excluir entradas de vocabulário personalizado existentes. O corpo da solicitação será semelhante a este:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. A resposta do `BatchDeleteCustomVocabularyItem` será semelhante a esta:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
```

```
            "displayAs": "string"
        }
    ]
}
```

4. Como essa é uma operação em lote, a solicitação não falhará se um dos itens não for excluído. A lista de erros conterá informações sobre o motivo pelo qual a operação falhou nessa entrada específica. A lista de recursos conterá todas as entradas que foram excluídas com êxito.
5. Para `BatchDeleteCustomVocabularyItem`, é possível encontrar estes tipos de erros:
   - `RESOURCE_DOES_NOT_EXIST`: a entrada de vocabulário personalizado que você está tentando excluir não existe.
   - `INTERNAL_SERVER_FAILURE`: houve um erro no back-end ao processar sua solicitação. Isso pode indicar uma interrupção do serviço ou outro problema.

Use a API **`BatchUpdateCustomVocabularyItem`** para atualizar as entradas de vocabulário personalizado existentes:

1. Se a localidade do bot ainda não tiver um vocabulário personalizado criado, siga as etapas em Usar o StartImport para criar um vocabulário personalizado.
2. Depois que o vocabulário personalizado for criado, use a operação `BatchUpdateCustomVocabularyItem` para atualizar entradas de vocabulário personalizado existentes. O corpo da solicitação será semelhante a este:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Observe que `weight` e `displayAs` são campos opcionais para o corpo da solicitação.
4. A resposta do `BatchUpdateCustomVocabularyItem` será semelhante a esta:

```
{
```

```
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Como essa é uma operação em lote, a solicitação não falhará se um dos itens não for excluído. A lista de erros conterá informações sobre o motivo pelo qual a operação falhou nessa entrada específica. A lista de recursos conterá todas as entradas que foram atualizadas com êxito.
6. Para `BatchUpdateCustomVocabularyItem`, é possível encontrar estes tipos de erros:

   - `RESOURCE_DOES_NOT_EXIST`: a entrada de vocabulário personalizado que você está tentando atualizar não existe.
   - `DUPLICATE_INPUT`: a lista de entradas contém itemIds duplicados.
   - `RESOURCE_ALREADY_EXISTS`: a frase fornecida para a entrada já existe no vocabulário personalizado.
   - `INTERNAL_SERVER_FAILURE`: houve um erro no back-end ao processar sua solicitação. Isso pode indicar uma interrupção do serviço ou outro problema.

Criar um arquivo de vocabulário personalizado

Um arquivo de vocabulário personalizado é uma lista de valores separada por tabulação que contém a frase a ser reconhecida, um peso para aumentar sua prioridade e um campo `displayAs` que substituirá a frase na transcrição da fala. Frases com um valor de prioridade mais alta têm maior probabilidade de serem usadas quando aparecem na entrada de áudio.

O arquivo de vocabulário personalizado deve ser nomeado como **`CustomVocabulary.tsv`** e compactado em um arquivo zip antes de ser importado. O arquivo .zip deve ter um tamanho menor que 300 MB. O número máximo de frases em um vocabulário personalizado é 500.

- phrase – Uma a quatro palavras que devem ser reconhecidas. Separe as palavras na frase com espaços. Não é possível ter frases duplicadas no arquivo. O campo phrase é obrigatório.
- weight – O grau de prioridade do reconhecimento da frase. O valor é um inteiro 0, 1, 2 ou 3. Se você não especificar um peso, o valor padrão será 1. Decida o peso com base na frequência com que a palavra não é reconhecida na transcrição e na raridade da palavra na entrada. O peso 0 significa que nenhuma prioridade será aplicada e que a entrada será usada apenas para realizar substituições usando o campo `displayAs`.
- displayAs – Define como você deseja que a frase apareça na saída da transcrição. Esse é um campo opcional no vocabulário personalizado.

O arquivo de vocabulário personalizado deve conter uma linha de cabeçalho com os cabeçalhos “phrase”, “weight” e “displayAs”. Os cabeçalhos podem estar em qualquer ordem, mas devem seguir a nomenclatura acima.

O exemplo a seguir é um arquivo de vocabulário personalizado. O caractere de tabulação necessário para separar a frase, o peso e o displayAs é representado pelo texto “[TAB]”. Se você usar esse exemplo, substitua o texto por um caractere de tabulação.

```
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
```

Melhorar o reconhecimento dos valores de slots com sugestões de runtime

Com as sugestões de runtime, é possível fornecer ao Amazon Lex V2 um conjunto de valores de slots com base no contexto para obter melhor reconhecimento em conversas de áudio e melhores resoluções de slots. Você pode usar as sugestões de runtime para fornecer uma lista de frases em runtime que se tornam candidatas à resolução de um valor de slot.

Por exemplo, se um usuário que interage com um bot de reserva de voos viaja com frequência para São Francisco, Jacarta, Seul e Moscou, é possível configurar sugestões de runtime com uma lista dessas quatro cidades ao solicitar o destino para melhorar o reconhecimento das cidades frequentemente visitadas.

As sugestões de runtime estão disponíveis somente nos idiomas inglês (EUA) e inglês (Reino Unido). Elas podem ser usadas com os seguintes tipos de slots:

- Tipos de slot personalizados
- AMAZON.City
- AMAZON.Country
- AMAZON.FirstName
- AMAZON.LastName
- AMAZON.State
- AMAZON.StreetName

Conceitos básicos de sugestões de runtime

- As sugestões de runtime são usadas somente ao solicitar um valor de slot de um usuário.
- Quando você usa sugestões de runtime, os valores das sugestões têm preferência sobre valores semelhantes. Por exemplo, para um bot de pedidos de comida, é possível definir uma lista de itens do menu como sugestões de runtime ao solicitar opções de comida em um slot personalizado para dar preferência a “filé” em vez de palavras semelhantes como “fila”.
- Se a entrada do usuário for diferente dos valores fornecidos nas sugestões de runtime, a entrada original do usuário será usada para o slot.
- Para tipos de slots personalizados, os valores fornecidos como sugestões de runtime serão usados para a resolução do slot, mesmo que não façam parte do slot personalizado durante a criação do bot.
- As sugestões de runtime são compatíveis somente com entradas de áudio de 8 kHz. Elas estão disponíveis com integrações de central de atendimento compatíveis com o Amazon Lex V2. Não são fornecidas sugestões de runtime para entrada de áudio da janela de teste no console do Amazon Lex V2, pois ela utiliza entrada de áudio de 16 kHz.

> **Note**
>
> Antes de usar sugestões de runtime com um bot existente, é necessário primeiro reconstruir o bot. As versões existentes do bot não são compatíveis com sugestões de runtime. É necessário criar uma nova versão do bot para usá-las.

Você pode enviar sugestões de runtime para o Amazon Lex V2 usando a operação PutSession, RecognizeText, RecognizeUtterance ou StartConversation. Você também pode adicionar sugestões de runtime usando uma função do Lambda.

Você pode enviar sugestões de runtime no início de uma conversa para configurar as sugestões para cada slot usado no bot ou pode enviar sugestões como parte do estado da sessão durante uma conversa. O atributo `runtimeHints` mapeia um slot para as sugestões desse slot.

Depois de enviar uma sugestão de runtime para o Amazon Lex V2, ela permanece presente em cada etapa da conversa até que a sessão termine. Se você enviar uma estrutura `runtimeHints` nula, as sugestões existentes serão usadas. É possível modificar as sugestões da seguinte forma:

- Enviar uma nova estrutura `runtimeHints` para o bot. O conteúdo da nova estrutura substitui os existentes.
- Enviar uma estrutura `runtimeHints` vazia para o bot. Isso limpa as sugestões de runtime do bot.

Adicionar valores de slot no contexto

Adicione contexto ao bot fornecendo os valores de slot esperados como sugestões de runtime quando a aplicação tiver informações sobre o próximo enunciado provável do usuário. Adicione um hook de código de diálogo do Lambda ao bot (consulte Habilitando a lógica personalizada com as funções do AWS Lambda para obter mais informações) e use o campo proposedNextState em Interpretar o formato do evento de entrada para determinar as sugestões de runtime que é preciso incluir para melhorar a conversa com o usuário.

Por exemplo, em um aplicativo bancário, é possível gerar uma lista de apelidos de conta para um usuário específico e, em seguida, usar a lista ao solicitar a conta que o usuário deseja acessar.

Envie sugestões de runtime no início da conversa quando tiver contexto para ajudar o bot a interpretar as informações do usuário. Por exemplo, se você tiver o número de telefone do usuário, poderá usar essa informação para localizar o usuário, permitindo assim o uso da operação

`PutSession` ou `StartConversation` para fornecer sugestões de nome e sobrenome ao bot, caso esteja solicitando o nome do usuário para validar suas credenciais.

Durante uma conversa, você pode coletar informações de um valor de slot que podem ajudar com outro valor de slot. Por exemplo, em um aplicativo de atendimento automotivo, quando você tem o número da conta do usuário, é possível fazer uma busca para descobrir os carros que o cliente possui e enviá-los como sugestões para outro slot.

Insira acrônimos ou outras palavras cujas letras devem ser pronunciadas individualmente como letras únicas separadas por um ponto ou um espaço. Não use letras individuais, a menos que elas façam parte de uma frase, como “J. P. Morgan” ou “A.W.S”. É possível usar letras maiúsculas ou minúsculas para definir um acrônimo.

Adicionar sugestões a um slot

Para adicionar sugestões de runtime a um slot, use a estrutura `runtimeHints` que faz parte da estrutura `sessionState`. Veja a seguir um exemplo da estrutura `runtimeHints`. Ela fornece sugestões para dois slots, “FirstName” e “LastName” para a intenção “MakeAppointment”.

```
{
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
```

```
                            "phrase": "Stiles"
                        },
                        {
                            "phrase": "Major"
                        }
                    ]
                }
            }
        }
    }
}
```

Você também pode usar uma função do Lambda para adicionar sugestões de runtime durante uma conversa. Para adicionar sugestões de runtime, adicione a estrutura `runtimeHints` ao estado da sessão na resposta que a função do Lambda envia para o Amazon Lex V2. Para mais informações, consulte Preparando o formato de resposta.

É necessário especificar um `intentName` e um `slotName` válidos na solicitação, caso contrário, o Amazon Lex V2 retornará um erro de runtime.

Capturar valores de slots com estilos de soletração

O Amazon Lex V2 fornece slots integrados para capturar informações específicas do usuário, como nome, sobrenome, endereço de e-mail ou identificadores alfanuméricos. Por exemplo, você pode usar o slot `AMAZON.LastName` para capturar sobrenomes como “Jackson” ou “Garcia”. No entanto, o Amazon Lex V2 pode se confundir com sobrenomes difíceis de pronunciar ou que não são comuns em uma localidade, como “Xiulan”. Para capturar esses nomes, você pode pedir ao usuário para fornecer as informações soletrando por letra ou soletrando por palavra.

O Amazon Lex V2 oferece três estilos de escolha de slots para você usar. Quando você define um estilo de escolha de slots, ele muda a forma como o Amazon Lex V2 interpreta a entrada do usuário.

Soletrar por letra – Com esse estilo, você pode instruir o bot a ouvir a soletração em vez da frase inteira. Por exemplo, para capturar um sobrenome como “Xiulan”, você pode pedir ao usuário para soletrar o sobrenome, uma letra por vez. O bot capturará a soletração e converterá as letras em uma palavra. Por exemplo, se o usuário disser “x i u l a n”, o bot captura o sobrenome como “xiulan”.

Soletrar por palavra – Em conversas de voz, especialmente usando o telefone, existem algumas letras, como “t”, “b”, “p”, que têm sons semelhantes. Quando a captura de valores alfanuméricos ou

soletração de nomes resulta em um valor incorreto, você pode solicitar que o usuário forneça uma palavra de identificação junto com a letra. Por exemplo, se a resposta de voz a uma solicitação de ID de reserva for “abp123”, seu bot poderá reconhecer a frase “abb123”. Se esse for um valor incorreto, você pode pedir ao usuário que forneça a entrada como “a como em alfa, b como em bola, p como em Pedro, um, dois, três”. O bot converterá a entrada em “abp123”.

Ao usar a soletração por palavra, é possível usar os seguintes formatos:

- “como em” (a como em maçã)
- “de” (a de maçã)
- “como” (a como maçã)

Padrão – Esse é o estilo natural de captura de slots que usa a pronúncia de palavras. Por exemplo, ele pode capturar nomes como “John Stiles” naturalmente. Se um estilo de escolha de slots não for especificado, o bot usará o estilo padrão. Para os tipos de slot de código `AMAZON.AlphaNumeric` e `AMAZON.UKPostal`, o estilo padrão é compatível com a entrada soletrada por letra.

Se o nome “Xiulan” for falado usando uma mistura de letras e palavras, como “x como em xarope, i, u, l como em leão, a, n”, o estilo de escolha do slot deve ser definido como soletração por palavra. O estilo de soletração por letra não o reconhecerá.

É necessário criar uma interface de voz que capture os valores dos slots com um estilo conversacional natural para uma melhor experiência. Para entradas que não foram capturadas corretamente usando o estilo natural, você pode solicitar novamente ao usuário e definir o estilo de escolha do slot como soletração por letra ou por palavra.

É possível usar estilos de soletração por palavra e soletração por letra para os seguintes tipos de slots nos idiomas inglês (EUA), inglês (Reino Unido) e inglês (Austrália):

- AMAZÔNIA. AlphaNumeric
- AMAZÔNIA. EmailAddress
- AMAZÔNIA. FirstName
- AMAZÔNIA. LastName
- AMAZON.UK PostalCode
- Tipos de slot personalizados

Ativação da soletração

Você ativa a soletração por letra e a soletração por palavra em runtime quando está escolhendo slots do usuário. Você pode definir o estilo de soletração com a operação PutSession, RecognizeText, RecognizeUtterance ou StartConversation. Também é possível ativar a soletração por letra e a soletração por palavra usando uma função do Lambda.

Defina o estilo de soletração usando o campo `dialogAction` do campo `sessionState` na solicitação de uma das operações de API mencionadas acima ou ao configurar a resposta do Lambda (consulte Preparando o formato de resposta para obter mais informações). Só é possível definir o estilo quando o tipo de ação da caixa de diálogo é `ElicitSlot` e quando o slot a ser escolhido é um dos tipos de slot compatíveis.

O código JSON a seguir mostra o campo `dialogAction` definido para usar o estilo de soletração por palavra:

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
    "type": "ElicitSlot"
}
```

O campo `slotElicitationStyle` pode ser definido como `SpellByLetter`, `SpellByWord` ou `Default`. Se você não especificar um valor, o valor padrão é definido como `Default`.

> **Note**
>
> Não é possível habilitar estilos de escolha de soletração por letra ou por palavra no console.

Código de exemplo

Geralmente, a alteração do estilo de soletração é realizada se a primeira tentativa de converter um valor de slot não funcionar. O exemplo de código a seguir é uma função do Lambda em Python que usa o estilo de soletração por palavra na segunda tentativa de resolver um slot.

Para usar o código de exemplo, é necessário ter:

- Um bot com um idioma, inglês (GB) (en_GB).

- Uma intenção, “CheckAccount” com um exemplo de enunciado, “Eu gostaria de verificar minha conta”. Certifique-se de que a opção Usar uma função do Lambda para inicialização e validação esteja selecionada na seção Hooks de código da definição da intenção.
- A intenção deve ter um slot, “PostalCode”, do tipo integrado `AMAZON.UKPostalCode`.
- Um alias com a função do Lambda definida. Para obter mais informações, consulte Criar e anexar uma função do Lambda a um alias de 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
```

Monitorar desempenho do bot

O monitoramento é importante para manter a confiabilidade, a disponibilidade e o desempenho dos chatbots do Amazon Lex V2. Este tópico descreve o uso de registros de conversas para monitorar conversas entre seus usuários e seus chatbots, usando estatísticas de enunciados para determinar os enunciados que seus bots detectam e ignoram, e como usar o Amazon CloudWatch Logs e monitorar AWS CloudTrail o Amazon Lex V2. Este tópico também descreve o runtime do Amazon Lex V2 e as métricas de associação de canais.

Use essas ferramentas e métricas para entender quais direções e ações você pode tomar para melhorar o desempenho de seus bots.

Tópicos

Medir o desempenho dos negócios com o Analytics

Com o Analytics, você pode avaliar o desempenho do seu bot com métricas relacionadas às taxas de sucesso e falha das interações dos seus bots com os clientes. Você também pode visualizar padrões de fluxos de conversa entre seu bot e os clientes. O Analytics simplifica sua experiência resumindo essas métricas em gráficos e tabelas. Ele fornece ferramentas para ajudar você a filtrar os resultados para identificar problemas envolvendo intenções, espaços, enunciados e conversas. Você pode usar esses dados para iterar e aprimorar seu bot para criar uma melhor experiência para o cliente.

> **Note**
>
> Para que um usuário acesse o Analytics, a política Política gerenciada pela AWS: AmazonLexFullAccess ou uma política personalizada que inclua permissões da API de análise devem ser anexadas ao perfil do IAM. Consulte Gerenciar permissões de acesso para análise para obter detalhes sobre como lidar com as permissões do usuário com uma política personalizada. Se Política gerenciada pela AWS: AmazonLexReadOnly estiver anexada ao perfil do IAM de um cliente, um erro exibirá as permissões ausentes que você

precisa adicionar ao perfil do IAM do usuário para que ele possa acessar os painéis do Analytics.

Para acessar o Analytics

1. Faça login AWS Management Console e abra o console Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. No painel de navegação, em Bots, selecione o bot que você deseja visualizar no Analytics.
3. Selecione a seção em Analytics que você deseja visualizar.

Tópicos

Definições chave

Este tópico fornece as principais definições que ajudarão você a interpretar suas análises de bots. Essas definições estão relacionadas ao desempenho do seu bot em quatro contextos: intenções, espaços, conversas e enunciados. Os campos a seguir são relevantes para muitas das métricas de desempenho:

- O campo `state` do objeto `Intent`.
- O `type`campo do `dialogAction` objeto dentro do SessionStateobjeto.

Intenções

O Amazon Lex V2 categoriza as intenções das seguintes maneiras:

- Sucesso – O bot atendeu com sucesso a intenção. Um dos valores a seguir é verdadeiro:

  - A intenção `state` é `ReadyForFulfillment` e o `type` de `dialogAction` é `Close`.
  - A intenção `state` é `Fulfilled` e o `type` de `dialogAction` é `Close`.
- Failed: o bot falhou em cumprir a intenção. O estado da intenção. Um dos valores a seguir é verdadeiro:
  - A intenção `state` é `Failed` e o `type` de `dialogAction` é `Close` (por exemplo, o usuário recusou a solicitação de confirmação).
  - O bot alterna para o `AMAZON.FallbackIntent` antes que a intenção seja concluída.
- Alternado – O bot reconhece uma intenção diferente alterna para ela, antes que a intenção original seja classificada como sucesso ou falha.
- Abandonado – O cliente não responde antes que a intenção seja classificada como sucesso ou falha.

Slots

O Amazon Lex V2 categoriza as intenções das seguintes maneiras:

- Sucesso — O bot preencheu o espaço e fez a transição com sucesso para outro espaço ou para a etapa de confirmação.
- Falha — O bot não conseguiu preencher o espaço, mesmo depois de atingir o número máximo de tentativas.
- Abandonado – O cliente não responde ou alterna para outra intenção antes que o slot seja classificado como sucesso ou falha.

Conversas

Quando um cliente faz uma chamada em runtime para o Amazon Lex V2, ele fornece um `sessionId` e o Amazon Lex V2 gera um `originatingRequestId`. Se o cliente não responder dentro do tempo limite da sessão (`idleSessionTTLInSeconds`) definido para o bot, a sessão expirará. Se um cliente retornar à sessão usando o mesmo `sessionId`, o Amazon Lex V2 gerará um novo `originatingRequestId`.

Para análise, uma conversa é uma combinação única de um `sessionId` e um `originatingRequestId`. O Amazon Lex V2 categoriza as conversas das seguintes maneiras:

- Sucesso – A intenção final da conversa é classificada como sucesso.

- Falha – A intenção final da conversa falhou. A conversa também falhará se o Amazon Lex V2 usar AMAZON.FallbackIntent como padrão.
- Abandonado – O cliente não responde antes que a conversa seja classificada como sucesso ou falha.

Enunciados

O Amazon Lex V2 categoriza enunciados das seguintes maneiras:

- Detectado – O Amazon Lex V2 reconhece o enunciado como uma tentativa de invocar uma intenção configurada para um bot.
- Ignorado – O Amazon Lex V2 não reconhece o enunciado.

Filtrar resultados

Na parte superior de cada página, você pode filtrar os resultados da análise do seu bot.
Opções de filtragem para análise.

Você também pode filtrar usando os seguintes parâmetros:

- Tempo – Você pode filtrar os resultados com base em um intervalo de tempo relativo ou absoluto. Quando você seleciona um horário de início e término, o Amazon Lex V2 recupera conversas que começaram após o horário de início e terminaram antes do horário de término.
  - Intervalo relativo — Selecione 1d para ver os resultados do dia anterior, 1w da semana anterior ou 1m do mês anterior.

    Para obter mais opções, selecione Personalizado e escolha uma duração no menu Intervalo relativo. Para obter mais controle sobre a duração, selecione Intervalo personalizado, insira um número no campo Duração e escolha uma Unidade de tempo no menu suspenso.
  - Intervalo absoluto – Selecione Personalizado e selecione o menu Intervalo absoluto para filtrar as conversas dentro de um intervalo de tempo especificado por você. Você pode escolher uma data de início e término no calendário ou inseri-la no formato AAAA/MM/DD.

> **Note**
>
> O período máximo em que você pode ver os resultados é de 30 dias.

- Filtros de bots – Para filtrar por localidade, alias e versão do seu bot, selecione os menus suspensos chamados Todas as localidades, Todos os aliases e Todas as versões.
- Modalidade – Selecione o ícone de engrenagem e selecione o menu suspenso Modalidade para escolher se deseja exibir os resultados de Fala ou Texto.
- Canal – Selecione o ícone de engrenagem e selecione o menu suspenso Canal para escolher o canal do qual você deseja exibir os resultados. Para mais informações sobre a integração de canais, consulte Integrar um bot do Amazon Lex V2 a uma plataforma de mensagens e centrais de contato do Amazon Connect

Visão geral: um resumo do desempenho do seu bot

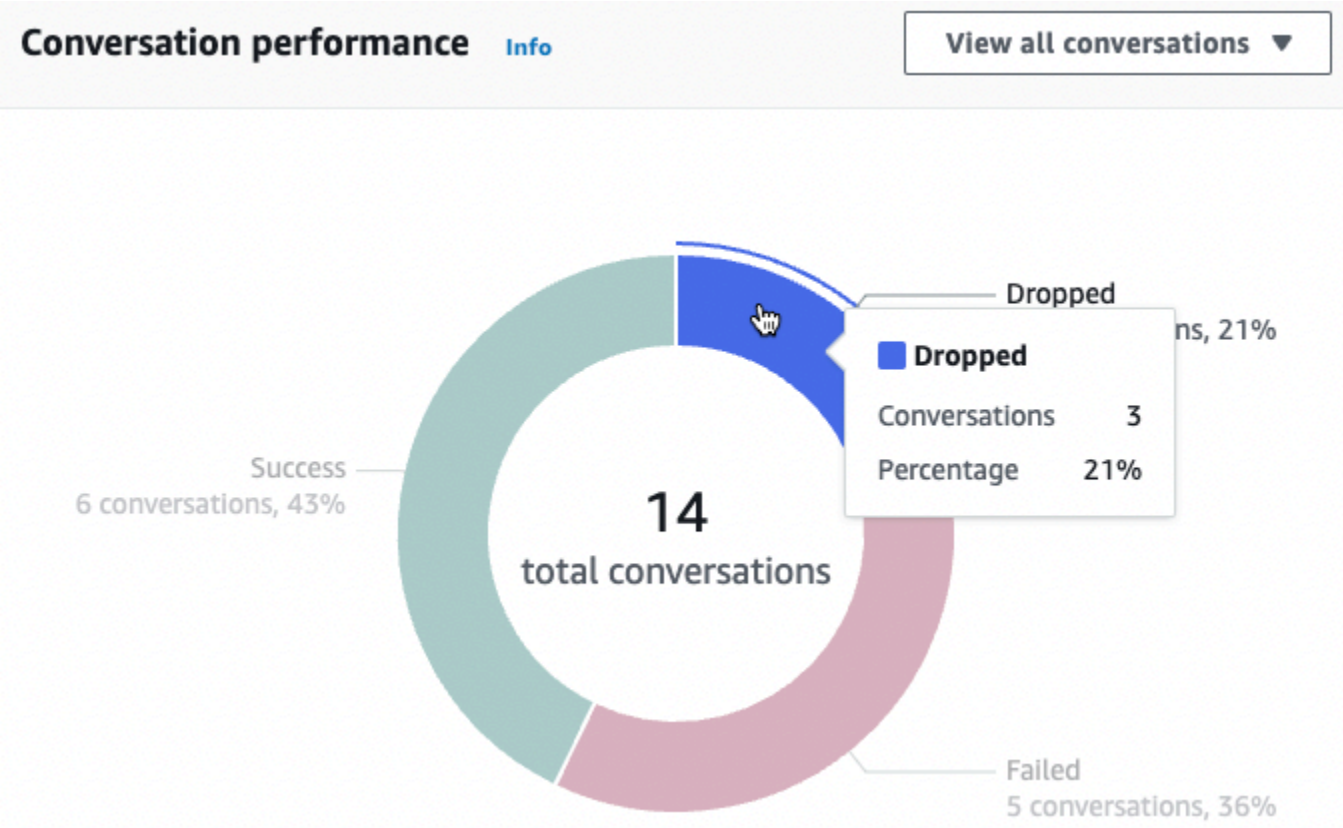
A página de visão geral resume o desempenho do seu bot em conversas, reconhecimento de enunciados e uso de intenções. A visão geral consiste nas seguintes seções:

- Desempenho da conversa
- Taxa de reconhecimento de enunciados
- Histórico de desempenho da conversa
- As 5 intenções mais usadas
- As 5 principais intenções com falha

Desempenho da conversa

Use esse gráfico para rastrear o número e a porcentagem de conversas que são categorizadas como sucesso, falha e abandonado. Para acessar uma lista de conversas, selecione Visualizar todas as conversas para revelar um menu suspenso. Você pode optar por ver uma lista de todas as conversas do usuário com o bot ou filtrar as conversas com um resultado específico (sucesso, falha ou abandonado). Esses links levam você à subseção Conversas do painel Conversa. Para ter mais informações, consulte Conversas.

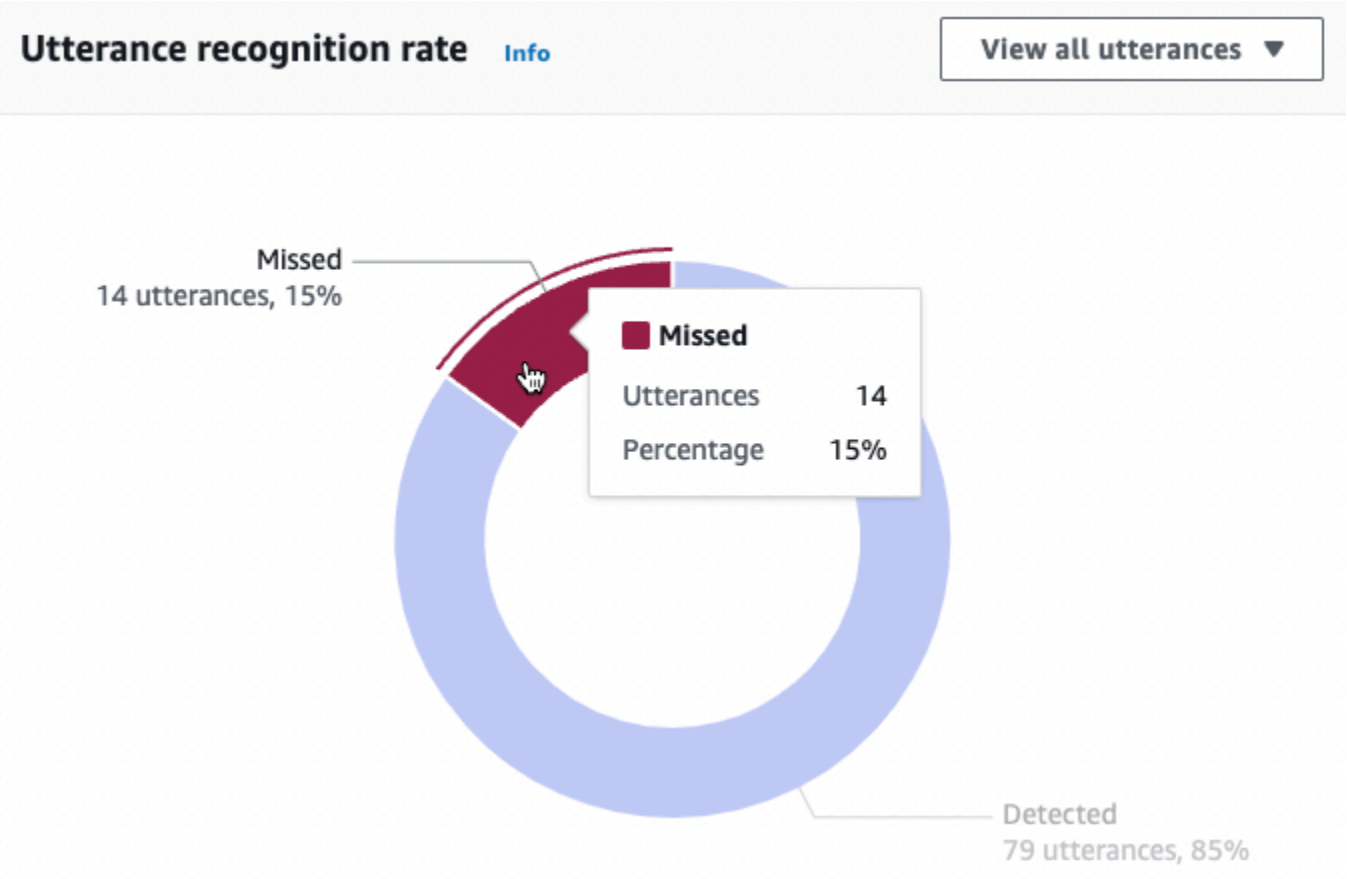
Para revelar uma caixa com a contagem e a porcentagem de conversas com esse resultado, passe o mouse sobre um segmento do gráfico, como na imagem a seguir.



Taxa de reconhecimento de enunciados

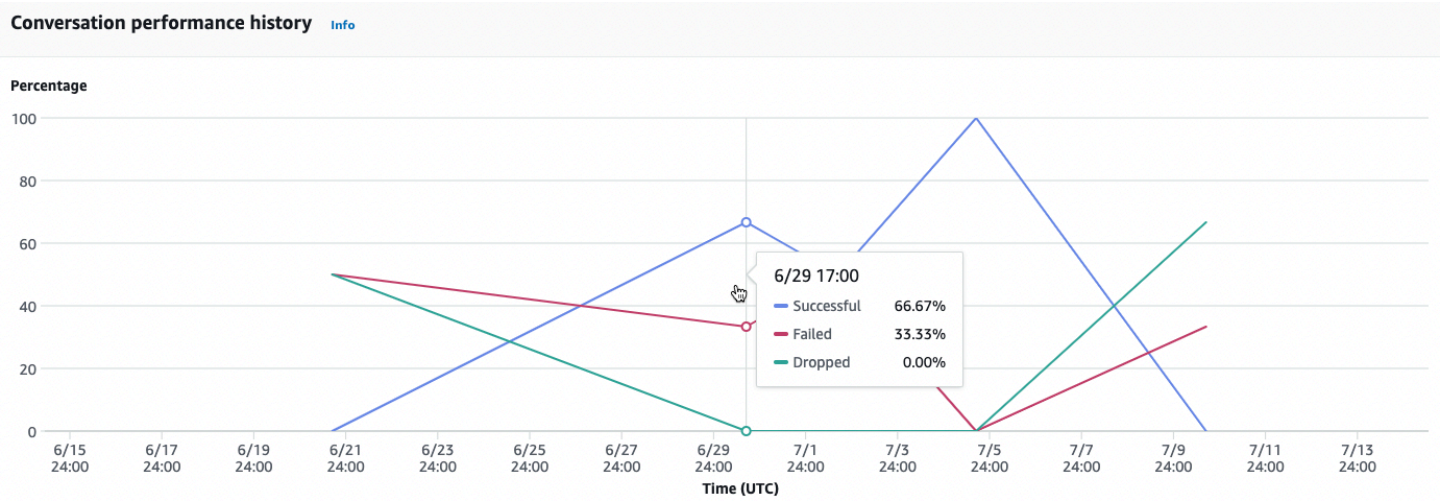
Use esse gráfico para rastrear o número e a porcentagem de enunciados que foram detectados e ignorados pelo seu bot. Para acessar uma lista de enunciados, selecione Exibir enunciados para revelar um menu suspenso. Você pode optar por ver uma lista de todos os enunciados do usuário ou filtrar os enunciados com um resultado específico (ignorado ou detectado). Esses links levam você à subseção Reconhecimento de enunciados do painel Desempenho. Para mais informações, consulte Visualizar enunciados para acessar Reconhecimento de enunciados.

Para revelar uma caixa com a contagem e a porcentagem de enunciados, passe o mouse sobre um segmento do gráfico, como na imagem a seguir.



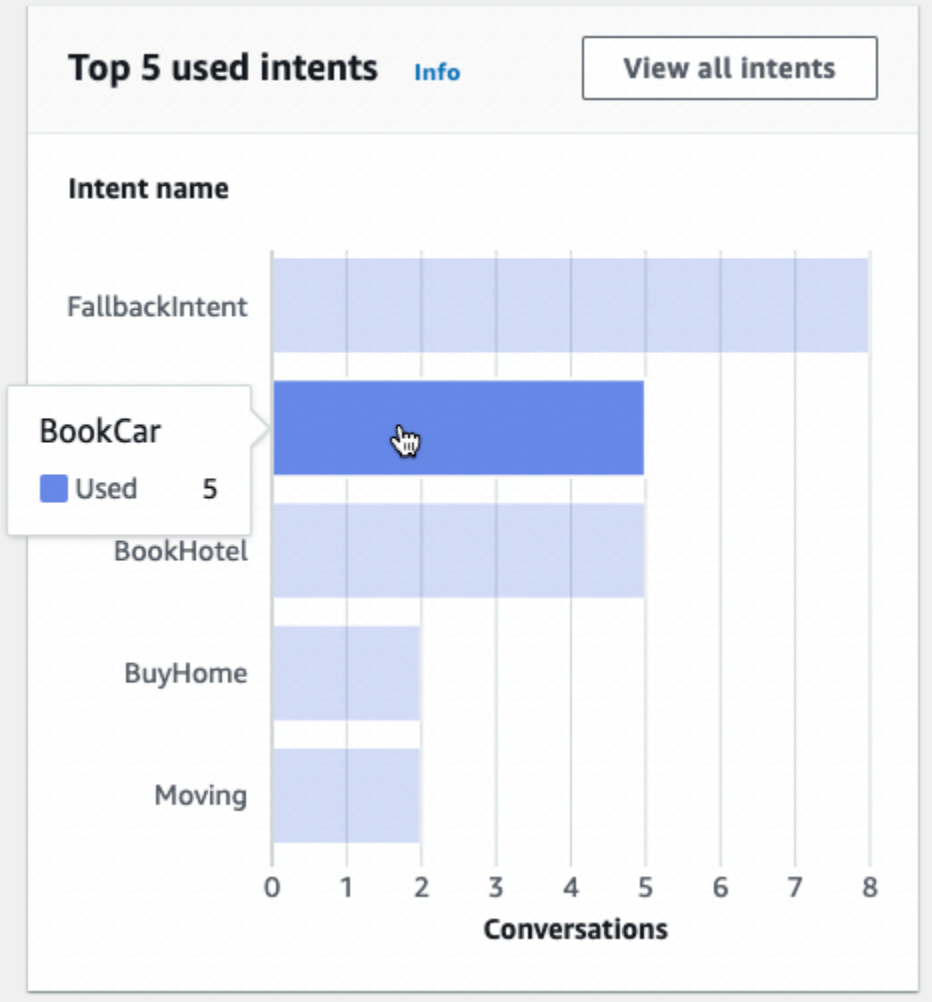
Histórico de desempenho da conversa

Use esse gráfico para rastrear a porcentagem de conversas categorizadas como sucesso, falha e abandonado no intervalo de tempo definido nos filtros. Para ver a porcentagem de conversas com um resultado específico em um intervalo de tempo, passe o mouse sobre esse intervalo, como na imagem a seguir.



As 5 intenções mais usadas

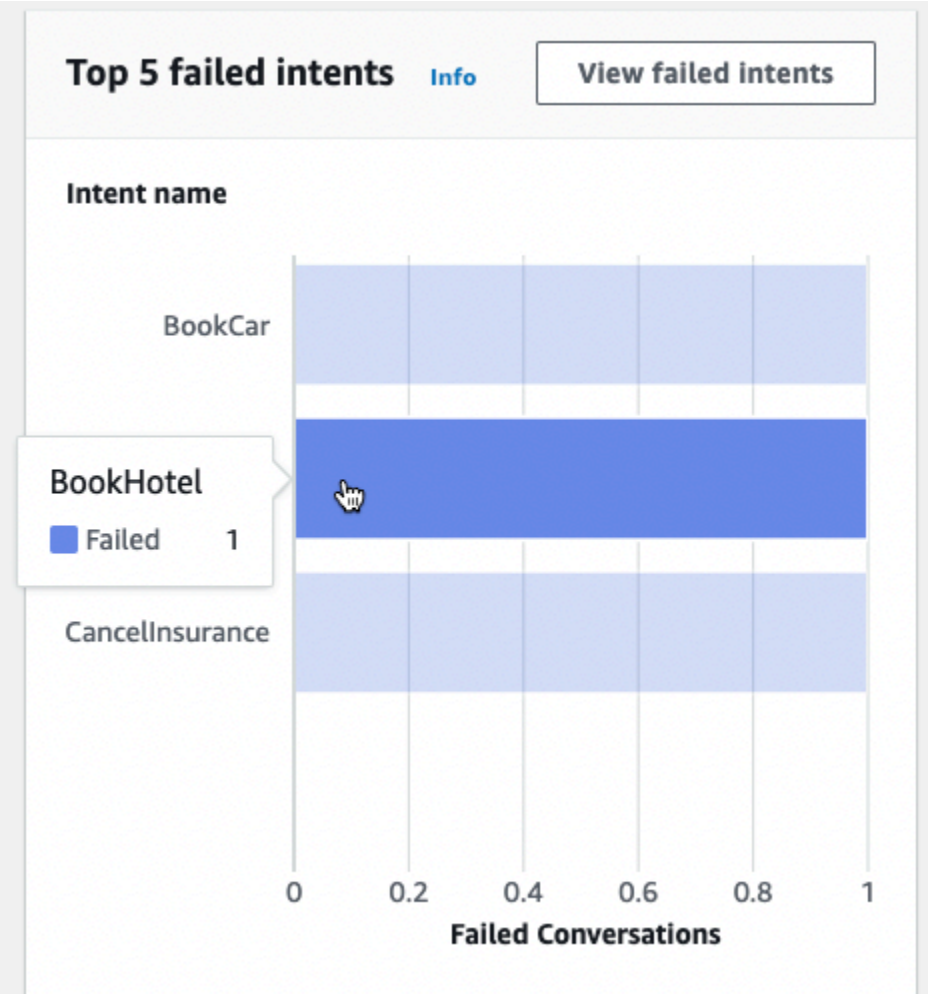
Use esse gráfico para identificar as cinco intenções que os clientes mais usaram com seu bot. Passe o mouse sobre uma barra para ver quantas vezes seu bot reconheceu essa intenção, como na imagem a seguir.



Selecione Exibir todas as intenções para navegar até a subseção Desempenho das intenções do painel Desempenho, onde você pode ver as métricas do desempenho do seu bot em termos de atendimento das intenções. Para ter mais informações, consulte Desempenho de intenção.

As 5 principais intenções com falha

Use esse gráfico para identificar as cinco principais intenções que seu bot falhou em atender (consulte Intenções para ver a definição de uma intenção com falha). Passe o mouse sobre uma barra para ver quantas vezes seu bot falhou em atender essa intenção, como na imagem a seguir.



Selecione Exibir intenções com falha para navegar até a subseção Desempenho das intenções do painel Desempenho, onde você pode ver as métricas sobre as intenções que seu bot falhou em atender. Para ter mais informações, consulte Desempenho de intenção.

Painel Conversa: um resumo das conversas do seu bot

O painel Conversa visualiza as métricas das conversas dos clientes (consulte Conversas para ver a definição de uma conversa) com seu bot.

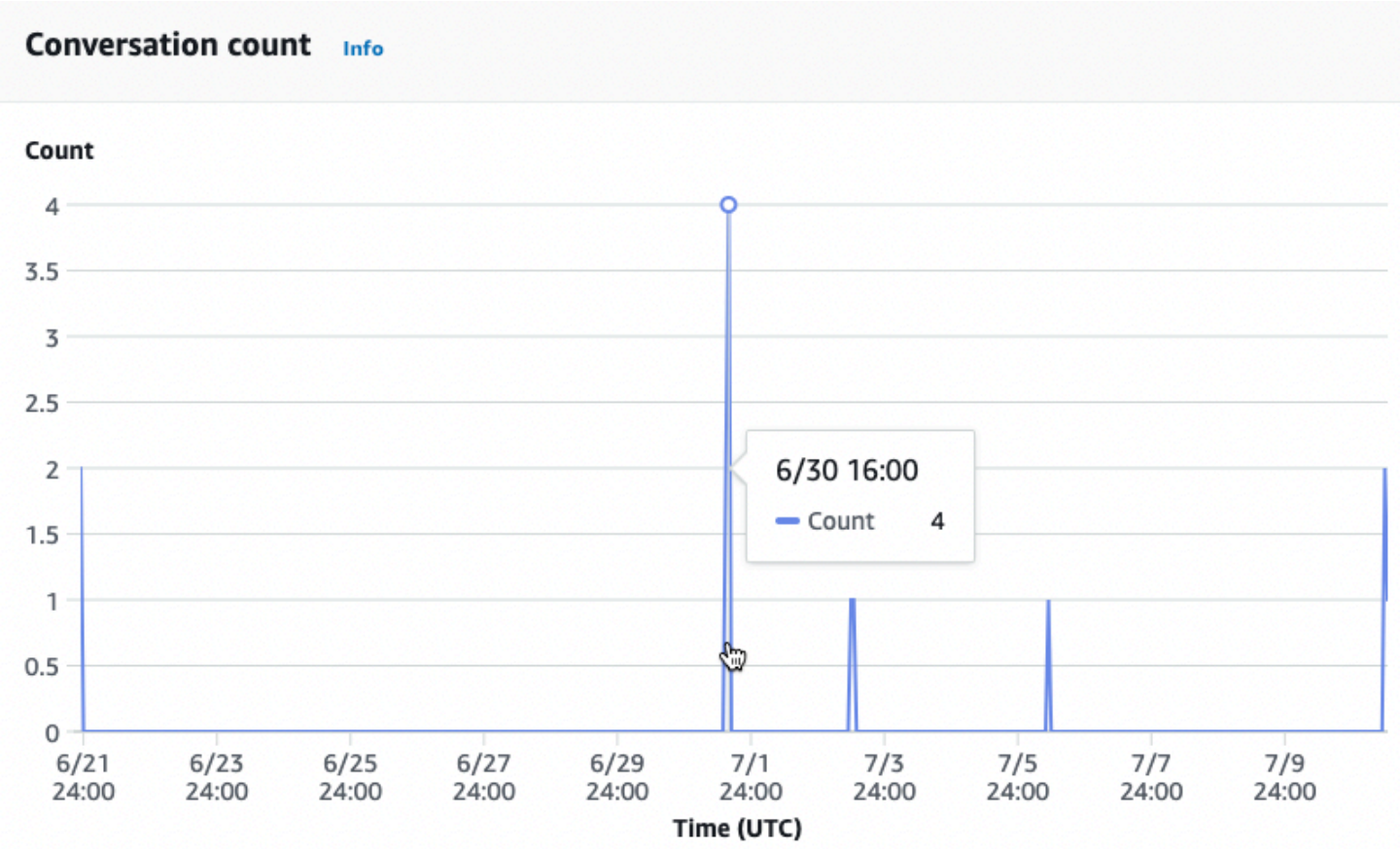
O Resumo contém as seguintes informações sobre as conversas do usuário com seu bot. Os números são calculados com base nas configurações do filtro.

- Total de conversas – O número total de conversas com o bot.
- Duração média da conversa – O tempo médio das conversas do usuário com o bot em minutos e segundos. O formato é mm:ss.
- Média de turnos por conversa – O número médio de turnos que uma conversa dura.

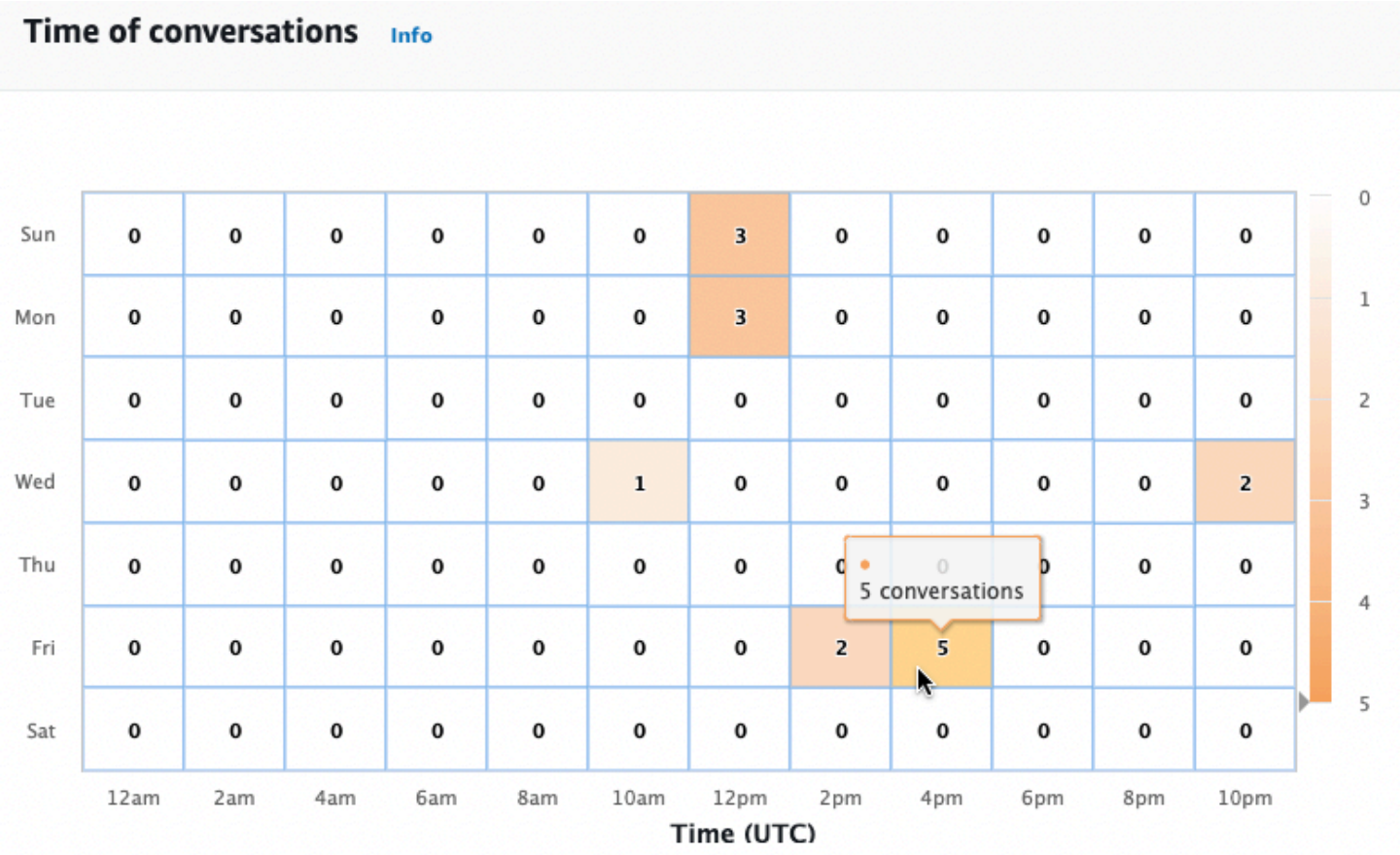
As seções Contagem de conversas e Contagem de mensagens contêm um gráfico que mostra o número de conversas e mensagens, respectivamente, no intervalo de tempo que você especifica em seus filtros. Passe o mouse sobre um segmento de tempo para ver o número de conversas ou mensagens nesse segmento. O tamanho do segmento de tempo depende do intervalo de tempo especificado:

- Menos de 1 semana – A contagem é exibida para cada hora.
- 1 semana ou mais – A contagem é exibida para cada dia.

Na imagem a seguir, veja um exemplo do comportamento de passar o mouse.



A seção Tempo das conversas apresenta o número de conversas que ocorreram entre seu bot e os clientes em cada intervalo de duas horas em cada dia da semana, dentro do intervalo de tempo especificado nos filtros. Células com um tom mais escuro indicam horários em que mais conversas ocorreram. Passe o mouse sobre uma célula para exibir o número de conversas nas 2 horas a partir desse horário. Por exemplo, a ação na imagem a seguir mostra o número de conversas que ocorrem entre 16h e 18h UTC.



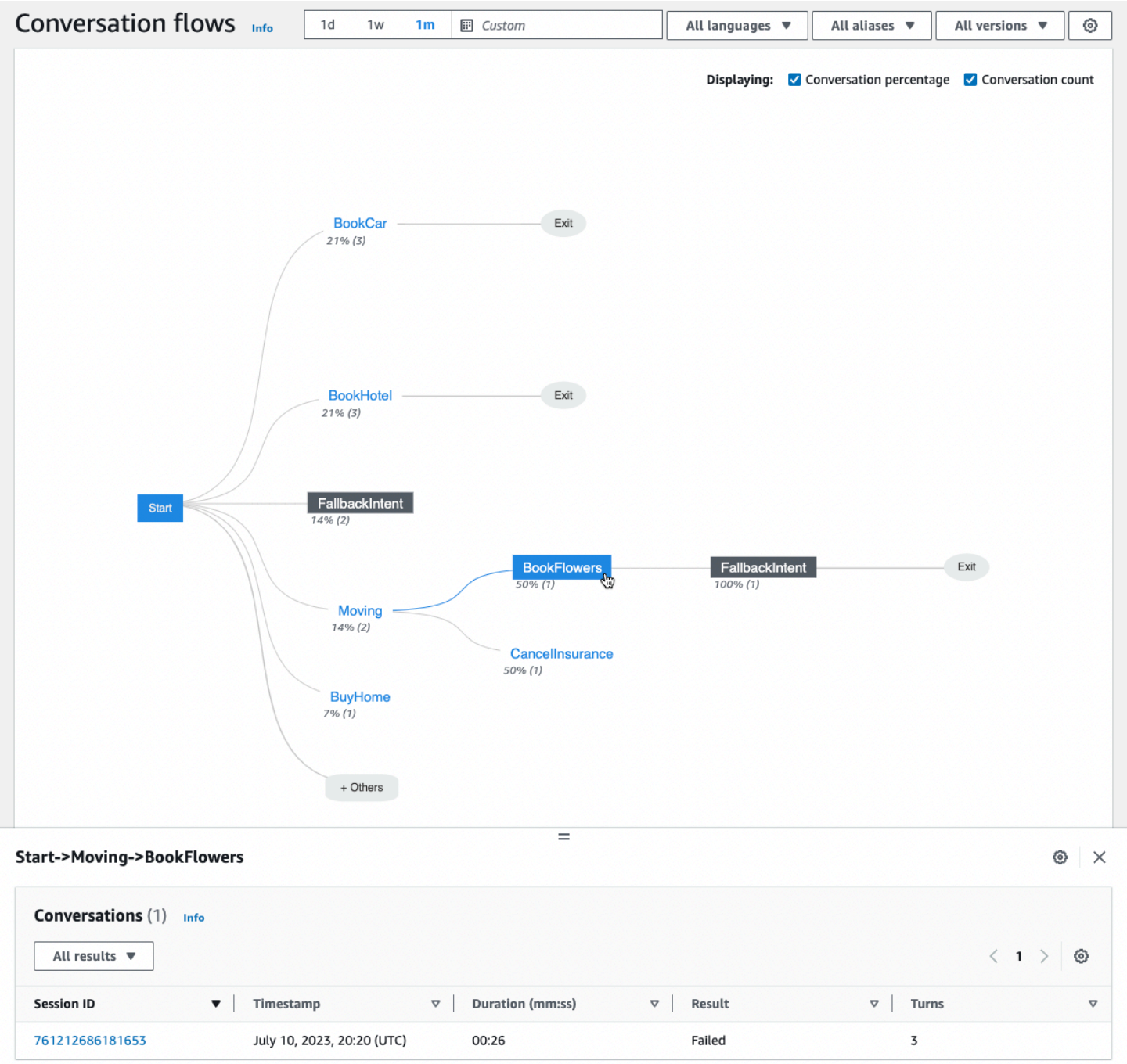
O painel Conversa contém duas ferramentas, Fluxos de conversa e Conversas. Acesse uma ferramenta selecionando-a em painel Conversa no painel de navegação esquerdo.

Fluxos de conversa

Use os Fluxos de conversa para visualizar as ordens de intenções que os clientes realizam nas conversas com seu bot. Abaixo de cada intenção está a porcentagem e a contagem de conversas que invocaram essa intenção naquele ponto da conversa. Você pode alternar entre a porcentagem e a contagem selecionando Porcentagem de conversas e Contagem de conversas na parte superior. Por padrão, as cinco intenções mais comuns nesse ponto da conversa são mostradas em ordem decrescente de frequência. Selecione + Outros para exibir todas as intenções.

Escolha uma intenção para expandir para uma nova coluna de ramificações que mostra uma lista das intenções tomadas naquele ponto da conversa, classificadas em frequência decrescente.

Ao selecionar um nó no fluxo de conversação, você pode expandir a janela abaixo para exibir uma lista de conversas que seguiram essa ordem de intenções. Selecione o ID da sessão correspondente a uma conversa para ver detalhes sobre essa conversa. A imagem a seguir mostra um fluxo de conversa e a janela Conversas expandida na parte inferior.



Conversas

A ferramenta Conversas exibe uma lista de conversas do seu bot. É possível selecionar uma coluna para usá-la para classificar em ordem crescente ou decrescente.

Para filtrar as conversas por resultado, selecione Todos os resultados e selecione Sucesso, Falha ou Abandonado.

Para filtrar as conversas por duração

1. Selecione a barra de pesquisa marcada como Filtrar conversas por duração
2. Defina o filtro de uma das seguintes maneiras:
   - Use as opções predefinidas.
     a. Selecione Duração.
     b. Escolha entre os operadores = (igual a), > (maior que) e < (menor que).
     c. Escolha um intervalo de tempo.
   - Insira uma entrada no formato “Duração {operador} {número} seg”. Por exemplo, para pesquisar todas as conversas com duração superior a 30 segundos, digite **`Duration > 30 sec`**. Especifique a duração em segundos.

Para ver informações detalhadas sobre a sessão, incluindo metadados, uso intencional e uma transcrição, selecione o ID da sessão de uma conversa.

> **Note**
>
> Como uma conversa é uma combinação exclusiva de um `sessionId` e `originatingRequestId`, o mesmo `sessionId` pode aparecer várias vezes na tabela.

A seção Detalhes contém os seguintes metadados:

- Carimbo de data e hora: especifica a data e a hora de início da conversa. A hora está no formato hh:mm:ss.
- Duração – Especifica quanto tempo a conversa durou no formato mm:ss. A duração não inclui a duração do tempo limite da sessão (`idleSessionTTLInSeconds`).
- Resultado – Especifica se a conversa foi categorizada como sucesso, falha ou abandonado. Consulte Conversas para obter mais detalhes sobre esses resultados.
- Modo – Especifica se a conversa foi `Speech`, `Text`, ou `DTMF` (pressionamentos do teclado numérico). Uma conversa que consiste em vários modos é `Multimode`.
- Canal – Especifica o canal em que a conversa ocorreu, se aplicável. Consulte Integrar um bot do Amazon Lex V2 a uma plataforma de mensagens.
- Idioma – Especifica o idioma do bot.

As intenções que o bot elicitou na conversa são mostradas abaixo de Detalhes. Selecione Ir para intenção para acessar essa intenção no editor de intenção. Selecione Encaixar na transcrição para rolar automaticamente a Transcrição até a primeira instância em que o bot elicitou a intenção.

Selecione a seta para a direita ao lado do nome da intenção para ver detalhes sobre os slots usados para a intenção, incluindo os nomes dos slots, o valor que o bot elicitou para cada espaço e o número de vezes que o bot tentou elicitar cada espaço.

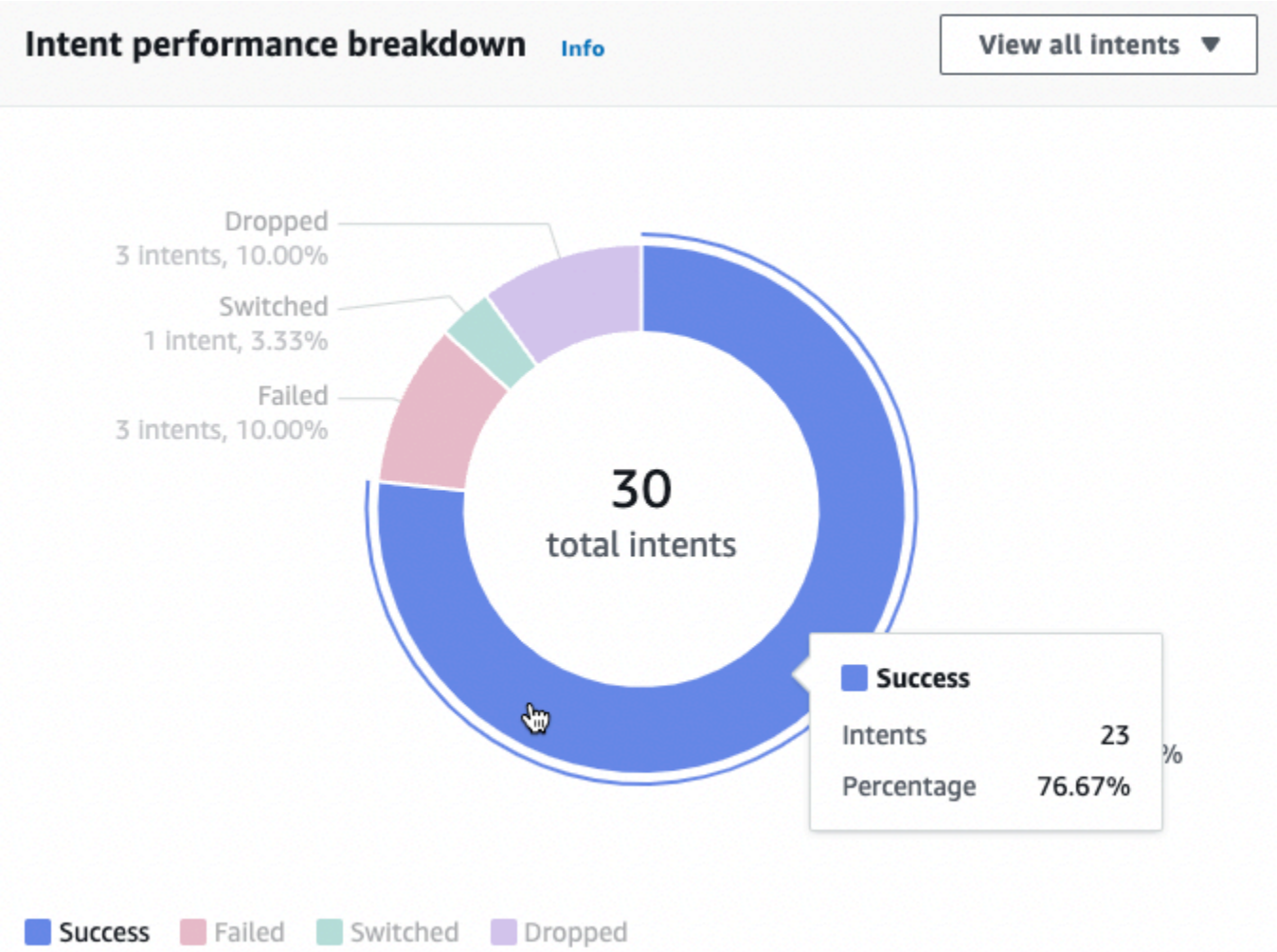
A Transcrição permite que você revise os enunciados da conversa e o comportamento do seu bot ao elicitar intenções e slots. Os enunciados do usuário são exibidos à esquerda e os enunciados do bot são exibidos à direita. Use a barra de pesquisa marcada como Filtrar transcrições nesta sessão para pesquisar texto na transcrição. Ao lado de Exibindo:, há três informações mostradas em cada turno de conversa que você pode selecionar para exibir ou não:

- Carimbo de data e hora – Especifica a hora do enunciado.
- Estado da intenção – Especifica a intenção que o bot está elicitando durante um enunciado e o resultado da intenção, se aplicável. Os estados possíveis são os seguintes.
  - Intenção invocada: *`nome da intenção`* – O bot identificou uma intenção que o cliente está invocando.
  - Intenção alternada: *`nome da intenção`* – O bot alternou para uma intenção diferente com base no enunciado.
  - *`nome da intenção`*: Sucesso – O bot atendeu a intenção.
- Estado do slot – Especifica o slot que o bot está elicitando durante um enunciado, se aplicável, e o valor que o cliente fornece.

Painel de desempenho: um resumo das métricas de intenção e expressão do seu bot

No painel Desempenho, você pode ver detalhes sobre o desempenho do atendimento da intenção e do reconhecimento de enunciados do seu bot.

A seção Detalhamento do desempenho da intenção exibe o número total de vezes que seu bot invocou uma intenção e detalha o número e a porcentagem de vezes em que as intenções foram categorizadas como sucesso, falha, abandonado e alternado. Consulte Intenções para obter uma explicação dessas definições. Passe o mouse sobre um segmento do gráfico, para revelar uma caixa com a contagem e a porcentagem de conversas com esse resultado, como na imagem a seguir.



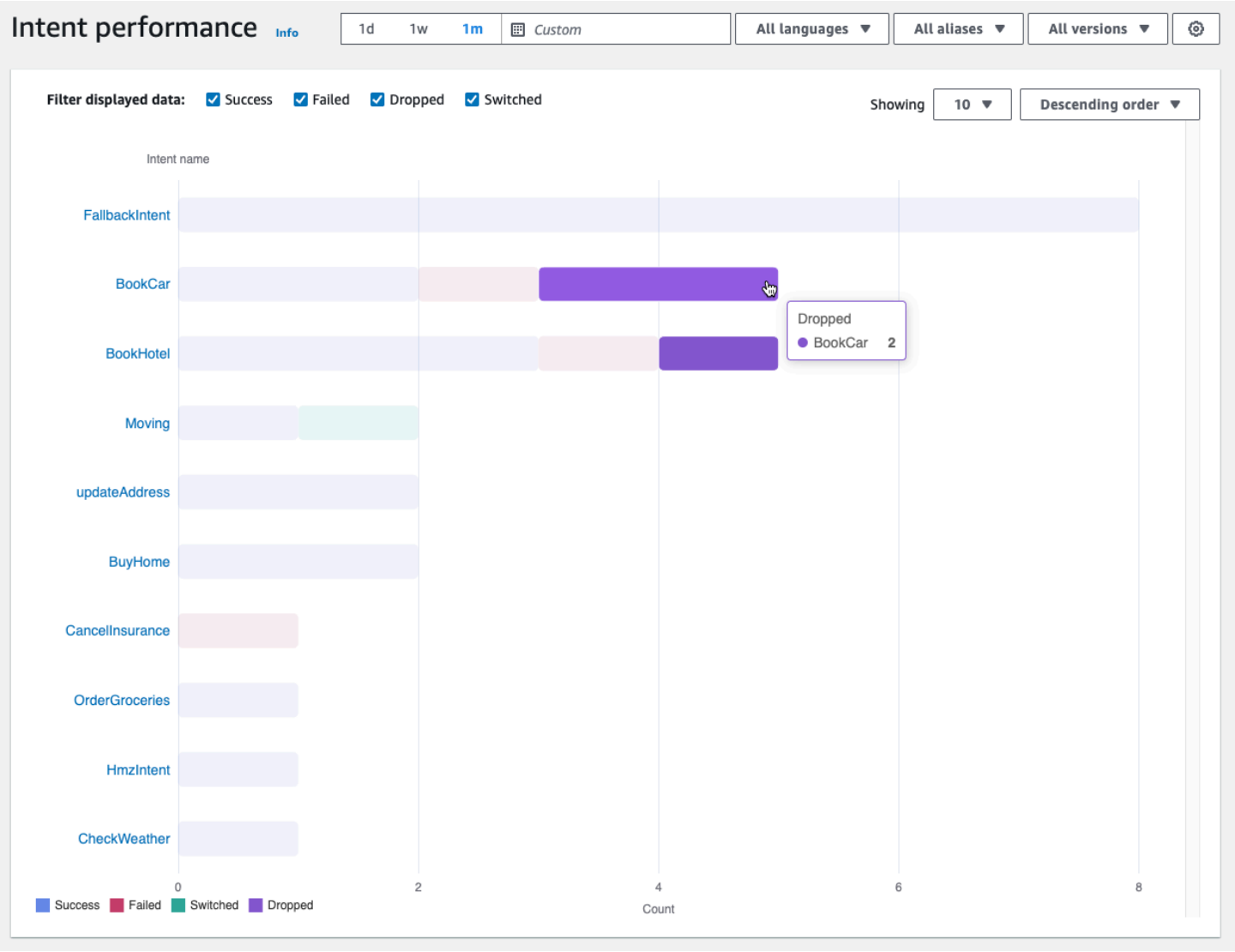
Selecione Exibir todas as intenções para revelar um menu suspenso, no qual você pode ver uma lista das intenções que o bot elicitou. Você também pode optar por visualizar as intenções com um resultado específico (sucesso, falha, abandonado ou alternado). Esses links levam você à subseção Desempenho de intenção do painel Desempenho. Para ter mais informações, consulte Desempenho de intenção.

A seção de Reconhecimento de enunciados resume o número de enunciados que foram ignorados e detectados. Selecione Exibir detalhes para navegar até uma lista de enunciados do bot. Selecione o número em Enunciados ignorados para ver uma lista de enunciados ignorados e o número em Enunciados detectados para ver uma lista de enunciados detectados do bot. Para ter mais informações, consulte Reconhecimento de enunciados.

Selecione Desempenho de intenções e Reconhecimento de enunciados no painel Desempenho na barra lateral esquerda para ver detalhes sobre intenções e enunciados no seu bot.

Desempenho de intenção

Esse painel resume o desempenho das intenções usadas com seu bot em ordem decrescente de frequência. A barra ao lado de cada intenção visualiza o número de vezes que a intenção foi categorizada como sucesso, falha, abandonado e alternado. Consulte Intenções para obter uma explicação dessas definições. Passe o mouse sobre um segmento da barra para ver o número de conversas que usam essa intenção com esse resultado, como na imagem a seguir:



> **Note**
>
> O painel mostra os 1.000 principais resultados de um conjunto de configurações de filtro. Para obter resultados mais direcionados, defina as configurações granulares do filtro.

Na parte superior do gráfico, você pode alternar os status de intenção que deseja visualizar com as caixas de seleção Sucesso, Falha, Abandonado e Alternado.

Selecione os menus suspensos à direita de Exibir para ajustar o número de intenções a serem exibidas e se as intenções devem ser exibidas em ordem crescente ou decrescente de frequência.

Selecione um nome de intenção para navegar até uma página que mostra três gráficos: Detalhamento do desempenho da intenção, Desempenho do slot e Alternâncias de intenção.

A seção Detalhamento do desempenho da intenção exibe o número total de vezes que seu bot usou a intenção e detalha o número e a porcentagem de vezes que o atendimento das intenções foi categorizado como sucesso, falha, abandonado e alternado. Consulte Intenções para obter uma explicação dessas definições. Passe o mouse sobre um segmento do gráfico para ver a contagem e a porcentagem de vezes que o atendimento da intenção gerou esse resultado.

A seção Desempenho do slot exibe métricas dos slots que pertencem à intenção atual. Para classificar por uma coluna, selecione essa coluna uma vez para classificá-la em ordem crescente e duas vezes para classificá-la em ordem decrescente. Você pode usar a barra de pesquisa para encontrar um slot específico ou usar os botões de número de página para navegar pelos slots.

> **Note**
>
> O painel mostra os 1.000 principais resultados de um conjunto de configurações de filtro. Para obter resultados mais direcionados, defina as configurações granulares do filtro.

A seção Alternâncias de intenção lista as instâncias em que o bot alternou da intenção atual para outra com as seguintes informações:

- Estágio – O estágio da conversa em que o bot alternou a intenção.
- Intenção alternada para – A intenção para a qual o bot alternou a intenção atual.
- Contagem de sessões – O número de sessões nas quais a combinação Estágio e Intenção alternada para ocorreu.

> **Note**
>
> O painel mostra os 1.000 principais resultados de um conjunto de configurações de filtro. Para obter resultados mais direcionados, defina as configurações granulares do filtro.

Reconhecimento de enunciados

Esta página lista todos os enunciados que foram ignorados e detectados pelo seu bot e fornece ferramentas para você adicionar exemplos de enunciados às intenções para ajudar a treinar seu bot. Consulte Enunciados para obter uma explicação dessas definições. Use as guias na parte superior para alternar entre uma lista de Enunciados perdidos e de Enunciados detectados.

> **Note**
>
> O painel mostra os 1.000 principais resultados de um conjunto de configurações de filtro. Para obter resultados mais direcionados, defina as configurações granulares do filtro.

Para adicionar enunciados a uma intenção:

1. Marque a caixa de seleção ao lado dos enunciados que você deseja adicionar como exemplos de enunciados para uma intenção.
2. Selecione Adicionar à intenção e escolha a intenção à qual você deseja adicionar os enunciados no menu suspenso em Intenção.
3. Selecione Adicionar.

Usar APIs para análise

Esta seção descreve as operações de API que você usa para recuperar análises de um bot.

> **Note**
>
> Para usar as ListUtterancemétricas e ListUtteranceAnalyticsData, sua função do IAM deve ter permissões para realizar a operação ListAggregatedEnunciados, que fornece acesso a análises relacionadas ao enunciado. Consulte Visualizar estatísticas de enunciado para obter detalhes sobre a política do IAM a ser aplicada ao perfil do IAM.

- As seguintes operações de API recuperam métricas resumidas de um bot:
  - ListSessionMétricas
  - ListIntentMétricas

  - ListIntentStageMetrics
  - ListUtteranceMétricas
- As seguintes operações de API recuperam uma lista de metadados para sessões e enunciados:
  - ListSessionAnalyticsData
  - ListUtteranceAnalyticsData
- A operação ListIntentPaths recupera métricas sobre uma ordem de intenções que os clientes adotam nas conversas com um bot.

Filtrar resultados

As solicitações de API de análise exigem que você especifique `startTime` e `endTime`. A API retorna sessões, intenções, estágios de intenção ou enunciados que começaram depois do `startTime` e que terminaram antes do `endTime`.

`filters` é um campo opcional nas solicitações da API de análise. Ele mapeia para uma lista de objetos AnalyticsSessionAnalyticsIntentFilter AnalyticsIntent StageFilter, AnalyticsUtterance Filter ou Filter. Em cada objeto, use os campos para criar uma expressão pela qual filtrar. Por exemplo, se você adicionar o filtro a seguir à lista, o bot pesquisará conversas com mais de 30 segundos.

```
{
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

Recuperar métricas para um bot

Use as operações `ListSessionMetrics`, `ListIntentMetrics`, `ListIntentStageMetrics` e `ListUtteranceMetrics` e para recuperar métricas resumidas de sessões, intenções, estágios de intenção e enunciados.

Para essas operações, preencha os seguintes campos obrigatórios:

- Forneça um `startTime` e `endTime` para definir um intervalo de tempo que você deseja usar para recuperar os resultados.
- Especifique as métricas que você deseja calcular`metrics`, uma lista de objetos AnalyticsSessionAnalyticsIntentmétricos AnalyticsIntent StageMetric, AnalyticsUtterance métricos

ou métricos. Em cada objeto, use o campo `name` para especificar a métrica para calcular o campo `statistic` para especificar se deseja calcular o número `Sum`, `Average` ou `Max`, e o campo `order` para especificar se os resultados devem ser classificados em ordem `Ascending` ou `Descending`.

> **Note**
>
> Tanto o objeto `metrics` quanto o `binBy` contêm um campo `order`. Você pode especificar a classificação `order` em somente um dos dois objetos.

Os campos restantes na solicitação são opcionais. É possível filtrar e organizar os resultados das seguintes maneiras:

- Filtrar resultados – Use o campo `filters` para filtrar os resultados. Consulte Filtrar resultados para obter mais detalhes.
- Agrupamento de resultados por categoria — Especifique o `groupBy` campo, uma lista contendo um único objeto AnalyticsSessionAnalyticsIntentResultado AnalyticsIntentStageResult, AnalyticsUtteranceResultado ou Resultado. No objeto, especifique o campo `name` com a categoria pela qual você deseja agrupar os resultados.

  Se você especificar um `groupBy` campo na solicitação, o `results` objeto na resposta `groupByKeys` conterá uma lista de objetos AnalyticsSessionGroupByAnalyticsIntentGroupByKey AnalyticsIntent StageGroupByKey, Key ou AnalyticsUtteranceGroupByKey, cada um com o `name` que você especificou na solicitação e um membro dessa categoria no `value` campo.
- Agrupando os resultados por hora — Especifique o `binBy` campo, uma lista contendo um único AnalyticsBinBySpecificationobjeto. No objeto, especifique o campo `name` com `ConversationStartTime` para agrupar os resultados de acordo com o início da conversa ou `UtteranceTimestamp` para agrupar os resultados com base no momento em que o enunciado ocorreu. Especifique o intervalo de tempo pelo qual você deseja agrupar os resultados no campo `interval` e se deseja classificar em ordem de tempo `Ascending` ou `Descending` ou no campo `order`.

  Se você especificar um `binBy` campo na solicitação, o `results` objeto na resposta `binKeys` conterá uma lista de objetos AnalyticsBinKey, cada um com o `name` que você especificou na solicitação e o intervalo de tempo que define esse compartimento no `value` campo.

> **Note**
>
> Tanto o objeto `metrics` quanto o `binBy` contêm um campo `order`. Você pode especificar a classificação `order` em somente um dos dois objetos.

Use os campos a seguir para lidar com a exibição da resposta:

- Especifique um número entre 1 e 1.000 no campo `maxResults` para limitar o número de resultados a serem retornados em uma única resposta.
- Se o número de resultados for maior que o número especificado no campo `maxResults`, a resposta conterá um `nextToken`. Faça a solicitação novamente, mas use esse valor no campo `nextToken` para retornar o próximo lote de resultados.

Se você estiver usando `ListUtteranceMetrics`, você pode especificar atributos a serem retornados no campo `attributes`. Esse campo é mapeado para uma lista contendo um único objeto AnalyticsUtteranceAttribute. Especifique `LastUsedIntent` no campo `name` para retornar a intenção que o Amazon Lex V2 está usando no momento do enunciado.

Na resposta, o `results` campo é mapeado para uma lista de objetos AnalyticsSessionAnalyticsIntentResultado AnalyticsIntentStageResult, AnalyticsUtteranceResultado ou Resultado. Cada objeto contém um campo `metrics` que retorna o valor de uma estatística resumida para uma métrica que você solicitou, além de quaisquer compartimentos ou grupos criados a partir dos métodos que você especificou.

Recuperar metadados para sessões e enunciados em um bot

Use as ListUtteranceAnalyticsDataoperações ListSessionAnalyticsDataand para recuperar metadados sobre sessões e declarações individuais.

Preencha os campos `startTime` e `endTime` obrigatórios para definir um intervalo de tempo para o qual você deseja recuperar os resultados.

Os campos restantes na solicitação são opcionais. Para filtrar e classificar os resultados:

- Filtrar resultados – Use o campo `filters` para filtrar os resultados. Consulte Filtrar resultados para obter mais detalhes.

- Classificando resultados — Classifique os resultados com o `sortBy` campo, que contém um UtteranceDataSortByobjeto SessionDataSortByou. Especifique o valor que você deseja usar para classificar no campo `name` e se deseja classificar em ordem `Ascending` ou `Descending` no campo `order`.

Use os campos a seguir para lidar com a exibição da resposta:

- Especifique um número entre 1 e 1.000 no campo `maxResults` para limitar o número de resultados a serem retornados em uma única resposta.
- Se o número de resultados for maior que o número especificado no campo `maxResults`, a resposta conterá um `nextToken`. Faça a solicitação novamente, mas use esse valor no campo `nextToken` para retornar o próximo lote de resultados.

Na resposta, o `utterances` campo `sessions` ou é mapeado para uma lista de SessionSpecificationou UtteranceSpecificationobjetos. Cada objeto contém metadados para uma única sessão ou enunciado.

Recuperar metadados para sessões e enunciados em um bot

Use a operação ListIntentPaths para recuperar métricas sobre uma ordem de intenções que os clientes realizam ao conversar com um bot.

Para essa operação, preencha os seguintes campos obrigatórios:

- Forneça um `startTime` e `endTime` para definir um intervalo de tempo que você deseja usar para recuperar os resultados.
- Forneça um `intentPath` para definir uma ordem de intenções para as quais você deseja recuperar métricas. Separe as intenções no caminho com uma barra para frente. Por exemplo, preencha o campo `intentPath` com **`/BookCar/BookHotel`** para ver detalhes sobre quantas vezes os usuários invocaram as intenções `BookCar` e `BookHotel` nessa ordem.

Use o campo `filters` opcional para filtrar os resultados. Para obter mais detalhes, consulte Filtrar resultados.

Visualizar estatísticas de enunciado

Você pode usar estatísticas de enunciados para determinar os enunciados que seus usuários estão enviando para o seu bot. Você pode ver tanto os enunciados que o Amazon Lex V2 detecta com sucesso quanto os enunciados que ele não detecta. É possível utilizar essas informações para ajustar seu bot.

Por exemplo, se você descobrir que seus usuários estão enviando um enunciado que está faltando no Amazon Lex V2, você pode adicionar o enunciado a uma intenção. A versão Rascunho da intenção é atualizada com o novo enunciado e você pode testá-la antes de implantá-la no seu bot.

O enunciado é detectado quando o Amazon Lex V2 reconhece o enunciado como uma tentativa de invocar uma intenção configurada para um bot. Um enunciado é perdido quando o Amazon Lex V2 não o reconhece e, em vez disso, invoca o `AMAZON.FallbackIntent`.

As estatísticas do enunciado podem ser visualizadas usando a API `ListUtteranceMetrics` e a API `ListAggregatedUtterance`.

As estatísticas de enunciado não são geradas usando a API `ListUtteranceMetrics` nas seguintes condições:

- A configuração da Lei de Proteção à Privacidade Online Infantil foi definida como Sim quando o bot foi criado com o console, ou o campo `childDirected` foi definido como verdadeiro quando o bot foi criado com a operação `CreateBot`.

A API `ListUtteranceMetrics` fornece recursos adicionais, incluindo:

- Mais informações disponíveis, como intenção mapeada para enunciados detectados.
- Mais capacidade de filtragem (incluindo canal e modo).
- Intervalo de datas de retenção mais longo (30 dias).
- Você pode usar a API mesmo se tiver optado por não armazenar dados. A funcionalidade do console para enunciados perdidos e detectados dependerá da API `ListUtteranceMetrics`.

As estatísticas de enunciado não são geradas usando a API `ListAggregatedUtterance` nas seguintes condições:

- A configuração da Lei de Proteção à Privacidade Online Infantil foi definida como Sim quando o bot foi criado com o console, ou o campo `childDirected` foi definido como verdadeiro quando o bot foi criado com a operação `CreateBot`.
- Você está usando a ofuscação de slots com um ou mais slots.

- Você optou por não participar da melhoria do Amazon Lex.

A API `ListAggregatedUtterance` fornece atributos adicionais, incluindo:

- Menos informações disponíveis (nenhuma intenção mapeada para enunciados detectados).
- Capacidade limitada de filtragem (incluindo canal e modo).
- Intervalo de datas de retenção curto (15 dias).

Usando as estatísticas de enunciado, você pode ver se um enunciado específico foi detectado ou não detectado, juntamente com a última vez em que o enunciado foi usado em uma interação de bot.

O Amazon Lex V2 armazena enunciados continuamente enquanto os usuários interagem com seu bot. Você pode consultar as estatísticas usando o console ou a operação `ListAggregatedUtterances`. Ele tem uma retenção de dados de 15 dias e não está disponível se o usuário tiver optado por não armazenar dados. Você pode excluir enunciados usando a operação `DeleteUtterances` ou desativando o armazenamento de dados. Todas as declarações serão excluídas se você fechar sua AWS conta. Os enunciados armazenados são criptografados com uma chave gerenciada pelo servidor.

Quando você exclui uma versão do bot, as estatísticas de enunciado ficam disponíveis para a versão por até 30 dias com `ListUtteranceMetrics` e 15 dias usando `ListAggregatedUtterances`. Você não pode ver as estatísticas da versão excluída no console do Amazon Lex V2. Para ver as estatísticas de versões excluídas, você pode usar as operações `ListAggregatedUtterances` e `ListUtteranceMetrics`.

Com as APIs `ListAggregatedUtterances` e `ListUtteranceMetrics`, os enunciados são agregados pelo texto do enunciado. Por exemplo, todas as instâncias em que o cliente usou a frase “Quero pedir uma pizza” são agregadas na mesma linha em uma resposta. Quando você usa a RecognizeUtteranceoperação, o texto usado é a transcrição de entrada.

Para usar as APIs `ListAggregatedUtterances` e `ListUtteranceMetrics`, aplique a política a seguir a uma função.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
```

```
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
            "Resource": "*"
        }
    ]
}
```

Gerenciar permissões de acesso para análise

Para fornecer ao usuário acesso às análises, anexe uma política a um perfil do IAM que permita que a função chame as operações de API para análise. Você pode anexar a Política gerenciada pela AWS: AmazonLexFullAccess ao perfil do IAM para fornecer acesso total às operações da API do Amazon Lex ou criar uma política personalizada que permita apenas permissões para análises e anexá-la a um perfil do IAM.

Para criar uma política personalizada contendo permissões para análise

1. Para criar um perfil do IAM, siga as etapas descritas em Criar um perfil para delegar permissões a um usuário do IAM.
2. Siga as etapas em Criar políticas do IAM para criar uma política usando o seguinte objeto JSON. Para permitir o acesso de análise a bots específicos para o perfil do IAM, adicione o ARN de cada bot ao campo `Resource`. Substitua a *`região`*, o *`ID da conta`* e o *`BOTID`* pelos valores correspondentes aos bots. Você também pode substituir o identificador da declaração, *`AnalyticsActions`*, por um nome de sua escolha.

```
{
    "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. Anexe a política que você criou ao perfil ao qual você deseja conceder permissões de análise seguindo as etapas em Adicionar e remover permissões de identidade do IAM.
4. Agora, o perfil deve ter permissões para visualizar as análises dos bots que você especificou.

Ativar logs de conversa

Use logs de conversas para armazenar conversas de usuários com seu bot. Analise esses logs para identificar problemas nas interações do seu bot com os usuários e modifique o comportamento do seu bot com esses insights. Esta seção também descreve como ofuscar os valores dos slots para proteger a privacidade dos usuários.

Tópicos

- Criar registros com logs de conversas
- Ocultar valores de slot nos logs de conversa
- Captura seletiva de log de conversa

Criar registros com logs de conversas

Você habilita logs de conversa para armazenar interações com o bot. É possível usar esses logs para revisar o desempenho do seu bot e solucionar problemas com conversas. Você pode registrar o texto da RecognizeTextoperação. Você pode registrar texto e áudio para a RecognizeUtteranceoperação. Ao habilitar os logs de conversa, você obtém uma visão detalhada das conversas que os usuários têm com seu bot.

Por exemplo, uma sessão com seu bot tem um ID de sessão. É possível usar esse ID para obter a transcrição da conversa, incluindo declarações do usuário e as respostas correspondentes do bot. Você também obtém metadados, como nome de intenção e valores de slot para uma declaração.

> **Note**
>
> Não é possível usar logs de conversa com um bot sujeito à Lei de Proteção de Privacidade Online das Crianças (COPPA).

Os logs de conversa são configurados para um alias. Cada alias pode ter configurações diferentes para seus logs de texto e áudio. É possível habilitar logs de texto, logs de áudio ou os dois para cada alias. Os registros de texto armazenam entradas de texto, transcrições de entrada de áudio e metadados associados nos registros. CloudWatch Os logs de áudio armazenam entrada de áudio no Amazon S3. É possível habilitar a criptografia de logs de texto e áudio usando CMKs gerenciados pelo cliente do AWS KMS .

Para configurar o registro, use o console ou a operação CreateBotAlias ou UpdateBotAlias. Depois de ativar os registros de conversação para um alias, usar a RecognizeUtteranceoperação RecognizeTextou para esse alias registra as declarações de texto ou áudio no grupo de registros de CloudWatch registros configurado ou no bucket do S3.

Tópicos

- Políticas do IAM para logs de conversa
- Configurar logs de conversa
- Visualização de registros de texto no Amazon CloudWatch Logs
- Acessar logs de áudio no Amazon S3
- Monitorando o status do registro de conversas com CloudWatch métricas

Políticas do IAM para logs de conversa

Dependendo do tipo de registro selecionado, o Amazon Lex V2 exige permissão para usar os buckets Amazon CloudWatch Logs e Amazon Simple Storage Service (S3) para armazenar seus registros. Você deve criar AWS Identity and Access Management funções e permissões para permitir que o Amazon Lex V2 acesse esses recursos.

Criar um perfil e políticas do IAM para logs de conversa

Para habilitar registros de conversas, você deve conceder permissão de gravação para CloudWatch Logs e Amazon S3. Se você habilitar a criptografia de objetos para seus objetos do S3, precisará conceder permissão de acesso às AWS KMS chaves usadas para criptografar os objetos.

Você pode usar o console do IAM, a API do IAM ou o AWS Command Line Interface para criar a função e as políticas. Essas instruções usam o AWS CLI para criar a função e as políticas.

> **Note**
>
> O código a seguir é formatado para Linux e MacOS. Para Windows, substitua o caractere de continuação de linha do Linux (\) pelo circunflexo (^).

Para criar um perfil do IAM para logs de conversa

1. Crie um documento no diretório atual chamado **`LexConversationLogsAssumeRolePolicyDocument.json`**, adicione o código a seguir a ele e salve-o. Esse documento de política adiciona o Amazon Lex V2 como uma entidade confiável ao perfil. Isso permite que o Amazon Lex assuma a função para entregar logs aos recursos configurados para logs de conversa.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. No AWS CLI, execute o comando a seguir para criar a função do IAM para registros de conversas.

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

Em seguida, crie e anexe uma política à função que permite que o Amazon Lex V2 grave em CloudWatch Logs.

Para criar uma política do IAM para registrar o texto da conversa no CloudWatch Logs

1. Crie um documento no diretório atual chamado **`LexConversationLogsCloudWatchLogsPolicy.json`**, adicione a ele a política do IAM a seguir e salve-o.

```
{
  "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. No AWS CLI, crie a política do IAM que concede permissão de gravação ao grupo de CloudWatch registros de registros.

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

3. Associe a política ao perfil do IAM criado para logs de conversa.

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

Se você estiver criando logs de áudio em um bucket do S3, crie uma política que permita ao Amazon Lex V2 gravar no bucket.

Como criar uma política do IAM para criar logs de áudio em um bucket do S3

1. Crie um documento no diretório atual chamado **`LexConversationLogsS3Policy.json`**, adicione a ele a política a seguir e salve-o.

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

2. No AWS CLI, crie a política do IAM que concede permissão de gravação ao seu bucket do S3.

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

3. Associe a política ao perfil criado para logs de conversa.

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

Conceder permissão para passar um perfil do IAM

Quando você usa o console AWS Command Line Interface, o ou um AWS SDK para especificar uma função do IAM a ser usada para registros de conversas, o usuário que especifica a função IAM dos registros de conversas deve ter permissão para passar a função para o Amazon Lex V2. Para permitir que o usuário passe o perfil ao Amazon Lex V2, é necessário conceder a permissão de `PassRole` ao usuário do IAM, ao perfil ou ao grupo do usuário.

A política a seguir define a permissão que será concedida ao usuário, ao perfil ou ao grupo. É possível usar as chaves de condição `iam:AssociatedResourceArn` e `iam:PassedToService` para limitar o escopo da permissão. Para obter mais informações, consulte Conceder permissões a um usuário para passar uma função para um AWS serviço e IAM e chaves de contexto de AWS STS condição no Guia do AWS Identity and Access Management usuário.

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

Configurar logs de conversa

Habilite e desabilite os logs de conversa usando o console ou o campo `conversationLogSettings` da operação `CreateBotAlias` ou `UpdateBotAlias`. É possível ativar ou desativar logs de áudio, logs de texto ou ambos. O registro em log começa em novas sessões do bot. As alterações nas configurações de log não são refletidas nas sessões ativas.

Para armazenar registros de texto, use um grupo de CloudWatch registros do Amazon Logs em sua AWS conta. É possível usar qualquer grupo de logs válido. O grupo de logs deve estar na mesma região que o bot do Amazon Lex V2. Para obter mais informações sobre a criação de um grupo de CloudWatch registros de registros, consulte Como trabalhar com grupos de registros e fluxos de registros no Guia do usuário do Amazon CloudWatch Logs.

Para armazenar registros de áudio, use um bucket do Amazon S3 em sua AWS conta. É possível usar qualquer bucket válido do S3. O bucket deve estar na mesma região que o bot do Amazon Lex V2. Para mais informações sobre como criar um bucket do Amazon S3, consulte Criar um bucket no Guia de conceitos básicos do Amazon Simple Storage Service.

Quando você gerencia logs de conversas usando o console, o console atualiza seu perfil de serviço para que tenha acesso ao grupo de logs e ao bucket do S3.

Se você não estiver usando o console, é necessário fornecer um perfil do IAM com políticas que permitam ao Amazon Lex V2 gravar no grupo de logs ou no bucket configurado. Se você criar uma função vinculada ao serviço usando o AWS Command Line Interface, deverá adicionar um sufixo personalizado à função usando a `custom-suffix` opção, como no exemplo a seguir. Para ter mais informações, consulte Criar um perfil e políticas do IAM para logs de conversa.

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.amazon.aws.com \
    --custom-suffix suffix
```

O perfil do IAM utilizado para habilitar logs de conversa deve ter a permissão `iam:PassRole`. A política a seguir deve ser anexada ao perfil:

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

Ativar logs de conversa

Para ativar os logs usando o console

1. Abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2.
2. Na lista, escolha um bot.
3. No menu à esquerda, selecione Aliases.
4. Na lista de aliases, escolha o alias para o qual você deseja configurar logs de conversa.
5. Na seção Logs de conversas, escolha Gerenciar logs de conversas.
6. Para registros de texto, escolha Ativar e insira o nome do grupo de CloudWatch registros do Amazon Logs.
7. Para logs de áudio, escolha Ativar e, em seguida, insira as informações do bucket do S3.
8. Opcional. Para criptografar registros de áudio, escolha a AWS KMS chave a ser usada para criptografia.

9. Escolha Salvar para iniciar o registro em log de conversas. Se necessário, o Amazon Lex V2 atualizará sua função de serviço com permissões para acessar o grupo de CloudWatch logs de registros e o bucket S3 selecionado.

Desativar logs de conversa

Como desativar os logs usando o console

1. Abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2.
2. Na lista, escolha um bot.
3. No menu à esquerda, selecione Aliases.
4. Na lista de aliases, escolha o alias para o qual você deseja configurar logs de conversa.
5. Na seção Logs de conversas, escolha Gerenciar logs de conversas.
6. Desative o log de texto, log de áudio ou ambos para desativar a criação de logs.
7. Escolha Salvar para interromper o registro em log de conversas.

Visualização de registros de texto no Amazon CloudWatch Logs

O Amazon Lex V2 armazena registros de texto para suas conversas no Amazon CloudWatch Logs. Para ver os registros, use o console de CloudWatch registros ou a API. Para obter mais informações, consulte Pesquisar dados de log usando padrões de filtro e sintaxe de consulta do CloudWatch Logs Insights no Guia do usuário do Amazon CloudWatch Logs.

Para visualizar os logs usando o console do Amazon Lex V2

1. Abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2.
2. Na lista, escolha um bot.
3. No menu à esquerda, escolha Analytics e, em seguida, escolha CloudWatch métricas.
4. Veja as métricas do seu bot na página de CloudWatch métricas.

Você também pode usar o CloudWatch console ou a API para visualizar suas entradas de registro. Para localizar as entradas de log, navegue até o grupo de logs configurado para o alias. Você pode encontrar o prefixo do stream de logs para seus registros no console do Amazon Lex V2 ou usando a operação DescribeBotAlias.

As entradas de log para uma expressão de usuário estão em vários fluxos de log. Uma declaração na conversa tem uma entrada em um dos fluxos de log com o prefixo especificado. Uma entrada no fluxo de log contém as seguintes informações.

message-version

A versão do esquema de mensagem.

bot

Detalhes sobre o bot com o qual o cliente está interagindo.

mensagens

A resposta que o bot enviou de volta ao usuário.

utteranceContext

Informações sobre o processamento desse enunciado.

- `runtimeHints`—contexto de runtime usado para transcrever e interpretar a entrada do usuário. Para ter mais informações, consulte Melhorar o reconhecimento dos valores de slots com sugestões de runtime.
- `slotElicitationStyle`—Estilo de elicitação de slots usado para interpretar a entrada do usuário. Para ter mais informações, consulte Capturar valores de slots com estilos de soletração.

sessionState

O estado atual da conversa entre o usuário e o bot. Para ter mais informações, consulte Gerenciar conversas.

interpretations

Uma lista de intenções que o Amazon Lex V2 determinou que poderiam satisfazer o enunciado do usuário. Usar pontuações de confiança.

interpretationSource

Indica se um slot foi resolvido pelo Amazon Lex ou pelo Amazon Bedrock. Valores: Lex | Bedrock

sessionId

O identificador da sessão do usuário que está tendo a conversa.

inputTranscript

Uma transcrição da entrada do usuário.

- Para entrada de texto, esse é o texto que o usuário digitou. Para entrada DTMF, essa é a chave que o usuário insere.
- Para entrada de voz, esse é o texto no qual o Amazon Lex V2 converte o enunciado do usuário para invocar uma intenção ou preencher um slot.

cru InputTranscript

A transcrição bruta da entrada do usuário antes de qualquer processamento de texto. Observação: o processamento de texto é somente para localidades en-US e en-GB.

transcriptions

Uma lista de possíveis transcrições da entrada do usuário. Para ter mais informações, consulte Como usar pontuações de confiança na transcrição de voz.

rawTranscription

Usar pontuações de confiança na transcrição de voz. Para ter mais informações, consulte Como usar pontuações de confiança na transcrição de voz.

missedUtterance

Indica se o Amazon Lex V2 conseguiu reconhecer o enunciado do usuário.

requestId

O Amazon Lex V2 gerou o ID de solicitação para a entrada do usuário.

timestamp

O carimbo de data e hora da entrada do usuário.

developerOverride

Indica se o fluxo da conversa foi atualizado usando um hook de código de diálogo. Para mais informações sobre como usar um hook de código de diálogo, consulte Habilitando a lógica personalizada com as funções do AWS Lambda.

inputMode

Indica o tipo de entrada. Pode ser áudio, DTMF ou texto.

requestAttributes

Os atributos da solicitação usados ao processar a entrada do usuário.

audioProperties

Se os logs de conversas de áudio estiverem ativados e a entrada do usuário estiver no formato de áudio, inclua a duração total da entrada de áudio, a duração da voz e a duração do silêncio no áudio. Também inclui um link para o arquivo de áudio.

bargeIn

Indica se a entrada do usuário interrompeu a resposta anterior do bot.

responseReason

O motivo pelo qual uma resposta foi gerada. Pode ser um dos seguintes:

- `UtteranceResponse` – resposta à entrada do usuário
- `StartTimeout` – resposta gerada pelo servidor quando o usuário não forneceu a entrada
- `StillWaitingResponse` – resposta gerada pelo servidor quando o usuário solicita que o bot espere
- `FulfillmentInitiated` – resposta gerada pelo servidor informando que o atendimento está prestes a ser iniciado
- `FulfillmentStartedResponse` – resposta gerada pelo servidor informando que o atendimento foi iniciado
- `FulfillmentUpdateResponse` – resposta periódica gerada pelo servidor enquanto o atendimento está em andamento
- `FulfillmentCompletedResponse` – resposta gerada pelo servidor quando o atendimento é concluído.

operationName

A API usada para interagir com o bot. Pode ser `PutSession`, `RecognizeText`, `RecognizeUtterance` ou `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"
}
```

O conteúdo da entrada de log depende do resultado de uma transação e da configuração do bot e da solicitação.

- Os campos `intent`, `slots` e `slotToElicit` não aparecerão em uma entrada se o campo `missedUtterance` for `true`.
- O campo `s3PathForAudio` não aparecerá se os logs de áudio estiverem desativados ou se o campo `inputDialogMode` for `Text`.
- O campo `responseCard` só aparecerá quando você tiver definido um cartão de resposta para o bot.
- O mapa `requestAttributes` só aparecerá se você tiver especificado atributos de solicitação na solicitação.
- O campo `kendraResponse` só está presente quando o `AMAZON.KendraSearchIntent` faz uma solicitação para pesquisar um índice do Amazon Kendra.

- O campo `developerOverride` é verdadeiro quando uma intenção alternativa foi especificada na função do Lambda do bot.
- O mapa `sessionAttributes` só aparecerá se você tiver especificado atributos de sessão na solicitação.
- O mapa `sentimentResponse` só aparecerá se você configurar o bot para retornar valores de sentimento.

> **Note**
>
> O formato de entrada pode mudar sem uma alteração correspondente em `messageVersion`. Seu código não deve gerar um erro se novos campos estiverem presentes.

Acessar logs de áudio no Amazon S3

O Amazon Lex V2 armazena logs de áudio para suas conversas em um bucket do S3.

É possível usar o console ou a API do Amazon S3 para acessar logs de áudio. Você pode ver o prefixo de chaves de objeto do S3 dos arquivos de áudio no console do Amazon Lex V2 ou no campo `conversationLogSettings` na resposta da operação `DescribeBotAlias`.

Monitorando o status do registro de conversas com CloudWatch métricas

Use CloudWatch a Amazon para monitorar as métricas de entrega dos seus registros de conversas. É possível definir alarmes em métricas para que você esteja ciente de problemas com o registro em log se eles ocorrerem.

O Amazon Lex V2 fornece quatro métricas no namespace `AWS/Lex` para logs de conversa:

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

As métricas de sucesso mostram que o Amazon Lex V2 gravou com êxito seus logs de áudio ou texto em seus destinos.

As métricas de falha mostram que o Amazon Lex V2 não conseguiu entregar logs de áudio ou texto ao destino especificado. Normalmente, este é um erro de configuração. Quando suas métricas de falha estiverem acima de zero, verifique o seguinte:

- Certifique-se de que o Amazon Lex V2 seja uma entidade confiável para o perfil do IAM.
- Para registro de texto, verifique se o grupo CloudWatch Registros existe. Para criar log de áudio, certifique-se de que o bucket do S3 exista.
- Certifique-se de que a função do IAM que o Amazon Lex V2 usa para acessar o grupo de CloudWatch logs de registros ou o bucket do S3 tenha permissão de gravação para o grupo de logs ou bucket.
- Certifique-se de que o bucket do S3 exista na mesma região que o bot do Amazon Lex V2 e pertença à sua conta.

Ocultar valores de slot nos logs de conversa

O Amazon Lex V2 permite ofuscar (ocultar) o conteúdo dos slots para que ele não fique visível. Para proteger dados confidenciais capturados como valores de slot, é possível ativar a ofuscação de slot para mascarar esses valores para o registro em log.

Ao optar por ofuscar valores de slot, o Amazon Lex V2 substitui o valor do slot pelo nome do slot nos logs de conversa. Para um slot chamado `full_name`, o valor do slot seria ofuscado da seguinte forma:

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

Se uma expressão contiver caracteres de colchete ({}), o Amazon Lex V2 insere um caractere de escape nos caracteres de colchete com duas barras invertidas (\\). Por exemplo, o texto `{John Stiles}` é ofuscado da seguinte forma:

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

Os valores de slot são ofuscados nos logs de conversa. Os valores de slot ainda estão disponíveis na resposta das operações `RecognizeText` e `RecognizeUtterance`, e os valores de slot estão disponíveis para suas funções do Lambda de validação e atendimento. Se você estiver usando valores de slot em seus prompts ou respostas, eles não serão ofuscados nos logs de conversa.

No primeiro turno de uma conversa, o Amazon Lex V2 ofuscará valores de slot se ele reconhecer um slot e um valor de slot no enunciado. Se nenhum valor de slot for reconhecido, o Amazon Lex V2 não ofuscará o enunciado.

No segundo turno e nos posteriores, o Amazon Lex V2 sabe qual slot elicitar e se o valor de slot deve ser ofuscado. Se o Amazon Lex V2 reconhecer o valor de slot, o valor será ofuscado. Se o Amazon Lex V2 não reconhecer um valor, toda a expressão será ofuscada. Nenhum valor de slot em declarações perdidas será ofuscado.

O Amazon Lex V2 também não ofusca valores de slot armazenados em atributos de solicitação ou sessão. Se você estiver armazenando valores de slot que devem ser ofuscados como um atributo, deverá criptografar ou ofuscar o valor.

O Amazon Lex V2 não ofusca o valor de slot no áudio. Ele ofusca o valor de slot na transcrição de áudio.

Você pode escolher quais slots ofuscar usando o console ou a API do Amazon Lex V2. No console, escolha Ofuscação de slot nas configurações de um slot. Se você estiver usando a API, defina o `obfuscationSetting` campo do slot como `DEFAULT_OBFUSCATION` quando você chama a UpdateSlotoperação CreateSlotou.

Captura seletiva de log de conversa

A captura seletiva do registro de conversas permite que o usuário selecione como os logs de conversas são capturados com dados de texto e áudio das conversas ao vivo.

Para ativar e capturar a saída do recurso de captura seletiva de logs de conversas, você deve ativar o recurso no console do Amazon Lex V2 e habilitar os atributos de sessão necessários nas configurações da API para capturar a saída selecionada dos logs.

Você pode selecionar as seguintes opções para a captura seletiva de log de conversas:

- somente texto
- somente áudio
- texto e áudio

Você pode capturar partes específicas da conversa e escolher se áudio, texto ou ambos são capturados para o log da conversa.

> **Note**
>
> A captura seletiva de log de conversa funciona somente no Amazon Lex V2.

Tópicos

Gerenciar a captura seletiva de log de conversa

Usando o console do Lex, você pode ativar as configurações de captura seletiva de registros de conversas e escolher em quais slots deseja habilitar a captura seletiva de log de conversas.

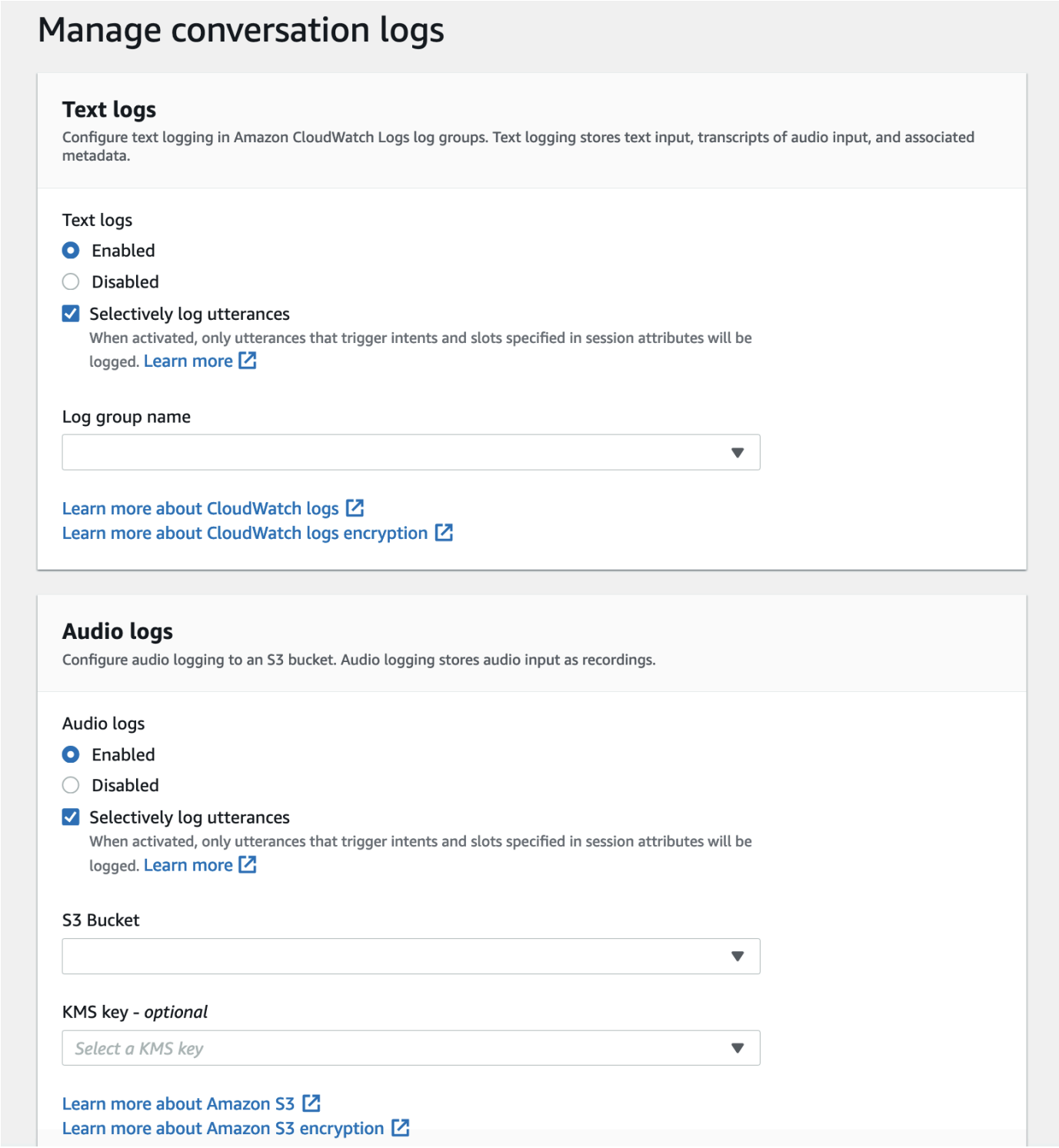
Ative a captura seletiva de logs de conversas no console Amazon Lex V2:

1. Faça login AWS Management Console e abra o console Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Selecione Bots nos painéis laterais esquerdos e escolha o bot que você deseja para ativar a captura seletiva do log de conversas. Use um bot existente ou crie um novo.
3. Escolha Aliases para o bot selecionado na seção Implantação no painel lateral esquerdo.
4. Escolha o alias do seu bot e selecione Gerenciar logs de conversas.
5. No painel Gerenciar logs de conversas, em Logs de texto, escolha se os logs de texto estão ativados ou desativados selecionando o botão de opção. Se você escolher Ativado para logs de texto, precisará inserir um Nome de grupo de logs ou escolher um nome de grupo de logs existente no menu suspenso. Marque a caixa de seleção Log seletivo de enunciados se você estiver criando logs seletivamente de arquivos de texto.

   > **Note**
   >
   > Ative registros de texto e/ou áudio marcando a caixa de seleção Registrar frases seletivamente nas configurações de registros de conversação (texto e/ou áudio) nas

configurações de tempo de construção. BotAlias Você deve configurar o grupo de CloudWatch logs e o bucket do Amazon S3 para selecionar essa opção.

6. Na seção Logs de áudio, escolha se os logs de áudio estão ativados ou desativados selecionando o botão de opção. Se você escolher Ativado para logs de áudio, precisará especificar a localização do bucket do Amazon S3 e (opcional) a chave KMS para criptografar seus dados de áudio. Marque a caixa de seleção Log seletivo de enunciados se você estiver criando logs seletivamente de arquivos de áudio.

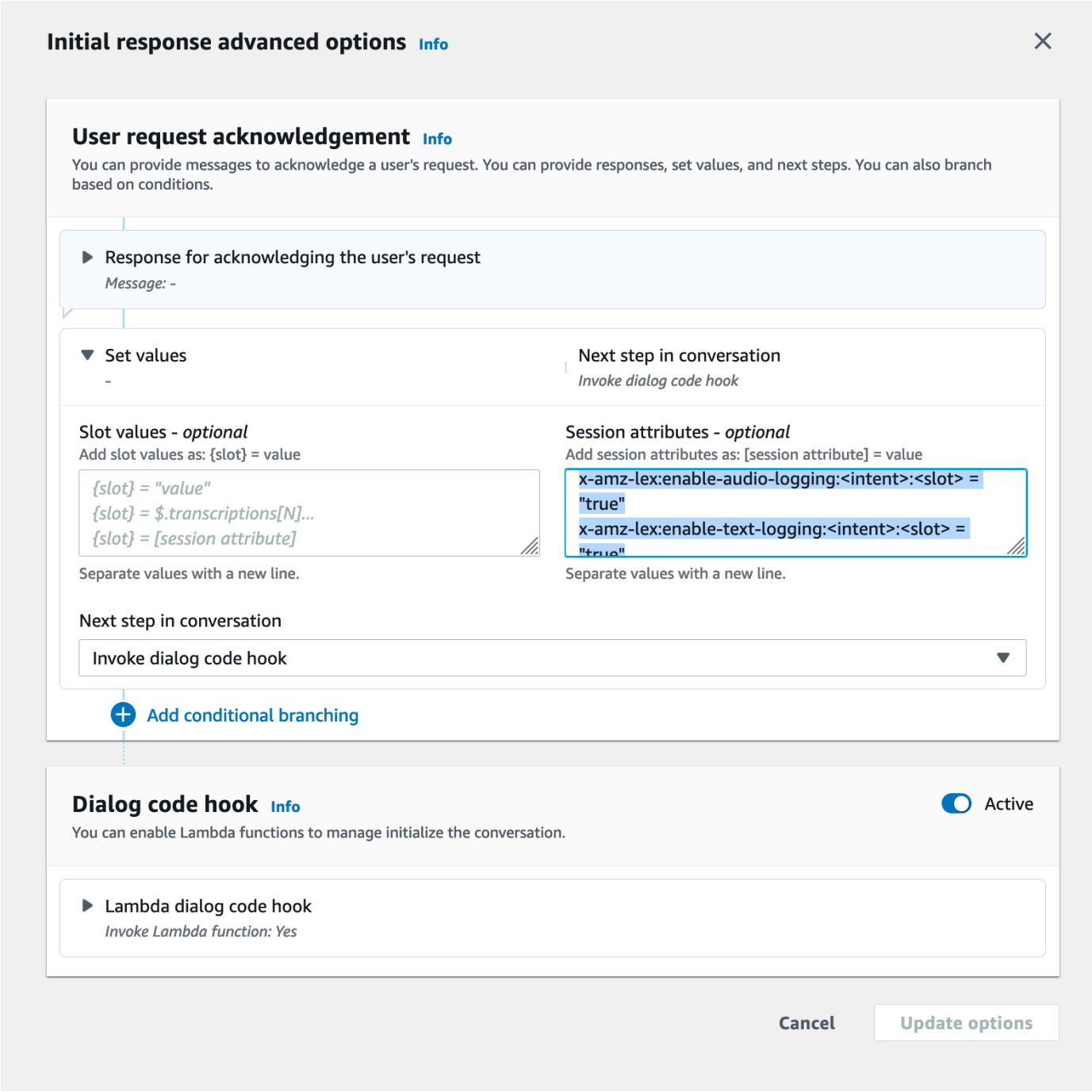


7. Selecione Salvar no canto inferior direito do painel para salvar suas configurações de captura seletiva de log de conversas.

Ative a captura seletiva de logs de conversas no console Lex:

1. Vá até Intenções e selecione o Nome da intenção, Resposta inicial, Configurações avançadas, os Valores definidos e Atributos da sessão.
2. Defina os seguintes atributos com base nas intenções e nos slots para os quais você deseja ativar a captura seletiva do log de conversas:

   - `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`*` = "true"`
   - `x-amz-lex:enable-text-logging:`*`intent`*`:`*`slot`*` = "true"`



> **Note**
>
> Defina `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`*` = "true"` para capturar enunciados que contenham somente um slot específico na conversa. A ação para criar log de um enunciado depende da avaliação de *`intenção`*: *`slot`* dentro do

enunciado, em comparação com os enunciados do atributo da sessão, e o valor do sinalizador correspondente. Para criar log de um enunciado, pelo menos um enunciado no atributo de sessão deve permitir isso, com o sinalizador Ativar log definido como `true`. O valor da *`intenção`* e do *`slot`* também pode ser `"*"`. Se o valor do slot e/ou da intenção for `"*"`, significa que qualquer slot e/ou valor de intenção de `"*"` corresponderá a ele. Semelhante a `x-amz-lex:enable-audio-logging`, um novo atributo de sessão chamado `x-amz-lex:enable-text-logging` será usado para controlar logs de texto.

3. Selecione Opções de atualização e crie o bot para incluir as configurações atualizadas.

> **Note**
>
> Seu perfil do IAM deve ter permissão de acesso para permitir que você grave dados no bucket do Amazon S3 e use uma chave KMS para criptografar os dados. Lex atualizará sua função do IAM com as permissões do Lex para acessar o grupo de CloudWatch registros de registros e o bucket Amazon S3 selecionado.

Diretrizes para usar a captura seletiva de logs de conversas:

Você só pode ativar a captura seletiva de logs de conversas para logs de texto e/ou áudio quando tiver habilitado logs de texto e/ou áudio nas Configurações de logs de conversa. Ao ativar a captura seletiva de logs de conversas para logs de texto e/ou áudio, você desativa a criação de logs para todas as intenções e slots na conversa. Para gerar logs de texto e/ou áudio para intenções e slots específicos, você deve definir os atributos da sessão de captura seletiva de log de conversas de texto e/ou áudio para essas intenções e slots como “verdadeiros”.

- Se a captura seletiva do registro de conversação estiver ativada e nenhum atributo de sessão com o prefixo x-amz-lex: enable-audio-logging estiver presente, o registro será desativado por padrão para todos os enunciados. Esse cenário também é verdadeiro em relação x-amz-lex a:enable-text-logging.
- Os logs de enunciados serão armazenados exclusivamente para os segmentos de conversa em texto e/ou áudio, se pelo menos um enunciado no atributo de sessão permitir.
- As configurações para captura seletiva de log de conversas de texto e/ou áudio, conforme definido nos atributos da sessão, serão efetivas somente quando a captura seletiva do log de conversas

para texto e/ou áudio estiver ativada nas Configurações do logs de conversa dentro do alias do bot; caso contrário, os atributos da sessão serão desconsiderados.

- Quando a captura seletiva do registro de conversas estiver ativada, quaisquer valores de slot em SessionState, Interpretações e Transcrições para os quais o registro não esteja ativado usando atributos de sessão serão ofuscados no registro de texto gerado.
- A decisão de produzir logs de áudio e/ou texto é avaliada combinando o slot obtido pelo bot com os atributos da sessão de captura seletiva do log de conversa, exceto pelo turno de elicitação de intenção, em que o usuário pode fornecer valores de slot junto com a elicitação de intenção. Em um turno de elicitação de intenção, os slots preenchidos no turno atual são comparados com os atributos da sessão de captura seletiva do log de conversas.
- Os slots considerados preenchidos são derivados do estado da sessão no final do turno. Portanto, quaisquer alterações feitas pelo Hook de código de diálogo do Lambda nos slots no estado da sessão influenciarão o comportamento da captura seletiva do log de conversa.
- Em um turno de elicitação de intenção, se vários valores de slot forem fornecidos pelo usuário, o log de texto e/ou áudio só será gerado se os atributos da sessão de texto/áudio permitirem a criação de log de todos os slots preenchidos nesse turno.
- A abordagem operacional recomendada é definir o atributo da sessão de captura seletiva do log de conversas no início da sessão e evitar modificá-lo durante a sessão.
- Se algum slot contiver dados confidenciais, você deve sempre habilitar a ofuscação do slot.

Exemplo de captura seletiva de log de conversa

Aqui está um exemplo de um caso de uso comercial para captura seletiva de logs de conversas.

Caso de uso:

Uma empresa fintech utiliza um bot do Amazon Lex V2 para dar suporte a seu sistema IVR, que permite aos usuários fazer pagamentos de contas. Para atender aos requisitos de conformidade e auditoria, ela deve reter gravações de áudio do consentimento de autorização fornecido pelo usuário. No entanto, habilitar registros gerais de áudio não é viável, pois os tornaria incompatíveis, pois não é possível ofuscar slots confidenciais CardNumber, como CVV e outras informações nos registros de áudio. Em vez disso, ela pode habilitar a captura seletiva de logs de conversas para logs de áudio e definir o atributo de sessão para produzir somente logs de áudio para enunciados que tenham consentimento de autorização.

BotAlias Configurações:

- Logs de texto ativados: verdadeiro
- Logs de texto seletivos ativado: falso
- Logs de áudio ativados: verdadeiro
- Logs de áudio seletivos ativado: verdadeiro

Atributos da sessão:

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

Exemplo de conversa:

- Usuário (entrada de áudio): “Quero pagar minha fatura com o número 35XU68.”
- Bot: “Qual é o valor devido em dólares?”
- Usuário (entrada de áudio): “235.”
- Bot: “Qual é o número do seu cartão de crédito?”
- Usuário (entrada de áudio): “9239829722200348.”
- Bot: “Você está pagando 235 dólares usando o número do seu cartão de crédito que termina em 0348. Por favor, diga 'Eu autorizo a pagar 235 dólares.'"
- Usuário (entrada de áudio): “Autorizo pagar 235 dólares.”
- Bot: “Sua conta foi paga.”

Saída de logs de conversas:

Nessa situação, logs de texto serão produzidos para todos os turnos. No entanto, os registros de áudio só serão gravados para um turno específico quando o AuthorizationConsentespaço dentro da PayBillintenção for obtido, e nenhum registro de áudio será produzido para nenhum outro turno.

Monitorar métricas operacionais

Amazon CloudWatch e AWS CloudTrail são dois AWS serviços que se integram ao Amazon Lex V2 para ajudar você a monitorar as interações do usuário com seu bot. Use esses serviços para registrar ações, enviar dados quase em tempo real e configurar notificações e ações automatizadas quando os critérios forem atendidos.

Tópicos

- Medindo métricas operacionais com a Amazon CloudWatch

- Visualizando eventos com AWS CloudTrail

Medindo métricas operacionais com a Amazon CloudWatch

Você pode monitorar o Amazon Lex V2 usando CloudWatch, que coleta dados brutos e os processa em métricas legíveis, quase em tempo real. Essas estatísticas são mantidas por 15 meses, de maneira que você possa acessar informações históricas e ter uma perspectiva melhor de como o aplicativo web ou o serviço está se saindo. Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

O serviço do Amazon Lex V2 relata as métricas a seguir no namespace do `AWS/Lex`.

Métrica	Descrição
`AssistedSlotResolutionModelAccessDeniedErrorCount`	O número de vezes que o Amazon Lex V2 teve seu acesso negado ao Amazon Bedrock Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation`: • BotId, BotAliasId, LocaleId, Operação, InputMode, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, InputMode, ModelType, Modelo Dimensões válidas para `RecognizeText`: • BotId, BotAliasId, LocaleId, Operação, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, ModelType, Modelo Unidade: Contagem
`AssistedSlotResolutionModelInvocationCount`	O número de vezes que o Amazon Bedrock foi invocado. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation`: • BotId, BotAliasId, LocaleId, Operação, InputMode, ModelType, Modelo

Métrica	Descrição
	• BotId, BotVersion, LocaleId, Operação, InputMode, ModelType, Modelo Dimensões válidas para `RecognizeText` : • BotId, BotAliasId, LocaleId, Operação, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, ModelType, Modelo Unidade: contagem
`AssistedSlotResolutionModelSystemErrorCount`	O número de vezes que um 5xx ocorreu ao chamar o Amazon Bedrock. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation` : • BotId, BotAliasId, LocaleId, Operação, InputMode, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, InputMode, ModelType, Modelo Dimensões válidas para `RecognizeText` : • BotId, BotAliasId, LocaleId, Operação, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, ModelType, Modelo Unidade: contagem

Métrica	Descrição
`AssistedSlotResolutionModelThrottlingErrorCount`	O número de vezes que a utilização do Amazon Lex foi controlada pelo Amazon Bedrock. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation`: • BotId, BotAliasId, LocaleId, Operação, InputMode, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, InputMode, ModelType, Modelo Dimensões válidas para `RecognizeText`: • BotId, BotAliasId, LocaleId, Operação, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, ModelType, Modelo Unidade: contagem
`AssistedSlotResolutionResolvedSlotCount`	O número de vezes que o Amazon Bedrock retornou um valor de slot. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation`: • BotId, BotAliasId, LocaleId, Operação, InputMode, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, InputMode, ModelType, Modelo Dimensões válidas para `RecognizeText`: • BotId, BotAliasId, LocaleId, Operação, ModelType, Modelo • BotId, BotVersion, LocaleId, Operação, ModelType, Modelo Unidade: contagem

Métrica	Descrição
KendraIndexAccessError	O número de vezes que o Amazon Lex V2 não conseguiu acessar seu índice do Amazon Kendra. • Operação, BotId, BotAliasId, LocaleId Unidade: contagem
KendraLatency	A quantidade de tempo que a Amazon Kendra leva para responder a uma solicitação do AMAZON.KendraSearchIntent . Dimensões válidas: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: milissegundos
KendraSuccess	O número de vezes que o Amazon Lex V2 não conseguiu acessar seu índice Amazon Kendra. Dimensões válidas: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: contagem
KendraSystemErrors	O número de vezes que o Amazon Lex V2 não conseguiu acessar seu índice do Amazon Kendra. Dimensões válidas: • Operação, BotId, BotAliasId, InputMode, LocaleId Unidade: contagem

Métrica	Descrição
`KendraThrottledEvents`	O número de vezes que o Amazon Kendra controlou a utilização das solicitações do `AMAZON.KendraSearchIntent` . Dimensões válidas: • Operação, BotId, BotAliasId, InputMode. LocaleId Unidade: contagem
`RuntimeConcurrency`	O número de conexões simultâneas no período especificado. `RuntimeConcurrency` é relatado como um `StatisticSet` . Dimensões válidas para as operações `RecognizeUtterance` ou `StartConversation` : • Operação, BotId, BotVersion, InputMode, LocaleId • Operação, BotId, BotAliasId, InputMode, LocaleId Dimensões válidas para outras operações: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: contagem
`RuntimeInvalidLambdaResponses`	O número de AWS Lambda respostas inválidas no período especificado. Dimensões válidas: • Operação, BotId, BotAliasId, InputMode, LocaleId Unidade: contagem

Métrica	Descrição
`RuntimeLambdaErrors`	O número de erros de runtime do Lambda no período especificado. Dimensões válidas: • Operação, BotId, BotAliasId, InputMode, LocaleId Unidade: contagem
`RuntimePollyErrors`	O número de respostas inválidas do Amazon Polly no período especificado. Dimensões válidas: • Operação, BotId, BotAliasId, InputMode, LocaleId Unidade: contagem
`RuntimeRequestCount`	O número de solicitações de runtime no período especificado. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation`: • Operação, BotId, BotVersion, InputMode, LocaleId • Operação, BotId, BotAliasId, InputMode, LocaleId Dimensões válidas para outras operações: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: contagem

Métrica	Descrição
`RuntimeRequestLength`	Duração total de uma conversa com um bot do Amazon Lex V2. Aplicável somente à StartConversationoperação. Dimensões válidas: • BotAliasId, BotId, LocaleId, Operação • BotId, BotAliasId, LocaleId, Operação Unidade: milissegundos
`RuntimeSuccessfulRequestLatency` **Important** Essa métrica é `RuntimeSuccessfulRequestLatency` e não `RuntimeSuccessfulRequestLatency` .	A latência de solicitações bem-sucedidas entre o horário em que a solicitação foi feita e a resposta foi passada. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation` : • Operação, BotId, BotVersion, InputMode, LocaleId • Operação, BotId, BotAliasId, InputMode, LocaleId Dimensões válidas para outras operações: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: milissegundos

Métrica	Descrição
`RuntimeSystemErrors`	O número de erros do sistema no período especificado. O intervalo de códigos de resposta para um erro do sistema vai de 500 até 599. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation` : • Operação, BotId, BotVersion, InputMode, LocaleId • Operação, BotId, BotAliasId, InputMode, LocaleId Dimensões válidas para outras operações: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: contagem

Métrica	Descrição
`RuntimeThrottledEvents`	O número de solicitações que passaram por controle de utilização. O Amazon Lex V2 controla a utilização de um evento ao receber mais solicitações do que o limite de transações por segundo definido para a conta. Se o limite definido para a conta for frequentemente excedido, você poderá solicitar um aumento no limite. Para solicitar um aumento, consulte Limites de serviço da AWS. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation` : • Operação, BotId, BotVersion, InputMode, LocaleId • Operação, BotId, BotAliasId, InputMode, LocaleId Dimensões válidas para outras operações: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: contagem

Métrica	Descrição
`RuntimeUserErrors`	O número de erros de usuário no período especificado. O intervalo de códigos de resposta para um erro de usuário vai de 400 até 499. Dimensões válidas para as operações `RecognizeUtterance` e `StartConversation`: • Operação, BotId, BotVersion, InputMode, LocaleId • Operação, BotId, BotAliasId, InputMode, LocaleId Dimensões válidas para outras operações: • Operação, BotId, BotVersion, LocaleId • Operação, BotId, BotAliasId, LocaleId Unidade: contagem

Há suporte para as dimensões a seguir para as métricas do Amazon Lex V2.

Dimensão	Descrição
`Operation`	O nome da operação do Amazon Lex V2 – `RecognizeText`, `RecognizeUtterance`, `StartConversation`, `GetSession`, `PutSession`, `DeleteSession` – que gerou a entrada.
`BotId`	O identificador alfanumérico exclusivo do bot.
`BotAliasId`	O identificador alfanumérico exclusivo do alias do bot.
`BotVersion`	A versão numérica do bot.
`InputMode`	O tipo de entrada para o bot – fala, texto ou DTMF.
`LocaleId`	O identificador da localidade do bot, como en-US ou fr-CA.

Dimensão	Descrição
`Model`	Indica a ID do grande modelo de linguagem no Amazon Bedrock.
`ModelType`	Indica o tipo de grande modelo de linguagem invocado no Amazon Bedrock.

Visualizando eventos com AWS CloudTrail

O Amazon Lex V2 é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Amazon Lex V2. CloudTrail captura chamadas de API para o Amazon Lex V2 como eventos. As chamadas capturadas incluem as chamadas do console do Amazon Lex V2 e as chamadas de código para as operações da API do Amazon Lex V2. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o Amazon Lex V2. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Amazon Lex V2, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Informações sobre o Amazon Lex V2 em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Amazon Lex V2, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para obter um registro contínuo dos eventos em sua AWS conta, incluindo eventos do Amazon Lex V2, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte as informações a seguir.

- Visão geral da criação de uma trilha
- CloudTrail serviços e integrações suportados
- Configurar notificações do Amazon SNS para o CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e Recebendo arquivos de CloudTrail log de várias contas

O Amazon Lex V2 oferece suporte ao registro em log para todas as ações listadas na API de criação de modelo V2.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário root ou do AWS Identity and Access Management IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte o elemento CloudTrail userIdentity.

Noções básicas sobre entradas de arquivos de log do Amazon Lex V2

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a ação CreateBotAlias.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ID of caller:temporary credentials",
        "arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
```

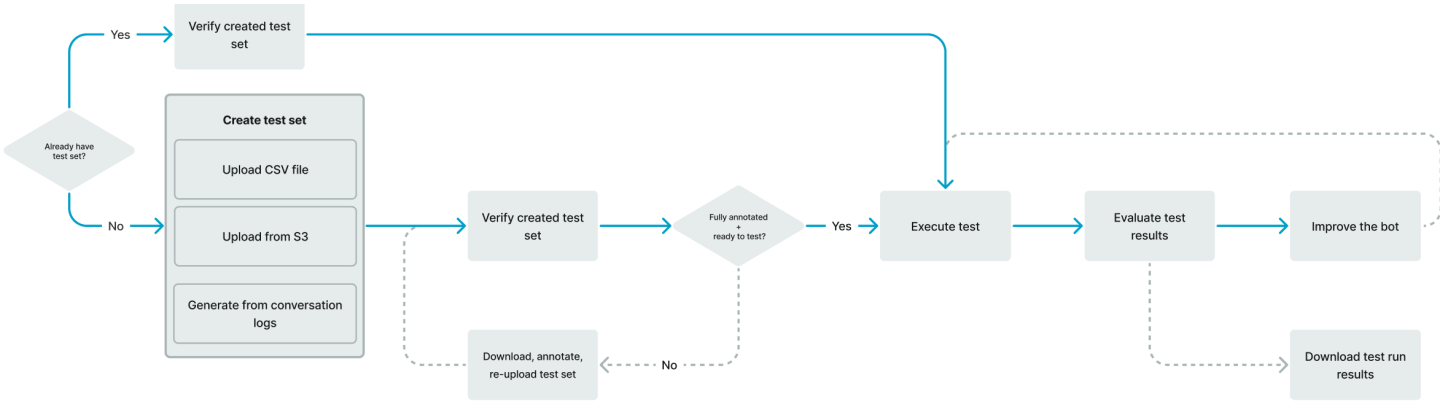
```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ID of caller",
                "arn": "arn:aws:iam::111122223333:role/role name",
                "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"
}
```

Avaliar desempenho do bot com o Test Workbench

Para melhorar o desempenho dos bots, você pode avaliar o desempenho deles em grande escala. Os resultados da avaliação do teste são exibidos em tabelas e gráficos simples.

Você pode usar o Test Workbench para criar conjuntos de testes de referência que usam dados de transcrição existentes. Você pode testar bots para avaliar o desempenho antes da implantação e visualizar os detalhamentos dos resultados do teste em grande escala.



Os usuários podem usar o Test Workbench para estabelecer o desempenho básico para bots. Isso abrange a intenção e o desempenho do slot para enunciados que estão na forma de entradas ou conversas únicas. Depois que um conjunto de testes for carregado com sucesso, você poderá executá-lo em seus bots de pré-produção ou produção existentes. O Test Workbench ajuda você a identificar oportunidades para melhorar o preenchimento de slots e a classificação de intenções.

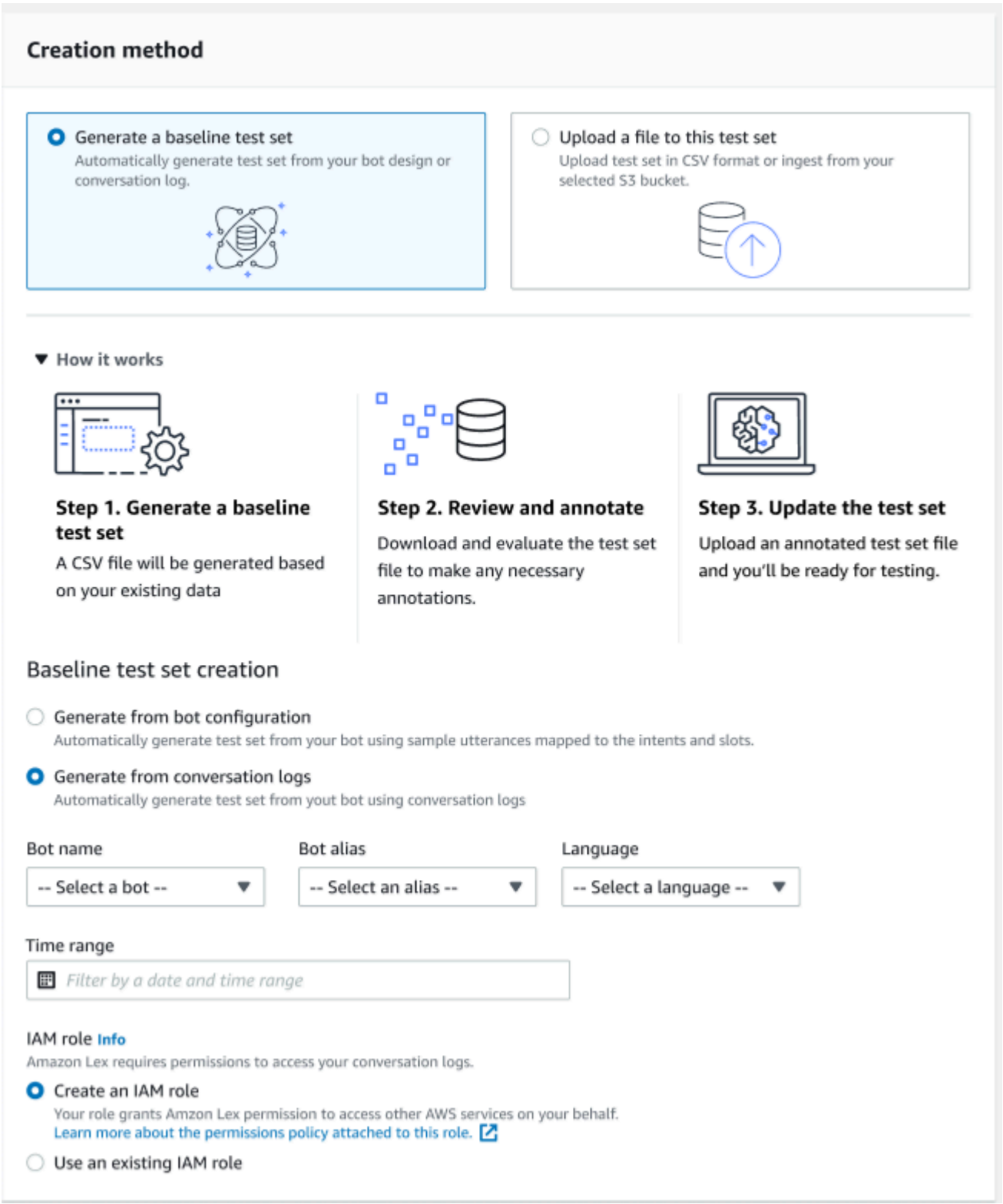
Tópicos

- Gerar um conjunto de teste
- Gerenciar conjuntos de teste
- Executar um teste
- Cobertura do conjunto de teste

- Visualizar resultados do teste
- Detalhes dos resultados do teste

Gerar um conjunto de teste

É possível criar um conjunto de testes para avaliar o desempenho do seu bot. Gere um conjunto de testes fazendo o upload de um conjunto de testes em formato de arquivo CSV ou gerando um conjunto de testes a partir de logs de conversas. O conjunto de teste pode conter entrada de áudio ou texto.



Se um conjunto de teste criar erros de validação, remova o conjunto de testes e substitua-o por outra lista de dados do conjunto de testes ou edite os dados no arquivo CSV usando um programa de edição de planilhas.

Para criar um conjunto de teste:

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Selecione Test Workbench no painel lateral esquerdo.
3. Selecione Conjuntos de teste nas opções em Test Workbench.
4. Selecione o botão Criar conjunto de teste no console.
5. Em Detalhes, digite o nome do conjunto de teste e uma descrição opcional.
6. Selecione Gerar um conjunto de teste de linha de base.
7. Selecione Gerar a partir de logs de conversas.
8. Selecione Nome do bot, Alias do bot e Idioma nos menus suspensos.
9. Se você estiver gerando um teste de linha de base a partir de um log de conversa, selecione Intervalo de tempo e Perfil do IAM, se necessário. Você pode criar um perfil com as permissões básicas do Amazon Lex V2 ou usar um perfil existente.
10. Selecione uma modalidade de Áudio ou Texto para o conjunto de teste que você está criando. OBSERVAÇÃO: o Test Workbench pode importar arquivos de texto de até 50k e até 5 horas de áudio.
11. Selecione um local do Amazon S3 para armazenar os resultados do teste e adicione uma chave KMS opcional para criptografar as transcrições de saída.
12. Escolha Criar.

Para fazer o upload de um conjunto de teste existente em formato de arquivo CSV ou para atualizar o conjunto de teste:

1. Selecione Test Workbench no painel lateral esquerdo.
2. Selecione Conjuntos de teste nas opções em Test Workbench.
3. Selecione Fazer upload de um arquivo para este conjunto de teste no console.
4. Escolha Fazer upload a partir do bucket do Amazon S3 ou Fazer upload a partir do seu computador. OBSERVAÇÃO: você pode carregar um arquivo CSV criado a partir de um modelo. Clique no modelo CSV para baixar um arquivo zip que contém os modelos.
5. Selecione Criar um perfil com as permissões básicas do Amazon Lex ou Usar um perfil existente para o ARN de perfil.

6. Selecione uma modalidade de Áudio ou Texto para o conjunto de teste que você está criando. OBSERVAÇÃO: o Test Workbench pode importar arquivos de texto de até 50k e até 5 horas de áudio.
7. Selecione um local do Amazon S3 para armazenar os resultados do teste e adicione uma chave KMS opcional para criptografar as transcrições de saída.
8. Escolha Criar.

Se a operação for bem-sucedida, a mensagem de confirmação indicará que o conjunto de teste está pronto para ser testado e o status exibirá Pronto para teste.

Dicas para criar um conjunto de teste bem-sucedido

- Você pode criar uma função do IAM para o Test Workbench no console ou configurar sua função step-by-step do IAM. Para mais informações, consulte Criar um perfil do IAM para o Test Workbench.
- Antes de executar um teste, valide o conjunto de teste e a definição do bot em busca de inconsistências usando o botão Validar discrepância. Se a intenção e as convenções de nomenclatura de slots usadas no conjunto de teste forem consistentes com o bot, execute o teste. Se alguma anomalia for identificada, revise o conjunto de teste, atualize o conjunto de teste e escolha Validar discrepância. Repita essa sequência novamente até que nenhuma inconsistência seja observada e, em seguida, execute o teste.
- O Test Workbench pode testar com diferentes formatos de valor de slot na coluna Slot de Saída Esperado. Para qualquer slot incorporado, você pode escolher o valor fornecido na entrada do usuário (por exemplo, Date = amanhã) ou fornecer seu valor absoluto resolvido (por exemplo, Date = 2023-03-21). Para mais informações sobre slots integrados e seus valores absolutos, consulte Slots integrados.
- Para consistência e legibilidade nas colunas do slot de saída esperado, siga a convenção de "SlotName = SlotValue" (por exemplo, AppointmentType = limpeza) com um espaço antes e depois do sinal de igual.
- Se o bot incluir slots compostos, em Slot de Saída Esperado, defina subslots com o nome do slot, separados por um ponto (por exemplo, “Car.Color”). Nenhuma outra sintaxe e pontuação funcionarão.
- Se o bot incluir slots de vários valores, em Slot de saída esperado, forneça vários valores de slot, separados por uma vírgula (” FlowerType = rosas, lírios”). Nenhuma outra sintaxe e pontuação funcionarão.

- Certifique-se de que o conjunto de teste seja criado a partir de logs de conversas válidos.
- O valor Slot:slot ficará na mesma coluna após as colunas de intenção no formato CSV.
- A entrada DTMF de um turno de usuário é interpretada como uma transcrição esperada e não lista uma localização do Amazon S3.

Criar um caso de teste dentro de um conjunto de teste

Os resultados do Test Workbench dependem da definição do bot e do conjunto de teste correspondente. Você pode gerar um conjunto de teste com as informações da definição do bot para identificar áreas que precisam ser aprimoradas. Crie um conjunto de dados de teste com exemplos que você suspeita (ou conhece) que serão difíceis para o bot interpretar corretamente, considerando o design atual do bot e seu conhecimento das conversas com os clientes.

Revise suas intenções com base nos aprendizados do seu bot de produção regularmente. Continue adicionando e ajustando os enunciados de amostra e os valores dos slots do bot. Considere melhorar a resolução do slot usando as opções disponíveis, como dicas de runtime. O design e o desenvolvimento do seu bot são um processo iterativo que é um ciclo contínuo.

Aqui estão algumas outras dicas para otimizar seu conjunto de teste:

- Selecione os casos de uso mais comuns com intenções e slots usados com frequência no conjunto de teste.
- Explore diferentes maneiras pelas quais um cliente pode se referir às suas intenções e vagas. Isso pode incluir entradas do usuário na forma de enunciados, perguntas e comandos que variam em tamanho, do mínimo ao estendido.
- Inclua entradas do usuário com um número variado de slots.
- Inclua sinônimos ou abreviações comumente usados de valores de slots personalizados aceitos pelo seu bot (por exemplo, “canal dentário”, “canal” ou “CD”).
- Inclua variações dos valores de slots integrados (por exemplo, “amanhã”, “o mais rápido possível” ou “no dia seguinte”).
- Examine a robustez do bot na modalidade falada coletando entradas do usuário que possam ser mal interpretadas (por exemplo, “tinta”, “tornozelo” ou “âncora”).

Criar um conjunto de teste a partir de um arquivo CSV

Você pode criar um conjunto de teste a partir do modelo de arquivo CSV fornecido no console do Amazon Lex V2 inserindo os valores diretamente usando um editor de planilhas CSV. O conjunto de teste é um arquivo de valores separados por vírgula (CSV) que consiste em enunciados de um único usuário e conversas de vários turnos gravadas nas seguintes colunas:

- Nº da linha – Esta coluna é um contador incremental que acompanha o total de linhas preenchidas a serem testadas.
- Nº da conversa – Esta coluna rastreia o número de turnos em uma conversa. Para entradas individuais, essa coluna pode ficar vazia, preenchida com “-” ou “N/D”. Para conversas, cada turno em uma conversa receberá o mesmo número de conversa.
- Fonte – Esta coluna está definida como “Usuário” ou “Agente”. Para entradas individuais, ela sempre será definida como “Usuário”.
- Entrada – Esta coluna inclui o enunciado do usuário ou as instruções do bot.
- Intenção de Saída Esperada – Esta coluna captura a intenção atendida na entrada.
- Slot de Saída Esperado da Intenção 1 – Esta coluna captura o primeiro slot obtido na entrada do usuário. O conjunto de teste deve incluir uma coluna chamada Slot de Saída Esperado X para cada slot na entrada do usuário.

Exemplo de um conjunto de teste com entradas individuais:

Nº da linha	Nº da conversa	Fonte	Entrada	Intenção de Saída Esperada	Slot de Saída Esperado 1	Slot de Saída Esperado 2
1		Usuário	marcar uma consulta de limpeza para amanhã	MakeAppointment	AppointmentType = limpeza	Date = amanhã
2	N/D	Usuário	marcar uma consulta	MakeAppointment	AppointmentType = limpeza	Date = 15/04/23

Nº da linha	Nº da conversa	Fonte	Entrada	Intenção de Saída Esperada	Slot de Saída Esperado 1	Slot de Saída Esperado 2
			de limpeza para 15 de abril			
3	N/D	Usuário	marcar uma consulta para primeiro de dezembro	MakeAppointment	Date = primeiro de dezembro	
4	N/D	Usuário	marcar uma consulta de limpeza	MakeAppointment	AppointmentType = limpeza	
1		Usuário	Você pode me ajudar a marcar uma consulta?	MakeAppointment		

Exemplo de um conjunto de teste com conversas

Nº da linha	Nº da conversa	Fonte	Entrada	Intenção de Saída Esperada	Slot de Saída Esperado 1	Slot de Saída Esperado 2	Slot de Saída Esperado 3
1	1	Usuário	marcar uma consulta	MakeAppointment			

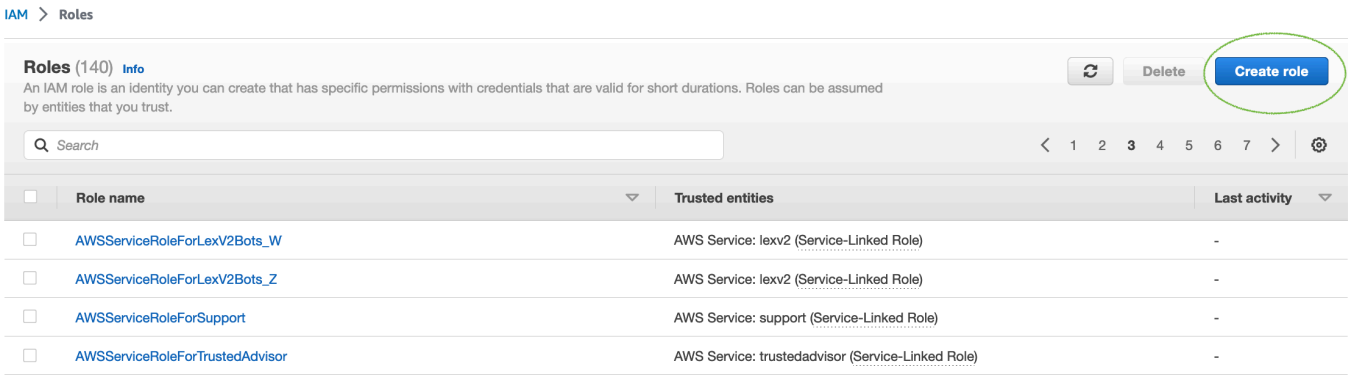
Nº da linha	Nº da conversa	Fonte	Entrada	Intenção de Saída Esperada	Slot de Saída Esperado 1	Slot de Saída Esperado 2	Slot de Saída Esperado 3
2	1	Atendente	Que tipo de consulta você deseja agendar?	MakeAppointment			
3	1	Usuário	limpeza	MakeAppointment	AppointmentType = limpeza		
4	1	Atendente	Para quando devo agendar sua consulta?	MakeAppointment			
5	1	Usuário	amanhã	MakeAppointment		Date = amanhã	
6	2	Usuário	marcar uma consulta para canal dentário para hoje	MakeAppointment	AppointmentType = canal radicular	Date = hoje	

Nº da linha	Nº da conversa	Fonte	Entrada	Intenção de Saída Esperada	Slot de Saída Esperado 1	Slot de Saída Esperado 2	Slot de Saída Esperado 3
7	2	Atendente	Para qual horário devo agendar sua consulta?	MakeAppointment			
8	2	Usuário	onze da manhã	MakeAppointment			Time = onze da manhã

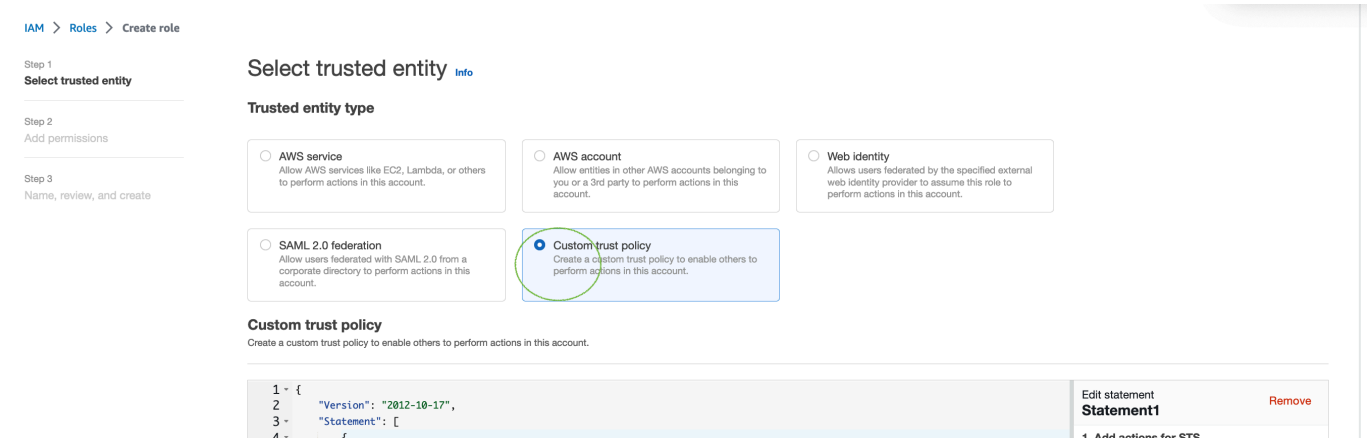
Criar um perfil do IAM para o Test Workbench

Para criar um perfil do IAM para o Test Workbench

1. Siga as etapas em Criar um usuário do IAM para criar um usuário do IAM que pode ser usado para acessar o console do test-workbench.
2. Selecione o botão Criar perfil.



3. Selecione a opção Política de confiança personalizada.



4. Insira a política de confiança abaixo e clique em Avançar.

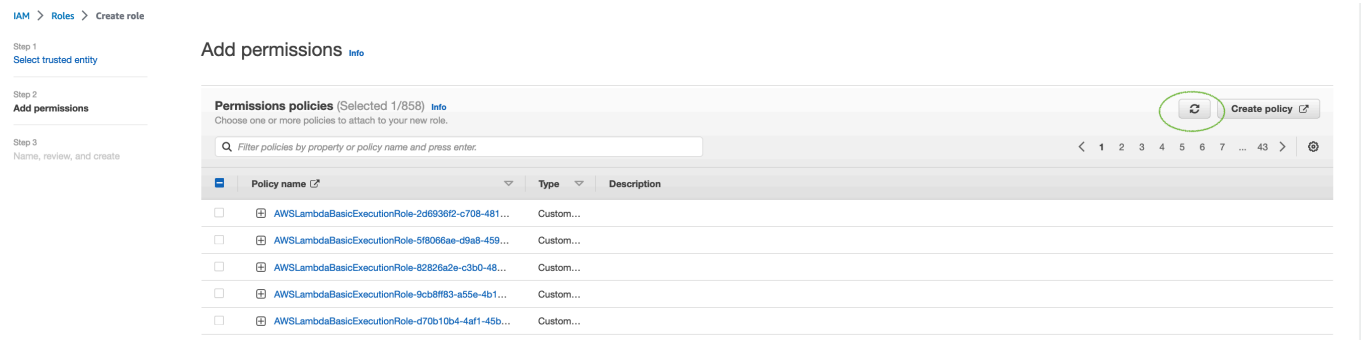
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. Selecione o botão Criar política.
6. Uma nova guia será aberta no seu navegador, onde você poderá inserir a política abaixo e clicar no botão Próximo: Tags.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Insira um nome de política, por exemplo LexTest WorkbenchPolicy " e clique em Criar política.
8. Volte para a guia anterior em seu navegador e atualize a lista de políticas clicando no botão Atualizar, conforme mostrado abaixo.



9. Pesquise na lista de políticas inserindo o nome da política que você usou na 6ª etapa e selecione a política.
10. Selecione o botão Avançar.
11. Insira o nome do perfil e clique no botão Criar perfil.
12. Escolha seu novo perfil do IAM quando solicitado no console do Amazon Lex V2 para o Test Workbench.

Gerenciar conjuntos de teste

Você pode baixar, atualizar e excluir conjuntos de teste na janela do conjunto de teste. Ou você pode usar a lista de conjuntos de teste disponíveis para editar ou anotar manualmente seu arquivo

de conjunto de teste. Em seguida, faça o upload mais uma vez para tentar novamente a validação, devido a erros ou outros problemas de entrada.

Para baixar o arquivo do conjunto de teste do registro do conjunto de testes:

1. Selecione o nome do conjunto de teste na lista de conjuntos de teste.
2. Na janela de registro do conjunto de teste, selecione o botão Download no lado direito da tela na seção Entradas de Teste.
3. Se houver algum detalhe do erro de validação na parte superior da janela em relação ao conjunto de teste, selecione o botão Download. O arquivo será salvo na sua pasta Downloads. Você pode corrigir os erros de validação no conjunto de teste a partir das mensagens de erro no arquivo CSV do conjunto de teste. Encontre o erro identificado na etapa de validação, corrija a linha ou remova-a e faça o upload do arquivo para repetir a etapa de validação.
4. Se você baixar o conjunto de teste com sucesso, uma mensagem em banner verde aparecerá.

Para baixar um conjunto de teste da lista de conjuntos de teste:

1. Na lista de conjuntos de teste, selecione o botão de opção ao lado do item do conjunto de teste que deseja baixar.
2. No menu Ação no canto superior direito, selecione Download.
3. Se você tiver baixado o conjunto de teste com sucesso, uma mensagem em banner verde aparecerá. O arquivo será salvo na sua pasta Downloads.

Visualizar erros de validação de teste

Você pode corrigir conjuntos de teste que relatam erros de validação. Esses erros de validação são gerados quando um conjunto de teste não está pronto para ser testado. O Test Workbench pode mostrar quais colunas obrigatórias no arquivo CSV de entrada do conjunto de teste não tinham um valor no formato esperado.

Para visualizar erros de validação de teste:

1. Na lista de conjuntos de teste, selecione o nome do conjunto de teste que relata um status de Erro de Validação que você deseja visualizar. Os nomes dos conjuntos de teste são links ativos que levam você aos detalhes sobre o conjunto de teste.
2. O registro do conjunto de teste exibe detalhes do erro de validação na parte superior da tela. Escolha Exibir detalhes para ver o relatório sobre erros de validação.

3. Na janela do relatório de erros, revise o número da linha e o tipo de erro para ver onde o erro ocorre. Para obter uma lista extensa de erros, você pode optar por fazer o Download do relatório de erros.
4. Compare os erros listados no arquivo CSV de entrada do conjunto de teste com o arquivo de teste original para corrigir quaisquer problemas e fazer o upload do conjunto de teste novamente.

A seguinte tabela lista as mensagens de erro de validação do CSV de entrada com cenários.

Cenário	Mensagem de erro	Observações
O tamanho do arquivo do conjunto de teste excede	O tamanho do arquivo do conjunto de teste é maior que 200 MB. Forneça um arquivo menor e tente fazer sua solicitação novamente.	
O conjunto de teste excede o máximo de registros	O arquivo de entrada tinha registros acima do número máximo permitido de 200.000.	
Carregar conjunto de teste vazio	O conjunto de teste importado está vazio. Forneça um conjunto de teste não vazio e tente fazer sua solicitação novamente.	
Nome do cabeçalho da coluna vazia	Linha de cabeçalhos de coluna: encontrou o nome da coluna vazia na coluna número 5.	
Nome do cabeçalho da coluna não reconhecido	Linha de cabeçalhos de coluna: não foi possível reconhecer o nome da coluna “fictícia” na coluna número 2.	

Cenário	Mensagem de erro	Observações
Nome do cabeçalho da coluna duplicado	Linha de cabeçalhos de coluna: encontrou várias colunas “link de áudio S3" e “link de áudio S3" que são iguais ou equivalentes. Remova ou renomeie uma dessas colunas.	
O nome da coluna de vários valores excedeu o limite	Linha de cabeçalhos de coluna: a contagem de colunas para o “Slot de Saída Esperado” excedeu a contagem máxima permitida: 6. Remova algumas colunas do “Slot de Saída Esperado” e tente novamente.	O número máximo de colunas permitidas pela coluna de valores múltiplos é 6.
O cabeçalho da coluna relacionado a texto ou áudio não está presente	Não foi possível encontrar colunas para conversas de texto ou áudio. Para conversas de texto, use as colunas {'Text input'}. Para conversas em áudio, use as colunas {'S3 audio link', 'Expected transcription'}.	Colunas obrigatórias de áudio: {S3 audio link', 'Expected transcription'} Colunas obrigatórias de texto: {'Text input'}
Existem cabeçalhos de coluna relacionados a texto e áudio	Colunas encontradas para conversas de texto e áudio. Você pode usar colunas {'Entrada de texto'} para conversas de texto ou colunas {'S3 link de áudio', 'Transcri ção esperada'} para conversas de áudio.	Colunas obrigatórias de áudio: {S3 audio link', 'Expected transcription'} Colunas obrigatórias de texto: {'Text input'}

Cenário	Mensagem de erro	Observações
A coluna obrigatória está ausente	Não foi possível encontrar as colunas obrigatórias ["Intenção de Saída Esperada"].	Colunas obrigatórias: {"Line #", "Source", "Expected Output Intent"}
Encontrou um dado na coluna sem cabeçalho	Foram encontrados dados na coluna número 8 para a linha número 6, mas a coluna correspondente não tinha um cabeçalho de coluna.	
Dados não encontrados para colunas obrigatórias	Linha=12: nenhum valor encontrado para as colunas obrigatórias: {"Source", “Expected Output Intent"}	
Identificação de conversa duplicada encontrada	o número da conversa '19' foi visto em uma conversa anterior na linha número 39.” Certifique-se de que o mesmo número de conversa não tenha sido fornecido para duas conversas. Você pode fazer isso garantindo que todas as linhas de um número de conversa estejam agrupadas.	
ID de conversa inválido fornecido	Foi encontrado o valor inválido 'test_conversation' na coluna 'Nº da conversa'. O valor dessa coluna deve ser numérico ou N/A (ou seja, Não aplicável) para uma linha de usuário.	

Cenário	Mensagem de erro	Observações
Valor não numérico fornecido para o número da linha	Foi encontrado o valor não numérico “test_line” na coluna “Nº da linha”. Seu valor deve ser numérico.	
ID da conversa não encontrada na linha do agente	Nenhum valor encontrado para a coluna “Nº da conversa”. Ele deve ser fornecido para uma linha de agentes.	
ID da conversa não encontrada na linha do agente	Foi encontrado o valor não numérico “test_conversation” na coluna “Nº da conversa”. Seu valor deve ser numérico para uma linha de agente.	
Local do S3 inválido	O valor inválido “bucket/folder” foi fornecido. O formato válido é S3://<bucketName>/<keyName>.	
Nome de bucket do S3 inválido	O nome inválido do bucket s3 “test_bucket” foi fornecido. Verifique o nome do bucket.	
A localização do áudio S3 é uma pasta	A localização de áudio fornecida “S3: //bucket/folder” é inválida. Ela aponta para uma pasta S3.	
Nome de intenção inválido	Caracteres inválidos estavam presentes na intenção “intent@name”. Verifique o nome da intenção.	Verificação de regex: ^([0-9a-zA-Z][_-]?)+$

Cenário	Mensagem de erro	Observações
Nome de slot inválido	Caracteres inválidos estavam presentes no slot “Slot@Name”. Verifique o nome do slot.	A Regex: ^([0-9a-zA-Z][_-]?)+$It não pode começar nem terminar com ponto (.)
Valor do slot fornecido para o slot principal	Os valores de slot foram fornecidos para o subslot “Address.City”, bem como para o slot principal “Address”. Os valores devem ser fornecidos somente para o subslot.	O slot principal no CST não deve ter valor de slot
Caractere inválido no nome do contexto	Caracteres inválidos estavam presentes no nome de contexto “context@1”. Verifique o nome do contexto.	Regex: ^([A-Za-z]_?)+$
Estilo de ortografia de slot inválido	O valor inválido “test” foi fornecido. Certifique-se de que estejam todas em maiúsculas. Os valores válidos são ["Default”, "SpellByLetter”, "SpellByWord"].	Valores suportados ["Padrão”, "SpellByCarta”, "SpellByPalavra”
O participante ou a fonte deve ser agente ou usuário	O valor inválido “bot” foi fornecido. Os valores válidos são ["Agent”, “User"].	Enums permitidos: “Agente”, “Usuário”
O número da linha não deve ser decimal	O valor inválido “10.1” foi fornecido. Deve ser um número válido sem nenhuma fração.	

Cenário	Mensagem de erro	Observações
O número da conversa não pode ser decimal	O valor inválido “10.1” foi fornecido. Deve ser um número válido sem nenhuma fração.	
O número da linha deve estar dentro do intervalo	O valor inválido “92233720 368547758071” foi fornecido . Ele deve ser maior que ou igual a 1 e menor que ou igual a 9223372036854775807.	
A coluna embutida aceita apenas o valor booleano	O valor inválido “test” foi fornecido. Deve ser um valor booleano válido, como “true” ou “false”. Alternativamente, “sim” e “não” podem ser usados.	Valores possíveis: "True", "true", "T", "Yes", "yEs", "Y", "1", "1.0", "False", "false", "F", "No", "no", "N", "0", "0.0"
O slot esperado, o atributo da sessão e o atributo da solicitação devem ser separados por igual a (=)	O valor “slotName:slotValue” não tem “=”. Esse valor deve ser fornecido como um par de valores-chave no formato "<key>=<value>”.	Por exemplo: slotName = slotType
O slot esperado, o atributo da sessão e o atributo da solicitação devem ter um par de valores-chave	“=slotValue” não tem uma chave antes de “=”. Esse valor deve ser fornecido como um par de valores-chave no formato "<key>=<value>”.	Por exemplo: slotName = slotType
Aspa inválida no final	Foram encontradas aspas incorretas em “Lenny's Burger'”. Começa com o caractere de aspa “'”, mas não termina com o mesmo caractere de aspa.	Por exemplo: ““Lenny's Burger”, KFC”

Cenário	Mensagem de erro	Observações
Aspa inválida no meio	Foram encontradas aspas incorretas em “"Lenny's" Burger, KFC”. Ele contém o caractere de aspa “"”dentro de seu conteúdo. Os valores que contêm aspas simples devem ser colocados entre aspas duplas e vice-versa.	Por exemplo: ““Lenny's Burger”, KFC”
Aspas obrigatórias	“chave = Lenny's Burger” contém aspas simples ou duplas, mas não foi colocado entre aspas. Os valores que contêm aspas simples devem ser colocados dentro de aspas duplas e vice-versa.	
Chave duplicada repetida na coluna	A chave “key1” foi repetida em duas colunas: “Atributo de sessão 3” e “Atributo de sessão 1”.	
Formato inválido na dica de runtime	Chave inválida `BookFlig ht.Car. “`fornecido para Runtime Hints. Para dicas de Runtime, a chave deve estar no formato <intentName>. <slotName>.	Se '.' precisar estar presente no meio da chave, o nome da intenção e o nome do slot não poderão ser extraídos dessa chave. Exemplos dessa formatação incorreta : "“BookFlight,”. BookFligh t.Carro”, “BookFlight.Carro”.
Nome de intenção inválido na chave de dica de runtime	Foi encontrada uma intenção inválida “intent@name” para dicas de runtime. Verifique o nome da intenção.	Verificação de regex: ^([0-9a-z A-Z][_-]?)+$

Cenário	Mensagem de erro	Observações
Nome de slot inválido na chave de dica de runtime	Foi encontrado um nome de slot inválido em “Slot@Nam e” para dicas de runtime. Verifique o nome do slot.	A Regex: ^([0-9a-zA-Z][_-]?)+$It não pode começar nem terminar com ponto (.)

Excluir um conjunto de teste

Você pode excluir facilmente um conjunto de teste da sua lista de conjuntos de teste.

Para excluir um conjunto de teste:

1. Vá até a lista de conjuntos de teste no menu do lado esquerdo para ver a lista de conjuntos de teste.
2. Na lista de conjuntos de teste, selecione o conjunto de teste que deseja excluir.
3. Vá para o menu suspenso Ações no canto superior direito e escolha Excluir.
4. Uma mensagem confirma que o conjunto de teste foi excluído.

Editar detalhes do conjunto de teste

Você pode editar o nome e os detalhes de um conjunto de teste na lista de conjuntos de teste. O nome ou os detalhes podem ser adicionados ou atualizados posteriormente. No entanto, você precisará atualizar seu conjunto de teste antes de executar o teste com seu bot ou dados de transcrição.

Para editar detalhes do conjunto de teste:

1. Vá até a lista de conjuntos de teste no menu do lado esquerdo para ver a lista de conjuntos de teste.
2. Na lista de conjuntos de teste, marque a caixa de seleção do conjunto de teste que deseja excluir.
3. Vá para o menu suspenso Ações no canto superior direito e escolha Editar detalhes.
4. Uma mensagem confirma que o conjunto de teste foi editado com sucesso.

Atualizar conjunto de teste

Você pode atualizar, corrigir, modificar ou excluir itens do conjunto de testes para otimizar os resultados da linha de base ou para corrigir outros erros que possam ter ocorrido no conjunto de teste

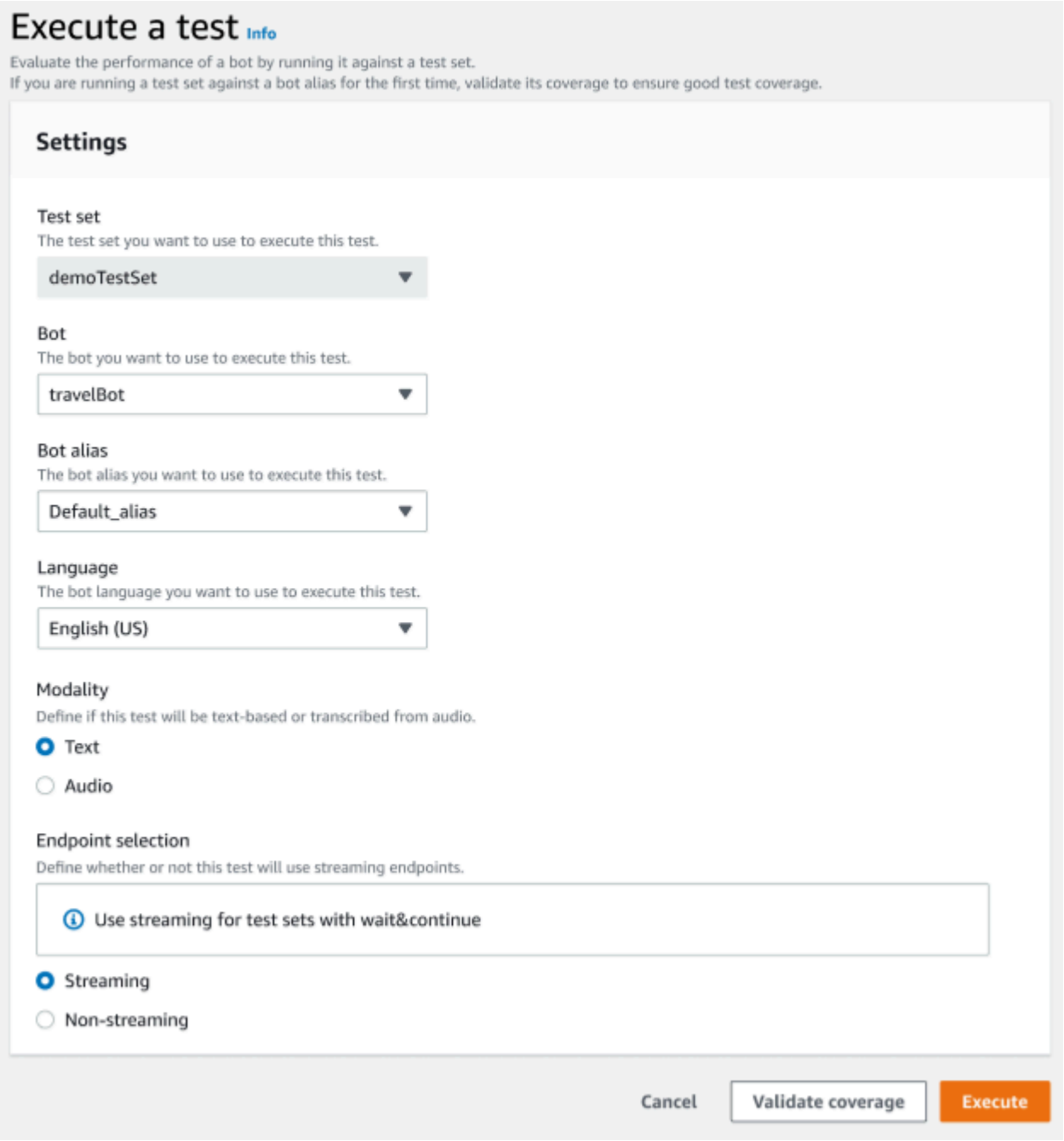
Você pode baixar um conjunto de teste e corrigir os erros de validação antes de fazer o upload do conjunto de teste corrigido. Consulte Visualizar erros de validação de teste.

Como atualizar um conjunto de teste:

1. No registro do conjunto de teste, escolha o botão Atualizar conjunto de teste no canto superior direito.
2. Escolha um arquivo para fazer upload a partir da sua conta do Amazon S3 ou faça upload de um arquivo de teste CSV do seu computador. OBSERVAÇÃO: a atualização de um conjunto de teste substituirá os dados existentes.
3. Selecione o botão Atualizar.
4. Uma mensagem confirma que o conjunto de teste foi atualizado com sucesso. OBSERVAÇÃO: essa operação pode levar alguns minutos, dependendo da complexidade e do tamanho do conjunto de teste.
5. Uma mensagem confirma que o conjunto de teste foi atualizado com êxito e o Status exibe Pronto para teste.

Executar um teste

Para executar um conjunto de teste, você deve escolher o bot apropriado para executar o teste no conjunto de teste. Você pode escolher um bot da sua AWS conta no menu suspenso em Conjunto de testes. Essa operação testará o bot selecionado em relação aos dados de teste validados para relatar métricas de desempenho em relação aos dados básicos do conjunto de teste.



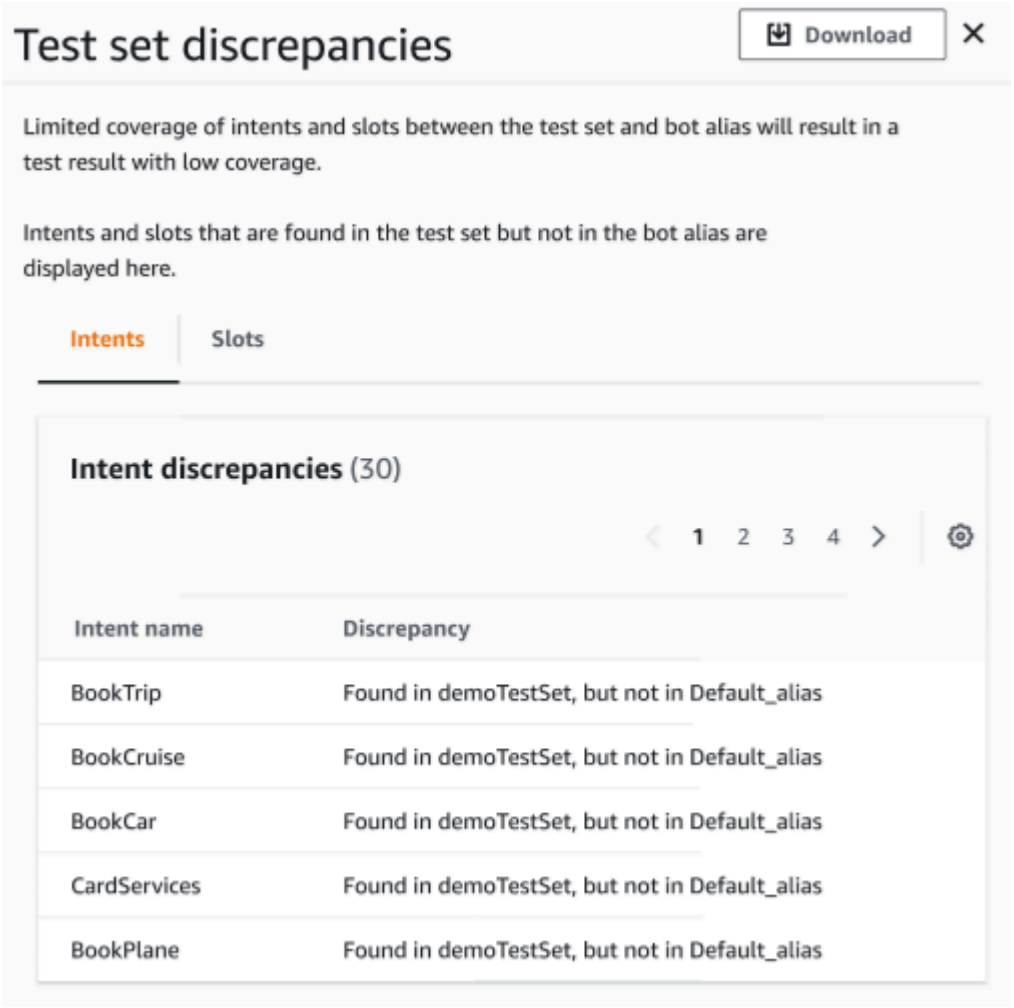
Para executar um teste no Test Workbench

1. Na página de registro do conjunto de teste, selecione Executar teste.
2. Selecione o conjunto de teste que você deseja usar no teste.
3. Selecione o nome do bot a ser usado no teste no menu suspenso Bot.
4. Escolha um alias de bot, se aplicável, no menu suspenso Alias de bot.
5. Na seleção de idiomas, escolha uma versão em inglês.

6. Selecione Texto ou Áudio para o tipo de modalidade.
7. Escolha sua localização do Amazon S3. (somente áudio)
8. Selecione sua Seleção de endpoint para seu bot. (somente streaming)
9. Selecione o botão Validar cobertura para confirmar que seu teste está pronto para ser executado. Se houver algum erro presente na etapa de validação, revise os parâmetros anteriores e faça as correções.
10. Selecione Executar para executar o teste.
11. Uma mensagem confirma que o conjunto de teste foi executado com sucesso.

Cobertura do conjunto de teste

A cobertura limitada de intenções e slots entre o conjunto de teste e o bot pode resultar em medidas de desempenho esperadas. Recomendamos que você revise a cobertura do conjunto de teste antes de executar o teste.



Como revisar a cobertura de validação

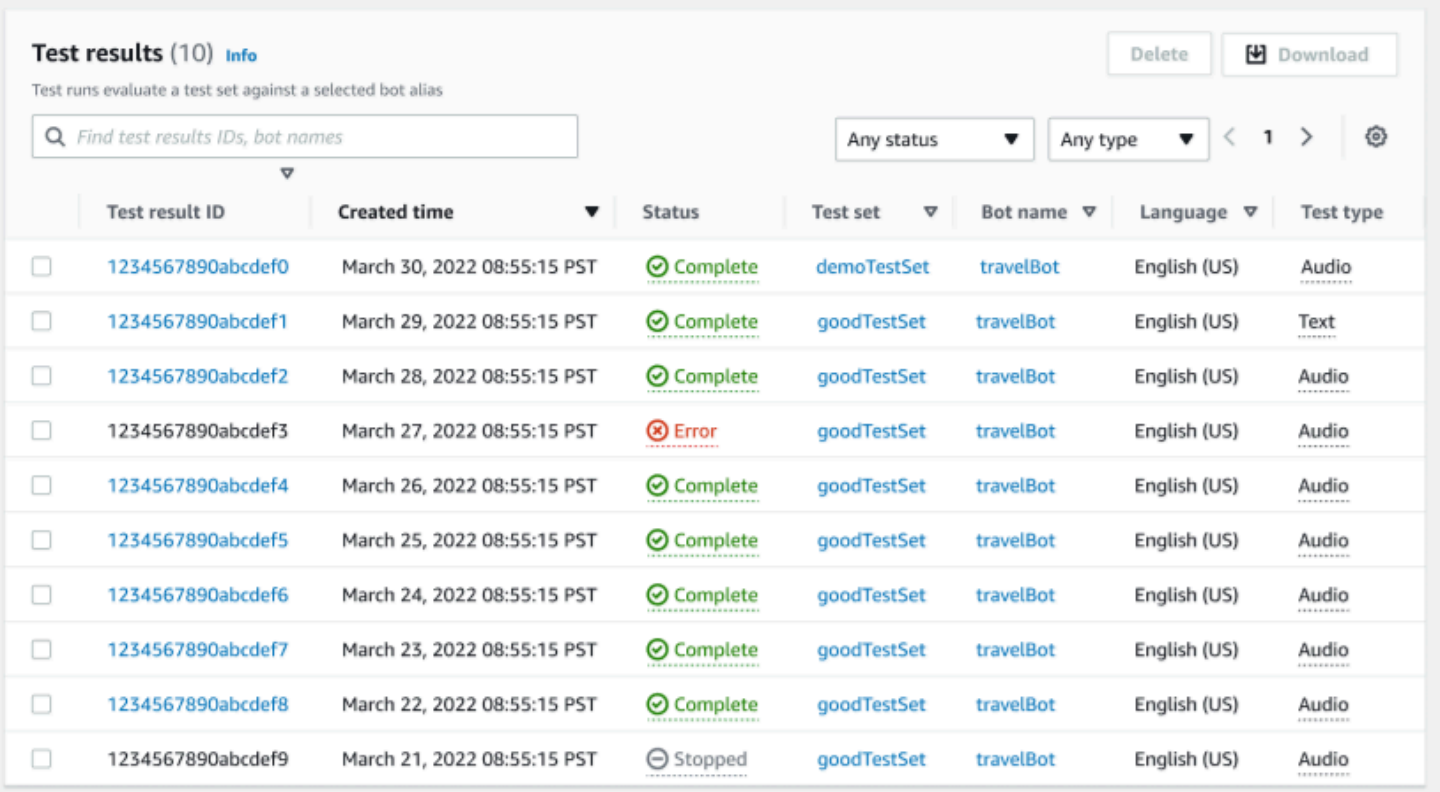
1. Nos registros do conjunto de teste, selecione o botão Validar cobertura.
2. A mensagem indica que está validando a cobertura entre o conjunto de teste e o bot selecionado.
3. Quando a operação for concluída, a mensagem indica Validação da cobertura bem-sucedida.
4. Selecione o botão Exibir detalhes na parte inferior da janela.
5. Veja as discrepâncias do conjunto de teste para intenções e slots escolhendo a guia para cada um. Você pode baixar esses dados em formato CSV escolhendo o botão Download.
6. Analise os resultados da validação dos dados, intenções do bot e slots do seu conjunto de teste. Identifique problemas e faça alterações na arquitetura do conjunto de teste do seu bot para melhorar os resultados. Faça o upload do conjunto de teste editado e do bot para executar o teste depois de fazer alterações no arquivo CSV. NOTA: a cobertura de validação é executada no conjunto de teste e não no bot. As intenções que estiverem no bot, mas não presentes no conjunto de testes, não serão abordadas.

Visualizar resultados do teste

Interprete os resultados do teste do Test Workbench para determinar onde a conversa entre seu bot e o cliente pode estar falhando ou exigindo que o cliente faça várias tentativas para atender a intenção.

Ao localizar esses problemas nos resultados do teste, você pode otimizar o desempenho do seu bot melhorando o desempenho da intenção usando diferentes dados de treinamento ou declarações que são mais consistentes com os valores de transcrição do bot em tempo real.

Você pode obter uma visão detalhada das intenções e dos slots que apresentavam discrepâncias de desempenho. Depois de identificar intenções ou espaços que apresentam discrepâncias, você pode detalhar e revisar as declarações e o fluxo da conversa.



Para revisar os resultados dos testes:

1. Vá para a lista de conjuntos de teste no menu do lado esquerdo para selecionar a opção Resultados do teste no Test Workbench. NOTA: os resultados do teste indicam um status de concluído se tiverem sido bem-sucedidos.
2. Selecione o ID do resultado do teste para os resultados do teste que você deseja revisar.

Detalhes dos resultados do teste

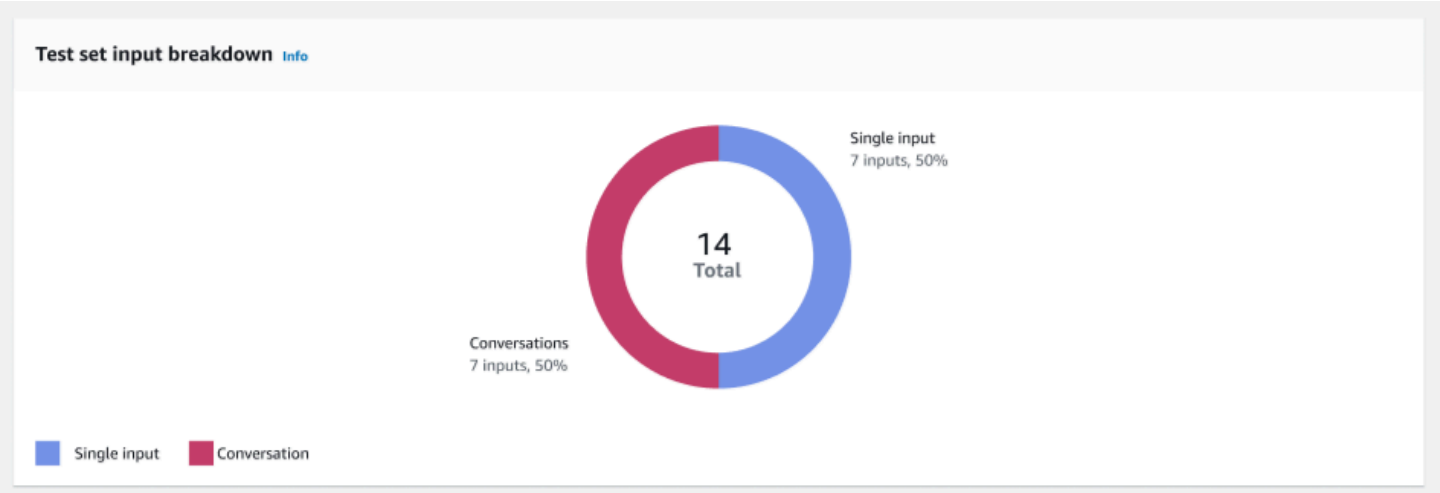
Os resultados do teste mostram os detalhes do conjunto de testes, as intenções usadas e os slots usados. Ele também fornece o detalhamento geral das entradas do conjunto de teste, incluindo os resultados gerais, os resultados da conversa, a intenção e os resultados do slot.

Os resultados do teste incluem todas as informações relacionadas ao teste, como:

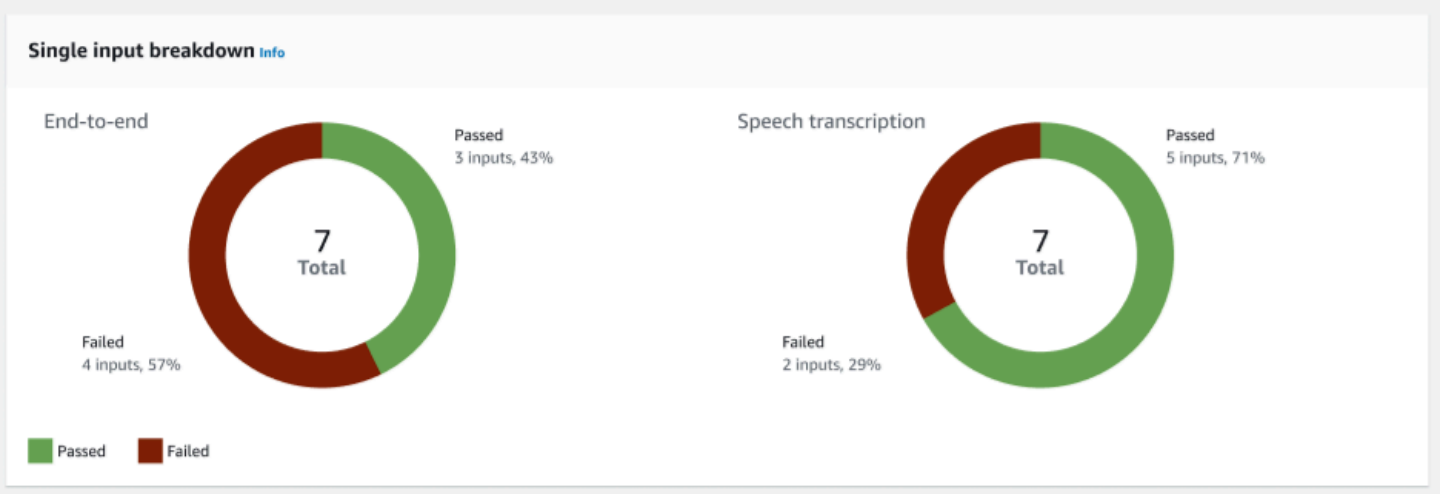
- Metadados de detalhes do teste
- Resultados gerais
- Resultados da conversa
- Intenção e resultados do slot

- Resultados detalhados

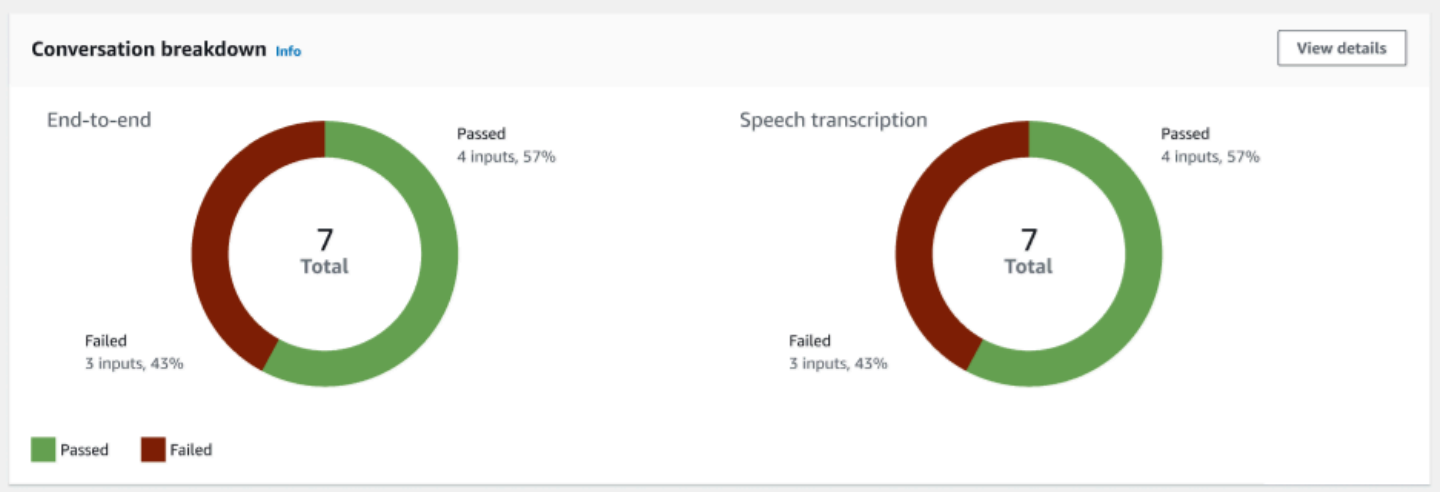
Guia de resultados gerais:



Detalhamento da entrada do conjunto de teste – Este gráfico mostra o detalhamento do número de conversas e enunciados de entrada única no conjunto de teste.

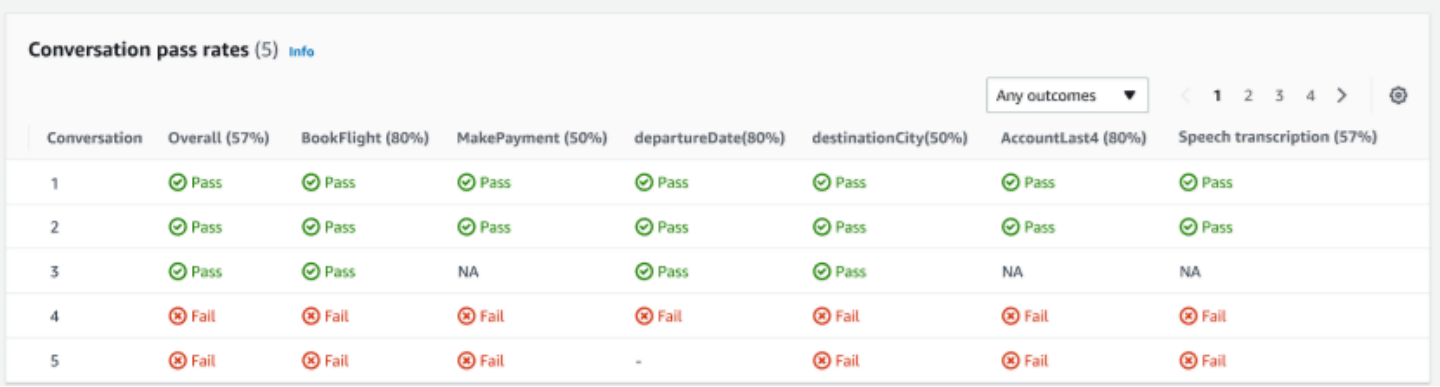


Detalhamento de entrada única — Exibe dois gráficos que incluem end-to-end conversas e transcrições de fala. O número de entradas aprovadas e reprovadas é indicado em cada gráfico. Observação: o gráfico de transcrição de fala estará visível somente para o conjunto de teste de áudio.



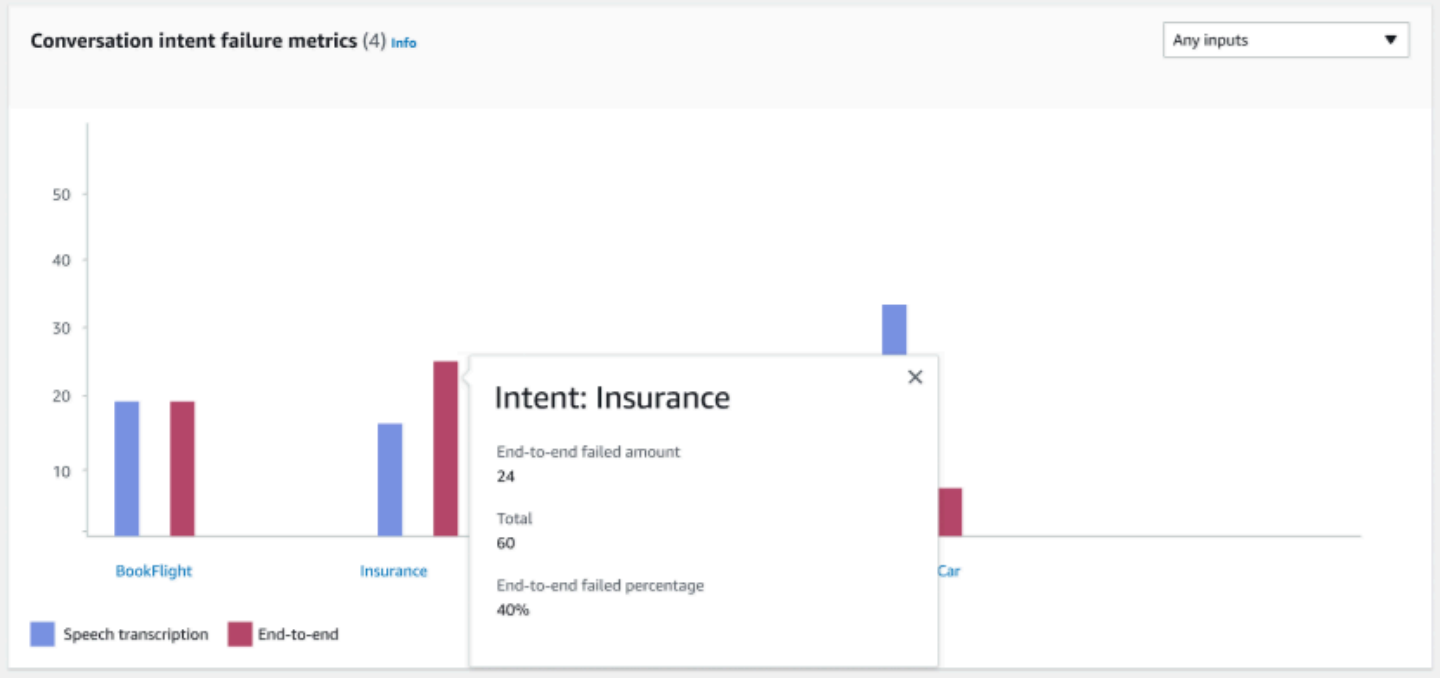
Detalhamento da conversa — Exibe dois gráficos que incluem end-to-end conversas e transcrições de discursos. O número de entradas aprovadas e reprovadas é indicado em cada gráfico. Observação: o gráfico de transcrição de fala estará visível somente para o conjunto de teste de áudio.

Guia de resultados da conversa:



Conversation	Overall (57%)	BookFlight (80%)	MakePayment (50%)	departureDate(80%)	destinationCity(50%)	AccountLast4 (80%)	Speech transcription (57%)
1	Pass	Pass	Pass	Pass	Pass	Pass	Pass
2	Pass	Pass	Pass	Pass	Pass	Pass	Pass
3	Pass	Pass	NA	Pass	Pass	NA	NA
4	Fail	Fail	Fail	Fail	Fail	Fail	Fail
5	Fail	Fail	Fail	-	Fail	Fail	Fail

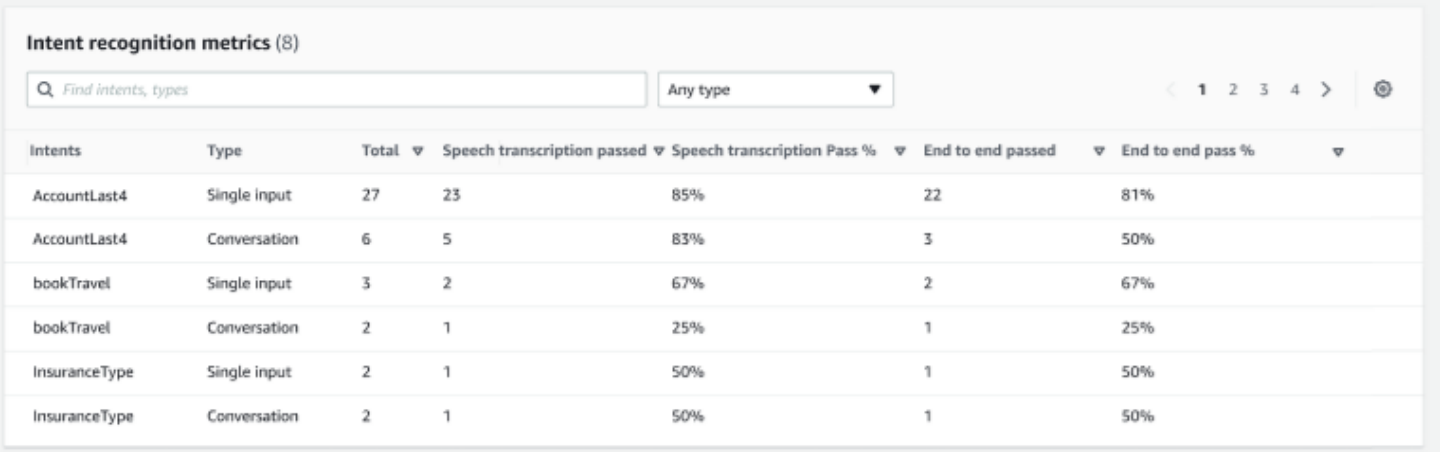
Taxas de aprovação da conversa – A tabela de taxas de aprovação da conversa é usada para ver quais intenções e slots são usados em cada conversa no conjunto de teste. Você pode visualizar onde a conversa falhou analisando qual intenção ou slot falhou, junto com a porcentagem de aprovação de cada intenção e slot.



Métricas de falha de intenção de conversa – Essa métrica mostra as 5 intenções com pior desempenho no conjunto de teste. Esse painel mostra um gráfico de qual porcentagem ou número de intenções foram bem-sucedidas ou falhas com base nos logs de conversas ou na transcrição do bot. Uma intenção bem-sucedida não significa que toda a conversa foi bem-sucedida. Essas métricas aplicam-se apenas ao valor das intenções, independentemente de qual intenção veio antes ou depois.

Métricas de falha de slot de conversa – Essa métrica mostra as 5 slots com pior desempenho no conjunto de teste. Indicou a taxa de sucesso de cada slot na intenção. O gráfico de barras mostra a transcrição da fala e end-to-end as conversas para cada espaço na intenção.

Guia de resultados de intenção e slot:

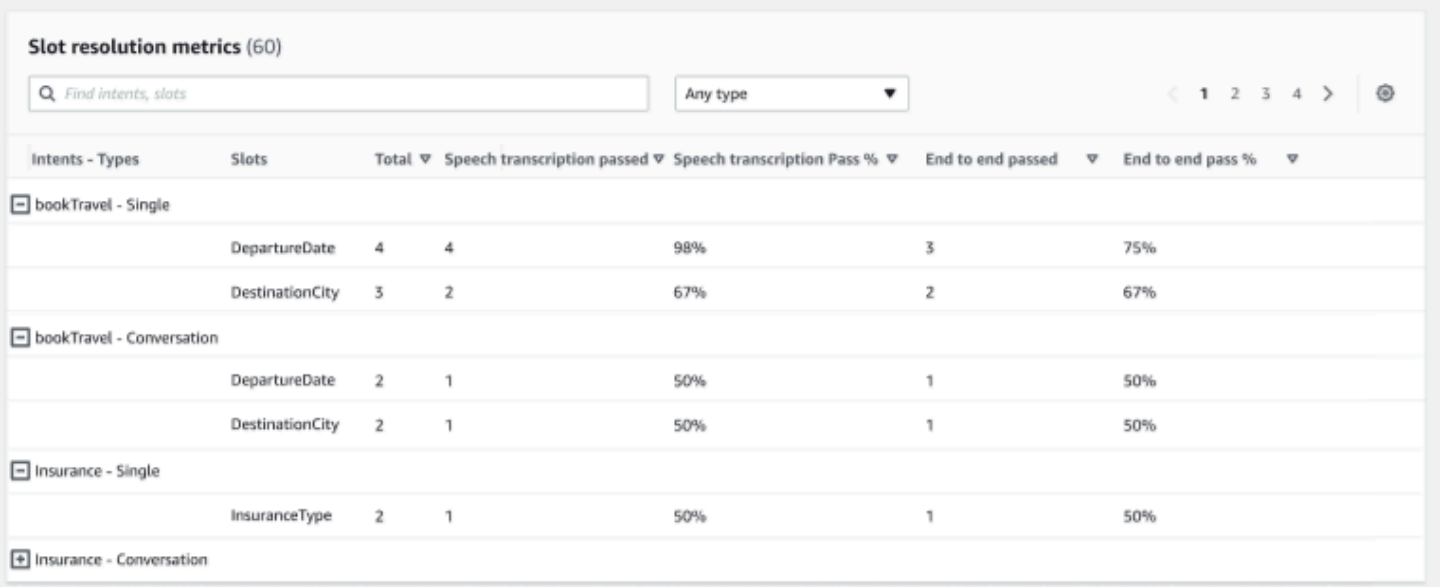


Intent recognition metrics (8)

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%

Métricas de reconhecimento de intenção – Mostra uma tabela de quantas intenções foram reconhecidas com sucesso. Exibe a taxa de aprovação da transcrição da fala e end-to-end das conversas.



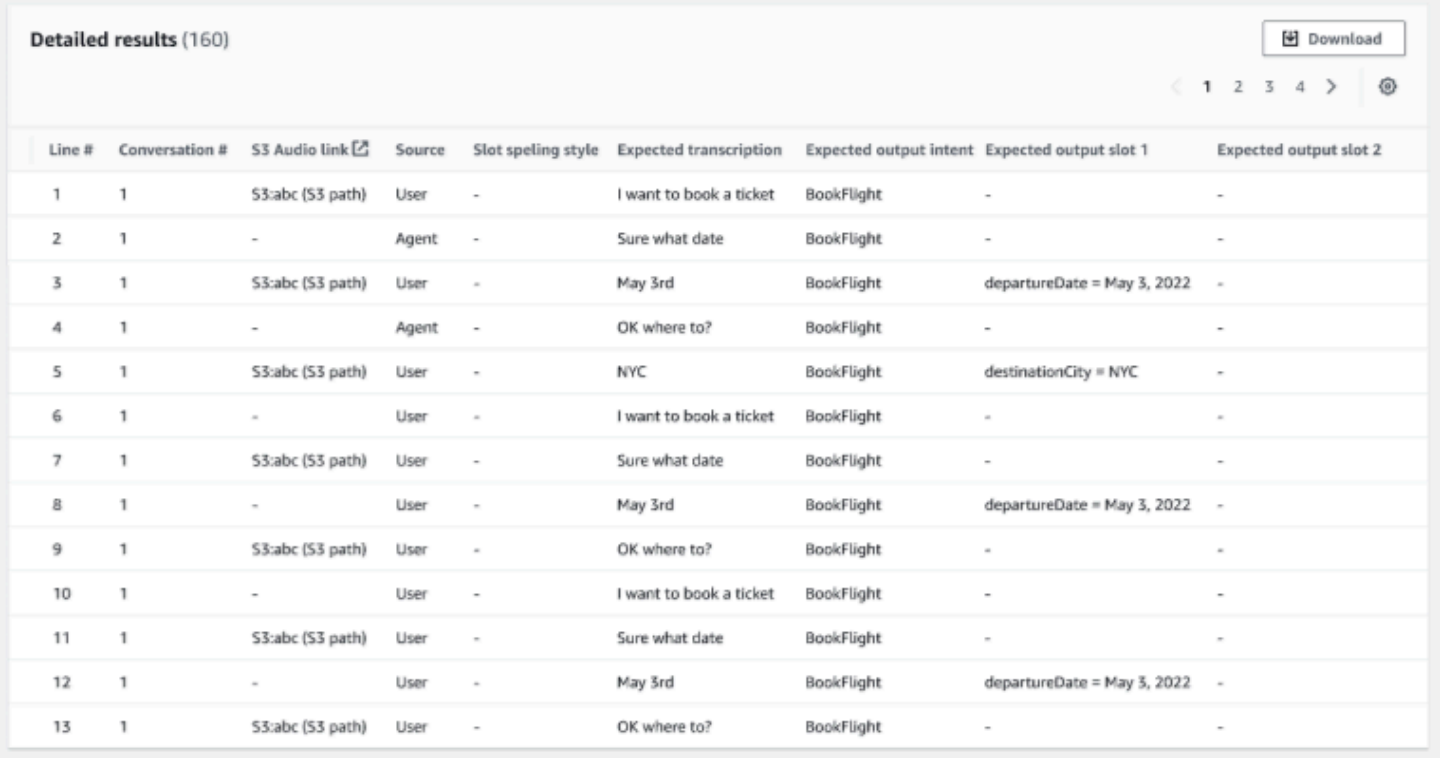
Slot resolution metrics (60)

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						

Métricas de resolução de slots – Mostra as intenções e os slots separadamente e a taxa de sucesso e falha de cada slot para cada intenção usada na conversa ou em uma única entrada. Exibe a taxa de aprovação da transcrição da fala e end-to-end das conversas.

Guia de resultados detalhados:



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	-	-

Resultados detalhados – Mostra uma tabela detalhada no log de conversas com os enunciados do usuário e do agente e a saída esperada e a transcrição esperadas para cada slot. Você pode baixar esse relatório selecionando o botão Download.

A seguinte tabela lista as mensagens de erro de falha no resultado com cenários.

Cenário	Mensagem de erro	Ação
Incompatibilidade de intenções	BookFlight Intenção esperada, mas era BookHotel intenção.	Ignorar outros turnos na conversa
Incompatibilidade de elicitação de slots	Esperava-se que o slot departureDate fosse obtido, mas era cabinType.	Ignorar outros turnos na conversa
Incompatibilidade do valor do slot	Incompatibilidade entre o valor esperado e o real do slot.	Continuar com outros turnos nas conversas
Falta ack-to-back o prompt do agente B	Esperava que o bot retornass e uma solicitação do agente	Ignorar outros turnos na conversa

Cenário	Mensagem de erro	Ação
	neste turno, mas ela não foi recebida.	
Incompatibilidade de transcrição	A transcrição esperada não correspondeu à transcrição real.	Continuar com outros turnos nas conversas
Slot opcional não elicitado	Esperava que fosse elicitar o slot cabinType no próximo turno, mas a intenção atual foi atendida antes disso.	Ignorar outros turnos na conversa
Slot não reconhecido	O slot departureDate esperado não foi reconhecido neste turno.	Ignorar outros turnos na conversa
Solicitação adicional back-to-back do agente	Esperava o turno de um usuário, mas era um prompt do agente	Ignorar outros turnos na conversa

Transmissão para um bot do Amazon Lex V2

Você pode usar a API de transmissão do Amazon Lex V2 para iniciar uma transmissão bidirecional entre um bot do Amazon Lex V2 e sua aplicação. Iniciar uma transmissão permite que o bot gerencie a conversa entre o bot e o usuário. O bot responde à entrada do usuário sem que você grave um código para lidar com as respostas do usuário. O bot pode:

- Lidar com as interrupções do usuário enquanto ele está reproduzindo uma mensagem. Para mais informações, consulte Permitir que seu bot seja interrompido pelo usuário.
- Aguardar até que o usuário forneça informações. Por exemplo, o bot pode esperar que o usuário colete as informações do cartão de crédito. Para mais informações, consulte Permitir que o bot espere que o usuário forneça mais informações.
- Usar a entrada de áudio e frequência múltipla de dois tons (DTMF) na mesma transmissão.
- Lidar com as pausas na entrada do usuário melhor do que se estivesse gerenciando a conversa a partir da sua aplicação.

O bot do Amazon Lex V2 não apenas responde aos dados enviados da sua aplicação, mas também envia informações sobre o estado da conversa à sua aplicação. Você pode usar essas informações para alterar a forma como sua aplicação responde aos clientes.

O bot do Amazon Lex V2 também monitora a conexão entre o bot e sua aplicação. Ele pode determinar se a conexão atingiu o tempo limite.

Para usar a API para iniciar uma transmissão para um bot do Amazon Lex V2, consulte Iniciar uma transmissão para um bot.

Ao iniciar a transmissão da sua aplicação para um bot do Amazon Lex V2, você pode configurar o bot para aceitar entrada de áudio ou texto do usuário. Você também pode escolher se o usuário receberá áudio ou texto em resposta à entrada.

Se você configurou o bot Amazon Lex V2 para aceitar entrada de áudio do usuário, ele não poderá receber entrada de texto. Se você configurou o bot para aceitar a entrada de texto, o usuário só poderá usar texto escrito para se comunicar com ele.

Quando um bot do Amazon Lex V2 usa uma entrada de streaming de áudio, o bot determina quando um usuário começa a falar e quando ele para de falar. Ele lida com quaisquer pausas ou

interrupções do usuário. Ele também pode receber entrada de frequência múltipla de dois tons (DTMF) e entrada de fala na mesma transmissão. Isso ajuda o usuário a interagir com o bot de forma mais natural. Você pode apresentar mensagens e avisos de boas-vindas aos usuários. Você também pode permitir que os usuários interrompam essas mensagens e avisos.

Quando você inicia uma transmissão bidirecional, o Amazon Lex V2 usa o protocolo HTTP/2. A aplicação e o bot trocam dados em uma única transmissão como uma série de eventos. Um evento pode ser um dos seguintes:

- Entrada de texto, de áudio ou de DTMF do usuário.
- Sinais da aplicação para o bot do Amazon Lex V2. Incluem uma indicação de que a reprodução de áudio de uma mensagem foi concluída, ou que o usuário se desconectou da sessão.

Para ter mais informações sobre eventos do , consulte Iniciar uma transmissão para um bot. Para obter informações sobre como codificar eventos, consulte Codificação de transmissão de evento.

Tópicos

Iniciar uma transmissão para um bot

Você usa a operação StartConversation para iniciar uma transmissão entre o usuário e o bot do Amazon Lex V2 em sua aplicação. A solicitação `POST` da aplicação estabelece uma conexão entre a aplicação e o bot do Amazon Lex V2. Isso permite que sua aplicação e o bot comecem a trocar informações entre si por meio de eventos.

A operação `StartConversation` é compatível somente com os seguintes SDKs:

- AWS SDK para C++
- AWS SDK para Java V2

- AWS SDK para JavaScript v3
- AWS SDK para Ruby V3

O primeiro evento que a aplicação deve enviar para o bot Amazon Lex V2 é um ConfigurationEvent. Esse evento tem informações como o formato do tipo de resposta. A seguir estão os parâmetros que você pode usar em um evento de configuração:

- responseContentType: determina se o bot responde à entrada do usuário com texto ou fala.
- sessionState: Informações relacionadas à sessão de transmissão com o bot, como intenção predeterminada ou estado da caixa de diálogo.
- welcomeMessages: especifica as mensagens de boas-vindas que são reproduzidas para o usuário no início da conversa com um bot. Essas mensagens são reproduzidas antes que o usuário forneça qualquer entrada. Para ativar uma mensagem de boas-vindas, você também deve especificar valores para os parâmetros `sessionState` e `dialogAction`.
- disablePlayback: determina se o bot deve esperar por uma sugestão do cliente antes de começar a ouvir a entrada do chamador. Por padrão, a reprodução é ativada, então o valor desse campo é `false`.
- requestAttributes: fornece informações adicionais para a solicitação.

Para obter informações sobre como especificar valores para os parâmetros anteriores, consulte o tipo de dados ConfigurationEvent da operação StartConversation.

Cada transmissão entre um bot e sua aplicação só pode ter um evento de configuração. Depois que sua aplicação enviar um evento de configuração, o bot poderá receber comunicação adicional da sua aplicação.

Se você especificou que seu usuário está usando áudio para se comunicar com o bot do Amazon Lex V2, sua aplicação pode enviar os seguintes eventos para o bot durante essa conversa:

- AudioInputEvent: contém um bloco de áudio com tamanho máximo de 320 bytes. A aplicação deve usar vários eventos de entrada de áudio para enviar uma mensagem do servidor para o bot. Cada evento de entrada de áudio na transmissão deve ter o mesmo formato de áudio.
- DTMFInputEvent: envia uma entrada de DTMF para o bot. Cada pressionamento de tecla de DTMF corresponde a um único evento.
- PlaybackCompletionEvent: informa ao servidor que uma resposta da entrada do usuário foi reproduzida para ele. Você deve usar um evento de conclusão de reprodução, se estiver enviando

uma resposta de áudio para o usuário. Se `disablePlayback` do seu evento de configuração for `true`, você não poderá usar esse atributo.

- DisconnectionEvent: informa ao bot que o usuário se desconectou da conversa.

Se você especificou que o usuário está usando texto para se comunicar com o bot, sua aplicação pode enviar os seguintes eventos para o bot durante essa conversa:

- TextInputEvent: texto enviado da sua aplicação para o bot. Você pode ter até 512 caracteres em um evento de entrada de texto.
- PlaybackCompletionEvent: informa ao servidor que uma resposta da entrada do usuário foi reproduzida para ele. Você deve usar esse evento se estiver reproduzindo áudio para o usuário. Se `disablePlayback` do seu evento de configuração for `true`, você não poderá usar esse atributo.
- DisconnectionEvent: informa ao bot que o usuário se desconectou da conversa.

Você deve codificar cada evento enviado para um bot do Amazon Lex V2 no formato correto. Para mais informações, consulte Codificação de transmissão de evento.

Cada evento tem um ID de evento. Para ajudar a solucionar quaisquer problemas que possam ocorrer na transmissão, atribua um ID de evento exclusivo a cada evento de entrada. Em seguida, você pode solucionar qualquer falha de processamento com o bot.

O Amazon Lex V2 também usa registros de data e hora para cada evento. Você pode usar esses carimbos de data/hora, além do ID do evento, para ajudar a solucionar qualquer problema de transmissão de rede.

Durante a conversa entre o usuário e o bot do Amazon Lex V2, o bot pode enviar os seguintes eventos de saída em resposta ao usuário:

- IntentResultEvent: contém a intenção que o Amazon Lex V2 determinou usando a declaração do usuário. Cada evento de resultado interno inclui:
  - inputMode: o tipo de declaração do usuário. Os valores válidos são `Speech`, `DTMF` ou `Text`.
  - interpretations: interpretações que o Amazon Lex V2 determina a partir da declaração do usuário.
  - requestAttributes: se você não modificou os atributos da solicitação usando uma função do lambda, esses são os mesmos atributos que foram passados no início da conversa.

  - sessionId: identificador de sessão usado para a conversa.
  - sessionState: o estado da sessão do usuário com o Amazon Lex V2.
- TranscriptEvent: se o usuário fornecer uma entrada para sua aplicação, esse evento conterá a transcrição da declaração do usuário para o bot. Sua aplicação não recebe um `TranscriptEvent` se não houver nenhuma entrada do usuário.

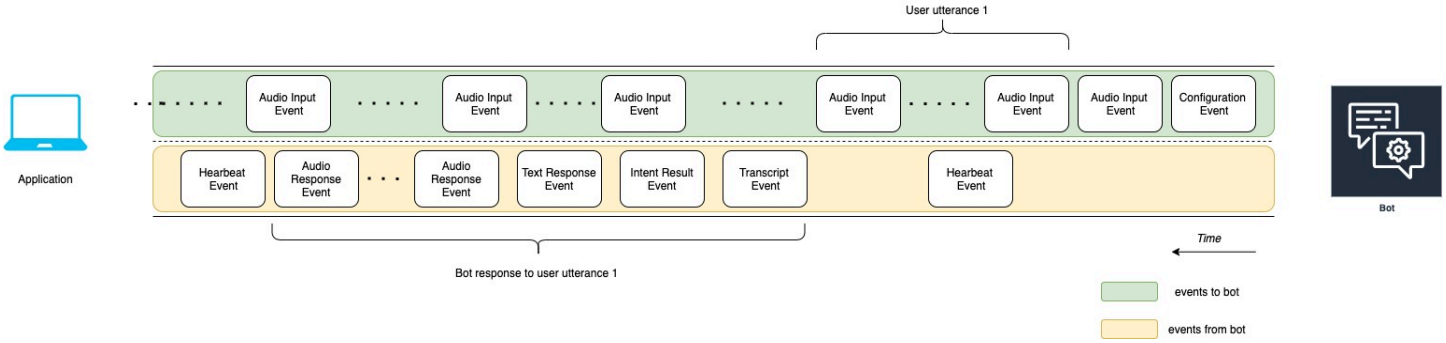
  O valor do evento de transcrição enviado à sua aplicação depende se você especificou áudio (fala e DMTF) ou texto como modo de conversa:
  - Transcrição da entrada de fala: se o usuário estiver falando com o bot, o evento de transcrição será a transcrição do áudio do usuário. É uma transcrição de todo o discurso desde o momento em que o usuário começa a falar até o momento em que termina de falar.
  - Transcrição da entrada DTMF: se o usuário estiver digitando em um teclado, o evento de transcrição conterá todos os dígitos que o usuário pressionou na entrada.
  - Transcrição da entrada de texto: se o usuário estiver fornecendo entrada de texto, o evento de transcrição conterá todo o texto na entrada do usuário.
- TextResponseEvent: contém a resposta do bot em formato de texto. Uma resposta de texto é retornada por padrão. Se você configurou o Amazon Lex V2 para retornar uma resposta de áudio, esse texto será usado para gerar uma resposta de áudio. Cada evento de resposta de texto contém uma matriz de objetos de mensagem que o bot retorna ao usuário.
- AudioResponseEvent: contém a resposta de áudio sintetizada a partir do texto gerado no `TextResponseEvent`. Para receber eventos de resposta de áudio, você deve configurar o Amazon Lex V2 para fornecer uma resposta de áudio. Todos os eventos de resposta de áudio têm o mesmo formato de áudio. Cada evento contém partes de áudio de no máximo 100 bytes. O Amazon Lex V2 envia um trecho de áudio vazio com o campo `bytes` definido como `null` para indicar o final do evento de resposta de áudio para a aplicação.
- PlaybackInterruptionEvent: quando um usuário interrompe uma resposta que o bot enviou à aplicação, o Amazon Lex V2 aciona esse evento para interromper a reprodução da resposta.
- HeartbeatEvent: o Amazon Lex V2 envia esse evento de volta periodicamente para evitar que a conexão entre a aplicação e o bot atinja o tempo limite.

Sequência de tempo de eventos para uma conversa em áudio

Os diagramas a seguir mostram uma conversa de streaming de áudio entre um usuário e um bot do Amazon Lex V2. A aplicação transmite áudio continuamente para o bot, e o bot procura a entrada do usuário a partir do áudio. Neste exemplo, tanto o usuário quanto o bot estão usando a fala para

se comunicar. Cada diagrama corresponde a uma declaração do usuário e à resposta do bot a essa declaração.

O diagrama a seguir mostra o início de uma conversa entre a aplicação e o bot. A transmissão começa no horário zero (t0).

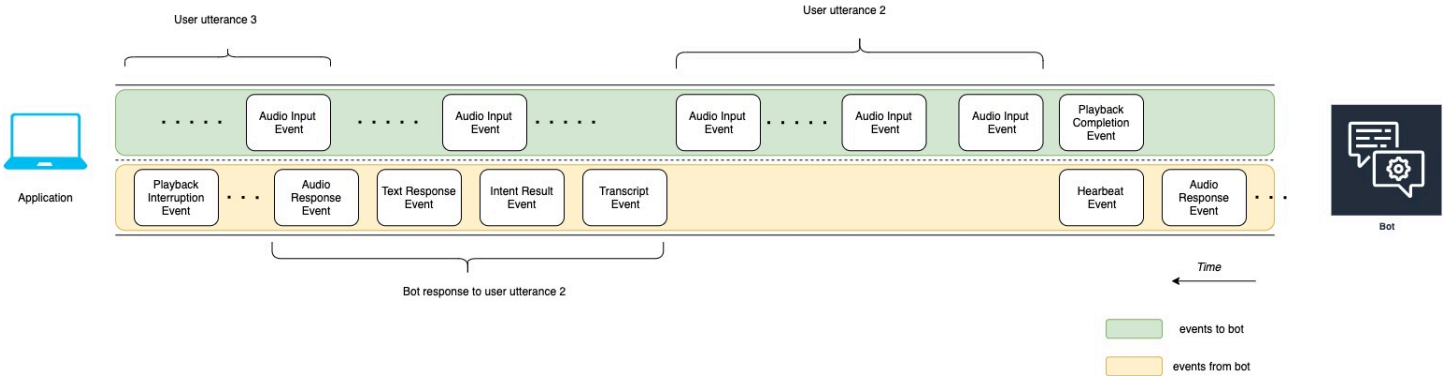


As seções a seguir descrevem os eventos do diagrama anterior.

- t0: a aplicação envia um evento de configuração ao bot para iniciar a transmissão.
- t1: a aplicação transmite dados de áudio. Esses dados são divididos em uma série de eventos de entrada da aplicação.
- t2: para a declaração do usuário 1, o bot detecta um evento de entrada de áudio quando o usuário começa a falar.
- t2: enquanto o usuário fala, o bot envia um evento de heartbeat para manter a conexão. Ele envia esses eventos de forma intermitente para garantir que a conexão não atinja o tempo limite.
- t3: o bot detecta o final da declaração do usuário.
- t4: o bot envia de volta um evento de transcrição que contém uma transcrição da fala do usuário para a aplicação. Este é o início da Resposta do bot à declaração do usuário 1.
- t5: o bot envia um evento de resultado de intenção para indicar a ação que o usuário deseja realizar.
- t6: o bot começa a fornecer sua resposta como texto em um evento de resposta de texto.
- t7: o bot envia uma série de eventos de resposta de áudio à aplicação para serem reproduzidos para o usuário.
- t8: o bot envia outro evento de heartbeat para manter a conexão de forma intermitente.

O diagrama a seguir é uma continuação do diagrama anterior. Ele mostra a aplicação enviando um evento de conclusão da reprodução ao bot para indicar que ele parou de reproduzir a resposta de áudio para o usuário. A aplicação reproduz a Resposta do bot à declaração do usuário 1 para

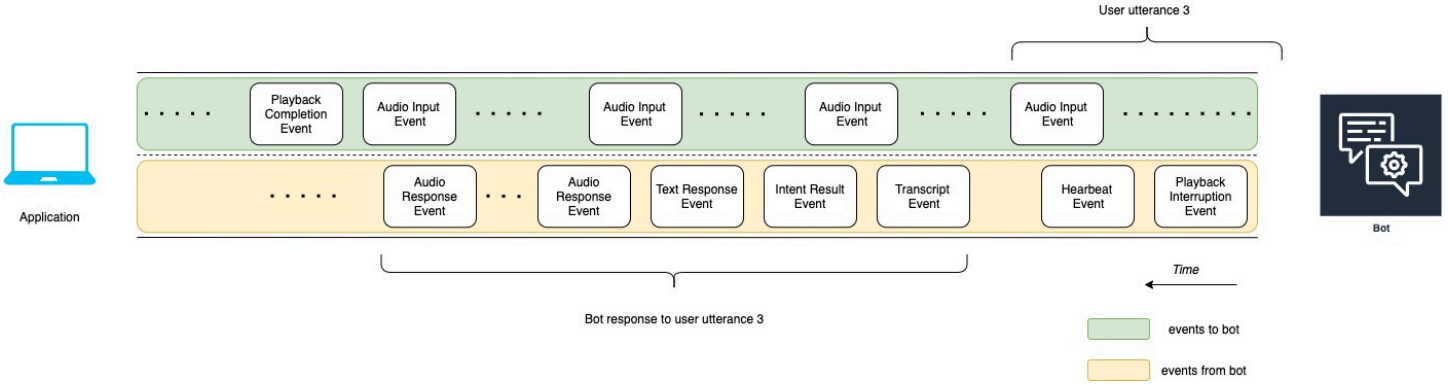
o usuário. O usuário responde à Resposta do bot à declaração do usuário 1 com Declaração do usuário 2.



As seções a seguir descrevem os eventos do diagrama anterior:

- t10: a aplicação envia um evento de conclusão da reprodução para indicar que terminou de reproduzir a mensagem do bot para o usuário.
- t11: a aplicação envia a resposta do usuário de volta ao bot como Declaração do usuário 2.
- t12: para Resposta do bot à declaração do usuário 2, o bot espera que o usuário pare de falar e, em seguida, começa a fornecer uma resposta de áudio.
- t13: enquanto o bot envia a Resposta do bot à declaração do usuário 2 para a aplicação, o bot detecta o início da Declaração do usuário 3. O bot interrompe a Resposta do bot à declaração do usuário 2 e envia um evento de interrupção da reprodução.
- t14: o bot envia um evento de interrupção da reprodução à aplicação para sinalizar que o usuário interrompeu a mensagem.

O diagrama a seguir mostra a Resposta do bot à declaração do usuário 3 e que a conversa continua depois que o bot responde à declaração do usuário.



Usar a API para iniciar uma conversa em streaming

Ao iniciar uma transmissão para um bot do Amazon Lex V2, você realiza as seguintes tarefas:

1. Criar uma conexão inicial com o servidor.
2. Configurar as credenciais de segurança e os detalhes do bot. Os detalhes do bot incluem se o bot usa DTMF e entrada de áudio ou entrada de texto.
3. Enviar eventos para o servidor. Esses eventos são dados de texto ou dados de áudio do usuário.
4. Processar eventos enviados do servidor. Nesta etapa, você determina se a saída do bot é apresentada ao usuário como texto ou fala.

Os exemplos de código a seguir inicializam uma conversa em streaming com um bot do Amazon Lex V2 e sua máquina local. Você pode modificar o código para atender às suas necessidades.

O código a seguir é um exemplo de solicitação usando o AWS SDK for Java para iniciar a conexão com um bot e configurar os detalhes e as credenciais do 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();
        }
    }
}
```

O código a seguir é um exemplo de solicitação usando o AWS SDK for Java para enviar eventos ao bot. O código neste exemplo usa o microfone do seu computador para enviar eventos de áudio.

```
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();
        }
    }
}
```

O código a seguir é um exemplo de solicitação usando o AWS SDK for Java para lidar com as respostas do bot. O código neste exemplo configura o Amazon Lex V2 para reproduzir uma resposta de áudio para você.

```
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;
    }

}
```

Para configurar um bot para responder aos eventos de entrada com áudio, você deve primeiro assinar os eventos de áudio do Amazon Lex V2 e depois configurar o bot para fornecer uma resposta de áudio aos eventos de entrada do usuário.

O código a seguir é um exemplo de AWS SDK for Java para assinatura de eventos de áudio do 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);
            }
        }
    }
}
```

O exemplo de AWS SDK for Java a seguir configura o bot Amazon Lex V2 para fornecer uma resposta de áudio aos eventos de entrada.

```
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();
    }
}
```

Codificação de transmissão de evento

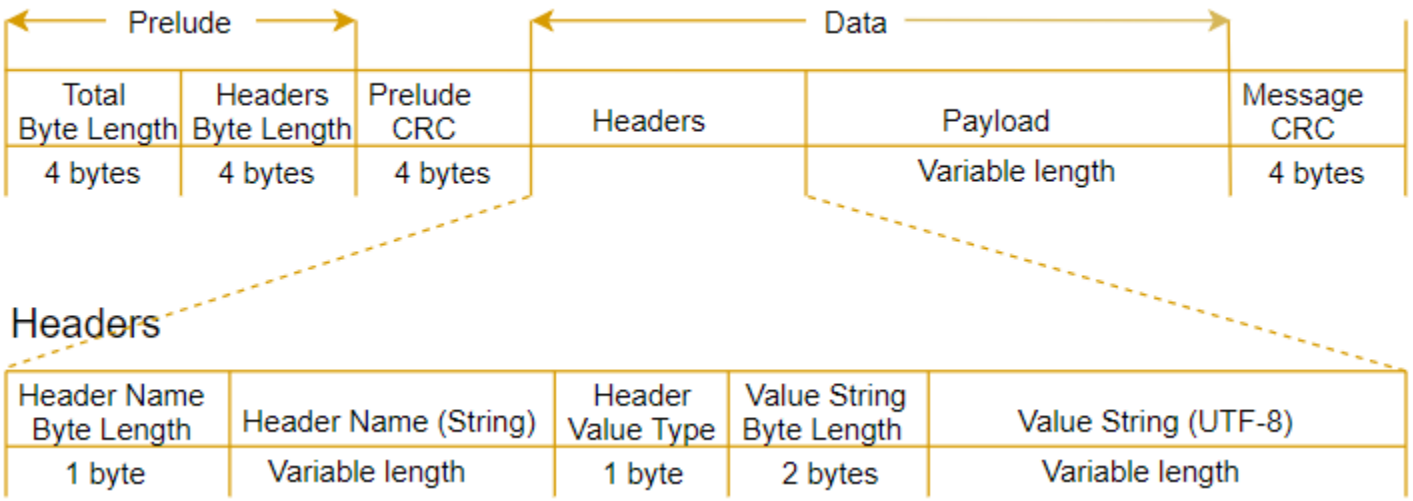
A codificação do fluxo de eventos fornece comunicação bidirecional usando mensagens entre um cliente e um servidor. Os quadros de dados enviados para o serviço de streaming do Amazon Lex V2 são codificados neste formato. A resposta do Amazon Lex V2 também usa essa codificação.

Cada mensagem consiste em duas seções: o prelúdio e os dados. A seção de prelúdio contém o tamanho total de bytes da mensagem e o comprimento combinado de todos os cabeçalhos. A seção de dados contém os cabeçalhos e uma carga útil.

Cada seção termina com uma soma de verificação CRC inteira big-endian de 4 bytes. A soma de verificação de CRC da mensagem inclui a seção de prelúdio e a seção de dados. O Amazon Lex V2 usa CRC32 (geralmente chamado de GZIP CRC32) para calcular os dois CRCs. Para obter mais informações sobre CRC32, consulte Especificação do formato de arquivo GZIP versão 4.3.

O total de sobrecarga de mensagem, incluindo o prelúdio e ambas as somas de verificação, é de 16 bytes.

O diagrama a seguir mostra os componentes que formam uma mensagem e um cabeçalho. Há vários cabeçalhos por mensagem.



Cada mensagem contém os seguintes componentes:

- Prelúdio: Sempre com tamanho fixo de 8 bytes, dois campos de 4 bytes cada.
  - Primeiros 4 bytes: O tamanho total de bytes. Este é o número inteiro big-endian de bytes da mensagem inteira, incluindo o próprio campo de comprimento de 4 bytes.
  - Segundo grupo de 4 bytes: O comprimento de bytes do cabeçalho. Este é o comprimento de bytes integral do big-endian da porção dos cabeçalhos da mensagem, excluindo o próprio comprimento dos cabeçalhos.
- CRC do prelúdio: a soma de verificação CRC de 4 bytes para a porção do prelúdio da mensagem, excluindo a própria CRC. O prelúdio tem uma CRC separada da CRC da mensagem para garantir que o Amazon Lex V2 possa detectar informações de comprimento de byte corrompido imediatamente sem causar erros como as sobrecargas de buffer.
- Cabeçalhos: Metadados que anotam a mensagem, como o tipo de mensagem, tipo de conteúdo, e assim por diante. As mensagens têm vários cabeçalhos. Cabeçalhos são pares de valor da chave onde a chave é uma sequência de caracteres UTF-8. Os cabeçalhos podem aparecer em qualquer ordem na parte dos cabeçalhos da mensagem e um determinado cabeçalho pode aparecer apenas uma vez. Para os tipos de cabeçalho necessários, consulte as seções a seguir.
- Carga útil: o conteúdo de áudio ou texto enviado para o Amazon Lex.
- CRC da mensagem: a soma de verificação CRC de 4 bytes desde o início da mensagem até o início da soma de verificação. Inclui tudo na mensagem, exceto a CRC em si.

Cada cabeçalho contém os seguintes componentes. Há vários cabeçalhos por quadro.

- O comprimento de byte do nome do cabeçalho: o comprimento de bytes do nome do cabeçalho.

- Nome do cabeçalho: o nome do cabeçalho que indica o tipo de cabeçalho. Para valores válidos, consulte as seguintes descrições de quadro.
- Tipo de valor de cabeçalho: uma enumeração que indica o tipo de valor de cabeçalho.
- Comprimento de byte da string de valor: o comprimento de byte da string de valor do cabeçalho.
- Valor do cabeçalho: o valor da string do cabeçalho. Os valores válidos para esse campo dependem do tipo de cabeçalho. Para valores válidos, consulte as seguintes descrições de quadro.

Permitir que seu bot seja interrompido pelo usuário

Ao iniciar uma transmissão de áudio bidirecional entre um bot do Amazon Lex V2 e sua aplicação, você pode configurar o bot para ouvir a entrada do usuário enquanto ele envia de volta uma mensagem. Com isso, o usuário pode interromper a mensagem antes que o bot termine de reproduzi-la. Você pode usar essa configuração para situações em que o usuário talvez já saiba a resposta para uma pergunta, como quando ele é solicitado a fornecer um código CVV.

Um bot sabe quando o usuário interrompe uma mensagem ao detectar a entrada do usuário antes que sua aplicação possa enviar de volta um evento `PlaybackCompletion`. Quando o usuário interrompe um bot, o bot envia um `PlaybackInterruptionEvent`.

Por padrão, o usuário pode interromper qualquer mensagem que o bot esteja transmitindo para sua aplicação. Você pode alterar essa configuração no console do Amazon Lex V2.

Você pode alterar a forma como um usuário pode responder a uma mensagem editando um slot. Um slot faz parte de uma intenção e é o meio pelo qual o usuário fornece as informações desejadas. Cada slot tem uma mensagem para que o usuário forneça essas informações. Para saber mais sobre slots, consulte Como funciona.

Para alterar se o usuário pode interromper uma mensagem (console)

1. Faça login no AWS Management Console e abra o console Amazon Lex V2 no console Amazon Lex V2.
2. Em Bots, selecione um bot.
3. Em Idioma, selecione o idioma do bot.
4. Selecione Exibir intenções.
5. Escolha a intenção .
6. Para Slots, escolha um slot.

7. Em Opções avançadas, escolha Mensagens de slot.
8. Escolha Mais opções de mensagem.
9. Marque ou desmarque a opção Usuários podem interromper a mensagem quando ela estiver sendo lida.

Você pode testar essa funcionalidade criando um bot com dois slots e especificando que os usuários não podem interromper a mensagem de um slot. Se você interromper uma mensagem interruptível, o bot enviará um evento de interrupção da reprodução. Se você interromper uma mensagem ininterruptível, a mensagem continuará sendo reproduzida.

Permitir que o bot espere que o usuário forneça mais informações

Ao iniciar uma transmissão bidirecional de um bot do Amazon Lex V2 para sua aplicação, você pode configurar o bot para esperar que o usuário forneça informações adicionais. Há circunstâncias em que um usuário pode não estar pronto para responder a uma solicitação. Por exemplo, um usuário pode não estar pronto para fornecer as informações do cartão de crédito porque a carteira está em outra sala.

Ao usar o comportamento Esperar e continuar do bot Amazon Lex V2, os usuários podem dizer frases como “espere um segundo” para fazer com que o bot espere até que eles encontrem as informações e as forneçam. Quando você ativa esse comportamento, o bot envia lembretes periódicos ao usuário para fornecer as informações. Ele não envia eventos de transcrição porque não há declarações do usuário para transcrever.

O bot do Amazon Lex V2 gerencia automaticamente uma conversa em streaming. Não é necessário escrever nenhum código adicional para ativar essa funcionalidade. Quando o usuário solicita que o bot espere, o `state` do `Intent` é `Waiting` e o `type` do `DialogAction` é`ElicitSlot`. Você pode usar essas informações para ajudar a personalizar sua aplicação de acordo com suas necessidades. Por exemplo, você pode configurar sua aplicação para reproduzir música quando o usuário estiver procurando o cartão de crédito.

Você ativa o comportamento de esperar e continuar para um slot individual. Para saber mais sobre slots, consulte Como funciona.

Para habilitar, aguardar e continuar

1. Faça login no AWS Management Console e abra o console Amazon Lex V2 no console Amazon Lex V2.

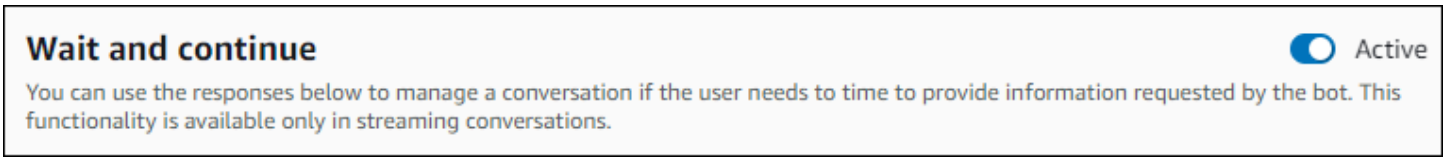
2. Em Bots, selecione um bot.
3. Em Idioma, selecione o idioma do bot.
4. Selecione Exibir intenções.
5. Escolha a intenção .
6. Em Slots, escolha um slot.
7. Em Opções avançadas, escolha Aguardar e continuar.
8. Em Aguardar e continuar, especifique os seguintes campos:

    - Resposta quando o usuário quer que o bot espere. É assim que o bot responde quando o usuário pede que ele espere pelas informações adicionais.
    - Resposta se o usuário precisar que o bot continue esperando. Essa é a resposta que o bot envia para lembrar ao usuário que ele ainda está esperando pelas informações. Você pode alterar a frequência com que o bot lembra o usuário.
    - Resposta quando o usuário quiser continuar. Essa é a resposta do bot quando o usuário tem as informações solicitadas.

Para cada resposta do bot, você pode fornecer várias variações da resposta, e uma é apresentada ao usuário aleatoriamente. Você também pode escolher se essas respostas podem ser interrompidas pelo usuário.

Para testar a funcionalidade de esperar e continuar, configure seu bot para aguardar a entrada do usuário e iniciar uma transmissão para um bot do Amazon Lex V2. Para obter informações sobre transmissão para um bot, consulte Usar a API para iniciar uma conversa em streaming. .

Talvez seja necessário desativar a espera e continuar as respostas. Use o botão Ativo para definir se as respostas de espera e continuação são usadas ou não.



Configurar atualizações do progresso do atendimento

Quando a função do Lambda de atendimento de uma intenção é chamada, o bot não envia uma resposta até que a função seja concluída. Se a função do Lambda levar mais do que alguns segundos para ser concluída, o usuário pode pensar que o bot não está respondendo. Para

resolver isso, você pode configurar seu bot para enviar atualizações ao usuário enquanto a função do Lambda de atendimento está em execução, para que o usuário saiba que o bot ainda está trabalhando em sua solicitação.

Quando você adiciona atualizações de atendimento a uma intenção, o bot responde no início do atendimento e periodicamente enquanto o atendimento está em andamento. Ao configurar a resposta inicial, você pode especificar um atraso antes que o bot envie a resposta. Com isso, você pode oferecer suporte a casos em que o atendimento não termina de forma relativamente rápida. Ao configurar uma resposta de atualização, você especifica a frequência na qual você deseja que as atualizações sejam enviadas. Você também configura um tempo limite para limitar o runtime da função de atendimento.

Você também pode adicionar respostas pós-atendimento a um bot. Isso permite que o bot envie uma resposta diferente dependendo se o atendimento foi bem-sucedido, falhou ou expirou.

As atualizações de atendimento são usadas somente ao interagir com um bot usando a operação StartConversation. Você pode usar a atualização pós-atendimento ao interagir com o bot usando as operações StartConversation, RecognizeText e RecognizeUtterance

Atualizações de atendimento

As atualizações de atendimento são enviadas enquanto sua função do Lambda está atendendo uma intenção. Ao ativar as atualizações de processamento, você fornece uma resposta inicial que é enviada no início do atendimento e uma resposta de atualização que é enviada periodicamente enquanto o processamento está em andamento.

Ao especificar uma resposta de atualização, você também especifica um tempo limite que determina por quanto tempo a função de atendimento pode ser executada. Você pode especificar um tempo limite de até 15 minutos (900 segundos).

Se você desativar as atualizações de atendimento, configurando `active` como falso no console ou usando a operação CreateIntent ou UpdateIntent, o tempo limite especificado para as atualizações de atendimento não será usado e, em vez disso, será usado o tempo limite padrão de 30 segundos.

Se a função de atendimento expirar, o Amazon Lex V2 executará uma das etapas a seguir:

- A resposta pós-atendimento está configurada e ativa: retorna a resposta de tempo limite.
- A resposta pós-atendimento está configurada e não está ativa: retorna uma exceção.
- A resposta pós-atendimento não está configurada: retorna uma exceção.

Iniciar resposta

O Amazon Lex V2 retorna a resposta inicial quando a função de atendimento do Lambda é chamada durante uma conversa de streaming. Normalmente, informa ao usuário que cumprir a intenção leva algum tempo e que ele deve esperar. A resposta inicial não é retornada quando você usa as operações `RecognizeText` ou `RecognizeUtterance`.

Você pode especificar até cinco mensagens de resposta. O Amazon Lex V2 escolhe uma das mensagens a reproduzir para o usuário.

Você pode configurar um atraso entre o momento em que a função do Lambda é chamada e o momento em que a resposta inicial é retornada. A resposta inicial não será retornada se a função do Lambda concluir seu trabalho antes que o atraso seja concluído.

Você pode usar o botão de alternância `active` no console ou na estrutura FulfillmentUpdatesSpecification para ativar e desativar a resposta inicial. Quando `active` for falso, a resposta inicial não será reproduzida.

Atualizar resposta

O Amazon Lex retorna a resposta de atualização periodicamente durante uma conversa de streaming enquanto a função de atendimento do Lambda está em execução. A resposta de atualização não é reproduzida quando você usa as operações `RecognizeUtterance` ou `RecognizeText`. Você pode configurar a frequência com que a resposta da atualização é reproduzida. Por exemplo, você pode reproduzir uma resposta de atualização a cada 30 segundos enquanto a função de atendimento é executada para informar ao usuário que o processo está em execução e que ele deve continuar aguardando.

Você pode especificar até cinco mensagens de atualização. O Amazon Lex V2 escolhe uma mensagem para ser reproduzida para o usuário. O uso de várias mensagens evita que as atualizações sejam repetitivas.

Se o usuário fornecer entrada via voz, DTMF ou texto enquanto a função do Lambda de atendimento estiver em execução, o Amazon Lex V2 retornará a resposta de atualização para o usuário.

Se a função do Lambda concluir seu trabalho antes do término do primeiro período de atualização, a resposta da atualização não será retornada.

Você pode usar o botão de alternância `active` no console ou a estrutura FulfillmentUpdatesSpecification para ativar e desativar a resposta de atualização. Quando `active` for falso, a resposta da atualização não será retornada.

Resposta pós-atendimento

O Amazon Lex V2 retorna uma resposta de pós-atendimento quando a função de atendimento termina. Uma resposta de pós-atendimento pode ser usada para cumprir qualquer intenção, não apenas ao transmitir conversas. A resposta pós-atendimento permite que o usuário saiba que a função está completa e o resultado.

Você pode usar o botão de alternância `active` no console ou a estrutura PostFulfillmentStatusSpecification para ativar e desativar a resposta pós-atendimento. Quando `active` for falso, a resposta não será reproduzida.

Existem três tipos de respostas pós-atendimento:

- Sucesso: retornada quando a função do Lambda de atendimento conclui seu trabalho com êxito. Se as respostas pós-atendimento não estiverem ativas. O Amazon Lex V2 executa a próxima ação configurada.
- Tempo limite: retornada se a função do Lambda não concluir seu trabalho antes que o período de tempo limite configurado termine. Se as respostas pós-atendimento não estiverem ativas, o Amazon Lex V2 retornará uma exceção.
- Falha: retornada quando a função do Lambda retorna o status `Failed` na resposta ou quando o Amazon Lex V2 encontra um erro ao cumprir a intenção. Se as respostas pós-atendimento não estiverem ativas, o Amazon Lex V2 retornará uma exceção.

Você pode especificar até cinco mensagens para cada tipo. O Amazon Lex V2 escolhe uma das mensagens a reproduzir para o usuário.

Diferentemente das respostas de início e atualização do processamento, as respostas pós-atendimento são reproduzidas tanto em conversas em transmissão como em outros modos.

Você também tem a opção de substituir essas mensagens configurando a função do Lambda para retornar uma mensagem pós-atendimento.

> **Note**
>
> Se a intenção tiver uma resposta final, ela será retornada após a resposta pós-cumprimento.

Exemplo de pós-atendimento

Para entender melhor a resposta pós-atendimento, vamos usar, como exemplo, um bot *BookTrip*, criado para ajudar a planejar uma viagem, com uma intenção de *BookFlight*, configurada com uma função do Lambda de atendimento que reserva o voo do cliente com uma companhia aérea. Depois que os slots do *BookFlight* forem obtidos, o Amazon Lex V2 invoca a função do Lambda de atendimento. Durante esse processo de atendimento, um dos três resultados a seguir pode ocorrer:

- Sucesso: o voo foi reservado com sucesso.
- Tempo limite: o processo de reserva demora mais do que o runtime configurado do Lambda de atendimento (por exemplo, se a companhia aérea não puder ser contatada dentro do tempo estipulado).
- Falha: a reserva falha por outro motivo.

Você pode aproveitar a resposta pós-atendimento para fornecer uma resposta mais significativa aos seus clientes em cada uma dessas situações. Os exemplos de cada situação são os seguintes:

- Resposta de sucesso: “Conseguimos reservar sua passagem com sucesso e enviamos um e-mail de confirmação. Sinta-se à vontade para entrar em contato conosco usando as informações de contato fornecidas nesse e-mail se tiver alguma dúvida.”
- Tempo limite de resposta: “Devido ao tráfego intenso em nossos sistemas, a reserva de sua passagem está demorando mais do que o esperado. Sua solicitação está em nossa fila e enviamos um e-mail com o número de referência correspondente a essa solicitação. Assim que reservarmos o bilhete, enviaremos uma confirmação da reserva. Sinta-se à vontade para entrar em contato conosco usando as informações de contato fornecidas nesse e-mail se tiver alguma dúvida.”

  > **Note**
  >
  > Se você não configurar uma mensagem de tempo limite, o Lex gerará um erro 4XX correspondente ao caso de uso.

- Falha na resposta: “Infelizmente, não foi possível reservar sua passagem. Enviamos um e-mail com detalhes sobre o problema que encontramos ao fazer sua reserva.”

Configurar tempos limite para capturar a entrada do usuário

A API de streaming do Amazon Lex V2 permite que um bot detecte automaticamente enunciados nas entradas do usuário. Ao criar uma intenção ou um espaço, você pode configurar aspectos de um enunciado, como a duração máxima de um enunciado, o tempo limite durante a espera pela entrada do usuário ou o caractere final da entrada DTMF. Você pode personalizar o comportamento de um bot para seu caso de uso. Por exemplo, você pode limitar o número de dígitos de um número de cartão de crédito a 16.

Você também pode configurar tempos limite por meio de atributos de sessão ao iniciar uma conversa com um bot e sobrescrevê-los em sua função do Lambda, se necessário.

As chaves de configuração de um atributo usam a sintaxe a seguir:

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

`InputType` pode ser **`audio`**, **`dtmf`** ou **`text`**.

Você pode definir as configurações padrão para todas as intenções ou slots em um bot especificando **`*`** como a intenção ou o nome do slot. Qualquer configuração específica de intenção ou slot tem precedência sobre as configurações padrão.

O Amazon Lex V2 fornece atributos de sessão predefinidos para gerenciar a forma como as operações do StartConversation funcionam com texto, voz ou entrada DTMF para seu bot. Todos os atributos predefinidos estão no namespace `x-amz-lex`.

Você pode definir as configurações padrão para todas as intenções, slots ou subslots em um bot especificando `*` como a intenção ou o nome do slot. Qualquer configuração específica de intenção ou slot tem precedência sobre as configurações padrão. Use esses padrões para todos os tempos limite abaixo.

Para um subslot de um slot composto, você pode separar por `.`. Por exemplo:

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

Expressão	Cenário
Intent:Slot.SubSlot	Aplicável somente ao subslot chamado 'SubSlot' dentro do slot composto chamado 'Slot'
Intent:Slot.*	Aplicável a qualquer subslot dentro do slot composto chamado 'Slot'
Intent:*.SubSlot	Aplicável somente ao subslot chamado 'SubSlot' dentro de qualquer slot composto
Intent:*.*	Aplicável a qualquer subslot dentro de qualquer slot composto

Comportamento de interrupção

Você pode configurar o comportamento de interrupção do bot. O atributo é definido pelo Amazon Lex V2.

Permitir interrupção

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

Define se o usuário pode interromper o prompt reproduzido pelo bot do Amazon Lex V2. Você pode desativá-lo seletivamente.

Padrão: Verdadeiro

Tempos limite para entrada de voz

Você pode definir valores de tempo limite para interação de voz com seu bot usando atributos de sessão. Os atributos são definidos pelo Amazon Lex V2. Esses atributos permitem que você especifique quanto tempo o Amazon Lex V2 espera até que um cliente termine de falar antes de coletar a fala de entrada.

Todos esses atributos estão no namespace `x-amz-lex:audio`.

Comprimento máximo do enunciado

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

Define quanto tempo o Amazon Lex V2 espera antes que a entrada de fala seja truncada e a fala retorne à sua aplicação. Você pode aumentar o tamanho da entrada quando houver expectativa de respostas longas, ou se quiser dar mais tempo para os clientes fornecerem informações.

Padrão: 13 mil milissegundos (13 segundos). O valor máximo é 15 mil milissegundos (15 segundos)

Se você definir o atributo `max-length-ms` para mais de 15 mil milissegundos, o valor padrão será 15 mil milissegundos.

Tempo limite de voz

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

Quanto tempo o bot espera antes de presumir que o cliente não vai falar. Você pode aumentar o tempo em situações nas quais o cliente precise de mais tempo para localizar ou recuperar informações antes de falar. Por exemplo, talvez queira dar mais tempo para os clientes encontrarem o cartão de crédito para inserir o número.

Padrão: 4 mil milissegundos (4 segundos)

Tempo limite do silêncio

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

Quanto tempo um bot espera após o cliente parar de falar para presumir que o enunciado foi concluído. Você pode aumentar o tempo em situações nas quais os períodos de silêncio são esperados ao fornecer entradas.

Padrão: 600 milissegundos (0,6 segundos)

Permitir entrada de áudio

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

Você pode habilitar esse atributo para que o bot aceite a entrada do usuário somente por meio da modalidade de áudio. O bot não aceitará entrada de áudio se esse sinalizador for definido como falso. Por padrão, o valor é definido como verdadeiro.

Padrão: Verdadeiro

Tempos limite para entrada de texto

Use o atributo de sessão a seguir para especificar como seu bot se comporta com o modo de conversação por texto.

Esse atributo está no namespace `x-amz-lex:text`.

Limite de tempo limite de início

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

Quanto tempo o bot espera antes de solicitar novamente ao cliente a entrada de texto. É possível aumentar o tempo em situações nas quais você deseja que o cliente tenha mais tempo para localizar ou recuperar informações antes de fornecer a entrada de texto. Por exemplo, você pode desejar que os clientes tenham mais tempo para encontrar detalhes do pedido. Como alternativa, você pode reduzir o limite para avisar os clientes mais cedo.

Padrão: 30 mil milissegundos (30 segundos)

Configuração para entrada DTMF

Use os seguintes atributos de sessão para especificar como seu bot do Amazon Lex V2 responde à entrada DTMF ao usar uma conversa de áudio.

Todos esses atributos estão no namespace `x-amz-lex:dtmf`.

Caractere de exclusão

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

O caractere DTMF que limpa os dígitos DTMF acumulados e encerra imediatamente a entrada.

Padrão: *

Caractere final

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

O caractere DTMF que encerra imediatamente a entrada. Se o usuário não pressionar esse caractere, a entrada terminará após o tempo limite de término.

Padrão: #

Tempo limite do fim

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

Quanto tempo o bot deve esperar a partir da última entrada do caractere DTMF antes de assumir que a entrada foi concluída.

Padrão: 5 mil milissegundos (5 segundos)

Número máximo de dígitos DTMF por enunciado

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

O número máximo de dígitos DTMF permitidos em um enunciado. Por exemplo, você pode definir esse valor como 16 para limitar o número de caracteres que podem ser inseridos para um número de cartão de crédito. Este valor não pode ser aumentado.

Padrão: 1024 caracteres

Permitir entrada do DTMF

Você pode definir o tipo de entrada que o bot pode aceitar usando os atributos da sessão. Os atributos são definidos pelo Amazon Lex V2.

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

Você pode habilitar esse atributo para que o bot aceite a entrada do usuário via modalidade DTMF. O bot não aceitará a entrada do DTMF se esse sinalizador estiver definido como falso. Por padrão, o valor é definido como verdadeiro.

Padrão: Verdadeiro

Importar e exportar

Você pode exportar uma definição de bot, um local de bot ou um vocabulário personalizado e depois importar novamente para criar outro recurso ou substituir um recurso de uma conta da AWS. Por exemplo, você pode exportar um bot de uma conta de teste e depois criar uma cópia do bot na sua conta de produção. Você pode copiar um bot de uma região AWS para outra região.

Você pode alterar os recursos do recurso exportado antes de importá-lo. Por exemplo, você pode exportar um bot e depois editar o arquivo JSON de um slot para adicionar ou remover declarações de elicitação do valor de um slot específico. Depois de editar a definição, importe o arquivo modificado.

Tópicos

Exportação

Você exporta um bot, o local do bot ou o vocabulário personalizado usando o console ou a operação `CreatExport`. Você especifica o recurso a ser exportado e pode fornecer uma senha opcional para proteger o arquivo zipado ao iniciar uma exportação. Depois de baixar o arquivo zipado, use a senha para acessar o arquivo. Para mais informações, consulte Como usar uma senha ao importar ou exportar.

A exportação é uma operação de natureza assíncrona. Depois de iniciar a exportação, use o console ou a operação `DescribeExport` para monitorar o progresso. Quando a exportação terminar, o console ou a operação `DescribeExport` mostrará o status `COMPLETED`, e o console baixará o arquivo zipado de exportação para seu navegador. Se você usar a operação `DescribeExport`, o Amazon Lex V2 fornecerá um URL pré-assinado do Amazon S3 para você baixar os resultados da exportação. O URL de download fica disponível por apenas cinco minutos. Para obter um novo URL, chame a operação `DescribeExport` novamente.

Você pode ver o histórico das exportações de um recurso com o console ou com a operação `ListExports`. Os resultados mostram as exportações junto com o status atual. Uma exportação fica disponível no histórico por sete dias.

Quando você exporta a versão `Draft` de um bot ou de um local de um bot, é possível que a definição no arquivo JSON fique em um estado inconsistente, pois é possível alterar a versão de `Draft` de um bot ou o local de um bot enquanto a exportação está em andamento. Se a versão de `Draft` for alterada durante a exportação, as alterações talvez não sejam incluídas no arquivo de exportação.

Quando você exporta o local de um bot, o Amazon Lex exporta todas as informações que definem o local, incluindo região, vocabulário personalizado, intents, tipos de slots e slots.

Quando você exporta um bot, o Amazon Lex exporta todos os locais definidos do bot, incluindo os intents, tipos de slots e slots. Os itens a seguir não são exportados com um bot:

- Aliases de bot
- ARN do perfil associado a um bot
- Tags associadas a bots e aliases de bots
- Hooks de código Lambda associados a um alias de bot

O ARN e as tags do perfil são inseridos como parâmetros de solicitação quando você importa um bot. Você precisa criar aliases de bot e atribuir hooks de código Lambda após a importação se necessário.

Você pode remover uma exportação e o arquivo zipado associado usando o console ou a operação `DeleteExport`.

Para ver um exemplo de exportação de um bot usando o console, consulte Como exportar um bot (console).

Permissões do IAM necessárias para exportar

Para exportar bots, locais de bots e vocabulários personalizados, o usuário que executa a exportação precisa ter as seguintes permissões do IAM.

API	• Ações de IAM necessárias	Recurso
CreateExport	• CreateExport	Bot

API	• Ações de IAM necessárias	Recurso
UpdateExport	• UpdateExport	Bot
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	Bot
DescribeExport para vocabulários personalizados	• DescribeExport • DescribeCustomVocabulary	bot
DeleteExport	• DeleteExport	Bot
ListExports	• ListExports	*

Para ver um exemplo de política do IAM, consulte Permitir que um usuário exporte bots e localidades de bots .

Como exportar um bot (console)

Você pode exportar um bot da lista de bots, da lista de versões ou da página de detalhes da versão. Quando você escolhe uma versão, o Amazon Lex V2 exporta essa versão. As instruções a seguir pressupõem que vai começar a exportar o bot da lista de bots; mas, quando você começa com uma versão, as etapas são as mesmas.

Exportar um bot usando o console

1. Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Na lista de bots, escolha o bot que quer exportar.
3. Em Ação, escolha Exportar.
4. Escolha a versão, a plataforma e o formato de exportação do bot.
5. (Opcional) Insira uma senha para o arquivo zipado. Com uma senha, você protege o arquivo de saída.
6. Escolha Exportar.

Depois de iniciar a exportação, retorne à lista de bots. Para monitorar o progresso da exportação, use a lista de histórico de importação/exportação. Quando o status da exportação é Complete, o console baixa automaticamente o arquivo zipado para seu computador.

Para baixar a exportação de novo, na lista de importação/exportação, escolha a exportação e, em seguida, Download. Você pode colocar uma senha no arquivo .zip obtido por download.

Exportar uma linguagem de bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Na lista de bots, escolha o bot cujo idioma deseja exportar.
3. Em Adicionar idiomas, escolha Visualizar idiomas.
4. Na lista Todos os idiomas, escolha o idioma para exportar.
5. Em Ação, escolha Exportar.
6. Escolha a versão, a plataforma e o formato do bot.
7. (Opcional) Insira uma senha para o arquivo zipado. Com uma senha, você protege o arquivo de saída.
8. Escolha Exportar.

Depois de iniciar a exportação, você retorna à lista de idiomas. Para monitorar o progresso da exportação, use a lista de histórico de importação/exportação. Quando o status da exportação é Complete, o console baixa automaticamente o arquivo zipado para seu computador.

Para baixar a exportação de novo, na lista de importação/exportação, escolha a exportação e, em seguida, Download. Você pode colocar uma senha no arquivo .zip obtido por download.

Importação

Para usar o console para importar um bot, um local de bot ou um vocabulário personalizado exportado anteriormente, forneça o local do arquivo no seu computador local e a senha opcional para desbloquear o arquivo. Para ver um exemplo, consulte Como importar um bot (console).

Quando você usa a API, importar um recurso é um processo de três etapas:

1. Crie um URL de upload usando a operação `CreateUploadUrl`. Não é necessário criar um URL de upload se estiver usando o console.
2. Carregue o arquivo zipado com a definição do recurso.
3. Inicie a importação com a operação `StartImport`.

O URL de upload é um URL pré-assinado do Amazon S3 com permissão de gravação. O URL fica disponível por cinco minutos após ser gerado. Se você proteger com senha o arquivo zipado, será preciso informar a senha para iniciar a importação. Para mais informações, consulte Como usar uma senha ao importar ou exportar.

Uma importação é um processo assíncrono. Monitore o andamento de uma importação usando o console ou a operação de `DescribeImport`.

Quando você importa um bot ou o local do bot, pode haver conflitos entre os nomes dos recursos no arquivo de importação e os nomes dos recursos existentes no Amazon Lex V2. O Amazon Lex V2 pode lidar com o conflito de três maneiras:

- Falha em caso de conflito — A importação é interrompida e nenhum recurso é importado do arquivo zipado de importação.
- Substituir — O Amazon Lex V2 importa todos os recursos do arquivo importado zipado e substitui qualquer recurso existente pela definição do arquivo de importação.
- Anexar — O Amazon Lex V2 importa todos os recursos do arquivo importado zipado e substitui qualquer recurso existente pela definição do arquivo de importação. Isso está disponível somente para o local do bot.

Você pode ver uma lista das importações para um recurso usando o console ou a operação `ListImports`. As importações permanecem na lista por sete dias. Você pode usar o console ou a operação `DescribeImport` para ver detalhes sobre uma importação específica.

Você também pode remover uma importação e o arquivo zipado associado usando o console ou a operação `DeleteImport`.

Para ver um exemplo de importação de um bot usando o console, consulte Como importar um bot (console).

Permissões do IAM necessárias para importar

Para importar bots, locais de bots e vocabulários personalizados, o usuário que executa a importação precisa ter as seguintes permissões do IAM.

API	Ações de IAM necessárias	Recurso
CreateUploadUrl	• CreateUploadUrl	*
StartImport para bots e local de bots	• StartImport • iam:PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary	1. Para importar um novo bot: bot, alias de bot. 2. Para substituir um bot existente: bot. 3. Para importar um novo local: bot.

API	Ações de IAM necessárias	Recurso
	• DeleteLocale • DeleteIntent • DeleteSlot • DeleteSlotType	
StartImport para vocabulários personalizados	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	bot
DescribeImport	• DescribeImport	Bot
DeleteImport	• DeleteImport	Bot
ListImports	• ListImports	*

Para ver um exemplo de política do IAM, consulte Permitir que um usuário importe bots e localidades de bots .

Como importar um bot (console)

Importar um bot usando o console

1. Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Em Ação, escolha Importar.
3. Em Arquivo de entrada, dê um nome ao bot e escolha o arquivo zipado que contém os arquivos JSON que definem o bot.
4. Informe a senha do arquivo zipado, se houver. Não é obrigatório usar uma senha no arquivo, mas isso ajuda a proteger o conteúdo.
5. Crie ou insira o perfil do IAM que define as permissões para o bot.
6. Indica se o seu bot está sujeito à Lei de Proteção à Privacidade Online para Crianças (COPPA).

7. Forneça uma configuração de tempo limite de inatividade para o bot. Se você não fornecer um valor, será usado o valor do arquivo zipado. Se o arquivo zipado não contiver uma configuração de tempo limite, o Amazon Lex V2 usa o padrão de 300 segundos (cinco minutos).
8. (Opcional) Adicione tags ao bot.
9. Escolha se deseja avisar sobre a substituição de bots existentes com o mesmo nome. Se você ativar os avisos de que o bot que você está importando sobrescreverá um bot existente, você receberá um aviso e o bot não será importado. Se você desativar os avisos, o bot importado substituirá o bot existente com o mesmo nome.
10. Escolha Import.

Depois de iniciar a importação, você vai retornar à lista de bots. Para monitorar o progresso da importação, use a lista de histórico de importação/exportação. Quando o status da importação for Complete, você poderá escolher o bot na lista de bots para modificar ou compilar o bot.

Importar uma linguagem de bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2/home.
2. Na lista de bots, escolha o bot para o qual você quer importar um idioma.
3. Em Adicionar idiomas, escolha Visualizar idiomas.
4. Em Ação, escolha Importar.
5. Em Arquivo de entrada, escolha o arquivo que contém o idioma a ser importado. Se você protegeu o arquivo zipado, informe a senha em Senha.
6. Em Idioma, escolha o idioma para importar. O idioma não precisa ser o mesmo do arquivo de importação. Você pode copiar os intents de um idioma para outro.
7. Em Voz, escolha a voz do Amazon Polly a ser usada para interação por voz ou escolha Nenhuma para um bot somente de texto.
8. Em Limite de pontuação de confiança, insira o limite em que o Amazon Lex V2 vai inserir `AMAZON.FallbackIntent` e `AMAZON.KendraSearchIntent` ou ambos ao retornar intents alternativos.
9. Escolha se deseja avisar sobre a substituição de um idioma existente. Se você ativar os avisos de que o idioma que você está importando sobrescreverá um idioma existente, você receberá um aviso e o idioma não será importado. Se você desativar os avisos, o idioma importado substituirá o idioma existente.

10. Escolha Importar para iniciar a importação. do idioma.

Depois de iniciar a importação, você vai retornar à lista de idiomas. Para monitorar o progresso da importação, use a lista de histórico de importação/exportação. Quando o status da importação for Complete, você poderá escolher o idioma na lista de bots para modificar ou compilar o bot.

Como usar uma senha ao importar ou exportar

O Amazon Lex V2 pode proteger com senha seus arquivos de exportação ou ler seus arquivos de importação protegidos usando a compactação padrão de arquivos zipados. Você deve sempre proteger com senha seus arquivos de importação e exportação.

O Amazon Lex V2 envia seu arquivo de exportação para um bucket do S3, que fica disponível para você com um URL pré-assinado do S3. O URL só está disponível por cinco minutos. O arquivo está disponível para qualquer pessoa com acesso ao URL de download. Para ajudar a proteger os dados no arquivo, forneça uma senha ao exportar o recurso. Se precisar obter o arquivo após o URL expirar, você pode usar o console ou a operação `DescribeExport` para obter um novo URL.

Se você perder a senha de um arquivo de exportação, poderá criar uma nova para um arquivo existente escolhendo Download na tabela do histórico de importação/exportação ou usando a operação `UpdateExport`. Se você escolher Download na tabela de histórico para uma exportação e não fornecer uma senha, o Amazon Lex V2 fará o download de um arquivo zip desprotegido.

Formato JSON para importação e exportação

Para importar e exporta bots, locais de bots ou vocabulários personalizados do Amazon Lex V2, use um arquivo zipado que contém estruturas JSON que descrevem as partes do recurso. Quando você exporta um recurso, o Amazon Lex V2 cria o arquivo zipado e o disponibiliza para você usando um URL pré-assinado do Amazon S3. Ao importar um recurso, você deve criar um arquivo zipado que contenha as estruturas JSON e carregá-lo em um URL pré-assinado do S3.

O Amazon Lex cria a seguinte estrutura de diretórios no arquivo zipado quando você exporta um bot. Quando você exporta um local de bot, somente a estrutura abaixo local é exportado. Quando você exporta um vocabulário personalizado, somente a estrutura sob o vocabulário personalizado é exportado.

```
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
                      ...
```

Estrutura de arquivo de manifesto

O arquivo de manifesto contém metadados para o arquivo de exportação.

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

Estrutura de arquivo do bot

O arquivo do bot contém as informações de configuração para o bot.

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

Estrutura de arquivo de local do bot

O arquivo de local do bot contém uma descrição do local ou do idioma de um bot. Quando você exporta um bot, pode haver mais de um arquivo de local do bot no arquivo zipado. Quando você exporta um local de bot, há somente um local no arquivo zipado.

```
{
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

Estrutura do arquivo do intent

O arquivo de intent contém as informações de configuração de um intent. Há um arquivo de intent no arquivo zipado para cada intent em um local específico.

Veja a seguir um exemplo de uma estrutura JSON para o intent da BookCar no exemplo de bot da BookTrip. Para ver um exemplo completo da estrutura JSON para um intent, consulte a operação CreateIntent.

```
{
    "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
        }
    ]
}
```

Estrutura de arquivo do slot

O arquivo de slot contém as informações de configuração de um slot em um intent. Há um arquivo de slot no arquivo zipado para cada slot definido para um intent em um local específico.

O exemplo a seguir é a estrutura JSON de um slot que permite ao cliente escolher o tipo de carro que deseja alugar no intento do BookCar no bot de exemplo da BookTrip. Para ver um exemplo completo da estrutura JSON para um slot, consulte a operação CreateSlot.

```
{
    "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,
    }
}
```

O exemplo a seguir mostra a estrutura JSON de um slot de composição.

```
{
  "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)"
  }
}
```

Estrutura do arquivo do tipo de slot

O arquivo do tipo de slot contém as informações de configuração de um tipo de slot personalizado usado em um idioma ou local. Há um arquivo de tipo de slot no arquivo zipado para cada tipo de slot personalizado em um local específico.

A seguir está a estrutura JSON para o tipo de slot que lista os tipos de carros disponíveis no bot de exemplo da BookTrip. Para ver um exemplo completo da estrutura JSON de um tipo de slot, consulte a operação CreateSlotType.

```
{
    "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
    }
}
```

O exemplo a seguir mostra a estrutura JSON de um tipo de slot de composição.

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

A seguir está um tipo de slot que usa uma gramática personalizada para entender as declarações do cliente. Para mais informações, consulte Tipo de slot de gramática.

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

Estrutura de arquivo de vocabulário personalizado.

O arquivo de vocabulário personalizado contém as entradas em um vocabulário personalizado para um único idioma ou local. Há um arquivo de vocabulário personalizado no arquivo zipado para cada localidade que tem um vocabulário personalizado.

A seguir está um arquivo de vocabulário personalizado de um bot que recebe pedidos de restaurantes. Há um arquivo por local no 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"
        }
    ]
}
```

Marcar recursos

Para ajudá-lo a gerenciar os bots e aliases de bot do Amazon Lex V2 é possível atribuir metadados a cada recurso como tags. Uma tag é um rótulo atribuído a um recurso da AWS. Cada tag consiste em uma chave e um valor.

As tags permitem categorizar seus recursos da AWS de diferentes formas, por exemplo, por finalidade, proprietário ou aplicação. As tags ajudam a:

- Identificar e organizar seus recursos da AWS. Muitos recursos da AWS oferecem suporte à marcação, portanto, é possível atribuir a mesma tag a recursos em diferentes serviços para indicar que os recursos são iguais. Por exemplo, é possível marcar um bot e as funções do Lambda que ele usa com a mesma tag.
- Aloque custos. É possível ativar as tags no painel do AWS Billing and Cost Management. A AWS usa as tags para categorizar seus custos e entregar um relatório mensal de alocação de custos para você. Para o Amazon Lex V2, é possível alocar custos para cada alias usando tags específicas para o alias. Para mais informações, consulte Usar etiquetas de alocação de custos no Guia do usuário do AWS Billing and Cost Management.
- Controle o acesso aos recursos da . Você pode usar tags com o Amazon Lex V2 para criar políticas para controlar o acesso aos recursos do Amazon Lex V2. Essas políticas podem ser anexadas a um usuário ou perfil do IAM para permitir o controle de acesso baseado em tags.

Você pode trabalhar com tags usando o AWS Management Console, o AWS Command Line Interface, ou a API do Amazon Lex V2.

Marcar recursos da

Se estiver usando o console do Amazon Lex V2, é possível marcar recursos ao criá-los ou adicionar as tags mais tarde. Também é possível usar o console para atualizar ou remover tags existentes.

Se você estiver usando a AWS CLI ou a API do Amazon Lex V2, use as seguintes operações para gerenciar tags para seus recursos:

- CreateBot e CreateBotAlias: aplica as tags ao criar um bot ou alias de bot.
- ListTagsForResource: visualiza as tags associadas a um recurso.
- TagResource: adiciona e modifica tags em um recurso existente.

- UntagResource: remove tags de um recurso.

Os seguintes recursos do Amazon Lex V2 são compatíveis com a marcação:

- Bots: use um nome do recurso da Amazon (ARN) como o seguinte:
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Aliases de bot: use um ARN como o seguinte:
  - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

Os valores `bot-id` e `bot-alias-id` são strings alfanuméricas em maiúsculas de 10 caracteres.

Restrições de tags

As restrições básicas a seguir se aplicam às tags nos recursos do Amazon Lex V2:

- Número máximo de chaves: 50 usando o console, 200 usando a API
- Tamanho máximo da chave – 128 caracteres
- Tamanho máximo do valor – 256 caracteres
- Caracteres válidos de chave e valor: a-z, A-Z, 0-9, espaço e os seguintes caracteres: _.:/=+- e @
- As chaves e os valores diferenciam letras maiúsculas de minúsculas
- Não use a `aws:` como um prefixo para chaves, pois ela é reservada para uso da AWS

Marcar recursos (console)

É possível usar o console para gerenciar tags em um bot ou alias de bot. É possível adicionar tags ao criar um recurso ou adicionar, modificar ou remover tags de recursos existentes.

Como adicionar uma tag ao criar um bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha Criar bot.
3. Na seção Configurações avançadas de Definir configurações do bot, escolha Adicionar nova tag. Você pode adicionar tags ao bot e ao alias do `TestBotAlias`.
4. Selecione Próximo para continuar a criação do seu bot.

Como adicionar uma tag ao criar um alias de bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot ao qual você deseja adicionar o alias de bot.
3. No menu esquerdo, escolha Aliases e depois Criar alias.
4. Em Informações gerais, escolha Adicionar nova tag em Tags.
5. Escolha Criar.

Como adicionar, remover ou modificar uma tag em um bot existente

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot que você deseja modificar.
3. No menu esquerdo, escolha Configurações e, em seguida, Editar.
4. Em Tags, faça suas alterações.
5. Escolha Salvar para salvar suas alterações no bot.

Para adicionar, remover ou modificar uma tag em um alias existente

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha o bot que você deseja modificar.
3. No menu à esquerda, escolha Aliases e, na lista de aliases, escolha o alias a ser modificado.
4. Em Detalhes do alias, em Tags, escolha Modificar tags.
5. Em Gerenciar tags, faça suas alterações.
6. Escolha Salvar para salvar as alterações no alias.

Segurança no Amazon Lex V2

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade que se aplicam ao Amazon Lex V2, consulte Serviços da AWS em escopo por programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon Lex V2. Os tópicos a seguir mostram como configurar o Amazon Lex V2 para atender aos seus objetivos de segurança e compatibilidade. Saiba também como usar outros serviços da AWS que ajudam você a monitorar e proteger os recursos do Amazon Lex V2.

Tópicos

Proteção de dados no Amazon Lex V2

O Amazon Lex V2 está em conformidade com o modelo de responsabilidade AWS compartilhada de , que inclui regulamentações e diretrizes para proteção de dados. AWS é responsável por proteger a infraestrutura global que executa todos os AWS serviços. AWS mantém o controle sobre os dados hospedados nessa infraestrutura, incluindo os controles de configuração de segurança para lidar com o conteúdo do cliente e os dados pessoais. AWS clientes e parceiros da APN, atuando como controladores ou processadores de dados, são responsáveis por todos os dados pessoais que colocam na AWS nuvem.

Para fins de proteção de dados, recomendamos proteger as credenciais da sua conta da AWS e configurar contas de usuário individuais com o AWS Identity and Access Management (IAM), de modo que cada usuário receba somente as permissões necessárias para cumprir suas funções. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS

- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções AWS de criptografia, juntamente com todos os controles de segurança padrão nos AWS serviços.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon S3.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como números de conta dos seus clientes, em campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Amazon Lex V2 ou outros AWS serviços usando o console, a API ou os AWS SDKs. AWS CLI Todos os dados que você insere no Amazon Lex V2 ou em outros serviços podem ser separados para inclusão em logs de diagnóstico. Ao fornecer um URL para um servidor externo, não inclua informações de credenciais no URL para validar a solicitação a esse servidor.

Para mais informações sobre proteção de dados, consulte a publicação Modelo de responsabilidade compartilhada da AWS e do GDPR no Blog de segurança da AWS .

Criptografia inativa

O Amazon Lex V2 criptografa os enunciados de usuário e outras informações que armazena.

Tópicos

Enunciados de amostra

Ao desenvolver um bot, você pode fornecer amostra de declarações para cada intenção e slot. Você também pode fornecer sinônimos e valores personalizados para slots. Essas informações são criptografadas em repouso e usadas apenas para criar o bot e criar a experiência do usuário.

Atributos da sessão

Atributos da sessão contêm informações específicas do aplicativo que são passadas entre o Amazon Lex V2 e os aplicativos clientes. O Amazon Lex V2 transmite atributos de sessão para todas as AWS Lambda funções configuradas para um bot. Se uma função do Lambda adicionar ou atualizar atributos da sessão, o Amazon Lex V2 passará as novas informações de volta para a aplicação cliente.

Os atributos da sessão permanecem em um armazenamento criptografado durante a sessão. Você pode configurar a sessão para permanecer ativa por um mínimo de 1 minuto e até 24 horas após a última declaração do usuário. O padrão, a duração da sessão é de 5 minutos.

Atributos de solicitação

Atributos de solicitação contêm informações específicas da solicitação e aplicam-se apenas à solicitação atual. Um aplicativo cliente usa atributos de solicitação para enviar informações para o Amazon Lex V2 em runtime.

Você usa atributos de solicitação para passar informações que não precisam ser mantidas durante toda a sessão. Como os atributos de solicitação não são são mantidos entre solicitações, eles não são armazenados.

Criptografia em trânsito

O Amazon Lex V2 usa o protocolo HTTPS para se comunicar com o aplicativo cliente. Ele usa HTTPS e AWS assinaturas para se comunicar com outros serviços, como o Amazon Polly AWS Lambda e em nome do seu aplicativo.

Gerenciamento de identidade e acesso para o Amazon Lex V2

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para utilizar os recursos do Amazon Lex V2. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Amazon Lex V2.

Usuário do serviço – Se você usar o serviço do Amazon Lex V2 para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões necessárias. À medida que mais atributos do Amazon Lex V2 forem usados para realizar o trabalho, talvez sejam necessárias permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões

corretas ao seu administrador. Se você não puder acessar um atributo no Amazon Lex V2, consulte Solução de problemas de identidade e acesso da Amazon Lex V2.

Administrador do serviço – Se você for o responsável pelos recursos do Amazon Lex V2 em sua empresa, provavelmente terá acesso total ao Amazon Lex V2. Cabe a você determinar quais funcionalidades e atributos do Amazon Lex V2 os usuários do serviço deverão acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como a empresa pode usar o IAM com o Amazon Lex V2, consulte Como o Amazon Lex V2 funciona com o IAM.

Administrador do IAM – Se você for um administrador do IAM, talvez deseje saber detalhes sobre como escrever políticas para gerenciar o acesso ao Amazon Lex V2. Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex V2 que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex V2.

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia AWS IAM Identity Center do usuário e Utilizar a autenticação multifator (MFA) na AWS no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do Usuário do IAM.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade. Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte “O que é o Centro de Identidade do IAM?” no Guia do usuário AWS IAM Identity Center .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e

chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do Usuário do IAM. Se você usar o Centro de identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário AWS IAM Identity Center .
- Permissões temporárias para usuários do IAM –– um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.

- Acesso entre contas –- é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Função de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
  - Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.
- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham

credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único

usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma

entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do Usuário do IAM.

- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte How SCPs work (Como os SCPs funcionam) no Guia do usuário do AWS Organizations .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do Usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o Amazon Lex V2 funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon Lex V2, entenda que atributos do IAM estão disponíveis para uso com o Amazon Lex V2.

Atributos do IAM que você pode usar com o Amazon Lex V2

Atributo do IAM	Suporte ao Amazon Lex V2
Políticas baseadas em identidade	Sim
Políticas baseadas em atributos	Sim
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de políticas	Não
ACLs	Não
ABAC (tags em políticas)	Sim
Credenciais temporárias	Não
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Funções vinculadas a serviço	Parcial

Para ter uma visão de alto nível de como o Amazon Lex V2 e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade do Amazon Lex V2

Suporta políticas baseadas em identidade	Sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos da política JSON do IAM no Guia do Usuário do IAM.

Exemplos de políticas baseadas em identidade para o Amazon Lex V2

Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex V2, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex V2.

Políticas baseadas em recursos no Amazon Lex V2

É compatível com políticas baseadas em atributos	Sim

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou serviços da AWS.

Não é possível usar políticas entre contas ou regiões com o Amazon Lex. Se você criar uma política para um recurso com um ARN entre contas ou regiões, o Amazon Lex retornará um erro.

O serviço Amazon Lex oferece suporte a políticas baseadas em recursos chamadas de política de bot e política de alias de bot, que são anexadas a um bot ou a um alias de bot. Essas políticas definem quais entidades principais podem realizar ações no bot ou no alias do bot.

As ações só podem ser usadas em recursos específicos. Por exemplo, a ação `UpdateBot` só pode ser usada em recursos de bots, a ação `UpdateBotAlias` só pode ser usada em recursos de alias de bots. Se você especificar uma ação em uma política que não possa ser usada no recurso

especificado na política, o Amazon Lex retornará um erro. Para obter a lista de ações e os recursos com os quais elas podem ser usadas, consulte a seguinte tabela.

Ação	Oferece suporte a políticas baseadas em recursos	Recurso
BuildBotLocalidade	Compatível	BotId
CreateBot	Não	
CreateBotPseudônimo	Não	
CreateBotChannel [somente permissão]	Compatível	BotId
CreateBotLocalidade	Compatível	BotId
CreateBotVersão	Compatível	BotId
CreateExport	Compatível	BotId
CreateIntent	Compatível	BotId
CreateResourcePolítica	Compatível	BotId, BotAliasId
CreateSlot	Compatível	BotId
CreateSlotTipo	Compatível	BotId
CreateUploadURL	Não	
DeleteBot	Compatível	BotId, BotAliasId
DeleteBotPseudônimo	Compatível	BotAliasIdentificação
DeleteBotChannel [somente permissão]	Compatível	BotId
DeleteBotLocalidade	Compatível	BotId
DeleteBotVersão	Compatível	BotId

Ação	Oferece suporte a políticas baseadas em recursos	Recurso
DeleteExport	Compatível	BotId
DeleteImport	Compatível	BotId
DeleteIntent	Compatível	BotId
DeleteResourcePolítica	Compatível	BotId, BotAliasId
DeleteSession	Compatível	BotAliasIdentificação
DeleteSlot	Compatível	BotId
DeleteSlotTipo	Compatível	BotId
DescribeBot	Compatível	BotId
DescribeBotPseudônimo	Compatível	BotAliasIdentificação
DescribeBotChannel [somente permissão]	Compatível	BotId
DescribeBotLocalidade	Compatível	BotId
DescribeBotVersão	Compatível	BotId
DescribeExport	Compatível	BotId
DescribeImport	Compatível	BotId
DescribeIntent	Compatível	BotId
DescribeResourcePolítica	Compatível	BotId, BotAliasId
DescribeSlot	Compatível	BotId
DescribeSlotTipo	Compatível	BotId
GetSession	Compatível	BotAliasIdentificação

Ação	Oferece suporte a políticas baseadas em recursos	Recurso
ListBotPseudônimos	Compatível	BotId
ListBotChannels [somente permissão]	Compatível	BotId
ListBotLocalidades	Compatível	BotId
ListBots	Não	
ListBotVersões	Compatível	BotId
ListBuiltInIntents	Não	
ListBuiltIntSlotTipos	Não	
ListExports	Não	
ListImports	Não	
ListIntents	Compatível	BotId
ListSlots	Compatível	BotId
ListSlotTipos	Compatível	BotId
PutSession	Compatível	BotAliasIdentificação
RecognizeText	Compatível	BotAliasIdentificação
RecognizeUtterance	Compatível	BotAliasIdentificação
StartConversation	Compatível	BotAliasIdentificação
StartImport	Compatível	BotId, BotAliasId
TagResource	Não	
UpdateBot	Compatível	BotId

Ação	Oferece suporte a políticas baseadas em recursos	Recurso
UpdateBotPseudônimo	Compatível	BotAliasIdentificação
UpdateBotLocalidade	Compatível	BotId
UpdateBotVersão	Compatível	BotId
UpdateExport	Compatível	BotId
UpdateIntent	Compatível	BotId
UpdateResourcePolítica	Compatível	BotId, BotAliasId
UpdateSlot	Compatível	BotId
UpdateSlotTipo	Compatível	BotId
UntagResource	Não	

Para saber como anexar uma política baseada em recurso a um bot ou alias de bot, consulte Exemplos de políticas baseadas em recursos para o Amazon Lex V2.

Exemplos de políticas baseadas em recursos no Amazon Lex V2

Para ver exemplos de políticas baseadas em recursos do Amazon Lex V2, consulte Exemplos de políticas baseadas em recursos para o Amazon Lex V2.

Ações de políticas para o Amazon Lex V2

Oferece compatibilidade com ações de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de

AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de ações do Amazon Lex V2, consulte Ações definidas pelo Amazon Lex V2 na Referência de autorização do serviço.

As ações de política no Amazon Lex V2 usam o seguinte prefixo antes da ação:

```
lex
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex V2, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex V2.

Recursos de políticas para o Amazon Lex V2

Oferece compatibilidade com recursos de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de atributos do Amazon Lex V2 e seus ARNs, consulte Tipos de atributos definidos pelo Amazon Lex V2 na Referência de autorização do serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Lex V2.

Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex V2, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex V2.

Chaves de condição de política para o Amazon Lex V2

Suporta chaves de condição de política específicas de serviço	Não

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um atributo somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos da política do IAM: variáveis e tags no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista de chaves de condição do Amazon Lex V2, consulte Chaves de condição do Amazon Lex V2 na Referência de autorização do serviço. Para saber com quais ações e atributos é possível usar a chave de condição, consulte Ações definidas pelo Amazon Lex V2.

Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex V2, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex V2.

Listas de controle de acesso (ACLs) no Amazon Lex V2

Oferece compatibilidade com ACLs	Não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Controle de acesso baseado em atributos (ABAC) com o Amazon Lex V2.

Oferece compatibilidade com ABAC (tags em políticas)	Sim

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a entidades do IAM (usuários ou funções) e a vários AWS recursos. A marcação de entidades e atributos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela estiver tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações onde o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do Usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

Usar credenciais temporárias com o Amazon Lex V2

Oferece compatibilidade com credenciais temporárias	Não

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS "Trabalhe com o IAM" no Guia do usuário do IAM.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar funções, consulte Alternar para um perfil (console) no Guia do usuário do IAM.

Você pode criar manualmente credenciais temporárias usando a AWS API AWS CLI ou. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para mais informações, consulte Credenciais de segurança temporárias no IAM.

Permissões de entidades principais entre serviços para o Amazon Lex V2

Suporte para o recurso Encaminhamento de sessões de acesso (FAS)	Sim

Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um

serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

Perfis de serviço para o Amazon Lex V2

Oferece compatibilidade com funções de serviço	Sim

O perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> A alteração das permissões de um perfil de serviço pode interromper a funcionalidade do Amazon Lex V2. Edite perfis de serviço somente quando o Amazon Lex V2 fornecer orientação para isso.

Perfis vinculados ao serviço para o Amazon Lex V2

Oferece suporte a perfis vinculados ao serviço	Parcial

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna

Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Exemplos de políticas baseadas em identidade para o Amazon Lex V2

Por padrão, usuários e perfis não têm permissão para criar ou modificar recursos do Amazon Lex V2. Eles também não podem realizar tarefas usando a AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) ou. Para conceder aos usuários permissão para executar ações nos recursos de que precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do Usuário do IAM.

Para obter detalhes sobre ações e tipos de recurso definidos pelo Amazon Lex V2, incluindo o formato dos ARNs para cada tipo de recurso, consulte Ações, recursos e chaves de condição do Amazon Lex V2 na Referência de autorização do serviço.

Tópicos

- Melhores práticas de política
- Usar o console do Amazon Lex V2
- Permitir que os usuários adicionem funções a um bot
- Permitir que os usuários adicionem canais a um bot
- Permitir que os usuários criem e atualizem bots
- Permitir que os usuários usem o Automated Chatbot Designer
- Permita que os usuários usem uma AWS KMS chave para criptografar e descriptografar arquivos
- Permitir que os usuários excluam bots
- Permitir que os usuários conversem com um bot
- Permitir que um usuário específico gerencie políticas baseadas em recursos
- Permitir que um usuário exporte bots e localidades de bots
- Permitir que um usuário exporte um vocabulário personalizado
- Permitir que um usuário importe bots e localidades de bots
- Permitir que um usuário importe um vocabulário personalizado
- Permitir que um usuário migre um bot do Amazon Lex para o Amazon Lex V2
- Permitir que usuários visualizem suas próprias permissões

- Permitir que um usuário desenhe o fluxo de conversa com o criador visual de conversas no Amazon Lex V2
- Permita que os usuários criem e visualizem réplicas de bots, mas não as excluam

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon Lex V2 em sua conta. Essas ações podem incorrer em custos para seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do Usuário do IAM.
- Aplique permissões de privilégio mínimo –- ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do Usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso –- você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode gravar uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte Elementos da política JSON do IAM: Condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais –- o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.

- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon Lex V2

Para acessar o console da Amazon Lex V2, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon Lex V2 em seu Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam a operação de API que estiverem tentando executar.

Para garantir que usuários e perfis ainda possam usar o console do Amazon Lex V2, os usuários precisam ter acesso ao Console. Para obter mais informações sobre como criar um usuário com acesso ao console, consulte Como criar um usuário do IAM em sua AWS conta no Guia do usuário do IAM.

Permitir que os usuários adicionem funções a um bot

Este exemplo mostra uma política que permite aos usuários do IAM adicionar permissões de consulta do Amazon Comprehend, da análise de sentimentos e do Amazon Kendra a um bot do 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*"
        }
    ]
}
```

Permitir que os usuários adicionem canais a um bot

Esse exemplo é uma política que permite que os usuários do IAM adicionem um canal de mensagens a um bot. O usuário deve ter essa política em vigor antes de poder implantar um bot em uma plataforma de mensagens.

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

Permitir que os usuários criem e atualizem bots

Este exemplo mostra um exemplo de política que permite que os usuários do IAM criem e atualizem qualquer bot. A política inclui permissões para concluir essa ação no console ou usando a AWS API AWS CLI ou.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateBot",
        "lex:UpdateBot".
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*]
    }
  ]
}
```

Permitir que os usuários usem o Automated Chatbot Designer

Este exemplo mostra um exemplo de política que permite que os usuários do IAM executem o 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>"
            ]
    ]
}
```

Permita que os usuários usem uma AWS KMS chave para criptografar e descriptografar arquivos

Este exemplo mostra um exemplo de política que permite que os usuários do IAM usem uma chave gerenciada pelo AWS KMS cliente para criptografar e descriptografar dados.

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

Permitir que os usuários excluam bots

Este exemplo mostra uma política que permite que os usuários do IAM excluam qualquer bot. A política inclui permissões para concluir essa ação no console ou usando a AWS API AWS CLI ou.

```
{
  "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/*"]
    }
  ]
}
```

Permitir que os usuários conversem com um bot

Este exemplo mostra uma política que permite que os usuários do IAM tenham uma conversa com qualquer bot. A política inclui permissões para concluir essa ação no console ou usando a AWS API AWS CLI ou.

```
{
  "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/*"
    }
  ]
}
```

Permitir que um usuário específico gerencie políticas baseadas em recursos

O exemplo a seguir concede permissão para um usuário específico gerenciar as políticas baseadas em recursos. Ele permite o acesso do console e da API às políticas associadas a bots e aliases de bots.

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

Permitir que um usuário exporte bots e localidades de bots

A política de permissão do IAM a seguir permite que um usuário crie, atualize e obtenha uma exportação para um bot ou uma localidade de bot.

```
{
    "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/*"]
        }
    ]
}
```

Permitir que um usuário exporte um vocabulário personalizado

A política de permissão do IAM a seguir permite que um usuário exporte um vocabulário personalizado de uma localidade de bot.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Permitir que um usuário importe bots e localidades de bots

A política de permissão do IAM a seguir permite que um usuário importe um bot ou a localidade do bot e verifique o status de uma importação.

```
{
    "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/*"
            ]
        }
    ]
}
```

Permitir que um usuário importe um vocabulário personalizado

A política de permissão do IAM a seguir permite que um usuário importe um vocabulário personalizado para uma localidade de 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/*"
            ]
        }
    ]
}
```

Permitir que um usuário migre um bot do Amazon Lex para o Amazon Lex V2

A política de permissão do IAM a seguir permite que um usuário comece a migrar um bot do Amazon Lex para o 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": "*"
        }
    ]
}
```

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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": "*"
        }
    ]
}
```

Permitir que um usuário desenhe o fluxo de conversa com o criador visual de conversas no Amazon Lex V2

A política de permissão do IAM a seguir permite que um usuário desenhe o fluxo de conversa com o criador visual de conversas no Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:UpdateIntent ",
                "lex:DescribeIntent "
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

Permita que os usuários criem e visualizem réplicas de bots, mas não as excluam

Você pode anexar as seguintes permissões a uma função do IAM para permitir que ela crie e visualize somente réplicas de bots. Ao omitir`lex:DeleteBotReplica`, você impede que a função exclua réplicas de bots. Para ter mais informações, consulte Permissões para replicar bots e gerenciar réplicas de bots.

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

Exemplos de políticas baseadas em recursos para o Amazon Lex V2

Uma política baseada em recursos é anexada a um recurso, como um bot ou um alias de bot. Com uma política baseada em recursos, você pode especificar quem tem acesso ao recurso e as ações que essas pessoas podem realizar nele. Por exemplo, você pode adicionar políticas baseadas em recursos que permitem que um usuário modifique um bot específico ou permita que um usuário use operações de runtime em um alias de bot específico.

Ao usar uma política baseada em recursos, você pode permitir que outros serviços da AWS acessem recursos em sua conta. Por exemplo, você pode permitir que o Amazon Connect acesse um bot do Amazon Lex.

Para saber como criar um bot ou alias de bot, consulte Compilar bots.

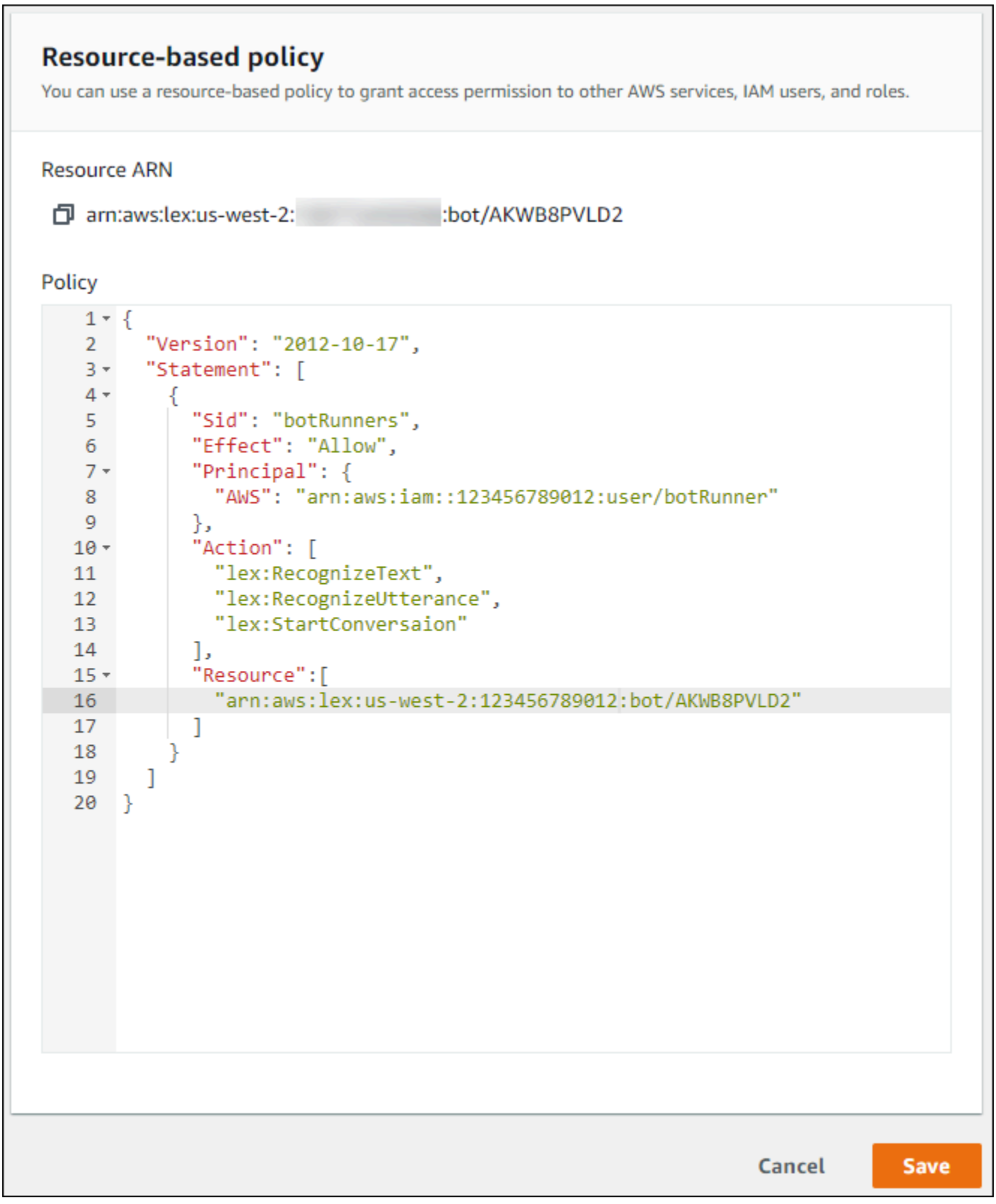
Tópicos

Use o console para especificar uma política baseada em recurso.

Você pode usar o console do Amazon Lex para gerenciar as políticas baseadas em recursos para seus bots e aliases de bot. Você insere a estrutura JSON de uma política e o console a associa ao recurso. Se já houver uma política associada a um recurso, você poderá usar o console para visualizar e modificar a política.

Quando você salva uma política com o editor de políticas, o console verifica a sintaxe da política. Se a política contiver erros, como um usuário inexistente ou uma ação que não seja permitida pelo recurso, ela retornará um erro e não salvará a política.

Veja a seguir o editor de políticas baseado em recursos para um bot no console. O editor de políticas para um alias de bot é semelhante.


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

Para abrir o editor de políticas para um bot

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de Bots, escolha o bot cuja política deseja editar.
3. Na seção Política baseada em recursos, escolha Editar.

Para abrir o editor de políticas para um alias de bot

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na lista de bots, escolha o bot que contém o alias que você deseja editar.
3. No menu à esquerda, selecione Aliases e, em seguida, escolha o alias que será editado.
4. Na seção Política baseada em recursos, escolha Editar.

Usar a API para especificar uma política baseada em recurso.

Você pode usar as operações de API para gerenciar as políticas baseadas em recursos para seus bots e aliases de bot. Existem operações para criar, atualizar e excluir políticas.

CreateResourcePolítica

Adiciona uma nova política de recursos com os enunciados de política definidas a um bot ou alias de bot.

CreateResourcePolicyStatement

Adiciona um novo enunciado de política de recursos de a um bot ou alias de bot.

DeleteResourcePolítica

Remove uma política de recursos de um bot ou alias de bot.

DeleteResourcePolicyStatement

Remove um enunciado de política de recursos de um bot ou alias de bot.

DescribeResourcePolítica

Obtém uma política de recursos e a revisão da política.

UpdateResourcePolítica

Substitui a política de recursos existente para um bot ou alias de bot por uma nova.

Exemplos

Java

O exemplo a seguir mostra como usar as operações de política baseada em recursos para gerenciar uma política baseada em recursos.

```
        /*
         * 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);
```

Permitir que um perfil do IAM atualize um bot e liste aliases de bots

O exemplo a seguir concede permissões para um perfil específico do IAM chamar operações de API de construção de modelos do Amazon Lex V2 para modificar um bot existente. O usuário pode listar aliases para um bot e atualizar o bot, mas não pode excluir o bot ou os aliases do 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"
            ]
        }
    ]
}
```

Permitir que um usuário converse com um bot

O exemplo a seguir concede permissão para que um usuário específico chame operações de API de runtime do Amazon Lex V2 em um único alias de um bot.

O usuário tem sua permissão especificamente negada para atualizar ou excluir o alias do 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"
            ]
        }
    ]
}
```

Permitir que um AWS serviço use um bot específico do Amazon Lex V2

O exemplo a seguir concede permissão para AWS Lambda que o Amazon Connect chame operações de API de tempo de execução do Amazon Lex V2.

O bloco de condições é necessário para entidades principais de serviço e deve usar as chaves de contexto globais `AWS:SourceAccount` e `AWS:SourceArn`.

O `AWS:SourceAccount` é o ID da conta que está chamando o bot do Amazon Lex V2.

O `AWS:SourceArn` é o ARN do recurso da instância de serviço do Amazon Connect ou da função do Lambda da qual a chamada para o alias de bot do Amazon Lex V2 se origina.

```
{
    "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 políticas gerenciadas para o Amazon Lex V2

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para obter mais informações, consulte Políticas gerenciadas pela AWS no Guia do usuário do IAM.

Política gerenciada pela AWS: AmazonLexReadOnly

É possível anexar a política `AmazonLexReadOnly` a suas identidades do IAM.

Essa política concede permissões somente de leitura que permitem aos usuários visualizar todas as ações no serviço de criação de modelos do Amazon Lex V2 e Amazon Lex.

Detalhes de permissão

Esta política inclui as seguintes permissões:

- `lex` – Acesso somente para leitura aos recursos do Amazon Lex V2 e do Amazon Lex no serviço de criação de modelos.

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

Política gerenciada pela AWS: AmazonLexRunBotsOnly

É possível anexar a política `AmazonLexRunBotsOnly` a suas identidades do IAM.

Essa política concede permissões somente de leitura que permitem o acesso para executar bots conversacionais do Amazon Lex V2 e do Amazon Lex. .

Detalhes de permissão

Esta política inclui as seguintes permissões:

- `lex` – Acesso somente para leitura a todas as ações no runtime do Amazon Lex V2 e do 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": "*"
        }
    ]
}
```

Política gerenciada pela AWS: AmazonLexFullAccess

É possível anexar a política `AmazonLexFullAccess` a suas identidades do IAM.

Essa política concede permissões administrativas que permitem ao usuário criar, ler, atualizar e excluir atributos do Amazon Lex V2 e do Amazon Lex e executar bots de conversação do Amazon Lex V2 e do Amazon Lex.

Detalhes de permissão

Esta política inclui as seguintes permissões:

- `lex` – Permite que as entidades principais tenham acesso de leitura e gravação a todas as ações nos serviços de runtime e construção de modelos do Amazon Lex V2 e do Amazon Lex.
- `cloudwatch`— Permite que os diretores visualizem CloudWatch métricas e alarmes da Amazon.
- `iam`: permite que os entidades principais criem e excluam funções vinculadas a serviços, passem funções e anexem e desanexem políticas a uma função. As permissões são restritas a “lex.amazonaws.com” para operações do Amazon Lex e a “lexv2.amazonaws.com” para operações do Amazon Lex V2.
- `kendra`: permite que as entidades principais listem índices do Amazon Kendra.
- `kms`: permite que as entidades principais descrevam chaves e aliases do AWS KMS .
- `lambda`: permite que as entidades principais listem funções AWS Lambda e gerenciem as permissões associadas a qualquer função do Lambda.
- `polly`: permite que as entidades principais descrevam vozes do Amazon Polly e sintetizem a fala.

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

Política gerenciada pela AWS: AmazonLexReplicationPolicy

Não é possível anexar `AmazonLexReplicationPolicy` às entidades do IAM. Essa política está vinculada a uma função vinculada ao serviço que permite que o Amazon Lex V2 execute ações em seu nome. Para ter mais informações, consulte Usar perfis vinculados ao serviço para o Amazon Lex V2.

Essa política concede permissões administrativas que permitem ao Amazon Lex V2 replicar AWS recursos entre regiões em seu nome. Você pode anexar essa política para permitir que uma função replique facilmente recursos, incluindo bots, localidades, versões, aliases, intenções, tipos de slots, slots e vocabulários personalizados.

Detalhes das permissões

Esta política inclui as seguintes permissões:

- `lex`— Permite que os diretores repliquem recursos em outras regiões.
- `iam`— Permite que os diretores passem funções do IAM. Isso é necessário para que o Amazon Lex V2 tenha permissões para replicar recursos em outras regiões.

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

Atualizações do Amazon Lex V2 para políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Amazon Lex V2 desde que esse serviço começou a monitorar essas alterações. Para obter alertas automáticos sobre alterações feitas nesta página, inscreva-se no feed RSS na página Histórico da documentação do Amazon Lex V2 do Amazon Lex V2.

Alteração	Descrição	Data
AmazonLexReadOnly: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir réplicas de acesso somente para leitura de recursos de bots.	10 de maio de 2024
AmazonLexFullAccess: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir a replicação de recursos de bots para outras regiões.	16 de abril de 2024
AmazonLexFullAccess: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir a replicação de recursos de bots para outras regiões.	31 de janeiro de 2024
AmazonLexReplicationPolicy – Nova política	O Amazon Lex V2 adicionou uma nova política para permitir a replicação de recursos de bots para outras regiões.	31 de janeiro de 2024
AmazonLexReadOnly: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir acesso somente	29 de novembro de 2022

Alteração	Descrição	Data
	leitura à lista de itens de vocabulário personalizados.	
AmazonLexFullAccess: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir acesso somente para leitura às operações do serviço de construção de modelos do Amazon Lex V2.	18 de agosto de 2021
AmazonLexReadOnly: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir acesso somente para leitura às operações de chatbot automatizado do Amazon Lex V2.	1º de dezembro de 2021
AmazonLexFullAccess: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir acesso somente para leitura às operações do serviço de construção de modelos do Amazon Lex V2.	18 de agosto de 2021
AmazonLexReadOnly: atualizar para uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir acesso somente para leitura às operações do serviço de construção de modelos do Amazon Lex V2.	18 de agosto de 2021

Alteração	Descrição	Data
AmazonLexRunBotsSomente — Atualização de uma política existente	O Amazon Lex V2 adicionou novas permissões para permitir acesso somente para leitura às operações do serviço de runtime do Amazon Lex V2.	18 de agosto de 2021
O Amazon Lex V2 passou a monitorar alterações	O Amazon Lex V2 passou a monitorar as alterações para as políticas gerenciadas pela AWS .	18 de agosto de 2021

Usar perfis vinculados ao serviço para o Amazon Lex V2

O Amazon Lex V2 usa funções AWS Identity and Access Management vinculadas a serviços (IAM). Um perfil vinculado ao serviço é um tipo especial de perfil do IAM vinculado diretamente ao Amazon Lex V2. As funções vinculadas ao serviço são predefinidas pelo Amazon Lex V2 e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Um perfil vinculado ao serviço facilita a configuração do Amazon Lex V2 porque você não precisa adicionar as permissões necessárias manualmente. O Amazon Lex V2 define as permissões dos perfis vinculados ao serviço e, a não ser que esteja definido de outra forma, somente o Amazon Lex V2 poderá assumir os perfis. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Para obter informações sobre outros serviços que oferecem suporte aos perfis vinculados ao serviço, consulte Serviços da AWS que funcionam com o IAM e procure os serviços com Sim na coluna Perfil vinculado ao serviço. Escolha um Sim com um link para exibir a documentação da função vinculada a serviço desse serviço.

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada a serviço. Para mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Você pode excluir uma função vinculada ao serviço somente após a primeira exclusão dos recursos relacionados. Isso protege seus recursos do Amazon Lex V2 porque você não pode remover inadvertidamente as permissões de acesso aos recursos.

Tópicos

Criar um perfil vinculado ao serviço para o Amazon Lex V2

Você não precisa criar manualmente uma função vinculada ao serviço, porque o Amazon Lex V2 cria a função vinculada ao serviço para você quando você executa a ação relevante (consulte Permissões de perfil vinculado ao serviço para o Amazon Lex V2 para obter mais informações) na AWS Management Console API, ou. AWS CLI AWS

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para criar um novo perfil em sua conta.

Editar um perfil vinculado ao serviço do Amazon Lex V2

O Amazon Lex V2 não permite que você edite funções vinculadas a serviços. Depois que você criar um perfil vinculado ao serviço, não poderá alterar o nome do perfil, pois várias entidades podem fazer referência ao perfil. No entanto, você pode editar a descrição de uma função usando o IAM. Para mais informações, consulte Editar uma função vinculada ao serviço no Guia do usuário do IAM.

Excluir um perfil vinculado ao serviço do Amazon Lex V2

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar os recursos de sua função vinculada ao serviço antes de excluí-la manualmente.

> **Note**
>
> Se o serviço do Amazon Lex V2 estiver usando o perfil quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para ver as etapas para excluir recursos para funções específicas vinculadas a serviços no Amazon Lex V2, consulte a seção específica da função em. Permissões de perfil vinculado ao serviço para o Amazon Lex V2

Para excluir manualmente uma função vinculada ao serviço usando o IAM

Depois de excluir recursos relacionados a uma função vinculada ao serviço, use o console do IAM AWS CLI, a ou a AWS API para excluir a função. Para mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Permissões de perfil vinculado ao serviço para o Amazon Lex V2

O Amazon Lex V2 usa funções vinculadas a serviços com os seguintes prefixos.

Tópicos

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

A função AWSServiceRoleForLexV2Bots _ dá permissões para conectar seu bot a outros serviços necessários. Essa função inclui uma política de confiança para permitir que o serviço lexv2.amazonaws.com assuma a função e inclui permissões para realizar as seguintes ações.

- Use o Amazon Polly para sintetizar a fala em todos os recursos do Amazon Lex V2 que a ação suporta.
- Se um bot estiver configurado para usar a análise de sentimentos do Amazon Comprehend, detecte o sentimento em todos os recursos do Amazon Lex V2 que a ação suporta.
- Se um bot estiver configurado para armazenar registros de áudio em um bucket do S3, coloque objetos em um bucket especificado.

- Se um bot estiver configurado para armazenar registros de áudio e texto, crie um fluxo de registros e coloque os registros em um grupo de registros especificado.
- Se um bot estiver configurado para usar uma AWS KMS chave para criptografar dados, gere uma chave de dados específica.
- Se um bot estiver configurado para usar a `KendraSearchIntent` intenção, consulte o acesso a um índice específico da Amazon Kendra.

Para criar a função

O Amazon Lex V2 cria uma nova função AWSServiceRoleForLexV2Bots _ com um sufixo aleatório na sua conta toda vez que você cria um bot. O Amazon Lex V2 modifica a função quando você adiciona recursos adicionais a um bot. Por exemplo, se você adicionar a análise de sentimentos do Amazon Comprehend a um bot, o Amazon Lex V2 adiciona permissão para a ação `lex:DetectSentiment` à função de serviço.

Para excluir a função

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. No painel de navegação esquerdo, selecione Bots e escolha o bot cuja função vinculada ao serviço você deseja excluir.
3. Selecione qualquer versão do bot.
4. A função de tempo de execução das permissões do IAM está nos detalhes da versão.
5. Volte para a página Bots e escolha o botão de rádio ao lado do bot para excluir.
6. Selecione Ação e, em seguida, escolha Excluir.
7. Siga as etapas em Excluir uma função vinculada ao serviço para excluir a função do IAM.

AWSServiceRoleForLexV2Channels_

A função AWSServiceRoleForLexV2Channels _ dá permissão para listar bots em uma conta e chamar APIs de conversação para um bot. Essa função inclui uma política de confiança para permitir que o serviço channels.lexv2.amazonaws.com assuma a função. Se um bot estiver configurado para usar um canal para se comunicar com um serviço de mensagens, a política de permissões AWSServiceRoleForLexV2Channels _ role permite que o Amazon Lex V2 conclua as seguintes ações.

- Liste as permissões em todos os bots em uma conta.
- Reconheça texto, obtenha sessão e coloque permissões de sessão em um alias de bot especificado.

Para criar a função

Quando você cria uma integração de canais para implantar um bot em uma plataforma de mensagens, o Amazon Lex V2 cria uma nova função vinculada ao serviço em sua conta para cada canal com um sufixo aleatório.

Para excluir a função

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. No painel de navegação esquerdo, selecione Bots.
3. Escolha um bot.
4. No painel de navegação esquerdo, escolha Integrações de canais em Implantações.
5. Selecione um canal cuja função vinculada ao serviço você deseja excluir.
6. A função de tempo de execução de permissões do IAM está na configuração geral.
7. Escolha Excluir e, em seguida, escolha Excluir novamente para excluir o canal.
8. Siga as etapas em Excluir uma função vinculada ao serviço para excluir a função do IAM.

AWSServiceRoleForLexV2Replication

A AWSServiceRoleForLexV2Replication função dá permissão para replicar bots em uma segunda região. Essa função inclui uma política de confiança para permitir que o serviço replication.lexv2.amazonaws.com assuma a função e também inclui a política AmazonLexReplicationPolicy AWS gerenciada, que permite permissões para as seguintes ações.

- Passe as funções do IAM do bot para o bot de réplica para duplicar as permissões apropriadas para o bot de réplica.
- Crie e gerencie bots e recursos de bots (versões, aliases, intenções, slots, vocabulários personalizados etc.) em outras regiões.

Para criar a função

Quando você ativa a resiliência global para um bot, o Amazon Lex V2 cria a função AWSServiceRoleForLexV2Replication vinculada ao serviço em sua conta. Certifique-se de ter as permissões corretas para conceder ao serviço Amazon Lex V2 permissões para criar a função vinculada ao serviço.

Para excluir os recursos do Amazon Lex V2 usados pelo AWSServiceRoleForLexV2Replication para que você possa excluir a função

1. Faça login AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Escolha um bot para o qual a Resiliência Global esteja ativada.
3. Selecione Resiliência global em Implantação.
4. Selecione Desativar resiliência global.
5. Repita o processo para todos os bots que têm a Resiliência Global ativada.
6. Siga as etapas em Excluir uma função vinculada ao serviço para excluir a função do IAM.

Regiões com suporte a perfis vinculados ao serviço do Amazon Lex V2.

O Amazon Lex V2 é compatível com perfis vinculados ao serviço em todas as regiões em que o serviço está disponível. Para mais informações, consulte Regiões e endpoints da AWS.

Solução de problemas de identidade e acesso da Amazon Lex V2

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon Lex V2 e o IAM.

Tópicos

- Não tenho autorização para executar uma ação no Amazon Lex V2
- Não estou autorizado a realizar iam: PassRole
- Sou administrador e quero permitir que outras pessoas tenham acesso ao Amazon Lex V2
- Conceder acesso programático a um usuário
- Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Amazon Lex V2

Não tenho autorização para executar uma ação no Amazon Lex V2

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `lex:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`my-example-widget`* usando a ação `lex:`*`GetWidget`*.

Não estou autorizado a realizar iam: PassRole

Caso receba uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, as políticas deverão ser atualizadas para permitir a transmissão de um perfil ao Amazon Lex V2.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon Lex V2. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Sou administrador e quero permitir que outras pessoas tenham acesso ao Amazon Lex V2

Para permitir que outras pessoas acessem o Amazon Lex V2, é necessário criar uma entidade do IAM (usuário ou perfil) para a pessoa ou o aplicativo que precisa do acesso. Elas usarão as credenciais dessa entidade para acessar a AWS. Você deve anexar uma política à entidade que concede a eles as permissões corretas na Amazon Lex V2.

Para começar a usar imediatamente, consulte Criar os primeiros usuário e grupo delegados pelo IAM no Guia do usuário do IAM.

Conceder acesso programático a um usuário

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporári as para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferrament as e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.

Qual usuário precisa de acesso programático?	Para	Por
IAM	Use credenciais temporárias para assinar solicitações programáticas para AWS SDKs ou APIs. AWS CLI AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Amazon Lex V2

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de

controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon Lex V2 é compatível com esses atributos, consulte Como o Amazon Lex V2 funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Registrar em log e monitorar no Amazon Lex V2

O monitoramento é uma parte importante da manutenção da confiabilidade, da disponibilidade e do desempenho do Amazon Lex V2 e de outras soluções da AWS. A AWS fornece as seguintes ferramentas de monitoramento para assistir o Amazon Lex V2, informar quando algo está errado e realizar ações automáticas quando apropriado:

- A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. É possível coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode CloudWatch rastrear o uso da CPU ou outras métricas de suas instâncias do Amazon EC2 e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.
- AWS CloudTrailcaptura chamadas de API e eventos relacionados feitos por ou em nome de sua AWS conta e entrega os arquivos de log para um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas ligaram AWS, o endereço IP de origem a partir do qual as chamadas foram feitas e quando elas ocorreram. Para mais informações, consulte o Guia do usuário do AWS CloudTrail.

Validação de conformidade para o Amazon Lex V2

Auditores terceirizados avaliam a segurança e a conformidade do Amazon Lex V2 como parte de vários programas de AWS conformidade. O Amazon Lex V2 é um serviço qualificado pela HIPAA. Ele é compatível com PCI, SOC e ISO.

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

> **Note**
>
> Nem todos Serviços da AWS são elegíveis para a HIPAA. Para mais informações, consulte a Referência dos serviços qualificados pela HIPAA.

- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).

- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon Lex V2

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenters tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, o Amazon Lex V2 oferece vários recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

> **Note**
>
> Para obter mais informações sobre resiliência global no Amazon Lex V2, que permite criar um bot replicado em uma segunda região em pares predeterminados, consulte Resiliência global.

Segurança da infraestrutura no Amazon Lex V2

Como um serviço gerenciado, o Amazon Lex V2 é protegido pelos AWS procedimentos de segurança de rede global que estão descritos no whitepaper Amazon Web Services: Visão geral dos processos de segurança.

Você usa chamadas de API AWS publicadas para acessar o Amazon Lex V2 pela rede. Os clientes devem oferecer compatibilidade com Transport Layer Security (TLS) 1.0 ou posterior. Recomendamos TLS 1.2 ou posterior. Os clientes também devem ter compatibilidade com conjuntos de criptografia com perfect forward secrecy (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece compatibilidade com esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Amazon Lex V2 e endpoints da VPC de interface (AWS PrivateLink)

É possível estabelecer uma conexão privada entre a VPC e o Amazon Lex V2 criando um endpoint da VPC de interface. Os endpoints de interface são alimentados por AWS PrivateLinkuma tecnologia que permite acessar de forma privada as APIs do Amazon Lex V2 sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão Direct Connect. AWS As instâncias na VPC não precisam de endereços IP públicos para se comunicar com as APIs do Amazon Lex V2. O tráfego de rede entre a VPC e o Amazon Lex V2 não deixa a rede da Amazon.

Cada endpoint de interface é representado por uma ou mais Interfaces de Rede Elástica nas sub-redes.

Para obter mais informações, consulte Interface VPC endpoints (AWS PrivateLink) no Guia do usuário da Amazon VPC.

Considerações sobre endpoints da VPC do Amazon Lex V2

Antes de configurar um endpoint da VPC de interface para o Amazon Lex V2, revise as Propriedades e limitações do endpoint de interface no Guia do usuário do Amazon VPC.

O Amazon Lex V2 é compatível com chamadas para todas as ações de API da VPC.

Criar um endpoint da VPC de interface para a API do Amazon Lex V2

Você pode criar um VPC endpoint para o serviço Amazon Lex V2 usando o console Amazon VPC ou o (). AWS Command Line Interface AWS CLI Para mais informações, consulte Criar um endpoint de interface no Guia do usuário da Amazon VPC.

Crie um endpoint da VPC para o Amazon Lex V2 usando o seguinte nome de serviço:

- com.amazonaws.*região*.models-v2-lex
- com.amazonaws.*região*.runtime-v2-lex

Se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API para o Amazon Lex V2 usando seu nome DNS padrão para a região, por exemplo, `runtime-v2-lex.us-east-1.amazonaws.com`.

Para mais informações, consulte Acessar um serviço por um endpoint de interface no Guia do usuário da Amazon VPC.

Criar uma política de endpoint da VPC para o Amazon Lex V2

É possível anexar uma política de endpoint ao endpoint da VPC que controla o acesso ao Amazon Lex V2. Essa política especifica as seguintes informações:

- A entidade principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos sobre os quais as ações podem ser realizadas.

Para mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Guia do usuário da Amazon VPC.

Exemplo: política de endpoint da VPC para ações do Amazon Lex V2

Veja a seguir um exemplo de política de endpoint para o Amazon Lex V2. Quando anexada a um endpoint, essa política concede acesso às ações listadas do Amazon Lex V2 para todas as entidades principais em todos os recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lex:RecognizeText",
            "lex:RecognizeUtterance",
            "lex:StartConversation",
            "lex:DeleteSession",
            "lex:GetSession",
            "lex:DeleteSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Diretrizes e práticas recomendadas

Consulte as diretrizes e as práticas recomendadas a seguir para otimizar o comportamento e as interações do seu bot com os clientes.

Assinatura de solicitações

Todas as solicitações de runtime e construção de modelos do Amazon Lex V2 na Referência da API usam a assinatura V4 para autenticar as solicitações. Para mais informações sobre a autenticação de solicitações, consulte Processo de cadastramento do Signature versão 4 na Referência geral da AWS.

Proteção das informações confidenciais

As operações da API runtime RecognizeText e RecognizeUtterance usam uma ID de sessão como parâmetro obrigatório. Os desenvolvedores podem configurar isso para qualquer valor que atenda às restrições descritas na API. Recomendamos que você não use esse parâmetro para enviar informações confidenciais como logins de usuário, e-mails ou números de seguro social. Esse ID é usado principalmente para identificar de forma exclusiva uma conversa com um bot.

Como capturar valores de slots a partir de declarações dos usuários

O Amazon Lex V2 usa os valores de enumeração fornecidos em uma definição de tipo de slot para treinar seus modelos de machine learning. Suponha que você defina uma intenção chamada `GetPredictionIntent` com o seguinte utterance de amostra:

```
"Tell me the prediction for {sign}"
```

em que {sign} é um slot com o tipo personalizado de `ZodiacSign` que tem 12 valores de enumeração: `Aries` até `Pisces`. Agora, suponha que o usuário diga “Diga-me a previsão para terra”:

- O Amazon Lex V2 infere que “terra” é um valor de ZodiacSign se você realizar uma das ações a seguir:
  - Definir o campo `valueSelectionStrategy` como`ORIGINAL_VALUE` usando a operação CreateSlotType
  - Selecionar Expandir valores no console
- O Amazon Lex V2 não reconhece o valor “terra” se você limitar o reconhecimento aos valores definidos para o tipo de slot executando uma das seguintes ações:

- Definir o campo `valueSelectionStrategy` como`TOP_RESOLUTION` usando a operação `CreateSlotType`
- Selecionar Restringir a valores de slot e sinônimos no console

Quando você define sinônimos para valores de slot, eles são reconhecidos como sendo iguais a um valor de slot. No entanto, o valor do slot é retornado em vez do sinônimo.

Como o Amazon Lex V2 passa esse valor para seu aplicativo cliente ou para a função do Lambda, você deve verificar se os valores dos slots são válidos antes de usá-los na atividade de atendimento.

Quando o Amazon Lex chama uma função do Lambda ou traz o resultado de uma interação de fala com o cliente, a capitalização dos valores do slot não é garantida. Em interações de texto, a capitalização dos valores de slot corresponde ao texto inserido ou ao valor de slot, dependendo do valor do campo `valueResolutionStrategy`.

Siglas nos valores dos slots

Quando for definir valores de slot que contêm siglas, use os seguintes padrões:

- Letras maiúsculas separadas por pontos (D.V.D.)
- Letras maiúsculas separadas por espaços (D V D)

Slots integrados para data e hora

Os tipos de slot integrado AMAZON.Date e AMAZON.Time capturam as datas e horas absolutas e relativas. As datas e horários relativos são resolvidos na hora e na data em que o Amazon Lex V2 recebe a solicitação e na região em que processa a solicitação.

Para o tipo de slot integrado `AMAZON.Time`, se o usuário não especificar que um horário é antes ou depois do meio-dia, o horário será ambíguo. Nesse caso, o Amazon Lex V2 fará uma nova solicitação ao usuário. Recomendamos que um horário absoluto seja escolhido para as solicitações. Por exemplo, use uma solicitação como "Em qual horário você deseja que a pizza seja entregue? Você pode dizer 18h ou 6 da tarde".

Como evitar ambiguidades nos dados de treinamento do bot

Fornecer dados de treinamento confusos no bot reduz a capacidade de o Amazon Lex V2 compreender a entrada do usuário. Suponha que você tenha dois intents (`OrderPizza` e `OrderDrink`) no bot e inclua “Quero fazer o pedido” como exemplo de declaração. Quando você

cria seu bot, o Amazon Lex V2 não consegue mapear essa expressão a um intent específico. Como resultado, quando um usuário insere esse enunciado em runtime, o Amazon Lex V2 não consegue escolher uma intenção com alto grau de confiança.

Se você tiver dois intents com o mesmo exemplo de declaração, use contextos de entrada para ajudar o Amazon Lex V2 a distinguir entre os dois intents no runtime. Para mais informações, consulte Configurar contexto de intenção.

Como usar o alias TSTALIASID

- O alias TSTALIASID do bot aponta para a versão de rascunho e deve ser usada somente para testes manuais. O Amazon Lex limita o número de solicitações de runtime que você pode fazer para o alias TSTALIASID do bot.
- Quando você atualiza a versão de rascunho do bot, o Amazon Lex encerra todas as conversas em andamento para qualquer aplicativo cliente usando o alias TSTALIASID do bot. Em geral, você não deve usar o alias TSTALIASID de um bot em produção, pois a versão de rascunho pode ser atualizada. Em vez disso, publique e use uma versão e um alias.
- Quando você atualiza um alias, o Amazon Lex leva alguns minutos para incorporar as alterações. Quando você modifica a versão de rascunho do bot, a alteração é capturada imediatamente pelo alias TSTALIASID.

Cotas

As cotas de serviço, também chamadas de limites, são o número máximo de recursos de serviço permitidos para sua AWS conta. Para mais informações, consulte AWS Service Quotas, na Referência geral da AWS .

Algumas cotas de serviço podem ser ajustadas ou aumentadas. Consulte a coluna Ajustável nas tabelas a seguir para ver se uma cota pode ser ajustada e a coluna Autoatendimento para ver se você pode solicitar um ajuste de cota por meio do console Service Quotas. Entre em contato AWS Support para aumentar uma cota ajustável, mas não por meio de autoatendimento. Pode demorar alguns dias para aumentar a cota de serviços. Se você estiver aumentando sua cota como parte de um projeto maior, não se esqueça de adicionar esse tempo ao seu plano.

> **Note**
>
> Os limites de caracteres são calculados como o número de unidades de código Unicode. Na maioria dos casos, um caractere Unicode é equivalente a uma unidade de código Unicode. Alguns caracteres especiais podem ser maiores que uma unidade e as contagens podem ser diferentes para codificações diferentes. Para obter mais informações sobre como calcular o tamanho da string, consulte esta documentação.

Cotas de tempo de construção

As cotas máximas a seguir são aplicadas quando você está criando um bot.

Descrição	Padrão	Ajustável	Autoatendimento
Bots por AWS conta	100	Sim	Sim
Associações de canais de bots por conta da AWS	5.000	Não	N/D
Bots por rede de bots	5	Não	N/D
Redes de bots por bot	25	Não	N/D

Descrição	Padrão	Ajustável	Autoatendimento
Versões por bot	100	Não	N/D
Intenções por localidade em cada bot	• 1 mil em en-AU, en-GB e en-US • 250 em todas as outras localidades	Sim	Não
Slots por localidade em cada bot	• 4 mil em en-AU, en-GB e en-US • 2 mil em todas as outras localidades	Não	N/D
Tipos de slots personalizados por localidade de bot	• 250 em en-AU, en-GB e en-US • 100 em todas as outras localidades	Não	N/D
Valores e sinônimos de tipo de slot personalizados por localidade em cada bot	50.000	Não	N/D
Total de caracteres em exemplos de enunciados por localidade em cada bot	• 2 milhões em en-AU, en-GB e en-US • 200 mil em todas as outras localidades	Não	N/D
Associações de canais por alias de bot	10	Não	N/D
Slots por intenção	100	Não	N/D

Descrição	Padrão	Ajustável	Autoatendimento
Exemplos de enunciados por intenção	1.500	Sim	Sim
Caracteres por amostra de enunciado	500	Não	N/D
Tamanho do texto de resposta	4.000	Não	N/D
Amostra de enunciados por slot	10	Sim	Sim
Caracteres por amostra de enunciados por slot	500	Não	N/D
Solicitações por slot	30	Não	N/D
Valores e sinônimos por tipo de slot personalizado	10.000	Não	N/D
Caracteres por valor do tipo de slot personalizado	500	Não	N/D
Caracteres em um nome de associação de canal	100	Não	N/D
Número de trabalhos simultâneos de análise do Automated Chatbot Designer em todos os bots da sua conta por região	10	Não	N/D

Descrição	Padrão	Ajustável	Autoatendimento
Tamanho do arquivo XML do tipo de slot de gramática personali zada	100 KB	Não	N/D

Cotas de runtime

As cotas máximas a seguir são aplicadas em runtime.

Descrição	Padrão	Ajustável	Autoatendimento
Tamanho do texto de entrada para RecognizeTexte RecognizeUtterance	1024 caracteres	Não	N/D
Duração da entrada de fala para a operação `RecognizeUtterance`	15 segundos	Sim	Não
Tamanho dos cabeçalhos `RecognizeUtterance`	16 KB	Não	N/D
Tamanho dos cabeçalhos combinados de solicitação e sessão para `RecognizeUtterance`	12 KB	Não	N/D
Número máximo de conversas simultâne	2	Não	N/D

Descrição	Padrão	Ajustável	Autoatendimento
as em modo de texto para `RecognizeText` `RecognizeUtterance` , ou para o `StartConversation` TestBotAlias			
Número máximo de conversas simultâneas em modo de texto para `RecognizeText` , `RecognizeUtterance` ou `StartConversation` para outros aliases	50	Sim	Não
Número máximo de conversas simultâneas no modo de voz para o `RecognizeUtterance` TestBotAlias	2	Não	N/D
Número máximo de conversas simultâneas no modo de voz para `RecognizeUtterance` para outros aliases	125	Sim	Não

Descrição	Padrão	Ajustável	Autoatendimento
Número máximo de conversas simultâneas no modo de voz para o `StartConversation` TestBotAlias	2	Não	N/D
Número máximo de conversas simultâneas no modo de voz para `StartConversation` para outros aliases	200	Sim	Não
Número máximo de operações simultâneas de gerenciamento de sessões (`PutSession` `GetSession`, ou`DeleteSession`) ao usar o TestBotAlias	2	Não	N/D
Número máximo de operações simultâneas de gerenciamento de sessões (`PutSession`, `GetSession` ou `DeleteSession`) ao usar outros aliases	50	Sim	Não

Descrição	Padrão	Ajustável	Autoatendimento
O tamanho máximo de entrada para uma função do Lambda	12 KB	Não	N/D
Tamanho máximo de saída de uma função do Lambda	50 KB	Não	N/D
Tamanho máximo dos atributos da sessão na saída da função do Lambda (após a codificação em base 64)	12 KB	Não	N/D
Tempo limite máximo de uma função Lambda	30 segundos	Sim	Não

Guia de migração do Amazon Lex V1 para a V2

O console e as APIs do Amazon Lex V2 facilitam a criação e o gerenciamento de bots. Use este guia para saber mais sobre as melhorias na API do Amazon Lex V2 à medida que você migra bots.

Você migra um bot usando o console ou a API do Amazon Lex. Para obter mais informações, consulte Migrating a bot no Guia do desenvolvedor do Amazon Lex.

Visão geral do Amazon Lex V2

Vários idiomas podem ser adicionados a um bot para que você possa gerenciá-los como um único recurso. Uma arquitetura de informações simplificada permite que você gerencie com eficiência suas versões de bots. Recursos como “fluxo de conversa”, gravação parcial da configuração do bot e upload em massa de enunciados oferecem mais flexibilidade.

Vários idiomas em um bot

Você pode adicionar vários idiomas com a API do Amazon Lex V2. Você adiciona, modifica e cria cada linguagem de forma independente. Recursos como tipos de slots são definidos com escopo no nível do idioma. Você pode alternar rapidamente entre diferentes idiomas para comparar e refinar as conversas. Você pode usar um painel no console para revisar os enunciados em todos os idiomas para acelerar a análise e as iterações. Um operador de bot pode gerenciar permissões e operações de registro em log para todos os idiomas com uma configuração de bot. Você deve fornecer uma linguagem como parâmetro de runtime para conversar com um bot do Amazon Lex V2. Para mais informações, consulte Idiomas e locais aceitos pelo Amazon Lex V2.

Arquitetura de informações simplificada

A API do Amazon Lex V2 segue uma arquitetura de informação simplificada (IA) com intenção e tipos de slots definidos com escopo para uma linguagem. Você cria versões no nível do bot para que recursos como intenções e tipos de slots não sejam versionados individualmente. Por padrão, um bot é criado com uma versão de rascunho que é mutável e usada para testar alterações. Você pode criar snapshots numerados a partir da versão de rascunho. Você escolhe os idiomas a serem incluídos em uma versão. Todos os recursos do bot (idiomas, intenções, tipos de slots) são arquivados como parte da criação de uma versão do bot. Para mais informações, consulte Versões.

Aumento da produtividade do construtor

Você tem ferramentas e recursos adicionais de produtividade do construtor que oferecem mais flexibilidade e controle do processo de design do seu bot.

Salvar configuração parcial

A API do Amazon Lex V2 permite que você salve alterações parciais durante o desenvolvimento. Por exemplo, você pode salvar um slot que faça referência a um tipo de slot excluído. Essa flexibilidade permite que você salve seu trabalho e retorne a ele mais tarde. Você pode resolver essas alterações antes de criar o bot. No Amazon Lex V2, a gravação parcial pode ser aplicada a slots, versões e aliases.

Renomear recursos

Com o Amazon Lex V2, você pode renomear um recurso após sua criação. Use um nome de recurso para associar metadados fáceis de usar a cada recurso. A API do Amazon Lex V2 atribui a cada recurso um ID de recurso exclusivo de 10 caracteres. Todos os recursos têm um nome de recurso. Você pode renomear os seguintes recursos:

- Bot
- Intenção
- Tipo de slot
- Slot
- Alias

É possível usar IDs de recursos para ler e modificar seus recursos. Se você estiver usando o AWS Command Line Interface ou a API do Amazon Lex V2 para trabalhar com o Amazon Lex V2, os IDs dos recursos serão necessários para determinados comandos.

Gerenciamento simplificado das funções do Lambda

Na API do Amazon Lex V2, você define uma função do Lambda por linguagem, em vez de uma função para cada intenção. A função do Lambda é configurada no alias da linguagem e é usada tanto para a caixa de diálogo quanto para o gancho do código de atendimento. Você ainda pode optar por habilitar ou desabilitar a caixa de diálogo e os ganchos do código de atendimento de forma independente para cada intenção. Para mais informações, consulte Habilitando a lógica personalizada com as funções do AWS Lambda.

Configurações granulares

A API do Amazon Lex V2 move o limite da pontuação de confiança da classificação de voz e intenção do bot para o escopo da linguagem. O sinalizador de análise de sentimentos passa do escopo do bot para o escopo do alias. O tempo limite da sessão e as configurações de privacidade no escopo do bot e os logs de conversas no escopo do alias permanecem inalterados.

Intenção de fallback padrão

A API Amazon Lex V2 adiciona uma intenção de fallback padrão quando você cria uma linguagem. Use-a para configurar o tratamento de erros para seu bot, em vez de solicitações específicas de tratamento de erros.

Atualização otimizada da variável de sessão

Com a API do Amazon Lex V2, você pode atualizar o estado da sessão diretamente com as operações RecognizeText e RecognizeUtterance sem depender das APIs da sessão.

Criar recursos do Amazon Lex V2 com o AWS CloudFormation

O Amazon Lex V2 é integrado à AWS CloudFormation, um serviço que ajuda você a modelar e configurar seus recursos da AWS, para que você possa passar menos tempo criando e gerenciando seus recursos e sua infraestrutura. Você cria um modelo que descreve todos os recursos da AWS desejados (como chatbots do Amazon Lex V2), e o AWS CloudFormation provisiona e configura esses recursos para você.

Quando você usa a AWS CloudFormation, é possível reutilizar o modelo para configurar os recursos do Amazon Lex V2 repetidamente e de forma consistente. Descreva seus recursos uma vez e depois provisione os mesmos recursos repetidamente em várias regiões e Contas da AWS.

Amazon Lex V2 e modelos da AWS CloudFormation

Para provisionar e configurar recursos para o Amazon Lex V2 e serviços relacionados, é preciso entender os modelos da AWS CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar nas suas pilhas do AWS CloudFormation. Se você não estiver familiarizado com JSON ou YAML, poderá usar o AWS CloudFormation Designer para ajudá-lo a começar a usar os modelos do AWS CloudFormation. Para obter mais informações, consulte O que é o AWS CloudFormation Designer? no Manual do usuário da AWS CloudFormation.

O Amazon Lex V2 oferece suporte à criação dos seguintes recursos em AWS CloudFormation:

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

Para obter mais informações, incluindo exemplos de modelos JSON e YAML para esses recursos, consulte Referência de tipo de recurso do Amazon Lex V2 no Guia do usuário da AWS CloudFormation.

Saiba mais sobre a AWS CloudFormation

Para saber mais sobre a AWS CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- Manual do usuário da AWS CloudFormation
- Referência da API da AWS CloudFormation
- Guia do usuário da interface de linha de comando da AWS CloudFormation

Histórico da documentação do Amazon Lex V2

- Última atualização da documentação: 10 de maio de 2024

A tabela a seguir descreve as alterações importantes em cada versão do Amazon Lex V2. Para receber notificações sobre atualizações dessa documentação, você poderá se inscrever em um feed RSS.

Alteração	Descrição	Data
Atualização da política AWS gerenciada	O Amazon Lex V2 adicionou novas permissões à política AmazonLex ReadOnlygerenciada para permitir acesso de leitura a recursos de bots que foram replicados em outras regiões.	10 de maio de 2024
Atualização da política AWS gerenciada	O Amazon Lex V2 adicionou novas permissões à política AmazonLexFullAcces sgerenciada para permitir a atualização de recursos de bots replicados para outras regiões.	15 de abril de 2024
Expansão de região	O Amazon Lex V2 agora está disponível em AWS GovCloud (Oeste dos EUA) (us-gov-w est-1).	22 de março de 2024
Atualização da política AWS gerenciada	O Amazon Lex V2 adicionou novas permissões à política AmazonLexReplicati onPolicygerenciada para permitir a atualização de	7 de março de 2024

	recursos de bots replicados para outras regiões.	
Nova função	Você pode usar a função fn.length () para determinar o tamanho do valor de um valor de string no Amazon Lex V2. Para obter mais informaçõ es, consulte Ramificação condicional - Funções.	4 de março de 2024
Atualização para o recurso	O slot QnA integrado para recursos generativos de IA no Amazon Lex V2 agora está disponível em disponibi lidade geral. Para obter mais informações, consulte Otimizar a criação de bots e o desempenho com IA generativ a.	28 de fevereiro de 2024
Atualização da política AWS gerenciada	O Amazon Lex V2 adicionou novas permissões à política AmazonLexReplicati onPolicygerenciada para permitir a atualização de recursos de bots replicados para outras regiões.	28 de fevereiro de 2024
Atualização da política AWS gerenciada	O Amazon Lex V2 adicionou novas permissões à política AmazonLexFullAcces sgerenciada para permitir a replicação de recursos de bots em outras regiões.	8 de fevereiro de 2024

Nova política gerenciada	O Amazon Lex V2 adicionou uma política gerenciada que fornece permissões para replicar recursos de bots em outras regiões. Para obter mais informações, consulte AmazonLexReplicationPolicy.	8 de fevereiro de 2024
Novo atributo	Você pode usar a resiliência global para replicar seu bot em uma segunda região da AWS no Amazon Lex V2. Para obter mais informações, consulte Resiliência global.	8 de fevereiro de 2024
Novo atributo	Agora você pode aproveitar os recursos de IA generativa no Amazon Lex V2. Para obter mais informações, consulte Otimizar a criação de bots e o desempenho com IA generativa.	29 de novembro de 2023
Novo atributo	O Amazon Lex V2 agora pode usar o registro seletivo para capturar texto e/ou áudio no nível do intent ou do slot. Para mais informações, consulte Registro em log seletivo.	8 de novembro de 2023

Novo atributo	O Amazon Lex V2 agora usa um slot integrado para determinar respostas Sim/Não/Talvez/Não sei usando `AMAZON.Confirmation`. Para mais informações, consulte Tipos de slots integrados.	17 de agosto de 2023
Novo atributo	Você pode visualizar as métricas de performance de intents e slots, além de outras métricas de conversação, usando o painel de análises. Para mais informações, consulte Analytics.	18 de julho de 2023
Novo atributo	Você pode melhorar a precisão e o sucesso de atendimento do seu bot com o Test Workbench. Para mais informações, consulte Test Workbench.	6 de junho de 2023
Novo atributo	Agora você pode criar bots a partir de um modelo para alguns setores de negócios populares. Para mais informações, consulte Modelos de bots.	23 de fevereiro de 2023
Novo atributo	Agora você pode combinar vários bots em uma rede de bots para oferecer aos clientes uma experiência integrada. Para mais informações, consulte Rede de bots.	9 de fevereiro de 2023

Novo atributo	O Amazon Lex V2 agora oferece suporte às regiões do Golfo Pérsico (Emirados Árabes Unidos) e aos idiomas cantonês (Hong Kong), finlandês (Finlândia), norueguês (Noruega), polonês (Polônia) e sueco (Suécia). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	6 de dezembro de 2022
Atualizado para a política AWS gerenciada	O Amazon Lex V2 adicionou novas permissões à política AmazonLex ReadOnlygerenciada para permitir a exibição de itens de vocabulário personalizados.	29 de novembro de 2022
Novo atributo	O Amazon Lex V2 exibe uma representação alternativa para uma frase ou palavra usando o console ou as APIs para personalizar a saída do recurso de fala em texto. Para mais informações, consulte Criar um vocabulário personalizado para reconhecimento de fala.	7 de novembro de 2022

Novo atributo	O Amazon Lex V2 pode adicionar um atributo de ponderação a um elemento do item que represent ará o quando a frase será aumentada durante o reconhecimento de fala. Para mais informações, consulte Ponderações gramaticais.	28 de outubro de 2022
Novo atributo	O Amazon Lex V2 pode ser usado para capturar entradas de formato livre do usuário final compostas por palavras ou caracteres usando `AMAZON.FreeFormInput` . Para mais informaçõ es, consulte Tipos de slots integrados.	19 de outubro de 2022
Novo atributo	O Amazon Lex V2 exibe uma representação alternativa para uma frase ou palavra na saída do recurso de fala em texto final. Para mais informações, consulte Criar um vocabulário personalizado para reconheci mento de fala.	19 de outubro de 2022
Novo atributo	O Amazon Lex V2 agora é compatível com Hindi (Índia) e Holandês (Holanda). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	14 de outubro de 2022

Novo atributo	Houve uma atualização na forma como o Amazon Lex V2 gerencia a entrada do usuário. Agora você pode escolher se o Amazon Lex V2 aceita entrada de texto, áudio ou DTMF em qualquer ponto do fluxo de conversação. Para mais informações, consulte Atributos configuráveis.	22 de setembro de 2022
Novo atributo	Houve uma atualização na forma como o Amazon Lex V2 gerencia os fluxos de conversação. O Visual Conversation Builder é um criador de conversas com recurso de arrastar e soltar usado para projetar e visualizar facilmente caminhos de conversação. Para mais informações, consulte Visual Conversation Builder.	14 de setembro de 2022
Novo atributo	Houve uma atualização na forma como o Amazon Lex V2 cria slots complexos. Agora você pode criar subslots complexos dentro de slots para gerenciar intents em um design complexo de conversação. Para mais informações, consulte Criar slots compostos .	9 de setembro de 2022

Novo atributo	Houve uma atualização na forma como o Amazon Lex V2 gerencia os fluxos de caminhos de conversação com seus usuários. Agora você pode criar caminhos de conversação complexos ordenando a próxima etapa da conversa. Para mais informações, consulte Criar caminhos de conversação.	17 de agosto de 2022
Novo atributo	Houve uma atualização na forma como o Amazon Lex V2 gerencia os fluxos de conversação com seus usuários. Agora você pode criar conversas complexas ordenando os prompts. Para mais informações, consulte Como configurar prompts.	5 de julho de 2022
Novo atributo	Houve uma atualização na forma como o Amazon Lex V2 gerencia os fluxos de conversação com seus usuários. Agora é possível criar conversas complexas usando condições. Para mais informações, consulte Noções básicas dos novos fluxos de conversação.	3 de maio de 2022

Novo atributo	Foram adicionados exemplos de gramáticas industriais para o tipo de slot gramatical incorporado. Para mais informações, consulte Gramáticas setoriais.	22 de março de 2022
Novo atributo	Foi adicionada documentação sobre como integrar o Amazon Lex V2 ao SDK do Amazon Chime. Para mais informações, consulte o SDK do Amazon Chime.	18 de março de 2022
Novo atributo	O Amazon Lex V2 agora contém pontuações de confiança para transcrições de voz. Use essa informação para determinar a resposta correta do usuário. Para mais informações, consulte Usar pontuações de confiança na transcrição de voz.	27 de janeiro de 2022
Novo atributo	Agora você pode adicionar dicas contextuais e dinâmicas aos slots para melhorar a precisão do seu bot. Para mais informações, consulte Como usar dicas para melhorar a precisão.	13 de janeiro de 2022

Novo atributo	O Amazon Lex V2 aceita vocabulários personalizados para melhorar o reconhecimento de fala de entradas de áudio. Para mais informações, consulte Como criar um vocabulário personalizado para melhorar o reconhecimento de fala.	12 de janeiro de 2022
Novo atributo	O Amazon Lex V2 agora oferece suporte AWS PrivateLink. Para obter mais informações, consulte VPC endpoints (AWS). PrivateLink	7 de janeiro de 2022
Novo atributo	O Amazon Lex V2 agora trabalha com Catalão (Espanha). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	3 de janeiro de 2022
Novo atributo	Agora você consegue criar tipos de slots usando sua própria gramática personalizada. Para mais informações, consulte Usar um tipo de slot de gramática personalizada.	20 de dezembro de 2021
Novo atributo	AWS CloudFormation agora é compatível com o Amazon Lex V2. Para mais informações, consulte Recursos do AWS CloudFormation.	20 de dezembro de 2021

Novo atributo	O Amazon Lex V2 agora trabalha com Português (Brasil), Português (Portugal) e Mandarim (RPC). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	16 de dezembro de 2021
Novo atributo	O Amazon Lex V2 agora fornece uma prévia do Designer de Chatbot Automatizado para ajudar você a começar a criar um chatbot a partir das transcriç ões da central de atendimen to. Para mais informações, consulte Como usa o Designer de Chatbot Automatizado (Pré-visualização).	1º de dezembro de 2021
Novo atributo	Agora você pode usar spell-by-letter e spell-by-word estilizar para inserir valores de slots que o Amazon Lex V2 tem dificuldade em entender. Para mais informações, consulte Como usar estilos de escrever para capturar valores de slots.	19 de novembro de 2021

Novo atributo	Agora você consegue usar vozes neurais de conversão de texto em fala (NTTS) do Amazon Polly para conversas de áudio com seus usuários. Para mais informações, consulte Vozes do Amazon Polly.	19 de novembro de 2021
Novo atributo	O Amazon Lex V2 agora é compatível com Inglês (África do Sul). Para mais informaçõ es, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	9 de novembro de 2021
Novo atributo	O Amazon Lex V2 agora é compatível com Alemão (Áustria). Para mais informaçõ es, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	5 de novembro de 2021
Novo atributo	Agora você pode fornecer aos usuários mensagens de atualização que são reproduzi das no início de uma função de atendimento e periodica mente enquanto a função é executada. Você também pode criar mensagens que informem ao usuário sobre o status do atendimento quando a função for concluída. Para mais informações, consulte Como configurar atualizações do progresso do atendimento.	7 de outubro de 2021

Expansão de região	O Amazon Lex V2 já está disponível na África (Cidade do Cabo) (af-south-1) e Ásia-Pacífico (Seul) (ap-north east-2).	22 de setembro de 2021
Novo atributo	Agora você pode ver as estatísticas das declarações que seus usuários enviam ao seu bot. Para mais informaçõ es, consulte Como visualizar estatísticas das declarações.	22 de setembro de 2021
Novo atributo	O Amazon Lex V2 agora aceita Coreano (Coreia). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	9 de setembro de 2021
Novo atributo	O Amazon Lex V2 agora fornece um tipo de slot integrado para códigos postais do Reino Unido. Para obter mais informações, consulte PostalCodeAMAZON.UK.	27 de julho de 2021
Novo atributo	O Amazon Lex V2 agora aceita Inglês (Índia). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	15 de julho de 2021

Novo atributo	O Amazon Lex V2 agora fornece uma ferramenta para migrar um bot do Amazon Lex V1 para a API Amazon Lex V2. Para mais informações, consulte Como migrar um bot no Guia do desenvolvedor do Amazon Lex.	13 de julho de 2021
Novo atributo	O Amazon Lex V2 agora permite que você aceite vários valores para um único slot no idioma inglês (EUA). Para mais informações, consulte Como usar vários valores em um slot.	15 de junho de 2021
Novo atributo	Agora você pode criar bots maiores para o idioma inglês. Para mais informações, consulte Cotas do .	11 de junho de 2021
Novo atributo	Use políticas baseadas em recursos do Amazon Lex V2 para gerenciar o acesso aos seus bots e aliases de bots. Para mais informações, consulte Políticas baseadas em recursos no Amazon Lex V2.	20 de maio de 2021

Novo atributo	O Amazon Lex V2 agora permite que você importe e exporte bots e locais de bots. Você pode usar esse recurso para copiar bots e localidades de bots entre contas e AWS regiões. Para mais informações, consulte Importar e exportar.	18 de maio de 2021
Expansão de região	O Amazon Lex V2 agora está disponível no Canadá (Central) (ca-central-1).	17 de maio de 2021
Novo atributo	O Amazon Lex V2 agora aceita Japonês (Japão). Para mais informações, consulte Idiomas e locais compatíveis com o Amazon Lex V2.	1º de abril de 2021
Novo atributo	O Amazon Lex V2 agora oferece suporte a três novos tipos de slots integrados: `AMAZON.City` , `AMAZON.Country` e `AMAZON.State` .	12 de março de 2021
Novo guia	Esta é a primeira versão do Guia do usuário do Amazon Lex V2.	21 de janeiro de 2021

Referência de API

A Referência de API agora é um documento separado.

Glossário do AWS

Para obter a terminologia mais recente da AWS, consulte o glossário da AWS na Referência do Glossário da AWS.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.