Guia do Desenvolvedor

Amazon Lex V1

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Lex V1: Guia do Desenvolvedor

Copyright © 2025 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

..... viii
O que é o Amazon Lex? 1
Está usando o Amazon Lex pela primeira vez? 3
Como funciona 4
Idiomas compatíveis 7
Idiomas e locais compatíveis 7
Idiomas e locais compatíveis com os atributos do Amazon Lex 8
Modelo de programação 8
Operações de API de criação de modelo 9
Operações de API de runtime 10
Funções do Lambda como hooks de código 11
Gerenciamento de mensagens 14
Tipos de mensagens 15
Contextos para a configuração de mensagens 16
Formatos de mensagem suportados 21
Grupos de mensagens 21
Cartões de resposta 23
Gerenciar contexto da conversa 27
Definir o contexto da intenção 28
Usar valores de slot padrão 30
Definição dos atributos da sessão 32
Definição de atributos de solicitação 34
Definição do tempo limite da sessão 37
Compartilhamento de informações entre intenções 37
Configuração de atributos complexos 38
Usar pontuações de confiança 39
Gerenciamento de sessões 42
Logs de conversa 43
Políticas do IAM para logs de conversa 44
Configurar logs de conversa 48
Criptografar logs de conversa 51
Visualização de registros de texto no Amazon CloudWatch Logs 53
Acessar logs de áudio no Amazon S3 57
Monitorando o status do registro de conversas com CloudWatch métricas 57

Gerenciamento de Sessões ... 58
Alternância entre intenções ... 60
Como retomar uma intenção anterior ... 60
Como iniciar uma nova sessão ... 61
Validação de valores de slot ... 62
Opções de implantação ... 62
Intenções integradas e tipos de slot ... 62
Intenções integradas ... 63
Tipos de slot integrados ... 81
Tipos de slot personalizados ... 92
Ofuscação de slot ... 94
Análise de sentimento ... 95
Marcação de recursos ... 96
Marcação de recursos ... 97
Restrições de tag ... 98
Marcação de recursos (console) ... 98
Marcação de recursos (AWS CLI) ... 100
Conceitos básicos ... 102
Etapa 1: Configurar uma conta ... 102
Inscreva-se para AWS ... 102
Criar um usuário ... 103
Próxima etapa ... 104
Etapa 2: Configurar o AWS CLI ... 104
... 105
Etapa 3: Conceitos básicos (console) ... 105
Exercício 1: Criar um bot usando um esquema ... 106
Exercício 2: criar bot personalizado ... 144
Exercício 3: publique uma versão e crie um alias ... 160
Etapa 4: Conceitos básicos (AWS CLI) ... 161
Exercício 1: Criar um bot ... 162
Exercício 2: Adicionar uma nova declaração ... 180
Exercício 3: Adicione uma função do Lambda ... 185
Exercício 4: Publicar uma versão ... 190
Exercício 5: Criar um alias ... 196
Exercício 6: Limpar ... 197
Versionamento e aliases ... 199

Versionamento ... 199
A versão $LATEST ... 199
Publicação de uma versão de recurso do Amazon Lex ... 200
Atualização de um recurso do Amazon Lex ... 201
Exclusão de um recurso ou uma versão do Amazon Lex. ... 201
Aliases ... 202
Uso de funções do Lambda ... 204
Evento de entrada de função do Lambda e formato de resposta ... 204
Formato de eventos de entrada ... 204
Formato de resposta ... 212
Amazon Lex e esquemas AWS Lambda ... 219
Atualização de um esquema para uma localidade específica ... 220
Implantação de bots ... 221
Implantação de um bot do Amazon Lex em uma plataforma de sistema de mensagens ... 221
Integrar com o Facebook ... 224
Integrar com o Kik ... 227
Integrar com o Slack ... 231
Integração com o SMS do Twilio ... 237
Implantação de um bot do Amazon Lex em aplicativos móveis ... 241
Importar e exportar ... 242
Exportar e importar em formato do Amazon Lex ... 242
Exportar em formato do Amazon Lex ... 243
Importar em formato do Amazon Lex ... 244
Formato JSON para importação e exportação ... 246
Exportar para uma habilidade do Alexa ... 249
Exemplos de bot ... 252
Agendar uma consulta ... 252
Visão geral do Bot Blueprint () ScheduleAppointment ... 255
Visão geral da função Lambda Blueprint () lex-make-appointment-python ... 256
Etapa 1: criar um bot do Amazon Lex ... 257
Etapa 2: criar uma função do Lambda ... 259
Etapa 3: atualizar a intenção - configuração de um hook de código ... 260
Etapa 4: implantar o bot na plataforma do Facebook Messenger ... 262
Detalhes do fluxo de informações ... 262
Reservar uma viagem ... 280
Etapa 1: análise de esquema ... 281

Etapa 2: criar um bot do Amazon Lex ... 284
Etapa 3: criar uma função do Lambda ... 287
Etapa 4: adicionar a função do Lambda como hook de código ... 288
Detalhes do fluxo de informações ... 292
Exemplo: uso de um cartão de resposta ... 312
Atualizar declarações ... 316
Exemplo: Integração com um site ... 318
Assistente de atendente do call center ... 319
Etapa 1: criar um índice do Amazon Kendra ... 320
Etapa 2: criar um bot do Amazon Lex ... 321
Etapa 3: adicionar uma intenção personalizada e integrada ... 322
Etapa 4: configurar o Amazon Cognito ... 323
Etapa 5: implantar seu bot como um aplicativo Web ... 325
Etapa 6: usar o bot ... 325
Migração de um bot ... 329
Migração de um bot (Console) ... 329
Migração de uma função do Lambda ... 330
Mensagens de migração ... 331
Intenção integrada ... 331
Tipo de slot integrado ... 331
Logs de conversa ... 331
Grupos de mensagens ... 332
Prompts e frases ... 332
Outros atributos do Amazon Lex V1 ... 333
Migração de uma função do Lambda ... 333
Lista de campos atualizados ... 335
Segurança ... 343
Proteção de dados ... 344
Criptografia em repouso ... 344
Criptografia em trânsito ... 346
Gerenciamento de chaves ... 346
Gerenciamento de Identidade e Acesso ... 346
Público ... 347
Autenticar com identidades ... 347
Gerenciar o acesso usando políticas ... 351
Como o Amazon Lex funciona com o IAM ... 354

Exemplos de políticas baseadas em identidade 365
Políticas gerenciadas pela AWS para o Amazon Lex 372
Uso de perfis vinculadas ao serviço 381
Solução de problemas 383
Monitoramento 385
Monitorando o Amazon Lex com CloudWatch 386
Registro de chamadas de API do Amazon Lex com AWS CloudTrail 398
Validação de conformidade 403
Resiliência 404
Segurança da infraestrutura 404
Diretrizes e cotas 406
Regiões do compatíveis 406
Diretrizes gerais 406
Cotas 410
Service Quotas de runtime 410
Cotas de criação de modelos 412
Referência da API 417
Ações 417
Serviço de criação de modelos do Amazon Lex 419
Serviço de runtime do Amazon Lex 631
Tipos de dados 676
Serviço de criação de modelos do Amazon Lex 677
Serviço de runtime do Amazon Lex 736
Histórico do documento 755
AWS Glossário 763

Aviso de fim do suporte: em 15 de setembro de 2025, o suporte para o Amazon Lex V1 AWS será interrompido. Depois de 15 de setembro de 2025, você não poderá mais acessar o console do Amazon Lex V1 ou os recursos do Amazon Lex V1. Se você estiver usando o Amazon Lex V2, consulte o guia do Amazon Lex V2 em vez disso.

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.

O que é o Amazon Lex?

O Amazon Lex é um serviço da AWS para a criação de interfaces de conversa em qualquer aplicação que usa voz e texto. Com o Amazon Lex, o mesmo mecanismo de conversa que está por trás do Amazon Alexa agora está disponível para todos os desenvolvedores, permitindo que você crie sofisticados chatbots de linguagem natural em seus aplicativos novos e existentes. O Amazon Lex 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 altamente envolventes com interações por conversa realistas e a criação de novas categorias de produtos.

O Amazon Lex permite que qualquer desenvolvedor crie chatbots de conversa rapidamente. Com o Amazon Lex, 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. O Amazon Lex 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 fornece integração predefinida com AWS Lambda, e você pode integrar-se facilmente com muitos outros serviços na plataforma da AWS, incluindo Amazon Cognito AWS Mobile Hub, Amazon e CloudWatch Amazon DynamoDB. A integração com o Lambda fornece aos bots acesso a conectores corporativos sem servidor pré-criados para vincular dados em aplicativos SaaS, como Salesforce ou Marketo. HubSpot

Alguns benefícios de usar o Amazon Lex incluem:

- Simplicidade: o Amazon Lex 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 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: com a mesma tecnologia da Alexa, o Amazon Lex fornece as tecnologias ASR e NLU para criar um sistema de compreensão de linguagem falada (SLU). Por meio do SLU, o Amazon Lex 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 de 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 enormes quantidades de dados e infraestrutura. O Amazon Lex coloca as tecnologias de aprendizado profundo ao alcance de todos os desenvolvedores, com a mesma tecnologia do Alexa. Os chatbots do Amazon Lex 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 escalonamento simples: com o Amazon Lex, você pode criar, testar e implantar seus chatbots diretamente no console do Amazon Lex. O Amazon Lex permite que você publique facilmente seus chatbots de voz ou texto para uso em dispositivos móveis, aplicativos Web e serviços de bate-papo (por exemplo, o Facebook Messenger). O Amazon Lex escala automaticamente para que você não precise se preocupar com o provisionamento de hardware e o gerenciamento da infraestrutura para potencializar sua experiência de bot.

- Integração integrada com a plataforma da AWS — O Amazon Lex tem interoperabilidade nativa com outros serviços da AWS, como Amazon Cognito AWS Lambda, Amazon e. CloudWatch AWS Mobile Hub Você pode aproveitar o poder da plataforma da AWS para segurança, monitoramento, autenticação do usuário, lógica de negócios, armazenamento e desenvolvimento de aplicativos móveis.

- Economia: com o Amazon Lex, não há custos iniciais nem taxas mínimas. Você será cobrado apenas pelas solicitações de texto ou fala feitas. O pay-as-you-go preço e o baixo custo por solicitação tornam o serviço uma forma econômica de criar interfaces conversacionais. Com o nível gratuito do Amazon Lex, você pode testar o Amazon Lex com facilidade e sem nenhum investimento inicial.

Está usando o Amazon Lex pela primeira vez?

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

1. Conceitos básicos do Amazon Lex - Nesta seção, você define sua conta e testa o Amazon Lex.
2. Referência da API - Esta seção fornece exemplos adicionais que você pode usar para explorar o Amazon Lex.

Amazon Lex: como funciona

O Amazon Lex permite criar aplicativos usando uma interface de texto ou de fala com a mesma tecnologia usada pelo Amazon Alexa. Veja a seguir as etapas comuns que você executa ao trabalhar com Amazon Lex:

1. Crie e configure um bot com uma ou mais intenções a que você deseja oferecer suporte. Configure o bot para que ele entenda o objetivo do usuário (intenção), 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.
3. Publique uma versão e crie um alias.
4. Implante o bot. Você pode implantar o bot em plataformas como aplicações móveis ou plataformas de mensagens, como Facebook Messenger.

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

- Bot: um bot executa tarefas automatizadas, como pedir uma pizza, reservar um hotel, encomendar flores, e assim por diante. Um bot do Amazon Lex possui os recursos de reconhecimento automático de voz (ASR) e compreensão de linguagem natural (NLU). Cada bot deve ter um nome exclusivo na sua conta.

  Os bots do Amazon Lex podem compreender a entrada do usuário fornecida por texto ou fala e conversar em linguagem natural. É possível criar funções do Lambda e adicioná-las como hooks de código em sua configuração de intenção para executar a validação de dados do usuário e tarefas de atendimento.

- 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 bot que peça pizza e bebidas. Para cada intenção, forneça as seguintes informações obrigatórias:

- Nome da intenção: um nome descritivo para a intenção. Por exemplo, **`OrderPizza`**. Os nomes de função devem ser exclusivos em sua conta.
- Amostra de declarações: como um usuário pode expressar a intenção. Por exemplo, um usuário pode dizer "Posso pedir uma pizza" ou "Quero pedir uma pizza".
- Como cumprir a intenção: como você deseja cumprir a intenção depois que o usuário fornecer todas as informações necessárias (por exemplo, fazer um pedido com uma pizzaria local). Recomendamos criar uma função do Lambda para atender à intenção.

  Opcionalmente, você pode configurar a intenção para que o Amazon Lex simplesmente retorne as informações de volta ao aplicativo cliente para executar o atendimento necessário.

Além de intenções personalizadas, como encomendar uma pizza, o Amazon Lex também fornece intenções integradas para configurar seu bot rapidamente. Para obter mais informações, consulte Intenções integradas e tipos de slot.

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

  Por exemplo, a intenção `OrderPizza` exige slots como tamanho da pizza, tipo de massa e número de pizzas. Na configuração de intenção, você adiciona esses slots. Para cada slot, você fornece o tipo de slot e um prompt para o Amazon Lex enviar ao cliente para obter os dados do usuário. Um usuário pode responder com um valor de slot que inclui palavras adicionais, como "pizza grande, por favor" ou "vamos querer a pequena". O Amazon Lex ainda consegue entender o valor pretendido do slot.

- Tipo de slot: cada slot possui um tipo. Você pode criar tipos de slot personalizados ou usar tipos de slot integrados. Cada tipo de slot deve ter um nome exclusivo na sua conta. 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 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 obter mais informações, consulte Intenções integradas e tipos de slot.

Para obter uma lista de regiões da AWS onde o Amazon Lex está disponível, consulte Regiões e endpoints da AWS na Referência geral do Amazon Web Services.

Os tópicos a seguir fornecem informações adicionais. Recomendamos que você analise-as em ordem e, em seguida, explore os exercícios Conceitos básicos do Amazon Lex.

Tópicos

- Idiomas compatíveis com o Amazon Lex
- Modelo de programação
- Gerenciamento de mensagens
- Gerenciar contexto da conversa
- Usar pontuações de confiança
- Logs de conversa
- Gerenciamento de sessões com a API do Amazon Lex
- Opções de implantação de bot
- Intenções integradas e tipos de slot
- Tipos de slot personalizados
- Ofuscação de slot
- Análise de sentimento
- Marcação de seus recursos do Amazon Lex

Idiomas compatíveis com o Amazon Lex

O Amazon Lex V1 oferece suporte a uma série de Idiomas e localidades. Os idiomas compatíveis e os atributos compatíveis estão listados nas tabelas a seguir.

O Amazon Lex V2 oferece suporte a outros idiomas, consulte Idiomas compatíveis no Amazon Lex V2

Idiomas e locais compatíveis

O Amazon Lex V1 oferece suporte aos seguintes idiomas e locais.

Código	Idioma e local
de-DE	Alemão (Alemanha)
en-AU	Inglês (Austrália)
en-GB	Inglês (Reino Unido)
en-IN	Inglês (Índia)
en-US	Inglês (EUA)
es-419	Espanhol (América Latina)
es-ES	Espanhol (Espanha)
es-US	Espanhol (EUA)
fr-CA	Francês (Canadá)
fr-FR	Francês (França)
it-IT	Italiano (Itália)
ja-JP	Japonês (Japão)
ko-KR	Coreano (Coreia)

Idiomas e locais compatíveis com os atributos do Amazon Lex

Todos os atributos do Amazon Lex são compatíveis em todos os idiomas e locais, exceto conforme listado nesta tabela.

Atributo	Idiomas e locais compatíveis
Definir o contexto da intenção	Inglês (EUA) (en-US)

Modelo de programação

Um bot é o principal tipo de recurso no Amazon Lex. Os outros tipos de recursos no Amazon Lex; são intenção, tipo de slot, alias e associação de canal de bot.

Você cria um bot usando o console do Amazon Lex ou a API de criação de modelos. O console fornece uma interface gráfica de usuário que você usa para criar um bot pronto para produção para seu aplicativo. Se preferir, você pode usar a API de criação de modelos por meio do AWS CLI ou de seu próprio programa personalizado para criar um bot.

Depois de criar um bot, você o implanta em uma das plataformas compatíveis ou o integra em seu próprio aplicativo. Quando um usuário interage com o bot, o aplicativo cliente envia solicitações ao bot usando a API de runtime do Amazon Lex. Por exemplo, quando um usuário diz "Quero encomendar uma pizza", o cliente envia essa entrada do usuário ao Amazon Lex usando uma das operações da API de runtime. Os usuários podem fornecer entrada de voz ou texto.

Você também pode criar funções do Lambda; e usá-las em uma intenção. Use esses hooks de código de função do Lambda para executar atividades em runtime, como inicialização, validação da entrada do usuário e atendimento da intenção. As seções a seguir fornecem informações adicionais.

Tópicos

- Operações de API de criação de modelo
- Operações de API de runtime
- Funções do Lambda como hooks de código

Operações de API de criação de modelo

Para criar programaticamente bots, intenções e tipos de slot, use a API de operações de criação de modelo. Você também pode usar a API de criação de modelo para gerenciar, atualizar e excluir os recursos para o seu bot. As operações de API de criação de modelo incluem:

- PutBot, PutBotAlias, PutIntente PutSlotType para criar e atualizar bots, aliases de bot, intenções e tipos de slot, respectivamente.
- CreateBotVersion, CreateIntentVersion e CreateSlotTypeVersion para criar e publicar versões de seus bots, intenções e tipos de slot, respectivamente.
- GetBot e GetBots para obter um bot específico ou uma lista de bots que você criou, respectivamente.
- GetIntent e GetIntents para obter uma intenção específica ou uma lista de intenções que você criou, respectivamente.
- GetSlotType e GetSlotTypes para obter um tipo de slot específico ou uma lista de tipos de slot que você criou, respectivamente.
- GetBuiltinIntent, GetBuiltinIntents e GetBuiltinSlotTypes para obter uma intenção integrada do Amazon Lex, uma lista de intenções integradas do Amazon Lex ou uma lista de tipos de slot integrado que você pode usar em seu bot, respectivamente.
- GetBotChannelAssociation e GetBotChannelAssociations para obter uma associação entre seu bot e uma plataforma de mensagens ou uma lista de associações entre seu bot e plataformas de mensagens, respectivamente.
- DeleteBot, DeleteBotAlias, DeleteBotChannelAssociation, DeleteIntent e DeleteSlotType para remover recursos desnecessários em sua conta.

Você pode usar a API de criação de modelo para criar ferramentas personalizadas a fim de gerenciar os recursos do Amazon Lex. Por exemplo, há um limite de 100 versões para bots, intenções e tipos de slot. Você pode usar a API de criação de modelo para criar uma ferramenta que exclui automaticamente versões antigas quando o bot se aproxima do limite.

Para garantir que apenas uma operação atualize um recurso por vez, o Amazon Lex usa somas de verificação. Quando usa uma operação de API `Put`, PutBot, PutBotAlias PutIntent ou PutSlotType, para atualizar um recurso, você deve passar a soma de verificação atual do recurso na solicitação. Se duas ferramentas tentarem atualizar um recurso ao mesmo tempo, elas oferecem a mesma soma de verificação atual. A primeira solicitação a alcançar o Amazon Lex corresponde à soma de verificação atual do recurso. Quando a segunda solicitação chega, a soma de verificação é diferente.

A segunda ferramenta recebe uma exceção `PreconditionFailedException` e a atualização é encerrada.

As operações `Get`, GetBot, GetIntent e GetSlotType, são eventualmente consistentes. Se você usar uma operação `Get` imediatamente após criar ou modificar um recurso com uma das operações `Put`, as alterações poderão não ser retornadas. Depois de uma operação `Get` retornar a atualização mais recente, ela sempre retornará esse recurso atualizado até que o recurso seja modificado novamente. Você pode determinar se um recurso atualizado foi retornado ao analisar a soma de verificação.

Operações de API de runtime

Os aplicativos cliente usam as seguintes operações de API em runtime para se comunicar com o Amazon Lex:

- PostContent - Usa entrada de texto ou fala e retorna informações de intenção e uma mensagem de texto ou fala para expressar ao usuário. No momento, o Amazon Lex é oferece suporte aos seguintes formatos de áudio:

  Formatos de áudio de entrada – LPCM e Opus

  Formatos de áudio de saída – MPEG, OGG e PCM

  A operação `PostContent` oferece suporte a entrada de áudio em 8 kHz e 16 kHz. Os aplicativos em que o usuário final se comunica com o Amazon Lex por telefone, como uma central de atendimento automatizada, podem passar áudio 8 kHz diretamente.

- PostText - Usa texto como entrada e retorna informações de intenção e uma mensagem de texto para transmitir ao usuário.

Seu aplicativo cliente usa a API de runtime para chamar um determinado bot do Amazon Lex para processar declarações entrada de texto ou de fala do usuário. Por exemplo, suponha que um usuário diga "Quero pizza". O cliente envia essa entrada do usuário a um bot usando uma das operações da API de runtime do Amazon Lex. A partir da entrada do usuário, o Amazon Lex; reconhece que a solicitação do usuário destina-se à intenção `OrderPizza` definida no bot. O Amazon Lex engaja o

usuário em uma conversa para obter as informações necessárias, ou dados de slot, como tamanho da pizza, cobertura e número de pizzas. Depois que o usuário fornece todos os dados necessários do slot, o Amazon Lex invoca o hook de código da função do Lambda para atender à intenção ou retorna os dados da intenção para o cliente, dependendo de como a intenção está configurada.

Use a operação PostContent quando seu bot usa entrada de voz. Por exemplo, um aplicativo de call center automatizado pode enviar fala para um bot do Amazon Lex, e não para um atendente, para resolver indagações do cliente. Você pode usar o formato de áudio de 8 kHz para enviar áudio diretamente do telefone para o Amazon Lex.

A janela de teste no console do Amazon Lex usa a API PostContent para enviar solicitações de texto e de fala para o Amazon Lex. Use esta janela de teste nos exercícios Conceitos básicos do Amazon Lex.

Funções do Lambda como hooks de código

Você pode configurar seu bot do Amazon Lex para invocar uma função do Lambda como um hook de código. O hook de código pode servir para várias finalidades:

- Personaliza a interação do usuário. Por exemplo, quando Joe pergunta pelas coberturas de pizza disponíveis, você pode usar o conhecimento anterior das opções de Joe para exibir um subconjunto de coberturas.
- Valida a entrada do usuário. Suponha que Jen queira escolher flores depois do horário de funcionamento. Você pode validar o horário que Jen inseriu e enviar uma resposta adequada.
- Atende à intenção do usuário. Depois de Joe fornecer todas as informações sobre seu pedido de pizza, o Amazon Lex pode invocar uma função do Lambda para fazer o pedido em uma pizzaria local.

Ao configurar uma intenção, você especifica funções do Lambda como hooks de código nos seguintes locais:

- Hook de código de diálogo para inicialização e validação: essa função do Lambda é invocada em cada entrada do usuário, supondo que o Amazon Lex tenha entendido a intenção do usuário.
- Hook de código de atendimento: essa função do Lambda;é invocada depois que o usuário fornece todos os dados de slot necessários para atender a intenção.

Você escolhe a intenção e define os hooks de código no console do Amazon Lex, conforme mostrado na captura de tela a seguir:

OrderFlowers Latest

Sample utterances

e.g. I would like to book a flight.

I would like to pick up flowers

I would like to order some flowers

Order flowers

Lambda initialization and validation

Initialization and validation code hook

Lambda Function Name

Slots

Priority	Required	Name	Slot type		Prompt
		e.g. Location	e.g. A…		e.g. What city?
1.	✓	FlowerType	Flowe…	1	What type of flow
2.	✓	PickupDate	AMA…	Built-in	What day do you
3.	✓	PickupTime	AMA…	Built-in	At what time do y

Confirmation prompt

Confirmation prompt

Confirm

Okay, your {FlowerType} will be ready for pickup by {Picku

Cancel (if the user says "no")

Okay, I will not place your order.

Fulfillment

AWS Lambda function Return parameters to client

Lambda Function Name

Response

Você também pode definir hooks de código usando os campos `dialogCodeHook` e `fulfillmentActivity` na operação PutIntent.

Uma função do Lambda pode executar a inicialização, a validação e o atendimento. Os dados do evento que a função do Lambda recebe têm um campo que identifica o chamador como um hook de código de diálogo ou de atendimento. É possível usar essas informações para executar a parte adequada de seu código.

Você pode usar uma função do Lambda para criar um bot que pode navegar por diálogos complexos. Você usa o campo `dialogAction` na resposta da função do Lambda para direcionar o Amazon Lex para executar ações específicas. Por exemplo, você pode usar a ação do diálogo `ElicitSlot` para dizer ao Amazon Lex para solicitar ao usuário um valor de slot que não é necessário. Se você tiver um prompt de esclarecimento definido, poderá usar a ação de diálogo `ElicitIntent` para obter uma nova intenção quando o usuário terminar de usar a anterior.

Para obter mais informações, consulte Uso de funções do Lambda.

Gerenciamento de mensagens

Tópicos

- Tipos de mensagens
- Contextos para a configuração de mensagens
- Formatos de mensagem suportados
- Grupos de mensagens
- Cartões de resposta

Ao criar um bot, você pode configurar mensagens de esclarecimento ou informativas que deseja que ele envie ao cliente. Considere os seguintes exemplos:

- Você pode configurar seu bot com a solicitação de esclarecimento a seguir:

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

  O Amazon Lex envia essa mensagem para o cliente, caso não entenda a intenção do usuário.

- Suponha que você crie um bot para oferecer suporte a uma intenção chamada `OrderPizza`. Para pedidos de pizza, você deseja que os usuários forneçam informações como tamanho da pizza, cobertura e tipo de massa. Você pode configurar as seguintes solicitações:

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

  Assim que o Amazon Lex determina a intenção do usuário de pedir uma pizza, ele envia estas mensagens ao cliente para obter informações do usuário.

Esta seção explica como projetar interações do usuário em sua configuração de bot.

Tipos de mensagens

A mensagem pode ser uma solicitação ou instrução.

- Normalmente, o prompt é uma pergunta que presume uma resposta do usuário.
- A instrução é apenas informativa. Ela não presume uma resposta.

Uma mensagem pode incluir referências a slot, atributos de sessão e atributos de solicitação. Em runtime, o Amazon Lex substitui essas referências com valores reais.

Para se referir a valores de slots que foram definidos, use a seguinte sintaxe:

```
{SlotName}
```

Para se referir a atributos de sessão, use a seguinte sintaxe:

```
[SessionAttributeName]
```

Para se referir a atributos de solicitação, use a seguinte sintaxe:

```
((RequestAttributeName))
```

As mensagens podem incluir valores de slot, atributos de sessão e atributos de solicitação.

Por exemplo, suponha que você configure a seguinte mensagem na OrderPizza intenção do seu bot:

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

Essa mensagem refere-se tanto ao slot (`PizzaTopping`) quanto aos atributos de sessão (`FirstName` e `DeliveryTime`). Em runtime, o Amazon Lex substitui estes espaços reservados por valores e retorna a seguinte mensagem para o cliente:

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

Para incluir colchetes ([]) ou chaves ({}) em uma mensagem, use o caractere de escape de barra invertida (\). Por exemplo, a seguinte mensagem inclui chaves e colchetes:

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

O texto retornado ao aplicativo cliente é semelhante a este:

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

Para obter mais informações sobre atributos de sessão, consulte as operações de API PostText e PostContent. Para obter um exemplo, consulte Reservar uma viagem.

As funções do Lambda também podem gerar mensagens e retorná-las ao Amazon Lex para envio ao usuário. Se você adicionar funções do Lambda ao configurar sua intenção, poderá criar mensagens dinamicamente. Se fornecer as mensagens ao configurar o bot, você poderá eliminar a necessidade de criar um prompt na função do Lambda.

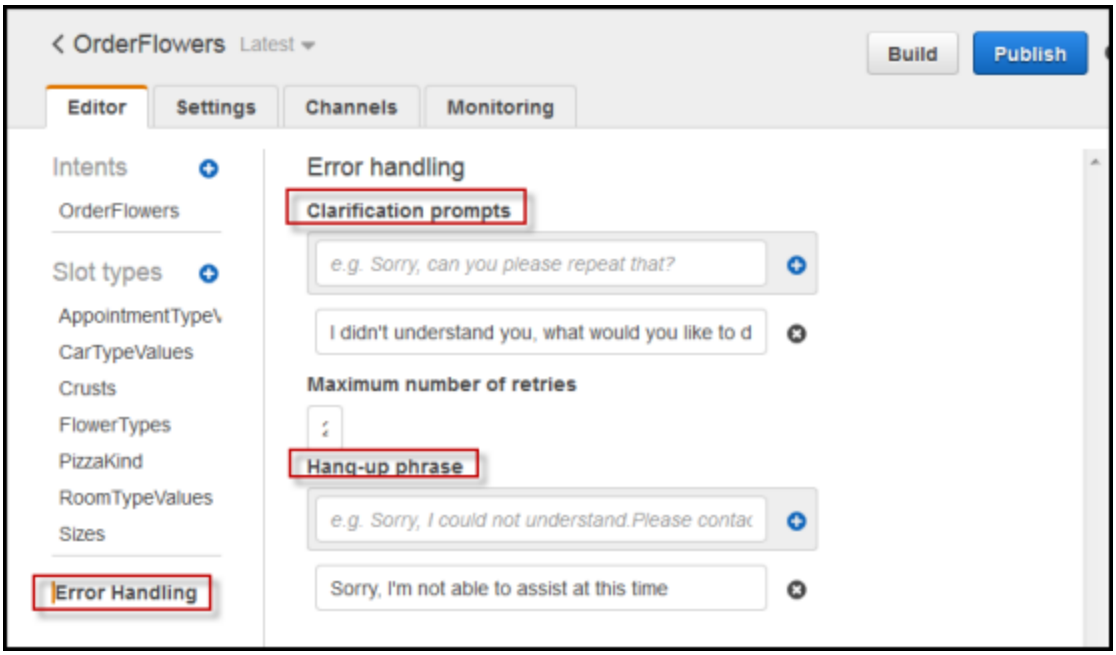
Contextos para a configuração de mensagens

Ao criar um bot, você pode criar mensagens em contextos diferentes, como solicitações de esclarecimento no bot, prompts de valores de slot e mensagens de intenções. O Amazon Lex escolhe uma mensagem apropriada em cada contexto para retornar ao usuário. Você pode fornecer um grupo de mensagens para cada contexto. Se você fizer isso, o Amazon Lex selecionará aleatoriamente uma mensagem no grupo. Você pode também especificar o formato da mensagem ou agrupar as mensagens. Para obter mais informações, consulte Formatos de mensagem suportados.

Se tiver uma função do Lambda associada a uma intenção, você poderá substituir qualquer uma das mensagens configuradas no momento da compilação. No entanto, para usar qualquer uma dessas mensagens, não é preciso ter uma função do Lambda.

Mensagens de bot

Você pode configurar seu bot com prompts de esclarecimento e mensagens de encerramento. Em runtime, o Amazon Lex usa o prompt de esclarecimento quando não compreende a intenção do usuário. Você pode configurar o número de vezes que o Amazon Lex solicita esclarecimentos antes de enviar a mensagem de fim da sessão. Você configura as mensagens em nível de bot na seção Tratamento de erros do console do Amazon Lex, da seguinte forma:



Com a API, você configura mensagens definindo os campos `clarificationPrompt` e `abortStatement` na operação PutBot.

Se você usar uma função do Lambda com uma intenção, a função do Lambda pode retornar uma resposta direcionando o Amazon Lex a solicitar uma intenção do usuário. Se a função do Lambda; não fornecer essa mensagem, o Amazon Lex usará o prompt de esclarecimento.

Prompts de slot

É necessário especificar pelo menos uma mensagem de prompt para os slots necessários em uma intenção. Em runtime, o Amazon Lex usa uma dessas mensagens para solicitar que o usuário forneça um valor para o slot. Por exemplo, para um slot `cityName`, o seguinte é um prompt válido:

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

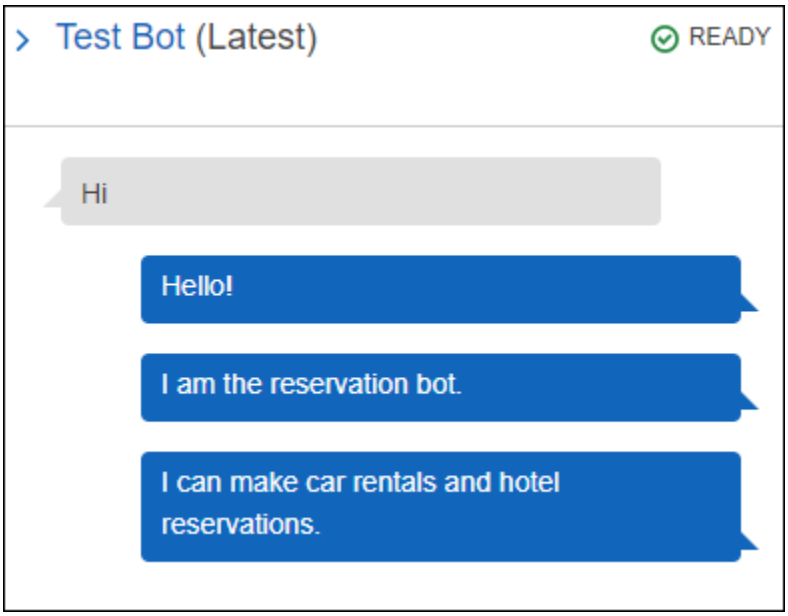
Você pode definir uma ou mais solicitações para cada slot usando o console. Você pode também criar grupos de solicitações usando a operação PutIntent. Para obter mais informações, consulte Grupos de mensagens.

Respostas

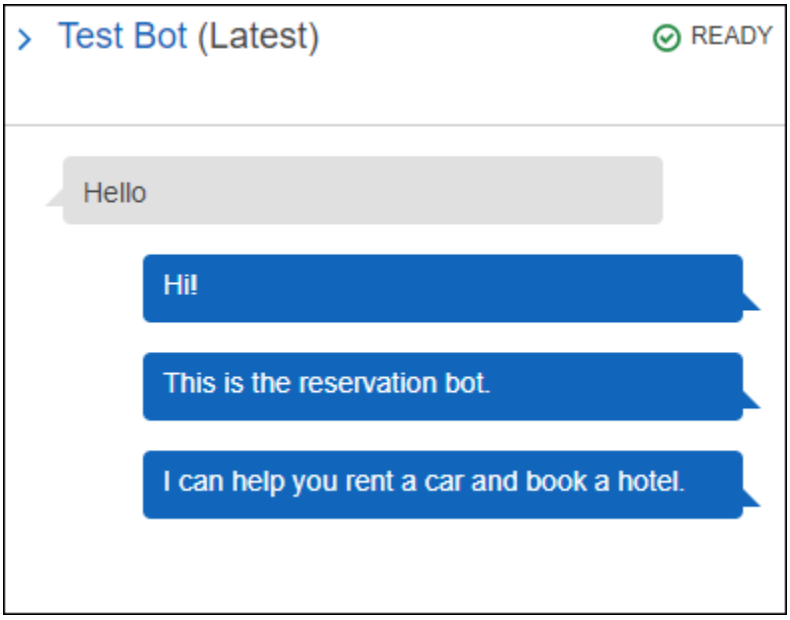
No console, use a seção Respostas para criar conversas dinâmicas e envolventes para seu bot. Você pode criar um ou mais grupos de mensagens para uma resposta. Em runtime, o Amazon Lex cria uma resposta selecionando uma mensagem de cada grupo de mensagens. Para obter mais informações sobre grupos de mensagens, consulte Grupos de mensagens.

Por exemplo, o primeiro grupo de mensagens pode conter diferentes saudações: "Olá", "Oi" e "Saudações". O segundo grupo de mensagens pode conter diferentes formas de introdução: "Sou o bot de reserva" e "Este é o bot de reserva". Um terceiro grupo de mensagens pode informar os recursos do bot: "Posso ajudar com aluguel de carro e reservas de hotéis", "Você pode alugar carros e fazer reservas de hotel" e "Eu posso ajudar você a alugar um carro e reservar um hotel".

O Lex usa uma mensagem de cada um dos grupos de mensagens para criar as respostas dinamicamente em uma conversa. Por exemplo, uma interação pode ser:

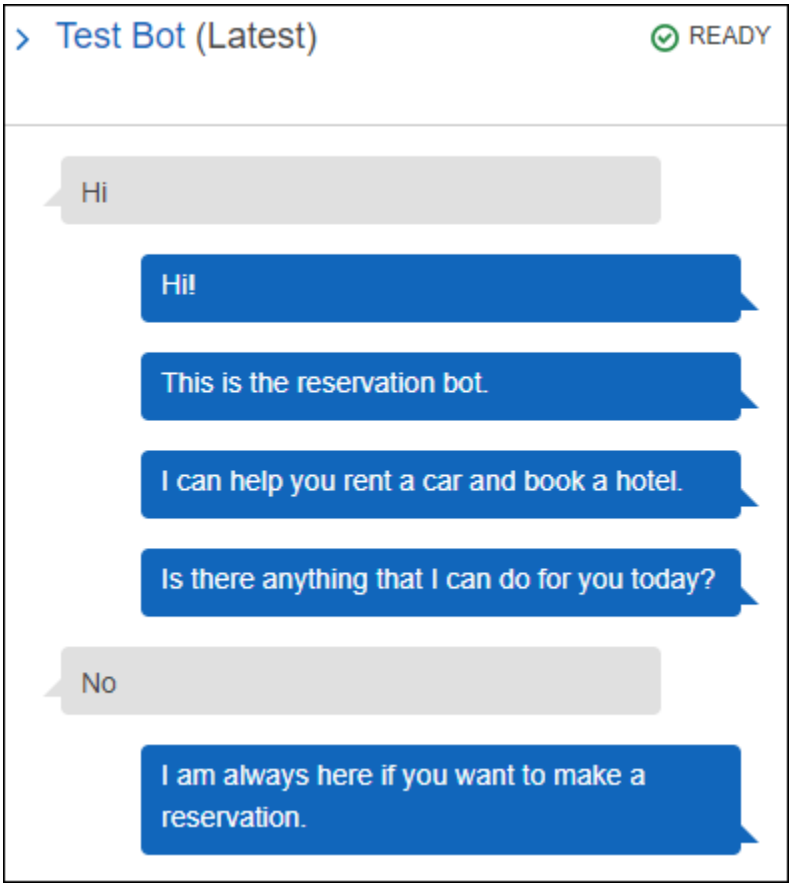


Outro poderia ser o seguinte:



Em qualquer um dos casos, o usuário pode responder com uma nova intenção, como a intenção `BookCar` ou `BookHotel`.

Você pode configurar o bot para fazer uma pergunta complementar na resposta. Por exemplo, para a interação anterior, você pode criar um quarto grupo de mensagens com as seguintes perguntas: "Posso ajudar com um carro ou um hotel?", "Você gostaria de fazer uma reserva agora?" e "Há algo que possa fazer para você?". Para mensagens que incluam "Não" como resposta, você pode criar um prompt de acompanhamento. A imagem seguir fornece um exemplo:



Para criar um prompt de acompanhamento, escolha Aguardar resposta do usuário. Em seguida, digite a mensagem ou as mensagens que você deseja enviar quando o usuário disser "Não". Ao criar uma resposta para usar como prompt de acompanhamento, você devem especificar também uma instrução apropriada quando a resposta para a instrução for "Não". Para obter um exemplo, veja a imagem a seguir.

Para adicionar respostas a uma intenção com a API, use a operação `PutIntent`. Para especificar uma resposta, defina o campo `conclusionStatement` na solicitação `PutIntent`. Para definir uma solicitação de acompanhamento, defina o campo `followUpPrompt` e inclua a instrução a ser enviada quando o usuário disser "Não". Você não pode definir o campo `conclusionStatement` e o campo `followUpPrompt` na mesma intenção.

Formatos de mensagem suportados

Quando você a operação PostText ou a operação PostContent com o cabeçalho `Accept` definido como `text/plain;charset=utf8`, o Amazon Lex oferece suporte a mensagens nos seguintes formatos:

- `PlainText`: a mensagem contém texto sem formatação UTF-8.
- `SSML`: a mensagem contém texto formatado para saída de voz.
- `CustomPayload`: a mensagem contém um formato personalizado que você criou para seu cliente. Você pode definir a carga útil para atender às necessidades de seu aplicativo.
- `Composite`: a mensagem é uma coleção de mensagens, uma de cada grupo de mensagens. Para obter mais informações sobre grupos de mensagens, consulte Grupos de mensagens.

Por padrão, o Amazon Lex retorna qualquer uma das mensagens definidas para determinado prompt. Por exemplo, se você definir cinco mensagens para obter um valor de slot, o Amazon Lex escolherá uma das mensagens aleatoriamente e a retornará para o cliente.

Se quiser que o Amazon Lex retorne um tipo específico de mensagem para o cliente em uma solicitação de runtime, defina o parâmetro de solicitação `x-amzn-lex:accept-content-types`. A resposta é limitada pelo tipo ou pelos tipos solicitados. Se houver mais de uma mensagem do tipo especificado, o Amazon Lex retornará uma aleatoriamente. Para obter mais informações sobre o cabeçalho `x-amz-lex:accept-content-types`, consulte Como configurar o tipo de resposta.

Grupos de mensagens

Um grupo de mensagens é um conjunto de respostas adequadas para determinada solicitação. Use grupos de mensagens quando desejar que seu bot crie respostas dinamicamente em uma conversa. Quando o Amazon Lex retorna uma resposta ao aplicativo cliente, ele escolhe aleatoriamente uma mensagem de cada grupo. Você pode criar no máximo cinco grupos de mensagens para cada resposta. Cada grupo pode conter no máximo cinco mensagens. Para obter exemplos de como criar grupos de mensagens no console, consulte Respostas.

Para criar um grupo de mensagens, você pode usar o console ou as operações PutBot, PutIntent ou PutSlotType para atribuir um número de grupo a uma mensagem. Se você não criar um grupo de mensagens ou se criar apenas um grupo, o Amazon Lex enviará uma única mensagem no campo `Message`. Os aplicativos cliente recebem várias mensagens em uma resposta somente quando você cria mais de um grupo de mensagens no console ou mais de um grupo de mensagens ao criar ou atualizar uma intenção com a operação PutIntent.

Quando o Amazon Lex envia uma mensagem de um grupo, o campo `Message` da resposta contém um objeto JSON de escape que contém as mensagens. O exemplo a seguir mostra o conteúdo do campo `Message` quando ele contém várias mensagens.

> **Note**
>
> O exemplo é formatado por motivo de legibilidade. Uma resposta não contém retornos de carro (CR).

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

Você pode definir o formato das mensagens. O formato pode ser um dos seguintes:

- PlainText—A mensagem está em texto UTF-8 simples.
- SSML: o formato da mensagem é Linguagem de marcação de síntese de fala (SSML).
- CustomPayload—A mensagem está em um formato personalizado que você especificou.

Para controlar o formato das mensagens que as operações `PostContent` e `PostText` retornam no campo `Message`, defina o atributo de solicitação `x-amz-lex:accept-content-types`. Por exemplo, se você definir o cabeçalho como a seguir, você receberá apenas texto sem formatação e mensagens SSML na resposta:

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

Se solicitar um formato específico de mensagem e um grupo de mensagens não contiver uma mensagem com esse formato, você obterá uma exceção `NoUsableMessageException`. Quando

usar um grupo de mensagens para agrupar mensagens por tipo, não use o cabeçalho `x-amz-lex:accept-content-types`.

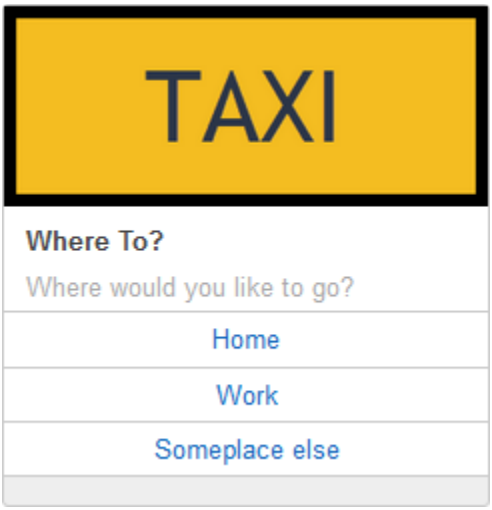
Para obter mais informações sobre o cabeçalho `x-amz-lex:accept-content-types`, consulte Como configurar o tipo de resposta.

Cartões de resposta

> **Note**
>
> Os cartões de resposta não funcionam com o chat do Amazon Connect. No entanto, consulte Adicionar mensagens interativas ao chat para obter uma funcionalidade semelhante.

Um cartão de resposta contém um conjunto de respostas adequadas a um prompt. Use cartões de resposta para simplificar interações para seus usuários e aumentar a precisão de seu bot ao reduzir erros tipográficos nas interações de texto. Você pode enviar um cartão de resposta para cada prompt que o Amazon Lex envia a seu aplicativo cliente. Você pode usar cartões de resposta com o Facebook Messenger, Slack e Twilio e com seus próprios aplicativos clientes.

Por exemplo, em um aplicativo de táxi, você pode configurar uma opção no cartão de resposta para "Casa" e defina o valor para o endereço residencial do usuário. Quando o usuário seleciona essa opção, o Amazon Lex recebe o endereço inteiro como o texto de entrada. Veja a imagem a seguir:



Você pode definir um cartão de resposta para os seguintes prompts:

- Declaração de conclusão
- Prompt de confirmação

- Prompt de acompanhamento
- Declaração de rejeição
- Declarações de tipo de slot

Você pode definir apenas um cartão de resposta para cada prompt.

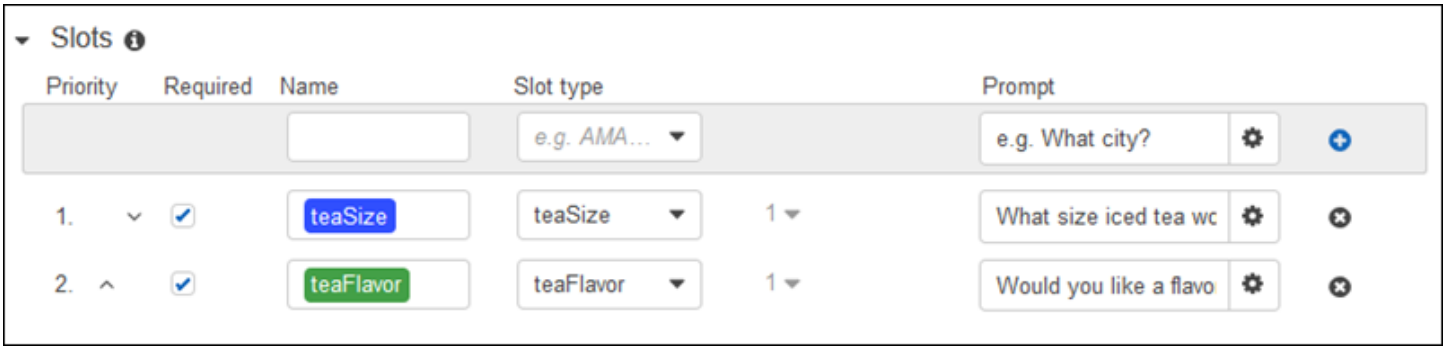
Você configura cartões de resposta ao criar uma intenção. Você pode definir um cartão de resposta estático no momento da criação usando o console ou a operação PutIntent. Ou você pode definir um cartão de resposta dinâmica em runtime em uma função do Lambda. Se você definir cartões de resposta estática e dinâmica, a o cartão de resposta dinâmica terá precedência.

O Amazon Lex envia cartões de resposta no formato que o cliente compreende. Ele transforma cartões de resposta para Facebook Messenger, Slack e Twilio. Para outros clientes, o Amazon Lex envia uma estrutura JSON na resposta PostText. Por exemplo, se o cliente for o Facebook Messenger, o Amazon Lex transformará o cartão de resposta para um modelo genérico. Para obter mais informações sobre os modelos genéricos do Facebook Messenger, consulte Modelo genérico no site do Facebook. Para ver um exemplo de estrutura JSON, consulte Geração dinâmica de cartões de resposta.

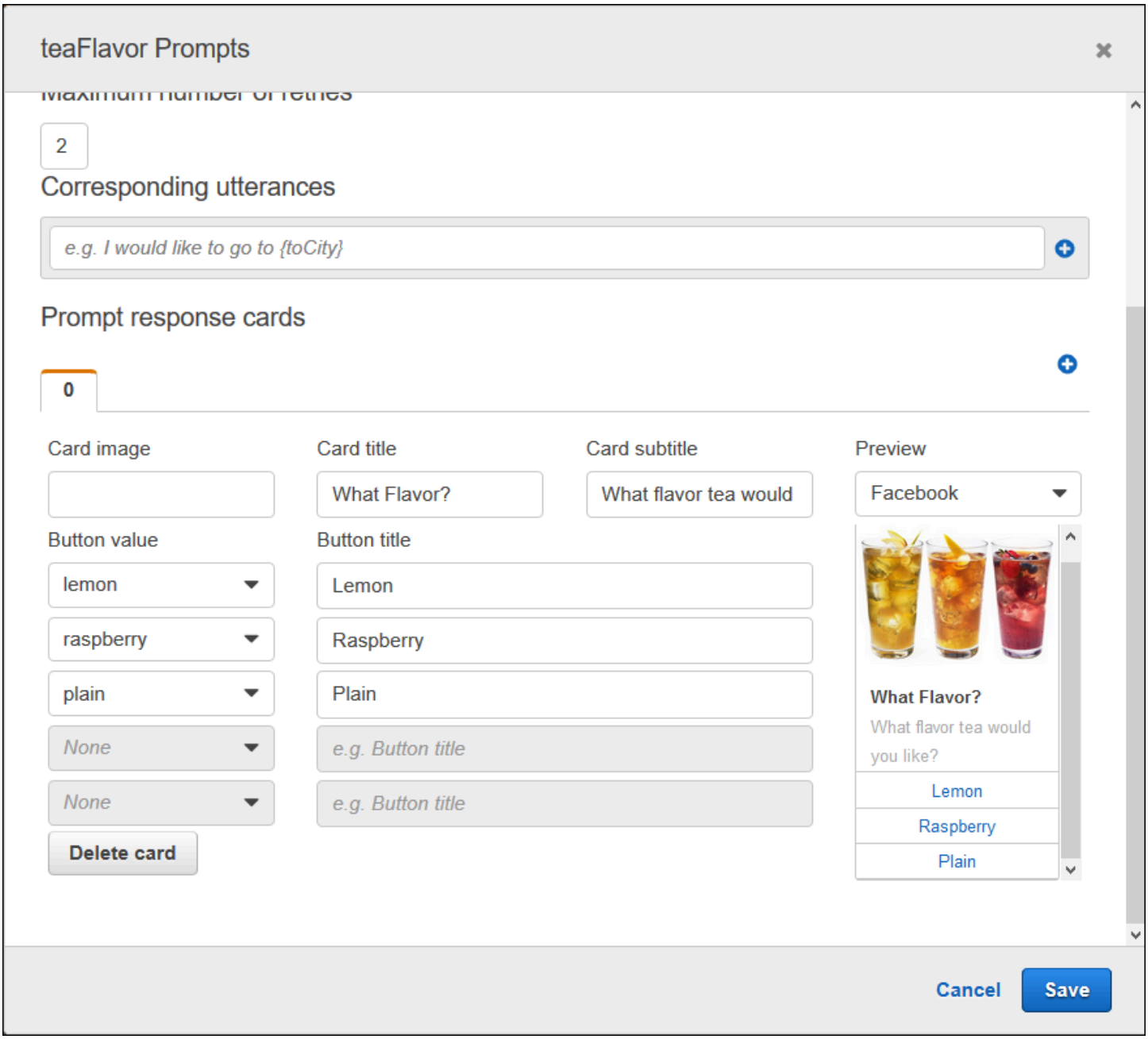
Você pode usar cartões de resposta somente com a operação PostText. Você não pode usar cartões de resposta com a operação PostContent.

Definição de cartões de resposta estática

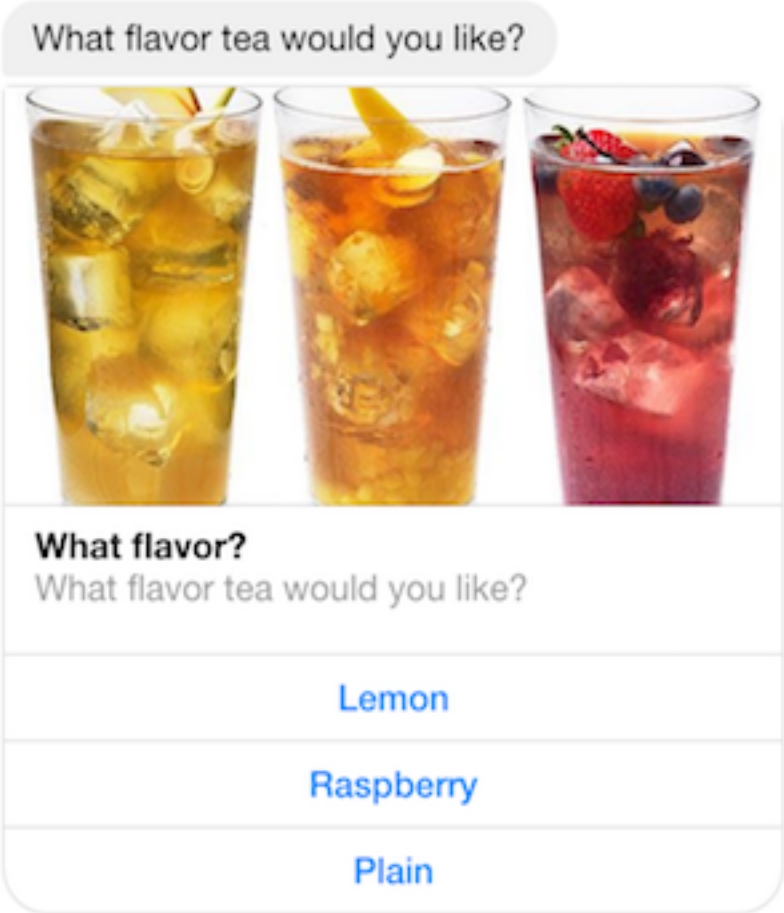
Defina cartões de resposta estática com a operação PutBot ou o console do Amazon Lex ao criar uma intenção. Um cartão de resposta estática é definido ao mesmo tempo que a intenção. Use um cartão de resposta estática quando as respostas são fixas. Suponha que você está criando um bot com uma intenção que tem um slot para sabor. Ao definir o slot de sabor, você especifica prompts, como mostrado na seguinte captura de tela do console:



Ao definir prompts, você tem a opção de associar um cartão de resposta e definir detalhes na operação PutBot ou no console do Amazon Lex, conforme mostrado no exemplo a seguir:



Agora, suponha que você integrou o bot ao Facebook Messenger. O usuário pode clicar nos botões para escolher uma sabor, como mostrado na ilustração a seguir:



Para personalizar o conteúdo de um cartão de resposta, você pode consultar atributos da sessão. Em runtime, o Amazon Lex substitui essas referências pelos valores adequados dos atributos de sessão. Para obter mais informações, consulte Definição dos atributos da sessão. Para obter um exemplo, consulte Exemplo: uso de um cartão de resposta.

Geração dinâmica de cartões de resposta

Para gerar cartões de resposta dinamicamente em runtime, use a função do Lambda de inicialização e validação para a intenção. Use um cartão de resposta dinâmica quando as respostas forem determinadas em runtime na função do Lambda. Em resposta à entrada do usuário, a função do Lambda gera um cartão de resposta e o retorna na seção `dialogAction` da resposta. Para obter mais informações, consulte Formato de resposta.

A seguir, veja uma resposta de uma função do Lambda; parcial que mostra o elemento `responseCard`. Ele gera uma experiência do usuário semelhante à mostrada na seção anterior.

```
responseCard: {
```

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

Para obter um exemplo, consulte Agendar uma consulta.

Gerenciar contexto da conversa

Contexto de conversa são as informações que um usuário, seu aplicativo 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 pelo aplicativo cliente e atributos de sessão que o aplicativo 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 PutIntent. Você só pode usar contextos na localidade em inglês (EUA) (en-US) e somente se definir o parâmetro `enableModelImprovements` para `true` quando criou o bot com a operação PutBot.

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 ao seu aplicativo na resposta da operação PostText ou PostContent, 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 `PostContent` ou `PostText`.

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 obter mais informações, consulte Evento de entrada de função do Lambda e formato de resposta.

Por exemplo, suponha que você tenha a intenção de reservar um carro alugado configurado para retornar um contexto de saída chamado "book_car_fulfilled". Quando a intenção é cumprida, o Amazon Lex define a variável de contexto de saída "book_car_fulfilled". Como "book_car_fulfilled" é um contexto ativo, uma intenção com o contexto "book_car_fulfilled" definido como um contexto de entrada agora é considerada para reconhecimento, desde que a declaração do usuário seja reconhecida como uma tentativa de obter essa intenção. Você pode usar isso para intenções 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 de slot para a intenção cumprida. Você pode usar esses parâmetros para preencher previamente os valores de slot 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 PutIntent. 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 PostText ou PostContent.

Veja a seguir como atribuir um contexto de saída a uma intenção usando o console.

Ao definir um contexto de saída, você também define seu tempo de vida, a duração ou o número de turnos em que o contexto é incluído nas respostas do Amazon Lex. Um turno é uma solicitação do seu aplicativo para o Amazon Lex. Depois que o número de turnos ou o tempo expirar, o contexto não ficará mais ativo.

Seu aplicativo pode usar o contexto de saída conforme necessário. Por exemplo, seu aplicativo pode usar o contexto de saída para:

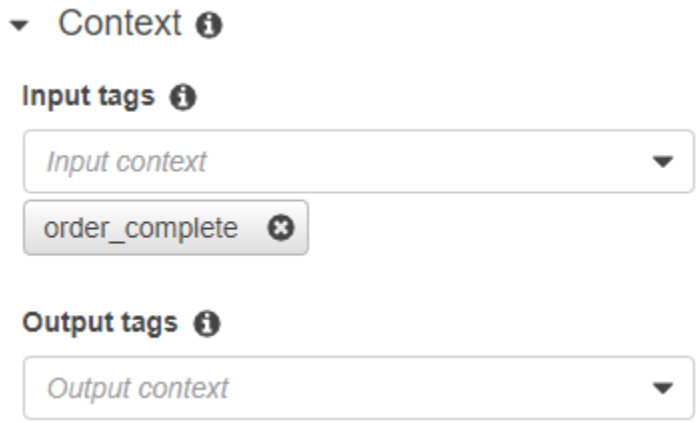
- Altere o comportamento do aplicativo com base no contexto. Por exemplo, um aplicativo de viagens pode ter uma ação diferente para o contexto "book_car_fulfilled" e "rental_hotel_fulfilled".

- Retorne o contexto de saída para o Amazon Lex como contexto de entrada para a próxima declaração. Se o Amazon Lex reconhecer a declaração 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 `PutIntent`. Uma intenção pode ter mais de um contexto de entrada. Veja a seguir como atribuir um contexto de entrada a uma intenção usando o console.



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 PostText, PostContent 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 obter 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.

Você pode usar o seguinte como fonte para seus valores padrão.

- Caixa de diálogo anterior (contextos): #context-name.parameter-name
- Atributos da sessão: [attribute-name]
- Atributos de solicitação: <attribute-name>
- Valor fixo: qualquer valor que não corresponda ao anterior

Ao usar a operação PutIntent 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": [
    {
        "name": "reservation-start-date",
        "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-fulfilled "" estiver ativo, o valor do parâmetro “StartDate” será usado como o valor padrão.
2. Se o contexto book-car-fulfilled "" não estiver ativo ou se o parâmetro “StartDate” não estiver definido, o valor do atributo de sessão reservationStartDate "" será usado como o 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 do aplicativo que são passadas entre o bot e o aplicativo cliente durante uma sessão. O Amazon Lex passa atributos da sessão para todas as funções função 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 o aplicativo cliente. Por exemplo:

- No exemplo Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)., o bot usa o atributo de sessão `price` para manter o preço das flores. A função do Lambda define esse atributo com base nos tipos de flores encomendados. Para obter mais informações, consulte Etapa 5 (opcional): Revise os detalhes do fluxo de informações (console).
- No exemplo Reservar uma viagem, o bot usa o atributo de sessão `currentReservation` para manter uma cópia dos dados do tipo de slot durante a conversa para fazer uma reserva em hotel ou alugar um carro. Para obter mais informações, consulte Detalhes do fluxo de informações.

Use atributos de sessão em suas funções do Lambda para inicializar um bot e personalizar prompts e cartões de resposta. Por exemplo:

- Inicialização: em um bot de pedido de pizza, o aplicativo cliente passa o local do usuário como um atributo de sessão na primeira chamada à operação PostContent ou PostText. Por exemplo, `"Location": "111 Maple Street"`. A função do Lambda usa essas informações para encontrar a pizzaria mais próxima para fazer o pedido.
- Personalizar prompts: configure prompts e cartões de resposta para fazer referência a atributos de sessão. Por exemplo, “Ei [FirstName], quais coberturas você gostaria?” Se você passar o nome do usuário como um atributo de sessão (`{"FirstName": "Jo"}`), o Amazon Lex substituirá o nome pelo espaço reservado. Em seguida, ele envia um prompt personalizado para o usuário: "Oi, Jo, quais coberturas você quer?"

Os atributos da sessão permanecem durante a vigência da sessão. Eles são criptografados e armazenados pelo Amazon Lex até que o final da sessão. O cliente pode criar atributos

de sessão em uma solicitação usando a operação PostContent ou PostText com o campo `sessionAttributes` definido como 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 o aplicativo 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 `PostContent` ou `PostText` 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 ao aplicativo cliente.

Se o aplicativo cliente ou uma função do Lambda passar atributos de sessão, o Amazon Lex atualizará as informações dos 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, você cria um string-to-string mapa dos atributos. As considerações a seguir mostram como mapear atributos de sessão:

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

Para a operação `PostText`, insira o mapa no corpo da solicitação usando o campo `sessionAttributes`, como a seguir:

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

Para a operação `PostContent`, codifique o mapa em base64 e o envie como o cabeçalho `x-amz-lex-session-attributes`.

Se você estiver enviando dados binários ou estruturados em um atributo de sessão, deve primeiro transformar os dados em uma string simples. Para obter mais informações, consulte Configuração de atributos complexos.

Definição de atributos de solicitação

Atributos de solicitação contêm informações específicas da solicitação e aplicam-se apenas à solicitação atual. Um aplicativo 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 uma the section called "PostContent" ou no campo `requestAttributes` em uma solicitação the section called "PostText". Como os atributos de solicitação não são persistentes entre as solicitações como os atributos de sessão, eles não são retornados em respostas `PostContent` ou `PostText`.

> **Note**
>
> Para enviar informações que são mantidas nas solicitações, use atributos de sessão.

O namespace `x-amz-lex:` é reservado para os atributos de solicitação predefinidos. Não crie atributos de solicitação com o prefixo `x-amz-lex:`.

Definição de atributos de solicitação predefinidos

O Amazon Lex fornece atributos de solicitação predefinidos para gerenciar a maneira como ele processa as informações enviadas a seu bot. Os atributos não são mantidos durante toda a sessão. Por isso, você deve enviar os atributos predefinidos em cada solicitação. Todos os atributos predefinidos estão no namespace `x-amz-lex:`.

Além dos atributos predefinidos a seguir, o Amazon Lex fornece atributos predefinidos para plataformas de sistemas de mensagens. Para obter uma lista desses atributos, consulte Implantação de um bot do Amazon Lex em uma plataforma de sistema de mensagens.

Como configurar o tipo de resposta

Se tiver dois aplicativos clientes com diferentes recursos, talvez seja necessário restringir o formato das mensagens em uma resposta. Por exemplo, talvez você queira restringir as mensagens enviadas a um cliente web a textos sem formatação, mas permitir que um cliente móvel use o texto sem formatação e Speech Synthesis Markup Language (SSML). Para definir o formato das mensagens retornadas pelas operações PostContent e PostText, use o atributo de solicitação `x-amz-lex:accept-content-types"`.

Você pode definir o atributo para qualquer combinação dos seguintes tipos de mensagem:

- `PlainText`: a mensagem contém texto sem formatação UTF-8.
- `SSML`: a mensagem contém texto formatado para saída de voz.
- `CustomPayload`: a mensagem contém um formato personalizado que você criou para seu cliente. Você pode definir a carga útil para atender às necessidades de seu aplicativo.

O Amazon Lex retorna apenas mensagens com o tipo especificado no campo `Message` da resposta. Você pode definir mais de um valor. Para isso, separe os valores com uma vírgula. Se estiver usando grupos de mensagens, cada um deverá conter pelo menos uma mensagem do tipo especificado. Do contrário, você receberá um erro `NoUsableMessageException`. Para obter mais informações, consulte Grupos de mensagens.

> **Note**
>
> O atributo de solicitação `x-amz-lex:accept-content-types` não tem efeito sobre o conteúdo do corpo HTML. O conteúdo da resposta de uma operação `PostText` sempre é texto sem formatação UTF-8. O corpo da resposta de uma operação `PostContent` contém dados no formato definido no cabeçalho `Accept` da solicitação.

Definição do fuso horário preferido

Para definir o fuso horário usado para resolver datas de forma que seja relativo ao fuso horário do usuário, use o atributo de solicitação `x-amz-lex:time-zone`. Se um fuso horário não está especificado no atributo `x-amz-lex:time-zone`, o padrão depende da região que você está usando para o seu bot.

Região	Fuso horário padrão
Leste dos EUA (N. da Virgínia)	`America/New_York`
Oeste dos EUA (Oregon)	`America/Los_Angeles`
Ásia-Pacífico (Singapura)	`Asia/Singapore`
Ásia-Pacífico (Sydney)	`Australia/Sydney`
Ásia-Pacífico (Tóquio)	`Asia/Tokyo`

Região	Fuso horário padrão
Europa (Frankfurt)	`Europe/Berlin`
Europa (Irlanda)	`Europe/Dublin`
Europa (Londres)	`Europe/London`

Por exemplo, se o usuário responde `tomorrow` em resposta ao prompt "Quando você gostaria que seu pacote fosse entregue?" a data em que o pacote será entregue depende do fuso horário do usuário. Por exemplo, quando ele é 01:00 de 16 de setembro em Nova York, são 22:00 de 15 de setembro em Los Angeles. Se uma pessoa no Leste dos EUA (Norte da Virgínia) encomendar um pacote a ser entregue "amanhã" usando o fuso horário padrão, o pacote será entregue no dia 17, e não no dia 16. No entanto, se você definir o atributo de solicitação `x-amz-lex:time-zone` para `America/Los_Angeles`, o pacote será entregue no dia 16.

Você pode definir o atributo para qualquer um nomes de fuso horário IANA (Internet Assigned Number Authority). Para ver a lista de nomes de fuso horário, consulte a Lista de fusos horários do banco de dados tz na Wikipédia.

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 cabeçalho `amz-lex-request-attributes` de uma solicitação `PostContent` ou no campo `requestAttributes` de uma solicitação `PostText`.

Para enviar atributos de solicitação para o Amazon Lex, você cria um string-to-string mapa 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 obter 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 ou ela pode fornecer os dados de slot restantes e concluir a compra.

No console do Amazon Lex, você define o tempo limite de sessão ao criar um bot. Com a interface de linha de comando da AWS (AWS CLI) ou a API, você define o tempo limite ao criar ou atualizar um bot com a PutBot operação definindo o campo TTLInIdleSession Seconds.

Compartilhamento de informações entre intenções

O Amazon Lex é compatível com o compartilhamento de informações entre intenções. Para compartilhar entre intenções, use atributos de sessão.

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.

Para obter um exemplo de compartilhamento de informações entre intenções, consulte Reservar uma viagem.

Configuração de atributos complexos

Os atributos de sessão e solicitação são string-to-string mapas 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"
         }
```

```
        }
    ]
}
```

Essa matriz de dados não se traduz bem em um string-to-string mapa. Nesse caso, você pode transformar um objeto em uma string simples para que você possa enviá-lo ao seu bot com as operações PostContent e PostText.

Por exemplo, se você estiver usando JavaScript, você pode usar a `JSON.stringify` operação para converter um objeto em JSON e a `JSON.parse` operação para converter texto JSON em um JavaScript objeto:

```
// 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 `PostContent` operação, você deve codificar os atributos em base64 antes de adicioná-los ao cabeçalho da solicitação, conforme mostrado no código a seguir: JavaScript

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

Você pode enviar dados binários para as operações `PostContent` e `PostText` 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"
}
```

Usar pontuações de confiança

Quando um usuário faz uma declaração, o Amazon Lex usa a compreensão de linguagem natural (NLU) para entender a solicitação do usuário e retornar a intenção correta. Por padrão, o Amazon Lex retorna a intenção mais provável definida pelo seu bot.

Em alguns casos, pode ser difícil para o Amazon Lex determinar a intenção mais provável. Por exemplo, o usuário pode fazer uma declaração ambígua 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 de uma lista de intenções alternativas. Uma pontuação de confiança é uma classificação fornecida pelo Amazon Lex que mostra o grau de confiança de que uma intenção é a intenção correta.

Para determinar a diferença entre duas intenções alternativas, 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, a primeira intenção provavelmente está 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 em seu aplicativo.

Você também pode usar pontuações de confiança para criar aplicativos de teste que determinam se as mudanças nas declarações de uma intenção fazem diferença no comportamento do bot. Por exemplo, você pode obter as pontuações de confiança das intenções de um bot usando um conjunto de declarações e, em seguida, atualizar as intenções com novas declarações. Em seguida, você pode verificar as pontuações de confiança para ver se houve uma melhora.

As pontuações de confiança que o Amazon Lex retorna são valores comparativos. Você não deve confiar neles como uma pontuação absoluta. Os valores podem mudar com base em melhorias no Amazon Lex.

Quando você usa pontuações de confiança, o Amazon Lex retorna a intenção mais provável e até 4 intenções alternativas com suas pontuações associadas em cada resposta. Se todas as pontuações de confiança forem menores que um limite, o Amazon Lex incluirá o `AMAZON.FallbackIntent`, o `AMAZON.KendraSearchIntent`, o ou ambos, se você os tiver configurado. É possível usar o limite padrão ou definir seu próprio limite.

O código JSON a seguir mostra o `alternativeIntents` campo na resposta da PostText operação.

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

```
        "slots": {
           "string" : "string"
        }
     }
  ],
```

Defina o limite ao criar ou atualizar um bot. Você pode usar a API ou o console do Amazon Lex. Para as regiões listadas abaixo, você precisa se inscrever para permitir melhorias de precisão e pontuações de confiança. No console, escolha pontuações de confiança na seção Opções avançadas. Usando a API, defina o parâmetro `enableModelImprovements` ao chamar a operação PutBot.

- Leste dos EUA (Norte da Virgínia) (us-east-1)
- Oeste dos EUA (Oregon) (us-west-2)
- Ásia-Pacífico (Sydney) (ap-southeast-2)
- Europa (Irlanda) (eu-west-1)

Em todas as outras regiões, melhorias na precisão e suporte à pontuação de confiança estão disponíveis por padrão.

Para alterar o limite de confiança, defina-o no console ou usando a operação PutBot. O limite deve ser um número entre 1,00 e 0,00.

Para usar o console, defina o limite de confiança ao criar ou atualizar seu bot.

Para definir o limite de confiança ao criar um bot (Console)

- Em Criar seu bot, insira um valor no campo Limite da pontuação de confiança.

Para atualizar o limite de confiança (Console)

1. Na lista de bots, escolha o bot a ser atualizado.
2. Escolha a guia Configurações.
3. Na barra de navegação à esquerda, selecione Geral.
4. Atualize o valor no campo Limite da pontuação de confiança.

Para definir ou atualizar o limite de confiança (SDK)

- Defina o parâmetro `nluIntentConfidenceThreshold` da operação PutBot. O código JSON a seguir mostra o parâmetro que está sendo definido.

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

Gerenciamento de sessões

Para alterar a intenção que o Amazon Lex usa em uma conversa com o usuário, você pode usar a resposta da função Lambda do gancho de código de diálogo ou usar o APIs gerenciamento de sessões em seu aplicativo personalizado.

Usar um URL da função do Lambda

Quando você usa uma função do Lambda, o Amazon Lex 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 identificou como a intenção mais provável para a declaração 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 atribuiu à intenção.

Para usar uma intenção alternativa, você a especifica na ação de diálogo `ConfirmIntent` ou `ElicitSlot` na função do Lambda.

Para obter mais informações, consulte Uso de funções do Lambda.

Usar a API de gerenciamento de sessões

Para usar uma intenção diferente da intenção atual, utilize a operação PutSession. Por exemplo, se você decidir que a primeira alternativa é preferível à intenção escolhida pelo Amazon Lex, use a operação `PutSession` para alterar as intenções de modo que a próxima intenção com a qual o usuário vai interagir seja aquela que você selecionou.

Para obter mais informações, consulte Gerenciamento de sessões com a API do Amazon Lex.

Logs de conversa

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 criar logs de texto para a operação PostText. É possível criar logs de texto e áudio para a operação PostContent. 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. Você pode ativar a criptografia de registros de texto e áudio usando o gerenciamento de AWS KMS clientes CMKs.

Para configurar o registro em log, use o console ou a operação PutBotAlias. Não é possível criar logs de conversas para o alias `$LATEST` do seu bot ou para o bot de teste disponível no console do Amazon Lex. Depois de ativar os registros de conversação para um alias, PostContent ou a PostText operação desse 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
- Criptografar 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 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 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 IAM AWS Management Console, 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. Para obter informações sobre como criar uma política com o console, consulte Criar políticas na guia JSON no Guia do usuário do AWS Identity and Access Management.

> **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 como uma entidade confiável ao perfil. Isso permite que o Lex assuma o perfil para entregar logs aos recursos configurados para logs de conversa.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "lex.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 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 criada 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 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. Para permitir que o usuário passe o perfil ao Amazon Lex, é necessário conceder a permissão de `PassRole` ao usuário, ao perfil ou ao grupo.

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": "lex.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 `conversationLogs` da operação `PutBotAlias`. 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. 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. Para obter mais informações sobre como criar um bucket do Amazon S3, consulte Criar um bucket no Guia de conceitos básicos do Amazon Simple Storage Service.

É necessário fornecer um perfil do IAM com políticas que permitam ao Amazon Lex gravar no grupo de logs ou no bucket configurado. Para obter mais informações, consulte Criar um perfil e políticas do IAM para logs de conversa.

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 a seguir:

```
aws iam create-service-linked-role \
    --aws-service-name lex.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 https://console.aws.amazon.com/lex.
2. Na lista, escolha um bot.
3. Selecione a guia Configurações e, no menu esquerdo, selecione Logs de conversa.
4. Na lista de aliases, escolha o ícone de configurações para o alias para o qual você deseja configurar logs de conversa.
5. Selecione se deseja registrar texto, áudio ou ambos.
6. Para registro de texto, insira o nome do grupo de CloudWatch registros do Amazon Logs.
7. Para registro de áudio em log, 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 uma perfil do IAM com as permissões necessárias.
10. Escolha Salvar para iniciar o registro em log de conversas.

Como ativar logs de texto usando a API

1. Chame a operação PutBotAlias com uma entrada no membro `logSettings` do campo `conversationLogs`
   - Defina o membro `destination` como `CLOUDWATCH_LOGS`
   - Defina o membro `logType` como `TEXT`
   - Defina o `resourceArn` membro como o Amazon Resource Name (ARN) do grupo de CloudWatch registros de registros que é o destino dos registros
2. Defina o membro `iamRoleArn` do campo `conversationLogs` como o nome do recurso da Amazon (ARN) de um perfil do IAM que tenha as permissões necessárias para habilitar logs de conversa nos recursos especificados.

Como ativar logs de áudio usando a API

1. Chame a operação PutBotAlias com uma entrada no membro `logSettings` do campo `conversationLogs`
   - Defina o membro `destination` como `S3`
   - Defina o membro `logType` como `AUDIO`
   - Defina o membro `resourceArn` como o ARN do bucket do Amazon S3 onde os logs de áudio são armazenados
   - Opcional. Para criptografar registros de áudio com uma AWS KMS chave específica, defina o `kmsKeyArn` membro do ARN da chave usada para criptografia.
2. Defina o membro `iamRoleArn` do campo `conversationLogs` como o nome do recurso da Amazon (ARN) de um perfil do IAM que tenha as permissões necessárias para habilitar logs de conversa nos recursos especificados.

Desativar logs de conversa

Como desativar os logs usando o console

1. Abra o console do Amazon Lex https://console.aws.amazon.com/lex.
2. Na lista, escolha um bot.
3. Selecione a guia Configurações e, no menu esquerdo, selecione Logs de conversa.
4. Na lista de aliases, escolha o ícone de configurações para o alias para o qual você deseja configurar logs de conversa.
5. Desmarque a verificação de texto, áudio ou ambos para desativar o registro em log.
6. Escolha Salvar para interromper o registro em log de conversas.

Como desativar os logs usando a API

- Chame a operação `PutBotAlias` sem o campo `conversationLogs`.

Como desativar os logs de texto usando a API

- - Se você estiver registrando áudio
    - Chame a operação PutBotAlias com uma entrada `logSettings` somente para `AUDIO`.

    - A chamada para a operação `PutBotAlias` não deve ter uma entrada `logSettings` para `TEXT`.
  - Se você não estiver registrando áudio em log
    - Chame a operação PutBotAlias sem o campo `conversationLogs`.

Como desativar os logs de áudio usando a API

- - Se você estiver registrando texto em log
    - Chame a operação PutBotAlias com uma entrada `logSettings` somente para `TEXT`.
    - A chamada para a operação `PutBotAlias` não deve ter uma entrada `logSettings` para `AUDIO`.
  - Se você não estiver registrando texto em log
    - Chame a operação PutBotAlias sem o campo `conversationLogs`.

Criptografar logs de conversa

É possível usar criptografia para ajudar a proteger o conteúdo dos logs de conversa. Para registros de texto e áudio, você pode usar o gerenciamento AWS KMS do cliente CMKs para criptografar dados no seu grupo de registros de CloudWatch registros e no bucket do S3.

> **Note**
>
> O Amazon Lex oferece suporte somente para sistemas simétricos CMKs. Não use uma CMK assimétrica para criptografar dados.

Você ativa a criptografia usando uma AWS KMS chave no grupo de CloudWatch registros de registros que o Amazon Lex usa para registros de texto. Você não pode fornecer uma AWS KMS chave nas configurações de registro para ativar a AWS KMS criptografia do seu grupo de registros. Para obter mais informações, consulte Criptografar dados de log em CloudWatch registros usando AWS KMS o Guia do usuário do Amazon CloudWatch Logs.

Para registros de áudio, você usa a criptografia padrão em seu bucket do S3 ou especifica uma AWS KMS chave para criptografar seus objetos de áudio. Mesmo que seu bucket do S3 use criptografia padrão, você ainda pode especificar uma AWS KMS chave diferente para criptografar seus objetos

de áudio. Para obter mais informações, consulte Criptografia padrão do Amazon S3 para buckets do S3 no Guia do desenvolvedor do Amazon Simple Storage Service.

O Amazon Lex exige AWS KMS permissões se você optar por criptografar seus registros de áudio. É necessário associar políticas adicionais ao perfil do IAM usado para logs de conversa. Se você usa criptografia padrão em seu bucket do S3, sua política deve conceder acesso à AWS KMS chave configurada para esse bucket. Se você especificar uma AWS KMS chave nas configurações do registro de áudio, deverá conceder acesso a essa chave.

Se você não criou uma função para logs de conversa, consulte Políticas do IAM para logs de conversa.

Para criar uma política do IAM para usar uma AWS KMS chave para criptografar registros de áudio

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

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

2. No AWS CLI, crie a política do IAM que concede permissão para usar a AWS KMS chave para criptografar registros de áudio.

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

```
    --role-name role-name
```

Visualização de registros de texto no Amazon CloudWatch Logs

O Amazon Lex armazena registros de texto para suas conversas no Amazon CloudWatch Logs. Para ver os registros, você pode usar 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

1. Abra o console do Amazon Lex https://console.aws.amazon.com/lex.
2. Na lista, escolha um bot.
3. Escolha a guia Configurações e, no menu esquerdo, selecione Logs de conversa.
4. Escolha o link em Registros de texto para ver os registros do alias no CloudWatch console.

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. Localize o prefixo de fluxo de log para seus logs no console do Amazon Lex ou usando a operação GetBotAlias.

As entradas de log para uma declaração 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.

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

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

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

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

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.

Você deve ter uma função e uma política definidas para permitir que o Amazon Lex grave em CloudWatch Logs. Para ter mais informações, consulte Políticas do IAM para logs de conversa.

Acessar logs de áudio no Amazon S3

O Amazon Lex armazena logs de áudio para suas conversas em um bucket do S3.

Para acessar logs de áudio usando o console

1. Abra o console do Amazon Lex https://console.aws.amazon.com/lex.
2. Na lista, escolha um bot.
3. Escolha a guia Configurações e, no menu esquerdo, selecione Logs de conversa.
4. Escolha o link em Logs de áudio para acessar os logs do alias no console do Amazon S3.

Também 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 ou no campo `resourcePrefix` na resposta da operação `GetBotAlias`.

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 fornece quatro métricas no namespace `AWS/Lex` para logs de conversa:

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

Para obter mais informações, consulte CloudWatch Métricas para registros de conversas.

As métricas de sucesso mostram que o Amazon Lex gravou com êxito seus logs de áudio ou texto em seus destinos.

As métricas de falha mostram que o Amazon Lex não conseguiu entregar os logs de áudio ou texto ao destino especificado. Normalmente, isso é um erro de configuração. Quando suas métricas de falha estiverem acima de zero, verifique o seguinte:

- Certifique-se de que o Amazon Lex 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 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 e pertença à sua conta.
- Se você estiver usando uma AWS KMS chave para criptografia S3, certifique-se de que não haja políticas que impeçam o Amazon Lex de usar sua chave e certifique-se de que a função do IAM fornecida tenha as AWS KMS permissões necessárias. Para obter mais informações, consulte Políticas do IAM para logs de conversa.

Gerenciamento de sessões com a API do Amazon Lex

Quando um usuário começa uma conversa com o bot, o Amazon Lex cria uma sessão. As informações trocadas entre o aplicativo e o Amazon Lex compõem o estado da sessão para a conversa. Quando você faz uma solicitação, a sessão é identificada por uma combinação do nome do bot e um identificador de usuário especificado por você. Para obter mais informações sobre o identificador de usuário, consulte o campo `userId` na operação PostContent ou PostText.

A resposta de uma operação de sessão inclui um identificador exclusivo que identifica uma sessão específica com um usuário. Você pode usar esse identificador durante o teste ou para ajudar a solucionar problemas no bot.

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 a próxima declaração.

Há duas maneiras de atualizar o estado da sessão. A primeira é usar uma função do Lambda com a operação `PostContent` ou `PostText` que é chamada após cada turno da conversa. Para obter mais informações, consulte Uso de funções do Lambda. A outra é usar a API de runtime do Amazon Lex no seu aplicativo para fazer alterações no estado da sessão.

A API de runtime do Amazon Lex oferece 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 das três últimas intenções com as quais o usuário interagiu.

A operação `PutSession` permite manipular diretamente o estado da sessão atual. Você pode definir o tipo de ação de diálogo que o bot realizará em seguida. Isso oferece a você controle sobre o fluxo da conversa com o bot. Defina o campo de ação de diálogo `type` como `Delegate` para que o Amazon Lex 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. É possível usar a operação `GetSession` para obter o estado do diálogo da intenção anterior do Amazon Lex e usar as informações para definir o estado do diálogo da intenção.

A resposta da operação `PutSession` contém as mesmas informações que a operação `PostContent`. Você pode usar essas informações para solicitar a próxima informação ao usuário, da mesma forma que faria com a resposta da operação `PostContent`.

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` como `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.

Alternância entre 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 utilizando a nova intenção.

Como retomar uma intenção anterior

Para retomar uma intenção anterior, use a operação `GetSession` para obter o resumo da intenção e, depois, use a operação `PutSession` para definir a intenção como seu estado de diálogo anterior.

- Chame a operação `GetSession`. A resposta da operação inclui um resumo do estado de diálogo das últimas três intenções com as quais o usuário interagiu.
- Usando as informações do resumo de intenção, 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.

É possível usar o parâmetro `checkpointLabel` da operação `PutSession` a fim de rotular uma intenção para que seja possível localizá-la mais tarde. Por exemplo, um bot que solicita informações a um cliente pode entrar em uma intenção `Waiting` enquanto o cliente coleta as informações. O bot cria um rótulo de ponto de verificação para a intenção atual e inicia a intenção `Waiting`. Quando o cliente retornar, o bot poderá encontrar a intenção anterior usando o rótulo de ponto de verificação e alternar de volta.

A intenção deve estar presente na estrutura `recentIntentSummaryView` retornada pela operação `GetSession`. Se você especificar um rótulo de ponto de verificação na solicitação da operação `GetSession`, ele retornará um máximo de três intenções com esse rótulo de ponto de verificação.

- Use a operação `GetSession` para obter o estado atual da sessão.
- Use a operação `PutSession` para adicionar um rótulo de ponto de verificação à última intenção. Se necessário, é possível usar esta chamada `PutSession` para alternar para uma intenção diferente.
- Quando for o momento de retornar à intenção rotulada, chame a operação `GetSession` para retornar uma lista de intenções recente. É possível usar o parâmetro `checkpointLabelFilter` para que o Amazon Lex retorne somente as intenções com o rótulo do ponto de verificação especificado.

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.

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`. Você também poderá definir os campos `message` e `messageFormat` se desejar alterar a mensagem usada pelo Amazon Lex para obter o valor de slot.

Opções de implantação de bot

No momento, o Amazon Lex fornece as seguintes opções de implantação do bot:

- SDK móvel da AWS — Você pode criar aplicativos móveis que se comunicam com o Amazon Lex usando o AWS Mobile SDKs.
- Facebook Messenger: você pode integrar sua página do Facebook Messenger ao bot do Amazon Lex para que os usuários finais no Facebook possam se comunicar com o bot. Na implementação atual, essa integração oferece suporte apenas as mensagens de entrada de texto.
- Slack: você pode integrar seu bot Amazon Lex a um aplicativo de mensagens Slack.
- Twilio: você pode integrar seu bot do Amazon Lex ao serviço Twilio Simple Messaging Service (SMS).

Para obter exemplos, consulte Implantação de bots do Amazon Lex.

Intenções integradas e tipos de slot

Para facilitar a criação dos bots, o Amazon Lex permite que você use as intenções integradas e os tipos de slot padrão.

Tópicos

- Intenções integradas
- Tipos de slot integrados

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 a amostra de declarações.

Na implementação atual, você não pode:

- Adicionar ou remover amostra de declarações da intenção básica
- Configurar slots para intenções integradas

Para adicionar uma intenção integrada a 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 o bot ao qual adicionar a intenção integrada.
3. No painel de navegação, escolha o sinal de adição (+) ao lado de Intenções.
4. Em Adicionar intenção, escolha Pesquisar intenções existentes.
5. Na caixa Intenções de pesquisa, insira o nome da intenção integrada a ser adicionada ao seu bot.
6. Em Copiar intenção integrada, dê um nome à intenção e escolha Adicionar.
7. Configure a intenção conforme necessário para o seu bot.

Tópicos

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

> **Note**
>
> Para a localidade em inglês (EUA) (en-US), o Amazon Lex oferece suporte a intenções integradas padrão da Alexa. Para obter uma lista de intenções integradas, consulte Intenções integradas padrão no Alexa Skills Kit.
> O Amazon Lex não oferece suporte às seguintes intenções:
>
> - `AMAZON.YesIntent`
> - `AMAZON.NoIntent`
> - As intenções na Biblioteca de intenções integradas do Alexa Skills Kit

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 do tipo de slot e outros atributos antes de encerrar a interação com o usuário.

Declarações 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 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 invocará a intenção de fallback para lidar com a resposta.

Inclua uma intenção de fallback adicionando o tipo de intenção `AMAZON.FallbackIntent` integrada ao bot. Você pode especificar a intenção usando a operação PutBot ou escolhendo a intenção na lista de intenções integradas no console.

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. Por exemplo, se o número máximo de tentativas para determinar uma intenção for 2, o bot retornará o prompt de esclarecimento do bot duas vezes antes de invocar a intenção de fallback.

O Amazon Lex 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 não consegue determinar qual intenção invocar.

A intenção de fallback é invocada quando:

- O bot não reconhece a entrada do usuário como uma intenção após o número configurado de tentativas para esclarecimento quando a conversa é iniciada.
- 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 pode usar o seguinte com uma intenção de fallback:

- Uma função do Lambda de atendimento
- Uma declaração de conclusão
- Um prompt de acompanhamento

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

- Enunciados
- Slots
- Uma função do Lambda de inicialização e validação
- Um prompt de confirmação

Se você tiver configurado uma instrução de anulação e uma intenção de fallback para um bot, o Amazon Lex usará a intenção de fallback. Se você precisar que seu bot tenha uma instrução de cancelamento, use a função de atendimento para que a intenção de fallback forneça o mesmo

comportamento que uma instrução de cancelamento. Para obter mais informações, consulte o parâmetro `abortStatement` da operação PutBot.

Usar prompts de esclarecimento

Se você fornecer ao bot um prompt de esclarecimento, o prompt será usado para solicitar uma intenção válida do usuário. O prompt de esclarecimento será repetido o número de vezes que você configurou. Depois disso, a intenção de fallback será invocada.

Se você não definir um prompt de esclarecimento ao criar um bot e o usuário não iniciar a conversa com uma intenção válida, o Amazon Lex chamará imediatamente sua intenção de fallback.

Quando você usa uma intenção de fallback sem um prompt de esclarecimento, o Amazon Lex não chama o fallback nestas circunstâncias:

- Quando o usuário responde a um prompt de acompanhamento, mas não fornece uma intenção. Por exemplo, em resposta a um prompt de acompanhamento que diz "Você quer mais alguma coisa hoje?", o usuário diz "Sim". O Amazon Lex retorna uma exceção 400 Bad Request porque não tem um prompt de esclarecimento para enviar ao usuário e obter uma intenção.
- Ao usar uma AWS Lambda função, você retorna um tipo `ElicitIntent` de diálogo. Como o Amazon Lex não tem um prompt de esclarecimento para obter uma intenção do usuário, ele retorna uma exceção 400 Bad Request.
- Ao usar a operação `PutSession`, envie um tipo de diálogo `ElicitIntent`. Como o Amazon Lex não tem um prompt de esclarecimento para obter uma intenção do usuário, ele retorna uma exceção 400 Bad Request.

Usar uma função do Lambda com uma intenção de fallback

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

- Retorna as informações de intenção para o aplicativo cliente.
- Chama a função do Lambda de atendimento. 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 `fulfillmentActivity` da operação PutIntent.

Se você usar a função do Lambda de atendimento em 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.

É possível executar qualquer ação em uma função do Lambda de intenção de fallback que possa ser executada na função de cumprimento para qualquer outra intenção. Para obter mais informações sobre como criar uma função de atendimento usando AWS Lambda, consulteUso de funções do Lambda.

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 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 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.

Declarações 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 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) (en-US) 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 chamará a intenção sempre que não conseguir determinar a declaração do usuário para um slot ou uma intenção. Por exemplo, se o bot estiver provocando uma resposta para um tipo de slot chamado "cobertura de pizza" e o usuário disser "O que é pizza?", o Amazon Lex chamará `AMAZON.KendraSearchIntent` para lidar com a pergunta. Se não houver resposta do Amazon Kendra, a conversa continuará conforme configurado no bot.

Quando `AMAZON.KendraSearchIntent` e `AMAZON.FallbackIntent` são usadas no mesmo bot, o Amazon Lex usa as intenções da seguinte forma:

1. O Amazon Lex chama `AMAZON.KendraSearchIntent`. A intenção chama a operação `Query` do Amazon Kendra.
2. Se o Amazon Kendra retornar uma resposta, o Amazon Lex exibirá o resultado para o usuário.
3. Se não houver resposta do Amazon Kendra, o Amazon Lex avisará novamente o usuário. A próxima ação depende da resposta do usuário.
   - Se a resposta do usuário contiver uma declaração reconhecida pelo Amazon Lex, 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 uma declaração reconhecida pelo Amazon Lex, o Amazon Lex fará outra chamada para a operação `Query`.

4. Se não houver resposta após o número configurado de novas tentativas, o Amazon Lex 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 chama o Amazon Kendra com a declaração 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 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 do diálogo. 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 do diálogo. É possível criar uma solicitação de consulta completa do Amazon Kendra enviada pelo Amazon Lex. 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 para o Amazon Kendra, é possível especificar uma consulta no campo `kendraQueryRequestPayload` na função do Lambda do diálogo. Se a consulta não for válida, o Amazon Lex 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 ao Amazon Lex assumir 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 criar o `AMAZON.KendraSearchIntent` usando o console Amazon Lex 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 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. 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 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/lex.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 precisa usar para chamar o Amazon Kendra deve ter as permissões `kendra:Query`. Quando você usa um perfil do IAM existente para obter 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 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.

Para especificar a função de tempo de execução da AMAZON. KendraSearchIntent

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 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 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 substituirá um valor antes de chamar o Amazon Kendra. Para mais informações sobre atributos de solicitação, consulte Definição de atributos de solicitação. Para 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 `conclusion` da intenção. A intenção deve ter uma instrução `conclusion`, a menos que uma função do Lambda de atendimento produza uma mensagem de conclusão.

O Amazon Kendra tem quatro tipos de respostas.

- `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.
- `x-amz-lex:kendra-search-response-document-<N>`: um trecho de um documento no índice relacionado ao texto da declaração.
- `x-amz-lex:kendra-search-response-document-link-<N>`: o URL de um documento no índice relacionado ao texto da declaração.
- `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 `conclusion` 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 criará uma consulta a partir da declaração 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, configura 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 com o `dialogAction` campo da resposta definido como`delegate`. O Amazon Lex 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 envia uma consulta ao Amazon Kendra, a resposta da consulta é retornada para a função do Lambda de cumprimento `AMAZON.KendraSearchIntent`. 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 usará a instrução `conclusion` para respostas.

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

Este exemplo cria um bot do Amazon Lex 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. Para criar o bot, você deve:

1. Criar um bot com o qual seus clientes vão interagir para obter respostas do bot.
2. Criar uma intenção personalizada. O bot requer pelo menos uma intenção com pelo menos uma declaração. Essa intenção permite que o bot crie, mas não seja usado de outra forma.
3. Adicionar a intenção `KendraSearchIntent` ao bot e configurá-lo para trabalhar com o índice do Amazon Kendra.
4. Testar o bot fazendo perguntas que são respondidas por documentos armazenados no índice do Amazon Kendra.

Antes de usar este exemplo, é necessário criar um índice do Amazon Kendra. Para obter mais informações, consulte Começar a usar um bucket do S3 (console) no Guia do desenvolvedor do Amazon Kendra.

Como criar um bot de perguntas frequentes

1. Faça login no 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.
4. Selecione Custom bot (Bot personalizado). Configure o bot da seguinte forma:
   - Nome do bot: dê ao bot um nome que indique sua finalidade, como **`KendraTestBot`**.
   - Voz de saída: escolha Nenhuma.
   - Tempo limite da sessão: insira **`5`**.
   - Análise de sentimento: escolha Não.
   - COPPA: escolha Não.
   - Armazenamento da declaração do usuário: escolha Não armazenar.
5. Escolha Criar.

Para criar um bot com êxito, é necessário criar pelo menos uma intenção com pelo menos uma amostra de declaração. Essa intenção é necessária para criar o bot do Amazon Lex, mas não é usada para a resposta de perguntas frequentes. A declaração para a intenção não deve se aplicar a nenhuma das perguntas feitas pelo cliente.

Como criar a intenção necessária

1. Na página Começar a usar o bot, escolha Criar intenção.
2. Em Adicionar intenção, escolha Criar intenção.
3. Na caixa de diálogo Criar intenção, dê um nome à intenção, como **`RequiredIntent`**.
4. Em Amostra de declarações, digite uma declaração, como **`Required utterance`**.
5. Selecione Salvar intenção.

Agora, crie a intenção de pesquisar um índice do Amazon Kendra e as mensagens de resposta que devem ser retornadas.

Para criar uma AMAZON. KendraSearchIntent mensagem de intenção e resposta

1. No painel de navegação, escolha o sinal de adição (+) ao lado de Intenções.
2. Em Adicionar intenção, escolha Pesquisar intenções existentes.
3. Na caixa Pesquisar intenções, insira **`AMAZON.KendraSearchIntent`** e escolha-a na lista.
4. Em Copiar intenção interna, dê um nome à intenção, como **`KendraSearchIntent`**, e escolha Adicionar.
5. No editor de intenções, escolha Consulta do Amazon Kendra para abrir as opções de consulta.
6. No menu Índice do Amazon Kendra, escolha o índice que a intenção deve pesquisar.
7. Na seção Resposta, adicione as três seguintes mensagens:

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

8. Escolha Salvar intenção e selecione Criar para criar o bot.

Por fim, use a janela de teste do console para testar as respostas do bot. Suas perguntas devem estar no domínio compatível com o índice.

Como testar o bot de perguntas frequentes

1. Na janela de teste do console, digite uma pergunta para o índice.
2. Verifique a resposta na seção de resposta da janela de teste.
3. Para redefinir a janela de teste para outra pergunta, escolha Limpar histórico de conversas.

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. Seu aplicativo ou função do Lambda precisa salvar dados de intenção em variáveis de sessão, ou você precisa usar a operação GetSession para recuperar dados de intenção ao retomar a intenção atual.

Declarações comuns:

- pausar
- pausar esse item

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.

Declarações comuns:

- repetir
- dizer isso de novo
- repetir isso

AMAZON.ResumeIntent

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

Declarações 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.

Declarações 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 do tipo de slot existentes e, em seguida, encerrar a interação.

Declarações comuns:

- parar
- desligar
- cale-se

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
AMAZON.Airport	Reconhece palavras que representam um aeroporto.	Todas as localidades
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.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	Todas as localidades

Tipo de slot	Descrição breve	Localidades compatíveis
	endereço de e-mail em um endereço de e-mail padrão.	
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
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
AMAZÔNIA.SpeedUnit	Converte palavras que representam uma unidade de velocidade em uma abreviatura padrão.	Inglês (EUA) (en-US)

Tipo de slot	Descrição breve	Localidades compatíveis
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, exceto inglês (EUA) (en-US)
AMAZON.TIME	Converte palavras que indicam horários em um formato de hora.	Todas as localidades
AMAZÔNIA. WeightUnit	Converte palavras que representam uma unidade de peso em uma abreviatura padrão	Inglês (EUA) (en-US)

> **Note**
>
> Para a localidade em inglês (EUA) (en-US), o Amazon Lex oferece suporte aos tipos de slots do Alexa Skill Kit. Para obter uma lista de tipos de slot integrados disponíveis, consulte Referência do tipo de slot na documentação do Alexa Skills Kit.
>
> - O Amazon Lex não oferece suporte ao tipo de slot integrado `AMAZON.LITERAL` ou ao `AMAZON.SearchQuery`.

AMAZON.Airport

Fornece uma lista de aeroportos. Os exemplos incluem:

- Aeroporto Internacional John F. Kennedy
- Aeroporto de Melbourne

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**

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 do Reino Unido ou do Canadá
- Números de carteira de motorista
- Números de identificação de veículo

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

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

O Amazon Lex 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 uma AMAZON. AlphaNumeric o tipo de slot 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 PutSlotType. A expressão regular é validada quando você salva o tipo de slot. Se a expressão não for válida, o Amazon Lex retornará uma mensagem de erro.

Quando você usa uma expressão regular em um tipo de slot, o Amazon Lex 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 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 não faz a conversão de uma variação para um nome oficial.

Exemplos:

- Nova York
- Reykjavik
- Tóquio
- Versalhes

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.

- Declaraçoes mapeadas para uma data específica, como "hoje", "agora" ou "vinte e cinco de novembro", são convertidas em uma data completa: `2020-11-25`. O padrão é datas iguais ou posteriores à data atual.
- Declarações mapeadas para uma semana específica, como "esta semana" ou "semana que vem", são convertidas 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 `2020-11-30`.
- Declarações mapeadas para um mês, mas não para um dia específico, como "próximo mês", são convertidas no último dia do mês. Por exemplo, se hoje for 25/11/2020, o "próximo mês" será convertido em `2020-12-31`.
- Declarações mapeadas para um ano, 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`.

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 (+).

AMAZÔNIA. FirstName

Nomes próprios comumente usados. Esse tipo de slot reconhece tanto nomes formais quanto apelidos informais. O nome enviado para sua intenção é o valor enviado pelo usuário. O Amazon Lex não converte do apelido para o nome formal.

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

Na localidade em inglês (EUA) (en-US), use o nome do slot Amazon.US_FIRST_NAME.

Exemplos:

- Emily
- John
- Sophie

AMAZÔNIA. LastName

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

Na localidade em inglês (EUA) (en-US), use o nome do slot Amazon.US_last_name.

Exemplos:

- Brosky
- Dasher
- Evers
- Parres
- Welt

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
50	50

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%

Entrada	Resposta
0,4%	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 1 (509) 555-1212	`+61744431061` `+15095551212`
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 7 dígitos sem código internacional ou código de área	555-1212	`5551212`

AMAZÔNIA. SpeedUnit

Converte palavras que representam unidades de velocidade na abreviatura correspondente. Por exemplo, "milhas por hora" é convertido em `mph`.

Esse tipo de slot está disponível somente na localidade em inglês (EUA) (en-US).

Os exemplos a seguir mostram como o tipo de slot `AMAZON.SpeedUnit` captura unidades de velocidade.

Unidade de velocidade	Abreviação
milhas por hora, mph, MPH, m/h	mph
quilômetros por hora, km por hora, kmph, KMPH, km/h	kmph
metros por segundo, mps, MPS, m/s	mps
milhas náuticas por hora, nós, nó	nó

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.

Esse tipo de slot não está disponível na localidade em inglês (EUA) (en-US).

Exemplos:

- Avenida Canberra
- Rua da frente

- Estrada do Mercado

AMAZON.TIME

Converte palavras que representam horários em valores de hora. Inclui resoluções para tempos ambíguos. Quando o usuário insere um horário ambíguo, o Amazon Lex usa o atributo `slotDetails` de um evento do Lambda 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 mapa `slots` é `null`, e a entidade `slotDetails` contém as duas possíveis resoluções de horário. O Amazon Lex insere o seguinte na função do Lambda.

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

Quando o usuário responde com um horário não ambíguo, o Amazon Lex envia o horário para a função do Lambda no atributo `slots` do evento do Lambda, e o atributo `slotDetails` fica vazio. Por exemplo, se o usuário responder ao prompt para um horário de entrega com "10:00 PM (22h)", o Amazon Lex inserirá o seguinte na função do Lambda:

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

Para obter mais informações sobre os dados enviados do Amazon Lex para uma função do Lambda, consulte Formato de eventos de entrada.

AMAZÔNIA. WeightUnit

Converte palavras que representam uma unidade de peso na abreviatura correspondente. Por exemplo, "quilograma" é convertido para `kg`.

Esse tipo de slot está disponível somente na localidade em inglês (EUA) (en-US).

Os exemplos a seguir mostram como o tipo de slot `AMAZON.WeightUnit` captura unidades de peso:

Unidade de peso	Abreviação
quilogramas, quilos, kg, KG	kg
gramas, g, g, G, g	g
miligramas, mg, mg	mg
libras, lb, LB	lb
onças, oz, OZ	oz
tonelada, tonelada, t	t
quilotonelada, kt	kt

Tipos de slot personalizados

Para cada intenção, você pode especificar parâmetros que indicam as informações de que a intenção precisa 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 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.`" Cada valor no tipo de slot é o nome de um gênero, "comédia", "aventura", "documentário", etc. Você pode definir um sinônimo para um valor de tipo de slot. Por exemplo, você pode definir os sinônimos "engraçado" e "humor" para o valor "comédia".

Você pode configurar o tipo de slot para restringir a resolução aos valores do slot. Os valores do slot serão usados como uma enumeração e o valor inserido pelo usuário será resolvido para o slot valor somente se ele for o mesmo que um dos valores de slot ou um sinônimo. Um sinônimo é resolvido

para o valor de slot correspondente. Por exemplo, se o usuário inserir "engraçado", isso será definido como o valor do slot "comédia".

Como alternativa, você pode configurar o tipo de slot para expandir os valores. Os valores do slot serão usados como dados de treinamento e o slot será resolvido para o valor fornecido pelo usuário se ele for semelhante aos valores e sinônimos do slot. Esse é o comportamento padrão.

O Amazon Lex mantém uma lista de possíveis resoluções para um slot. Cada entrada na lista fornece um valor de resolução que o Amazon Lex reconhece como possibilidades adicionais para o slot. Um valor de resolução é o melhor esforço para corresponder ao valor do slot. A lista contém até cinco valores.

Quando o valor inserido pelo usuário é um sinônimo, a primeira entrada na lista de valores de resolução é o valor do tipo do slot. Por exemplo, se o usuário insere "engraçado", o campo `slots` contém "engraçado" e a primeira entrada no campo `slotDetails` é "comédia". Você pode configurar o `valueSelectionStrategy` quando cria ou atualiza um tipo de slot com a operação PutSlotType para que o valor do slot seja preenchido com o primeiro valor na lista de resolução.

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

```
"slots": {
   "MovieGenre": "funny";
},
"slotDetails": {
   "Movie": {
      "resolutions": [
         "value": "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 do tipo 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. Se exceder esses limites, você receberá um `LimitExceededException` quando chamar a operação PutBot.

Ofuscação de slot

O Amazon Lex permite ofuscar ou ocultar o conteúdo dos slots para que não fique visível. Para proteger dados confidenciais capturados como valores de slot, é possível ativar a ofuscação de slot para mascarar esses valores para os logs de conversa.

Ao optar por ofuscar valores de slot, o Amazon Lex substitui o valor de 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 obfuscation:
    My name is John Stiles
After obfuscation:
    My name is {full_name}
```

Se uma declaração contiver caracteres de colchete ({}), o Amazon Lex 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 obfuscation:
    My name is {John Stiles}
After obfuscation:
    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 `PostContent` e `PostText`, 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 ofuscará valores de slot se ele reconhecer um slot e um valor de slot na declaração. Se nenhum valor de slot for reconhecido, o Amazon Lex não ofuscará a declaração.

No segundo turno e nos posteriores, o Amazon Lex sabe qual slot elicitar e se o valor de slot deve ser ofuscado. Se o Amazon Lex reconhecer o valor de slot, o valor será ofuscado. Se o Amazon Lex não reconhecer um valor, toda a declaração será ofuscada. Nenhum valor de slot em declarações perdidas será ofuscado.

O Amazon Lex 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 não ofusca o valor de slot no áudio. Ele ofusca o valor de slot na transcrição de áudio.

Não é necessário ofuscar todos os slots em um bot. Você pode escolher quais slots ofuscar usando o console ou a API do Amazon Lex. No console, escolha Ofuscação de slot nas configurações de um slot. Se você estiver usando a API, defina o campo `obfuscationSetting` do slot como `DEFAULT_OBFUSCATION` ao chamar a operação PutIntent.

Análise de sentimento

É possível usar a análise de sentimento para determinar os sentimentos expressos em uma declaração 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 passar uma conversa a um atendente humano.

O Amazon Lex integra-se 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 da declaração 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 usando a API do Amazon Lex. No console do Amazon Lex, selecione a guia Configurações do bot e defina a opção Análise de Sentimento como Sim. Se você estiver usando a API, chame a operação PutBot com o campo `detectSentiment` definido como `true`.

Quando a análise de sentimento está ativada, a resposta das operações PostContent e PostText retorna um campo chamado `sentimentResponse` na resposta do bot com outros metadados. O campo `sentimentResponse` tem dois campos, `SentimentLabel` 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 `PostText` ou `PostContent`. O campo `SentimentScore` é uma string que contém as pontuações para a resposta.

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

O Amazon Lex chama o Amazon Comprehend em seu nome para determinar o sentimento em cada declaração processada 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.

Marcação de seus recursos do Amazon Lex

Para ajudar você a gerenciar os aliases de bot, canais de bot e bots do Amazon Lex, é possível atribuir metadados a cada recurso como tags. Uma tag é um rótulo que você atribui a um AWS recurso. 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 aplicativo. As tags ajudam a:

- Identifique e organize seus AWS recursos. Muitos AWS recursos oferecem suporte à marcação, então você pode atribuir a mesma tag a recursos em serviços diferentes para indicar que os recursos estão relacionados. Por exemplo, é possível marcar um bot e as funções do Lambda que ele usa com a mesma tag.
- Alocar custos. Você ativa as tags no Gerenciamento de Faturamento e Custos da AWS painel. AWS usa as tags para categorizar seus custos e entregar um relatório mensal de alocação de custos para você. Para o Amazon Lex, é possível alocar custos para cada alias usando tags específicas para o alias, exceto para o alias `$LATEST`. Aloque custos para o alias `$LATEST` usando tags para o bot do Amazon Lex. Para obter mais informações, consulte Usar tags de alocação de custos no Guia do usuário do Gerenciamento de Faturamento e Custos da AWS .

- Controle o acesso aos seus recursos. É possível usar tags no Amazon Lex para criar políticas para controlar o acesso aos recursos do Amazon Lex. Essas políticas podem ser anexadas a um perfil do IAM ou um usuário para habilitar o controle de acesso baseado em tags. Para obter mais informações, consulte ABAC com o Amazon Lex. Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Uso de uma tag para acessar um recurso.

Você pode trabalhar com tags usando a AWS Management Console AWS Command Line Interface, a ou a API Amazon Lex.

Marcação de recursos

Se estiver usando o console do Amazon Lex, 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 API AWS CLI ou a Amazon Lex API, você usa as seguintes operações para gerenciar tags para seus recursos:

- ListTagsForResource: visualizar as tags associadas a um recurso.
- PutBot e PutBotAlias: aplicar as tags ao criar um bot ou alias de bot.
- TagResource: adicionar e modificar tags em um recurso existente.
- UntagResource: remover as tags de um recurso.

Os seguintes recursos do Amazon Lex são compatíveis com a marcação:

- Bots: use um nome do recurso da Amazon (ARN) como o seguinte:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}`
- Aliases de bot – use um ARN como o seguinte:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}:${bot-alias}`
- Canais de bot – use um ARN como o seguinte:
  - `arn:${partition}:lex:${region}:${account}:bot-channel:${bot-name}:${bot-alias}:${channel-name}`

Restrições de tag

As restrições básicas a seguir aplicam-se às tags nos recursos do Amazon Lex:

- Número máximo de tags – 50
- 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 `aws:` como prefixo para chaves, pois ele é reservado para uso da AWS.

Marcação de recursos (console)

É possível usar o console para gerenciar tags em um bot, alias de bot ou recurso de canal 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 para criar um novo bot.
3. Na parte inferior da página Criar seu bot, escolha Tags.
4. Escolha Adicionar tag e adicione uma ou mais tags ao bot. É possível adicionar até 50 tags.

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. Escolha Configurações.
4. Adicione o nome do alias, escolha a versão do bot e escolha Adicionar tags.
5. Escolha Adicionar tag e adicione uma ou mais tags ao alias de bot. É possível adicionar até 50 tags.

Como adicionar uma tag ao criar um canal 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 canal de bot.
3. Escolha Canais e escolha o canal que deseja adicionar.
4. Adicione os detalhes do canal de bot e escolha Tags.
5. Escolha Adicionar tag e adicione uma ou mais tags ao canal de bot. É possível adicionar até 50 tags.

Como adicionar uma tag ao importar 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 Ações e Importar.
3. Escolha o arquivo zip para importar o bot.
4. Escolha Tags e escolha Adicionar Tags para adicionar uma ou mais tags ao bot. É possível adicionar até 50 tags.

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. No menu esquerdo, escolha Bots e escolha o bot que deseja modificar.
3. Escolha Configurações e, no menu esquerdo, escolha Geral.
4. Escolha Tags e adicione, modifique ou remova tags para o bot.

Como adicionar, remover ou modificar uma tag em 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. No menu esquerdo, escolha Bots e escolha o bot que deseja modificar.
3. Escolha Configurações e, no menu esquerdo, escolha Aliases.

4. Escolha Gerenciar tags para o alias que você deseja modificar e adicione, modifique ou remova tags para o alias de bot.

Como adicionar, remover ou modificar uma tag em um canal de bot existente

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. No menu esquerdo, escolha Bots e escolha o bot que deseja modificar.
3. Escolha Canais.
4. Escolha Tags e adicione, modifique ou remova tags para o canal de bot.

Marcação de recursos (AWS CLI)

Você pode usar o AWS CLI para gerenciar tags em um bot, um alias de bot ou um recurso de canal de bot. É possível adicionar tags ao criar um bot ou um alias de bot ou adicionar, modificar ou remover tags de um bot, alias de bot ou canal de bot.

Todos os exemplos são formatados para Linux e macOS. Para usar o comando no Windows, substitua o caractere de continuação do Linux (\) por um acento circunflexo (^).

Como adicionar uma tag ao criar um bot

- O `put-bot` AWS CLI comando abreviado a seguir mostra os parâmetros que você deve usar para adicionar uma tag ao criar um bot. Para realmente criar um bot, é necessário fornecer outros parâmetros. Para obter mais informações, consulte Etapa 4: Conceitos básicos (AWS CLI).

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

Como adicionar uma tag ao criar um alias de bot

- O `put-bot-alias` AWS CLI comando abreviado a seguir mostra os parâmetros que você deve usar para adicionar uma tag ao criar um alias de bot. Para criar um alias de bot, é necessário fornecer outros parâmetros. Para obter mais informações, consulte Exercício 5: Criar um alias (AWS CLI).

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

Como listar as tags em um recurso

- Use o `list-tags-for-resource` AWS CLI comando para mostrar os recursos associados a um bot, alias de bot e canal de bot.

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

Como adicionar ou modificar as tags em um recurso

- Use o `tag-resource` AWS CLI comando para adicionar ou modificar um bot, um alias de bot ou um canal de bot.

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

Como remover as tags de um recurso

- Use o `untag-resource` AWS CLI comando para remover tags de um bot, alias de bot ou canal de bot.

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

Conceitos básicos do Amazon Lex

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

- AWS SDK — Ao usar o, SDKs suas solicitações para o Amazon Lex são automaticamente assinadas e autenticadas usando as credenciais que você fornece. Esta é a escolha recomendada para a criação de suas aplicações.
- AWS CLI — Você pode usar o AWS CLI para acessar qualquer recurso do Amazon Lex sem precisar escrever nenhum código.
- Console da AWS: o console é a maneira mais fácil de começar a testar e usar o Amazon Lex

Se você é novo no Amazon Lex, recomendamos que leia Amazon Lex: como funciona antes de continuar.

Tópicos

- Etapa 1: configurar uma AWS conta e criar um usuário administrador
- Etapa 2: Configurar o AWS Command Line Interface
- Etapa 3: Conceitos básicos (console)
- Etapa 4: Conceitos básicos (AWS CLI)

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

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

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

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. A cobrança incorrerá apenas pelos serviços utilizados.

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

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

Para criar uma AWS conta

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma ligação ou mensagem de texto e inserir 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

Serviços em AWS, como o Amazon Lex, 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. O console requer sua senha. No entanto, não recomendamos que você acesse AWS usando as credenciais da sua AWS conta. Em vez disso, recomendamos o uso de:

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

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

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 conta da AWS . Para obter instruções, consulte Criar seu primeiro grupo de administradores e usuário 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

Próxima etapa

Etapa 2: Configurar o AWS Command Line Interface

Etapa 2: Configurar o AWS Command Line Interface

Se você preferir usar o Amazon Lex com o AWS Command Line Interface (AWS CLI), faça o download e configure-o.

> **Important**
>
> Você não precisa AWS CLI executar as etapas dos exercícios de introdução. No entanto, alguns dos últimos exercícios neste guia usam a AWS CLI. Se você preferir começar usando o console, ignore esta etapa e vá para Etapa 3: Conceitos básicos (console). Mais tarde, quando precisar do AWS CLI, volte aqui para configurá-lo.

Para configurar o AWS CLI

1. Faça download e configure a AWS CLI. Para obter instruções, consulte os seguintes tópicos no Manual do usuário do AWS Command Line Interface :

   - Configurando o AWS Command Line Interface
   - Configurar a AWS Command Line Interface

2. Adicione um perfil nomeado para o usuário administrador ao final do arquivo de AWS CLI configuração. Você usa esse perfil ao executar AWS CLI comandos. Para obter mais informações sobre perfis nomeados, consulte Perfis nomeados no Guia do usuário da AWS Command Line Interface .

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

   Para obter uma lista das AWS regiões disponíveis, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

3. Verifique a configuração digitando o comando de ajuda na solicitação de comando:

   ```
   aws help
   ```

Etapa 3: Conceitos básicos (console)

Etapa 3: Conceitos básicos (console)

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

- Exercício 1: Criar um bot do Amazon Lex 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.

  Além disso, você usa o blueprint da função Lambda, fornecido por AWS Lambda, para criar uma função Lambda. A função é um hook de código que usa um código predefinido compatível com o bot.

- Exercício 2: criar um bot personalizado criando e configurando manualmente um bot. Você também pode criar uma função do Lambda como um hook de código. Um código de exemplo é fornecido.
- Exercício 3: publicar um bot e, em seguida, criar uma nova versão. Como parte deste exercício, você criará um alias que aponte para a versão do bot.

Tópicos

- Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console).
- Exercício 2: criar um bot personalizado do Amazon Lex
- Exercício 3: publique uma versão e crie um alias

Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console).

Neste exercício, você faz o seguinte:

- Crie seu primeiro bot do Amazon Lex e teste-o no console do Amazon Lex.

  Para este exercício, você usa o OrderFlowersmodelo. Para obter mais informações sobre esquemas, consulte Amazon Lex e esquemas AWS Lambda.

- Crie uma AWS Lambda função e teste-a no console Lambda. Ao processar uma solicitação, seu bot chama a função do Lambda. Para este exercício, você usa um blueprint do Lambda (lex-order-flowers-python) fornecido no AWS Lambda console para criar sua função do Lambda. O código de esquema ilustra como você pode usar a mesma função do Lambda para realizar a inicialização e validação, e cumprir a intenção `OrderFlowers`.

- Atualize o bot para adicionar a função do Lambda como o hook de código para cumprir a intenção. Teste a end-to-end experiência.

As seções a seguir explicam o que os esquemas fazem.

Bot do Amazon Lex: visão geral do esquema

Você usa o OrderFlowersblueprint para criar um bot do Amazon Lex. Para obter mais informações sobre a estrutura de um bot, consulte. Amazon Lex: como funciona O bot é pré-configurado da seguinte forma:

- 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 FlowerTypes personalizado)
  - `PickupDate` (tipo integrado AMAZON.DATE)
- Utterance: a seguinte amostra de declaração indica 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?"
  - Solicite a `PickupDate` vaga — “Em que dia você quer que o {FlowerType} seja retirado?”
  - Solicitar o `PickupTime` slot — “A que horas você deseja que o {FlowerType} seja retirado?”
  - Declaração de confirmação — “Ok, seu {FlowerType} estará pronto para retirada até {PickupTime} em {PickupDate}. Tudo bem?"

AWS Lambda Função: Resumo do plano

A função do Lambda neste exercício executa a inicialização e a validação e as tarefas de atendimento. Portanto, após criar a função do Lambda, você deve atualizar a configuração da intenção especificando a mesma função do Lambda como um hook de código para lidar com a inicialização e a validação e as tarefas de atendimento.

- Como hook de inicialização e validação, a função do Lambda executa uma validação básica. Por exemplo, se o usuário fornecer uma hora de retirada fora do horário comercial normal, a função do Lambda direcionará o Amazon Lex para solicitar que o usuário insira o horário novamente.

- Como parte do hook de código de atendimento, a função do Lambda retorna uma mensagem resumida indicando que o pedido de flores foi feito (ou seja, a intenção é atendida).

Próxima etapa

Etapa 1: criar um bot Amazon Lex (console)

Etapa 1: criar um bot Amazon Lex (console)

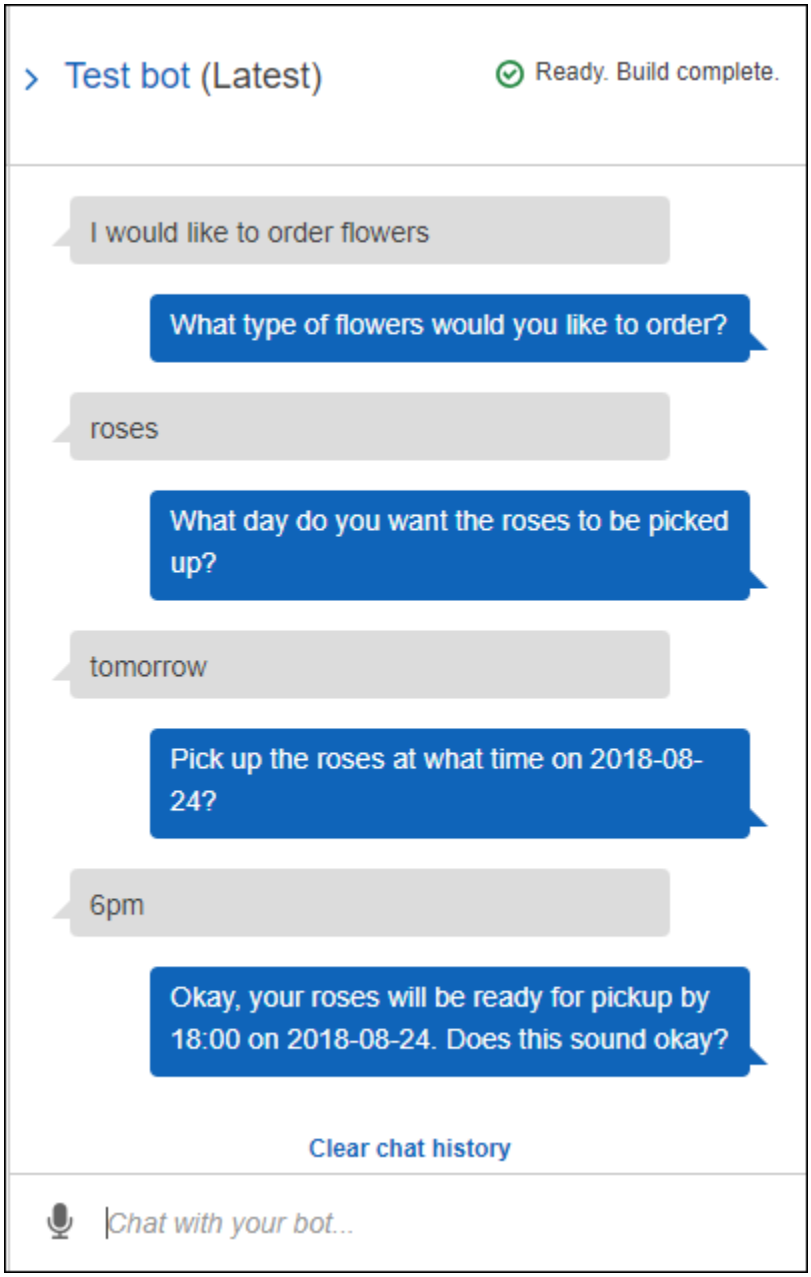
Para este exercício, crie um bot para pedir flores, chamado OrderFlowersBot.

Para criar um bot Amazon Lex (console)

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Se este for o seu primeiro bot, selecione Get Started (Conceitos básicos); caso contrário, na página Bots, escolha Criar.
3. Na página Create your Lex bot, forneça as informações a seguir e escolha Create.
   - Escolha o esquema OrderFlowers.
   - Deixe o nome padrão do bot (OrderFlowers).
   - Em COPPA, selecione **No**.
   - Para Armazenamento de declarações do usuário, escolha a resposta apropriada.
4. Escolha Criar. O console faz as solicitações necessárias ao Amazon Lex para salvar a configuração. Em seguida, o console exibe a janela do editor de bot.
5. Aguarde a confirmação de criação do bot.
6. Teste o bot.

> **Note**
> Você pode testar o bot digitando o texto na janela de teste ou, em navegadores compatíveis, selecionando o botão de microfone na janela de teste e falando.

Use o texto de exemplo a seguir para começar uma conversa com o bot e encomendar flores:



Nesta entrada, o bot infere a intenção `OrderFlowers` e solicita os dados de slot. Quando você fornecer todas os dados de slot necessários, o bot cumprirá a intenção (`OrderFlowers`) retornando todas as informações para o aplicativo cliente (neste caso, o console). O console mostra as informações na janela de teste.

Especificamente:

- Na instrução "Em que dia você deseja que as rosas sejam entregues?", o termo "rosas" aparece porque a solicitação para o slot `pickupDate` é configurada usando substituições, `{FlowerType}`. Verifique isso no console.

- A instrução "Está bem, as rosas estarão prontas..." é a solicitação de confirmação que você configurou.
- A última instrução ("`FlowerType:roses...`") contém apenas os dados do slot que são retornados ao cliente, neste caso, na janela de teste. No próximo exercício, você usará uma função do Lambda para atender à intenção. Nesse caso, você recebe uma mensagem indicando que o pedido foi atendido.

Próxima etapa

Etapa 2 (opcional): revisar os detalhes do fluxo de informações (console)

Etapa 2 (opcional): revisar os detalhes do fluxo de informações (console)

Esta seção explica o fluxo de informações entre um cliente e o Amazon Lex para cada entrada do usuário em nossa conversa de exemplo.

O exemplo usa a janela de teste do console para a conversa com o bot.

Para abrir a janela de teste do Amazon Lex

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 a ser testado.
3. No lado direito do console, escolha Testar chatbot.

Para ver o fluxo de informações do conteúdo falado ou digitado, escolha o tópico apropriado.

Tópicos

- Etapa 2a (opcional): revisar os detalhes do fluxo de informações falado (console)
- Etapa 2b (opcional): revisar os detalhes do fluxo de informações digitado (console)

Etapa 2a (opcional): revisar os detalhes do fluxo de informações falado (console)

Esta seção explica o fluxo de informações entre o cliente e o Amazon Lex quando o cliente usa fala para enviar solicitações. Para obter mais informações, consulte PostContent.

1. O usuário diz: Eu gostaria de encomendar flores.

a. O cliente (console) envia a seguinte solicitação PostContent para o Amazon Lex:

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


Request body
input stream
```

O URI de solicitação e o corpo fornecem informações ao Amazon Lex:

- URI da solicitação fornece o nome do bot (`OrderFlowers`), o alias do bot (`$LATEST`) e o nome do usuário (uma string aleatória que identifica o usuário). `content` indica que esta é uma solicitação da API `PostContent` (não uma solicitação `PostText`).
- Cabeçalhos de solicitação
  - `x-amz-lex-session-attributes`: o valor codificado em base64 representa "{}". Quando o cliente faz a primeira solicitação, não há atributos de sessão.
  - `Content-Type` – reflete o formato de áudio.
- Corpo da solicitação – o stream de áudio de entrada do usuário ("Eu gostaria de encomendar flores.").

> **Note**
>
> Se o usuário escolher enviar texto ("Eu gostaria de encomendar flores") para a API `PostContent` em vez falar, o corpo da solicitação será a entrada do usuário. O cabeçalho `Content-Type` é definido adequadamente:
>
> ```
> POST /bot/OrderFlowers/alias/$LATEST/
> user/4o9wwdhx6nlheferh6a73fujd3118f5w/content HTTP/1.1
> x-amz-lex-session-attributes: "e30="
> Content-Type: "text/plain; charset=utf-8"
> Accept: accept
>
> Request body
> ```

```
input stream
```

b. No fluxo de entrada, o Amazon Lex detecta a intenção (`OrderFlowers`). Em seguida, ele escolhe um dos slots da intenção (neste caso, o `FlowerType`) e um de seus prompts de seleção de valor e envia uma resposta com os seguintes cabeçalhos:

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

Os valores do cabeçalho fornecem as seguintes informações:

- `x-amz-lex-input-transcript` – fornece a transcrição do áudio (entrada do usuário) da solicitação
- `x-amz-lex-message`: fornece a transcrição do áudio que Amazon Lex retornou na resposta
- `x-amz-lex-slots` – a versão codificada em base64 dos slots e valores:

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

- `x-amz-lex-session-attributes`: a versão codificada em base64 dos atributos de sessão ({})

O cliente reproduz o áudio no corpo da resposta.

2. O usuário diz: rosas

a. O cliente (console) envia a seguinte solicitação PostContent para o Amazon Lex:

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

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

O corpo da solicitação é o stream de áudio de entrada do usuário (rosas). O `sessionAttributes` permanece vazio.

b. O Amazon Lex interpreta o fluxo de entrada no contexto da intenção atual (ele lembra que pediu ao usuário informações a respeito do slot `FlowerType`). Primeiro, o Amazon Lex atualiza o valor de slot para a intenção atual. Em seguida, ele escolhe outro slot (`PickupDate`), juntamente com uma das mensagens de solicitação (Quando você deseja pegar as rosas?) e retorna uma resposta com os seguintes cabeçalhos:

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

Os valores do cabeçalho fornecem as seguintes informações:

- `x-amz-lex-slots`: a versão codificada em base64 dos slots e valores:

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

- `x-amz-lex-session-attributes`: a versão codificada em base64 dos atributos de sessão ({})

O cliente reproduz o áudio no corpo da resposta.

3. O usuário diz: amanhã

a. O cliente (console) envia a seguinte solicitação PostContent para o Amazon Lex:

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

```
Accept: "audio/mpeg"


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

O corpo da solicitação é o stream de áudio de entrada do usuário ("amanhã"). O `sessionAttributes` permanece vazio.

b. O Amazon Lex interpreta o fluxo de entrada no contexto da intenção atual (ele lembra que pediu ao usuário informações a respeito do slot `PickupDate`). O Amazon Lex atualiza o valor de slot (`PickupDate`) para a intenção atual. Em seguida, ele escolhe outro slot para escolher o valor para (`PickupTime`) e um dos prompts de seleção de valor (Quando você deseja receber as rosas em 18/03/2017?), e retorna uma resposta com os seguintes cabeçalhos:

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

Os valores do cabeçalho fornecem as seguintes informações:

- `x-amz-lex-slots`: a versão codificada em base64 dos slots e valores:

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

- `x-amz-lex-session-attributes`: a versão codificada em base64 dos atributos de sessão ({})

O cliente reproduz o áudio no corpo da resposta.

4. O usuário diz: 18h

   a. O cliente (console) envia a seguinte solicitação PostContent para o Amazon Lex:

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


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

O corpo da solicitação é o stream de áudio de entrada do usuário ("18h"). O `sessionAttributes` permanece vazio.

b. O Amazon Lex interpreta o fluxo de entrada no contexto da intenção atual (ele lembra que pediu ao usuário informações a respeito do slot `PickupTime`). Primeiro, ele atualiza o valor do slot para a intenção atual.

Agora, o Amazon Lex detecta que tem informações de todos os slots. No entanto, a intenção `OrderFlowers` é configurada com uma mensagem de confirmação. Portanto, o Amazon Lex precisa de uma confirmação explícita do usuário para poder prosseguir para o atendimento da intenção. Ele envia uma resposta com os seguintes cabeçalhos solicitando a confirmação antes de encomendar as flores:

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

Os valores do cabeçalho fornecem as seguintes informações:

- `x-amz-lex-slots`: a versão codificada em base64 dos slots e valores:

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

- `x-amz-lex-session-attributes`: a versão codificada em base64 dos atributos de sessão ({})

O cliente reproduz o áudio no corpo da resposta.

5. O usuário diz: Sim

   a. O cliente (console) envia a seguinte solicitação PostContent para o Amazon Lex:

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


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

   O corpo da solicitação é o stream de áudio de entrada do usuário ("Sim"). O `sessionAttributes` permanece vazio.

   b. O Amazon Lex interpreta o fluxo de entrada e entende que o usuário deseja prosseguir com o pedido. A intenção `OrderFlowers` é configurada com `ReturnIntent` como a atividade de cumprimento. Isso orienta o Amazon Lex a retornar todos os dados de intenção ao cliente. O Amazon Lex retorna uma resposta com o seguinte:

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

   O cabeçalho de resposta `x-amz-lex-dialog-state` é definido como `ReadyForFulfillment`. O cliente pode, então, cumprir a intenção.

6. Agora, teste novamente o bot. Para estabelecer um novo contexto (usuário), escolha o link Clear no console. Forneça os dados para a intenção `OrderFlowers` e inclua alguns dados inválidos. Por exemplo:

- Jasmim como o tipo de flor (não é um dos tipos de flor suportados)
- Ontem como o dia em que você deseja receber as flores

Observe que o bot aceita esses valores, pois você não tem nenhum código para inicializar e validar os dados do usuário. Na próxima seção, adicione uma função do Lambda para fazer isso. Observe o seguinte sobre a função do Lambda:

- Ela valida dados de slot após cada entrada do usuário. Ela cumpre a intenção no final. Ou seja, o bot processa o pedido de flores e retorna uma mensagem para o usuário, em vez de simplesmente retornar dados do slot para o cliente. Para obter mais informações, consulte Uso de funções do Lambda.
- Ela também define os atributos da sessão. Para mais informações sobre atributos de sessão, consulte PostText.

  Ao concluir a seção Conceitos básicos, você pode fazer os exercícios adicionais (Exemplos adicionais: criação de bots do Amazon Lex). O Reservar uma viagem usa atributos de sessão para compartilhar informações entre intenções para iniciar uma conversa dinâmica com o usuário.

Próxima etapa

Etapa 3: Crie uma função do Lambda (console)

Etapa 2b (opcional): revisar os detalhes do fluxo de informações digitado (console)

Esta seção explica o fluxo de informações entre o cliente e o Amazon Lex, em que o cliente usa a API `PostText` para enviar solicitações. Para obter mais informações, consulte PostText.

1. O usuário digita: Eu gostaria de encomendar flores.

   a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

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

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

```
}
```

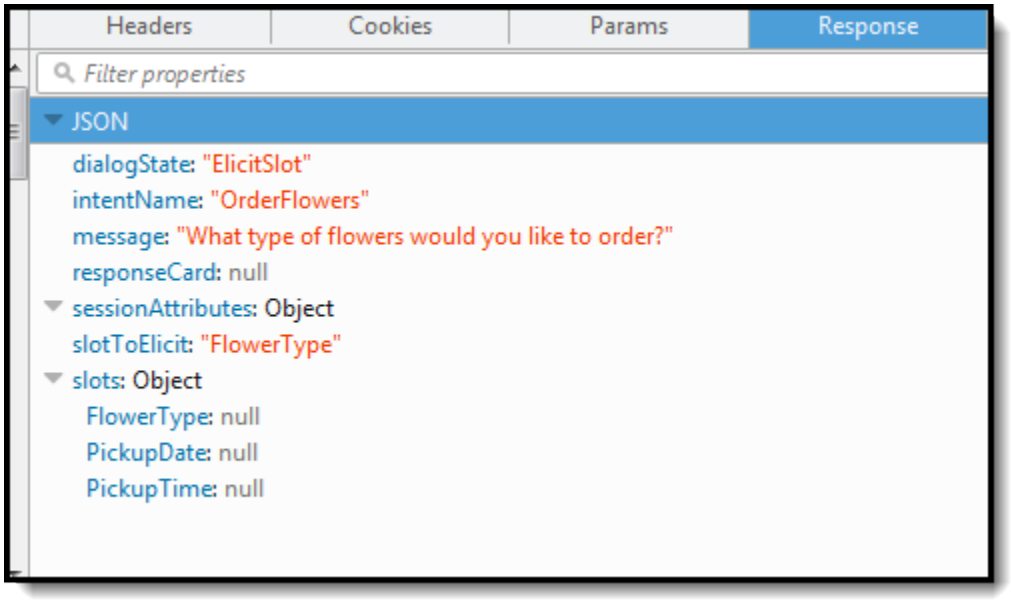
O URI de solicitação e o corpo fornecem informações ao Amazon Lex:

- URI de solicitação: fornece o nome do bot (`OrderFlowers`), o alias do bot (`$LATEST`) e o nome do usuário (uma string aleatória que identifica o usuário). O `text` final indica que esta é uma solicitação de API `PostText` (e não `PostContent`).

- Corpo da solicitação: inclui a entrada do usuário (`inputText`) e `sessionAttributes` vazio. Quando o cliente faz a primeira solicitação, não há atributos de sessão. A função do Lambda as inicia posteriormente.

b. A partir de `inputText`, o Amazon Lex detecta a intenção (`OrderFlowers`). Essa intenção não tem nenhum hook de código (ou seja, as funções do Lambda) para inicialização e validação do atendimento ou da entrada do usuário.

O Amazon Lex escolhe um dos slots (`FlowerType`) da intenção para obter o valor. Ele também seleciona uma das solicitações de escolha de valor para o slot (tudo parte da configuração de intenção) e, em seguida, envia a resposta a seguir de volta para o cliente. O console exibe a mensagem na resposta para o usuário.



O cliente exibe a mensagem na resposta.

2. O usuário digita: rosas

a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

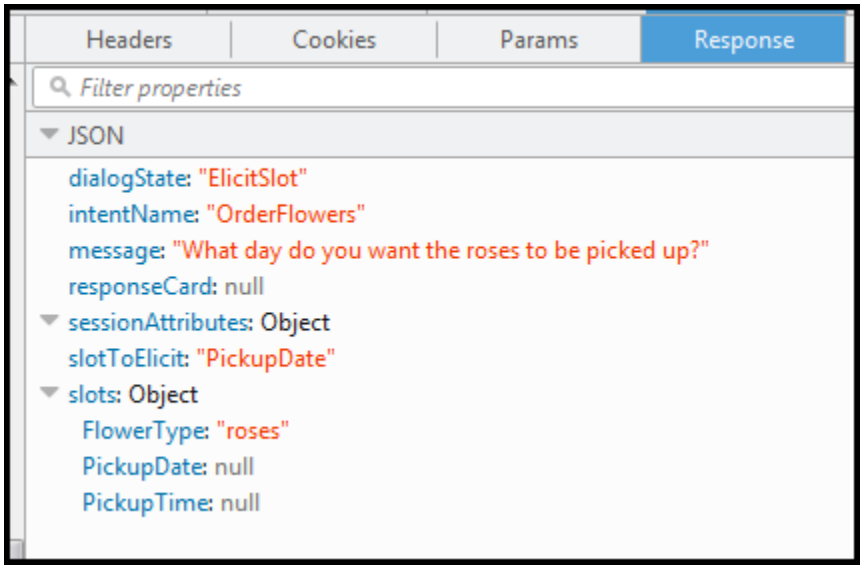
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

O `inputText` no corpo da solicitação fornece entradas do usuário. O `sessionAttributes` permanece vazio.

b. Primeiro, o Amazon Lex interpreta o `inputText` no contexto da intenção atual (o serviço lembra que pediu ao usuário específico informações sobre o slot `FlowerType`). O Amazon Lex atualiza o valor de slot para a intenção atual e escolhe outro slot (`PickupDate`) junto com um de seus prompts de mensagens, “Em que dia você deseja que as rosas sejam entregues?”, para o slot.

Em seguida, o Amazon Lex retorna a seguinte resposta:



O cliente exibe a mensagem na resposta.

3. O usuário digita: amanhã

a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

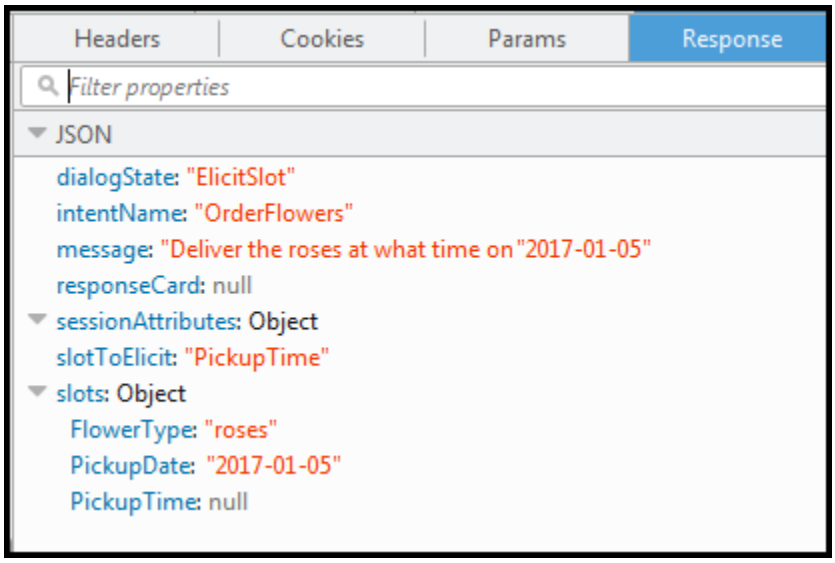
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

O `inputText` no corpo da solicitação fornece entradas do usuário. O `sessionAttributes` permanece vazio.

b. Primeiro, o Amazon Lex interpreta o `inputText` no contexto da intenção atual (o serviço lembra que pediu ao usuário específico informações sobre o slot `PickupDate`). O Amazon Lex atualiza o valor de slot (`PickupDate`) para a intenção atual. Ele escolhe outro slot para obter o valor para (`PickupTime`). Ele retorna uma das solicitações de escolha de valor, “Entregar as rosas a que horas em 05/01/2017?” para o cliente.

Em seguida, o Amazon Lex retorna a seguinte resposta:



O cliente exibe a mensagem na resposta.

4. O usuário digita: 18h

a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

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

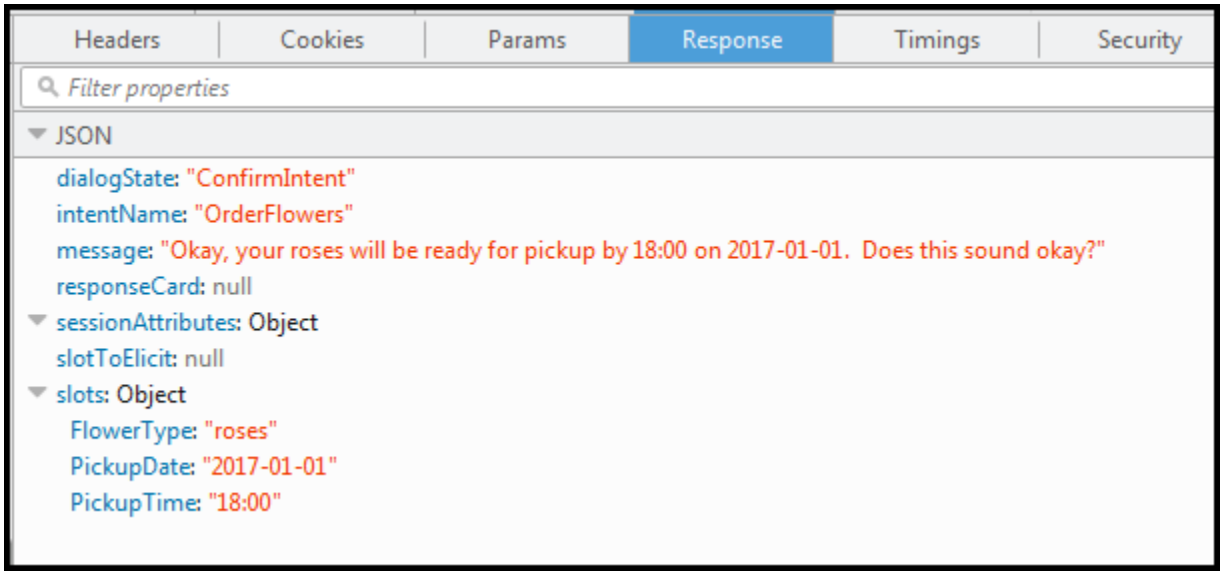
```
"Content-Encoding":"amz-1.0"

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

O `inputText` no corpo da solicitação fornece entradas do usuário. O `sessionAttributes` permanece vazio.

b. Primeiro, o Amazon Lex interpreta o `inputText` no contexto da intenção atual (o serviço lembra que pediu ao usuário específico informações sobre o slot `PickupTime`). Primeiro, o Amazon Lex atualiza o valor de slot para a intenção atual. Agora, o Amazon Lex detecta que tem informações de todos os slots.

A intenção `OrderFlowers` é configurada com uma mensagem de confirmação. Portanto, o Amazon Lex precisa de uma confirmação explícita do usuário para poder prosseguir para o atendimento da intenção. O Amazon Lex envia a seguinte mensagem para o cliente solicitando confirmação antes de encomendar as flores:



O cliente exibe a mensagem na resposta.

5. O usuário digita: Sim

a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

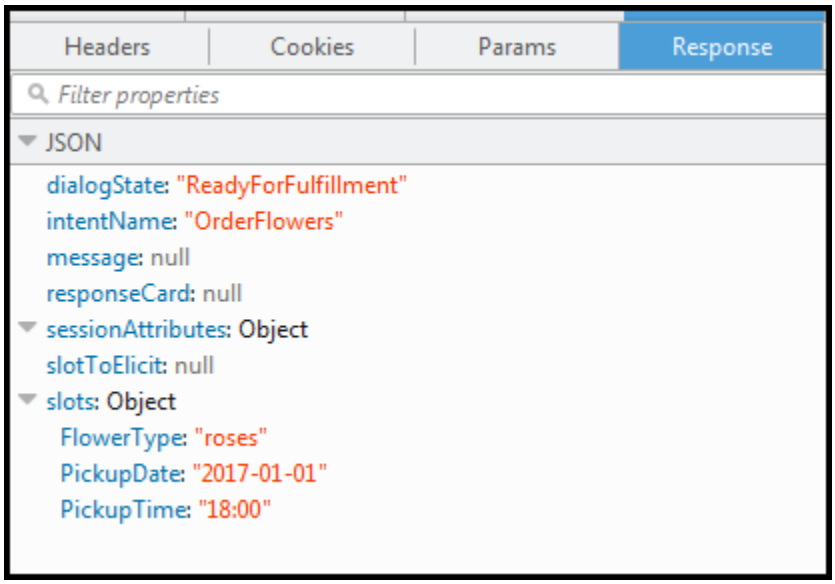
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
```

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

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

O `inputText` no corpo da solicitação fornece entradas do usuário. O `sessionAttributes` permanece vazio.

b. O Amazon Lex interpreta o `inputText` no contexto de confirmação da intenção atual. Ele entende que o usuário deseja prosseguir com o pedido. A intenção `OrderFlowers` é configurada com `ReturnIntent` como a atividade de cumprimento (não há função do Lambda para cumprir a intenção). Portanto, o Amazon Lex retorna os dados do slot ao cliente.



O Amazon Lex definiu `dialogState` como `ReadyForFulfillment`. O cliente pode, então, cumprir a intenção.

6. Agora teste o bot novamente. Para isso, selecione o link Clear no console para estabelecer um novo contexto (de usuário). Agora, conforme você fornece dados para a intenção de encomenda de flores, tente fornecer dados inválidos. Por exemplo:

   - Jasmim como o tipo de flor (não é um dos tipos de flor suportados).
   - Ontem como o dia em que você deseja receber as flores.

Observe que o bot aceita esses valores, pois você não tem nenhum código para inicializar/validar os dados do usuário. Na próxima seção, adicione uma função do Lambda para fazer isso. Observe o seguinte sobre a função do Lambda:

- A função do Lambda valida os dados do slot depois de cada entrada do usuário. Ela cumpre a intenção no final. Ou seja, o bot processa o pedido de flores e retorna uma mensagem para o usuário, em vez de simplesmente retornar dados de slot para o cliente. Para obter mais informações, consulte Uso de funções do Lambda.
- A função do Lambda também define os atributos da sessão. Para mais informações sobre atributos de sessão, consulte PostText.

  Ao concluir a seção Conceitos básicos, você pode fazer os exercícios adicionais (Exemplos adicionais: criação de bots do Amazon Lex). O Reservar uma viagem usa atributos de sessão para compartilhar informações entre intenções para iniciar uma conversa dinâmica com o usuário.

Próxima etapa

Etapa 3: Crie uma função do Lambda (console)

Etapa 3: Crie uma função do Lambda (console)

Crie uma função Lambda (usando o lex-order-flowers-pythonblueprint) e execute a invocação de teste usando dados de eventos de amostra no console. AWS Lambda

Você retorna ao console do Amazon Lex e adiciona a função do Lambda como o hook de código para atender à intenção `OrderFlowers` no `OrderFlowersBot` que criou na seção anterior.

Para criar a função do Lambda (console)

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função.
3. Na página Create function (Criar função), selecione Usar um blueprint (Usar um esquema). Digite **`lex-`** na caixa de texto do filtro e pressione `Enter` para localizar o esquema e selecione o esquema `lex-order-flowers-python`.

Os esquemas da função do Lambda são fornecidos em Node.js e em Python. Para este exercício, use o esquema baseado em Python.

4. Na página Basic information (Informações básicas), faça o seguinte:
   - Digite o nome de uma função do Lambda (`OrderFlowersCodeHook`).
   - Em Perfil de execução, escolha Criar um novo perfil com as permissões básicas do Lambda.
   - Deixe os outros valores padrão.
5. Escolha a opção Criar função.
6. Se você estiver usando uma localidade diferente do inglês (EUA) (en-US), atualize os nomes das intenções conforme descrito em Atualização de um esquema para uma localidade específica.
7. Testar a função do Lambda.
   a. Selecione Select a test event (Selecionar um evento de teste), Configure test events (Configurar eventos de teste).
   b. Selecione Amazon Lex Order Flowers (Flores do pedido do Amazon Lex) da lista Event template (Modelo do evento). Este evento de exemplo corresponde ao modelo de solicitação/resposta do Amazon Lex (consulte Uso de funções do Lambda). Dê um nome ao evento de teste (`LexOrderFlowersTest`).
   c. Escolha Criar.
   d. Selecione Testar para testar o hook de código.
   e. Verifique se a função do Lambda foi executada com êxito. A resposta, neste caso, corresponde ao modelo de resposta do Amazon Lex.

Próxima etapa

Etapa 4: Adicione a função do Lambda como hook de código (console)

Etapa 4: Adicione a função do Lambda como hook de código (console)

Nesta seção, você atualiza a configuração da OrderFlowers intenção de usar a função Lambda da seguinte forma:

- Primeiro, use a função do Lambda como um hook de código para executar o cumprimento da intenção `OrderFlowers`. Você pode testar o bot e verificar se recebeu uma mensagem de

cumprimento da função do Lambda. O Amazon Lex invoca a função do Lambda somente depois de você fornecer dados para todos os slots necessários para encomendar flores.

- Configure a mesma função do Lambda como um hook de código para executar a inicialização e a validação. Teste e verifique se a função do Lambda executa a validação (conforme você fornece dados de slot).

Para adicionar a função do Lambda como um hook de código (console)

1. No console do Amazon Lex, selecione o OrderFlowersbot. O console mostra a OrderFlowersintenção. Verifique se a versão da intenção está definida como `$LATEST`, pois essa é a única versão que podemos modificar.
2. Adicione a função do Lambda como o hook de código de atendimento e teste-a.

   a. No Editor, escolha função AWS Lambda como Atendimento e selecione a função do Lambda que você criou na etapa anterior (`OrderFlowersCodeHook`). Escolha OK para ter permissão do Amazon Lex para invocar a função do Lambda.

      Você está configurando essa função do Lambda como um hook de código para cumprir a intenção. O Amazon Lex invoca essa função apenas após ter todos os dados de slot necessários do usuário para cumprir a intenção.

   b. Especifique uma Goodbye message.

   c. Escolha Criar.

   d. Teste o bot usando a conversa anterior.

   A última declaração "Obrigado, seu pedido de rosas..." é uma resposta da função do Lambda que você configurou como um hook de código. Na seção anterior, não havia função do Lambda. Agora, você está usando uma função do Lambda para realmente cumprir a intenção `OrderFlowers`.

3. Adicione a função do Lambda como um hook de código de inicialização e validação e teste.

   O exemplo de código da função do Lambda que você está usando pode executar a validação e o atendimento da entrada do usuário. O evento de entrada que a função do Lambda recebe tem um campo (`invocationSource`) que o código usa para determinar qual parte do código executar. Para obter mais informações, consulte Evento de entrada de função do Lambda e formato de resposta.

a. Selecione a versão $LATEST da intenção `OrderFlowers`. Essa é a única versão que você pode atualizar.

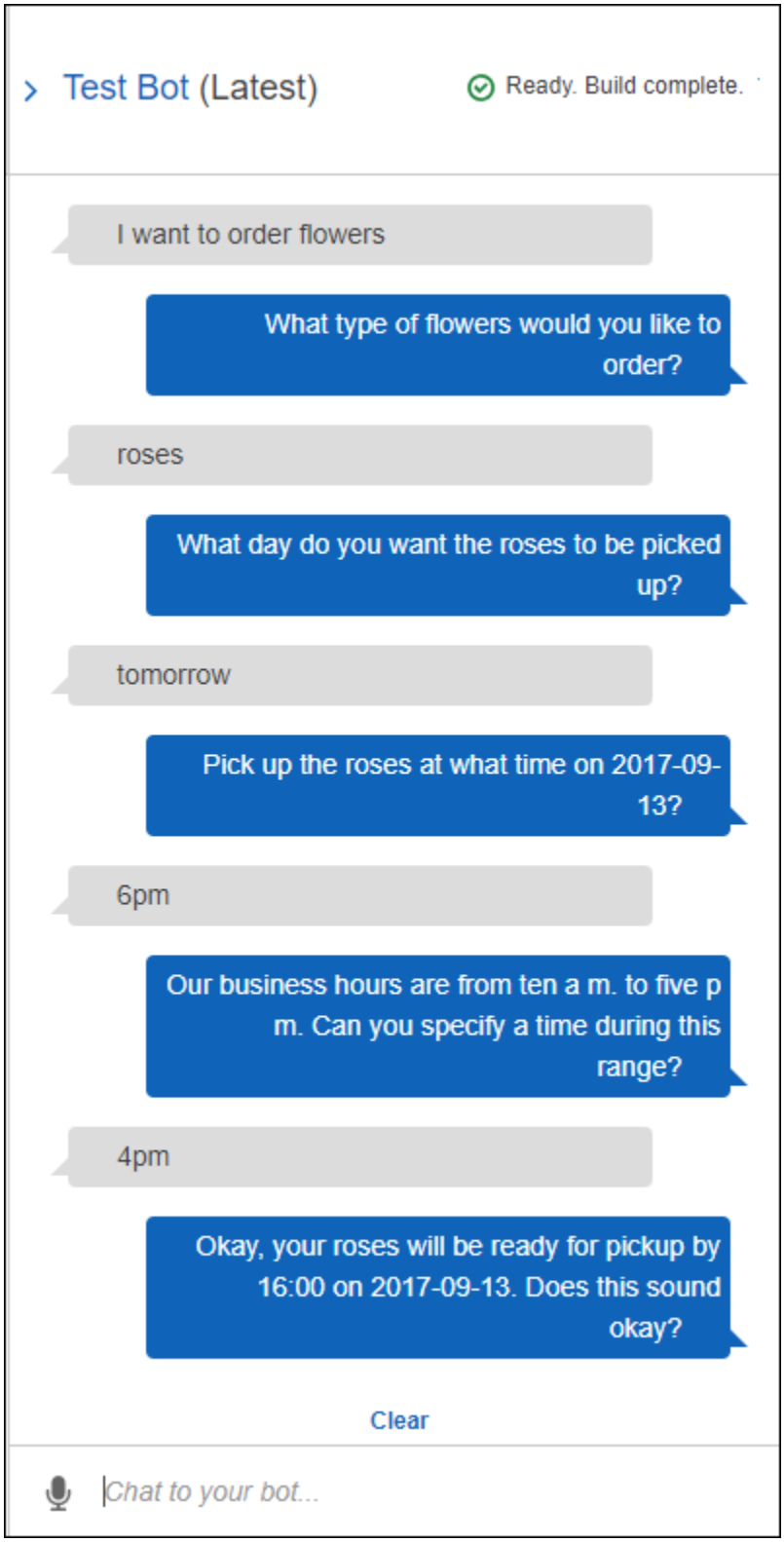
b. No Editor, escolha Initialization and validation em Options.

c. Novamente, selecione a mesma função do Lambda.

d. Escolha Criar.

e. Teste o bot.

   Agora você está pronto para conversar com o Amazon Lex do seguinte modo. Para testar a parte da validação, escolha 6 PM, e a função do Lambda retornará uma resposta ("Our business hours are from 10 AM to 5 PM.”), e enviará prompts à você novamente. Depois de fornecer todos os dados de slot válidos, a função do Lambda atende a ordem.



Próxima etapa

Etapa 5 (opcional): Revise os detalhes do fluxo de informações (console)

Etapa 5 (opcional): Revise os detalhes do fluxo de informações (console)

Esta seção explica o fluxo de informações entre o cliente e o Amazon Lex para cada entrada do usuário, incluindo a integração da função do Lambda.

> **Note**
>
> A seção supõe que o cliente envia solicitações ao Amazon Lex usando a API de runtime `PostText` e mostra os detalhes da solicitação e da resposta adequadamente. Para obter um exemplo do fluxo de informações entre o cliente e o Amazon Lex no qual o cliente usa a API `PostContent`, consulte Etapa 2a (opcional): revisar os detalhes do fluxo de informações falado (console) .

Para obter mais informações sobre a API de runtime `PostText` e detalhes adicionais sobre as solicitações e respostas mostradas nas etapas a seguir, consulte PostText.

1. Usuário: "Eu gostaria de encomendar flores."

   a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

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

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

   O URI de solicitação e o corpo fornecem informações ao Amazon Lex:

   - URI de solicitação: fornece o nome do bot (`OrderFlowers`), o alias do bot (`$LATEST`) e o nome do usuário (uma string aleatória que identifica o usuário). O `text` final indica que esta é uma solicitação de API `PostText` (e não `PostContent`).
   - Corpo da solicitação: inclui a entrada do usuário (`inputText`) e `sessionAttributes` vazio. Quando o cliente faz a primeira solicitação, não há atributos de sessão. A função do Lambda as inicia posteriormente.

b. A partir de `inputText`, o Amazon Lex detecta a intenção (`OrderFlowers`). Essa intenção é configurada com uma função do Lambda como um hook de código para inicialização e validação dos dados do usuário. Portanto, o Amazon Lex invoca aquela função do Lambda especificando as seguintes informações como dados de evento:

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

Para obter mais informações, consulte Formato de eventos de entrada.

Além das informações enviadas pelo cliente, o Amazon Lex também inclui os seguintes dados adicionais:

- `messageVersion`: atualmente, o Amazon Lex oferece suporte apenas à versão 1.0.
- `invocationSource`: indica o objetivo da invocação da função do Lambda. Nesse caso, é para executar os dados de inicialização e validação do usuário. Neste ponto, o Amazon Lex sabe que o usuário não forneceu todos os dados do slot para atender à intenção.
- Informações `currentIntent` com todos os valores de slot definidos como nulo.

c. No momento, todos os valores de slot são nulos. Não há nada para a função do Lambda validar. A função do Lambda retorna a seguinte resposta para Amazon Lex:

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

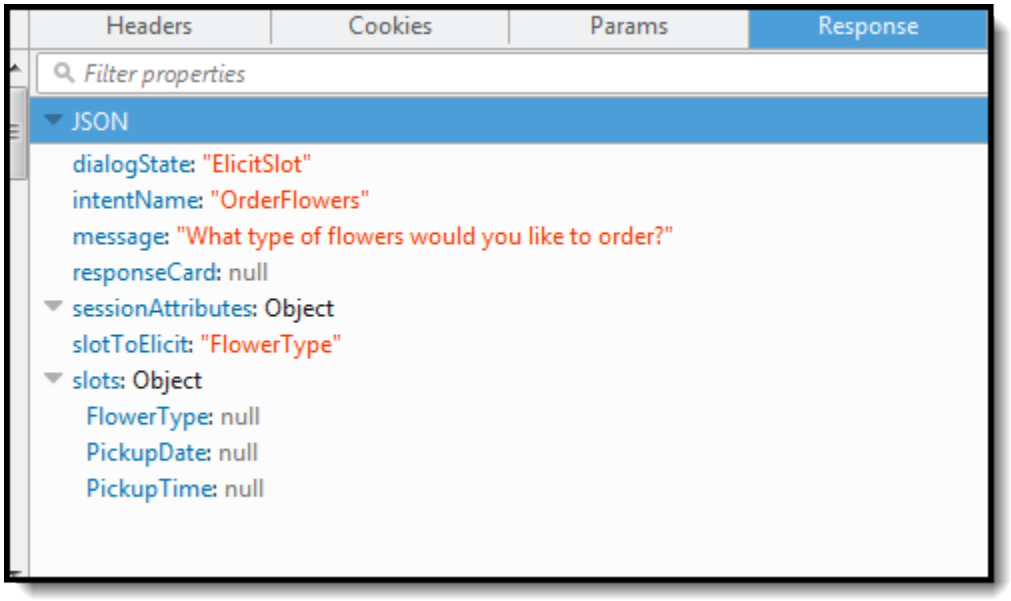
Para obter mais informações sobre o formato de resposta, consulte Formato de resposta.

Observe o seguinte:

- `dialogAction.type`: ao definir esse valor para `Delegate`, a função do Lambda delega a responsabilidade de decidir a próxima ação a Amazon Lex.

> **Note**
>
> Se a função do Lambda detectar qualquer coisa na validação dos dados do usuário, ela instruirá o Amazon Lex sobre o que fazer em seguida, conforme mostrado nas próximas etapas.

d. De acordo com o `dialogAction.type`, o Amazon Lex decide a próxima ação. Como nenhum dos slots estão preenchidos, ele decide escolher o valor para o slot `FlowerType`. Ele seleciona uma das solicitações de escolha de valor ("What type of flowers would you like to order? (Que tipo de flores você deseja encomendar?)") para este slot e envia a resposta a seguir de volta ao cliente:



O cliente exibe a mensagem na resposta.

2. Usuário: rosas

   a. O cliente envia a seguinte solicitação PostText para o Amazon Lex:

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

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

   No corpo da solicitação, o `inputText` fornece entradas do usuário. O `sessionAttributes` permanece vazio.

   b. O Amazon Lex primeiro interpreta o `inputText` no contexto da intenção atual. O serviço lembra que havia solicitado ao usuário específico informações sobre o slot `FlowerType`. Ele atualiza o valor de slot na intenção atual e invoca a função do Lambda com os seguintes dados do evento:

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

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

Observe o seguinte:

- `invocationSource` – continua sendo `DialogCodeHook` (estamos simplesmente validando os dados do usuário).
- `currentIntent.slots`: o Amazon Lex atualizou o slot `FlowerType` para rosas.

c. De acordo com o valor `invocationSource` de `DialogCodeHook`, a função do Lambda executa a validação dos dados do usuário. Ele reconhece `roses` como um valor de slot válido (e define `Price` como um atributo de sessão) e retorna a seguinte resposta ao Amazon Lex.

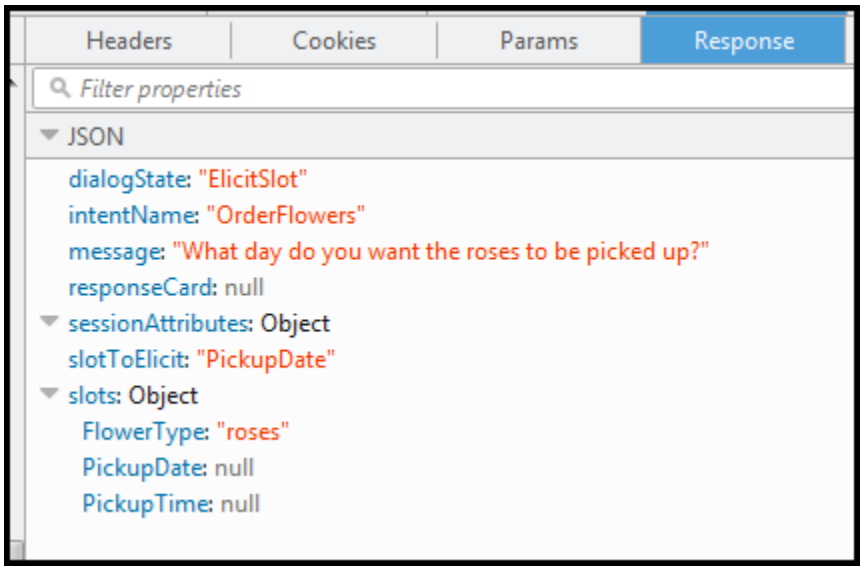
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": null
        }
    }
```

```
}
```

Observe o seguinte:

- `sessionAttributes`: a função do Lambda adicionou o `Price` (das rosas) como atributo de sessão.
- `dialogAction.type`: é definido como `Delegate`. Os dados do usuário eram válidos, então a função do Lambda direciona o Amazon Lex para escolher a próxima ação.

d. De acordo com o `dialogAction.type`, o Amazon Lex escolherá a próxima ação. O Amazon Lex sabe que precisa de mais dados de slot, então escolhe o próximo slot não preenchido (`PickupDate`) com a maior prioridade de acordo com a configuração de intenção. O Amazon Lex seleciona um dos prompts de mensagens ("Em que dia você quer que as rosas sejam retiradas?") para esse slot, de acordo com a configuração da intenção, e envia a seguinte resposta de volta ao cliente:



O cliente simplesmente exibe a mensagem na resposta – "Em que dia você deseja receber as rosas?."

3. Usuário: amanhã

a. O cliente envia a seguinte solicitação PostText para o Amazon Lex:

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

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

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

No corpo da solicitação, `inputText` fornece entradas do usuário e o cliente passa os atributos de sessão de volta ao serviço.

b. O Amazon Lex lembra do contexto para o qual estava escolhendo dados para o slot `PickupDate`. Nesse contexto, ele sabe que o valor `inputText` é para o slot `PickupDate`. Em seguida, o Amazon Lex invoca a função do Lambda enviando o seguinte evento:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {
        "Price": "25"
    },
    "bot": {
        "name": "OrderFlowersCustomWithRespCard",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        },
        "confirmationStatus": "None"
    }
}
```

O Amazon Lex atualizou o `currentIntent.slots` definindo o valor `PickupDate`. Observe também que o serviço passa o `sessionAttributes` como é para a função do Lambda.

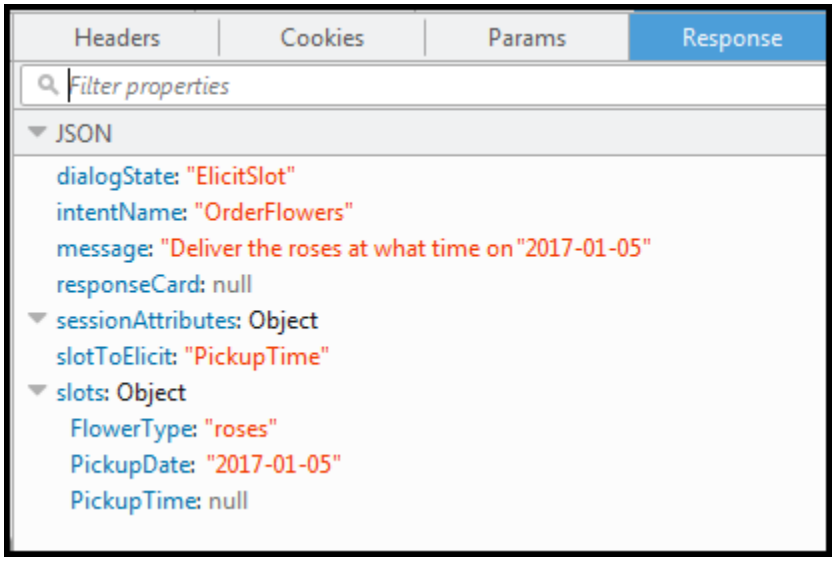
c. De acordo com o valor `invocationSource` de `DialogCodeHook`, a função do Lambda executa a validação dos dados do usuário. Ela reconhece que o valor de slot `PickupDate` é válido e retorna a seguinte resposta a Amazon Lex:

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

Observe o seguinte:

- `sessionAttributes` – sem alteração.
- `dialogAction.type`: é definido como `Delegate`. Os dados do usuário eram válidos, e a função do Lambda direciona Amazon Lex para escolher a próxima ação.

d. De acordo com o `dialogAction.type`, o Amazon Lex escolherá a próxima ação. O Amazon Lex sabe que precisa de mais dados de slot, então escolhe o próximo slot não preenchido (`PickupTime`) com a maior prioridade de acordo com a configuração de intenção. O Amazon Lex seleciona uma das mensagens de aviso ("Entregar as rosas a que horas em 05/01/2017?") para esse slot de acordo com a configuração de intenção e envia a seguinte resposta de volta ao cliente:



O cliente exibe a mensagem na resposta: "Entregar as rosas em que horário em 05/01/2017?"

4. Usuário: 16h

   a. O cliente envia a seguinte solicitação PostText para o Amazon Lex:

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

{
    "inputText": "4 pm",
    "sessionAttributes": {
        "Price": "25"
    }
}
```

   No corpo da solicitação, `inputText` fornece entradas do usuário. O cliente passa o `sessionAttributes` na solicitação.

   b. O Amazon Lex entende o contexto. Ele entende que estava escolhendo dados para o slot `PickupTime`. Neste contexto, ele sabe que o valor `inputText` é para o slot `PickupTime`. Em seguida, o Amazon Lex invoca a função do Lambda enviando o seguinte evento:

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

```
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {
        "Price": "25"
    },
    "bot": {
        "name": "OrderFlowersCustomWithRespCard",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        },
        "confirmationStatus": "None"
    }
}
```

O Amazon Lex atualizou o `currentIntent.slots` definindo o valor `PickupTime`.

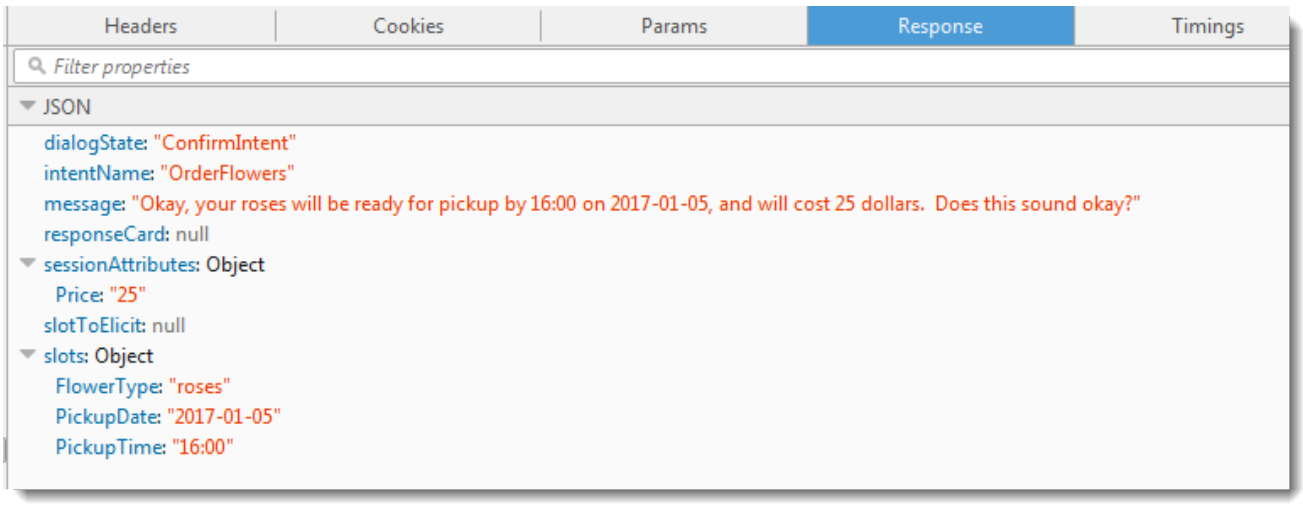
c. De acordo com o valor `invocationSource` de `DialogCodeHook`, a função do Lambda executa a validação dos dados do usuário. Ela reconhece que o valor de slot `PickupDate` é válido e retorna a seguinte resposta a Amazon Lex.

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

Observe o seguinte:

- `sessionAttributes` – nenhuma alteração no atributo de sessão.
- `dialogAction.type`: é definido como `Delegate`. Os dados do usuário eram válidos, então a função do Lambda direciona o Amazon Lex para escolher a próxima ação.

d. Neste ponto, o Amazon Lex sabe que tem todos os dados do slot. Essa intenção é configurada com um prompt de confirmação. Portanto, o Amazon Lex envia a seguinte resposta ao usuário solicitando confirmação antes de atender a intenção:



O cliente simplesmente exibe a mensagem na resposta e aguarda a resposta do usuário.

5. Usuário: Sim

a. O cliente envia a seguinte solicitação PostText para o Amazon Lex:

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

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

b. O Amazon Lex interpreta o `inputText` no contexto de confirmação da intenção atual. O Amazon Lex entende que o usuário deseja prosseguir com o pedido. Dessa vez, o Amazon Lex invoca a função do Lambda para cumprir a intenção enviando o seguinte evento, que

define o `invocationSource` como `FulfillmentCodeHook` caso ele envie para a função do Lambda. O Amazon Lex também define o `confirmationStatus` como `Confirmed`.

```
{
    "messageVersion": "1.0",
    "invocationSource": "FulfillmentCodeHook",
    "userId": "ignw84y6seypre4xly5rimopuri2xwnd",
    "sessionAttributes": {
        "Price": "25"
    },
    "bot": {
        "name": "OrderFlowersCustomWithRespCard",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "OrderFlowers",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        },
        "confirmationStatus": "Confirmed"
    }
}
```

Observe o seguinte:

- `invocationSource`: dessa vez, o Amazon Lex define esse valor como `FulfillmentCodeHook`, direcionando a função do Lambda a atender à intenção.
- `confirmationStatus`: é definido como `Confirmed`.

c. Dessa vez, a função do Lambda cumpre a intenção `OrderFlowers` e retorna a seguinte resposta:

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

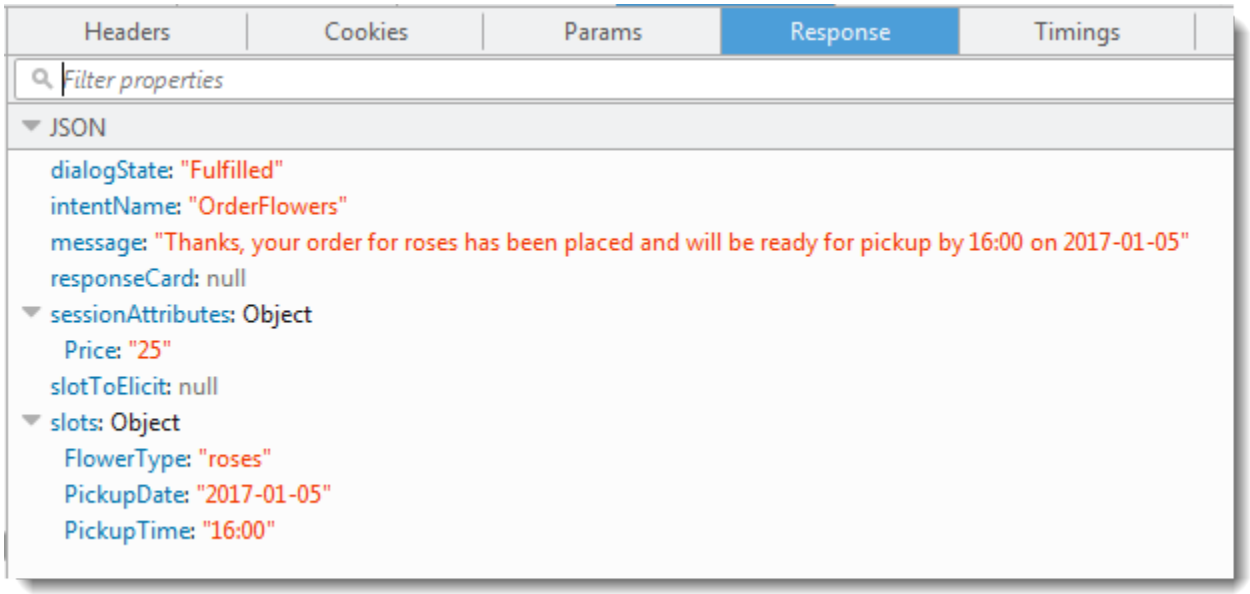
```
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}
```

Observe o seguinte:

- Define o `dialogAction.type`: a função do Lambda define esse valor como `Close`, direcionando Amazon Lex para não esperar uma resposta do usuário.
- `dialogAction.fulfillmentState` – é definido como Fulfilled e inclui uma `message` adequada para transmitir para o usuário.

d. Amazon Lex analisa o `fulfillmentState` e envia a resposta seguinte de volta para o cliente.

Em seguida, o Amazon Lex retorna o seguinte ao cliente:



Observe que:

- `dialogState`: o Amazon Lex define esse valor como `fulfilled`.
- `message`: é a mesma mensagem que a função do Lambda forneceu.

O cliente exibe a mensagem.

6. Agora teste o bot novamente. Para estabelecer um novo contexto (de usuário), selecione o link Clear na janela de teste. Agora, forneça dados de slot inválidos para a intenção `OrderFlowers`. Desta vez, a função do Lambda executa a validação dos dados, redefine os valores dos dados do slot inválidos como nulos e pede ao Amazon Lex para solicitar dados válidos ao usuário. Por exemplo, tente o seguinte:

   - Jasmim como o tipo de flor (não é um dos tipos de flor suportados).
   - Ontem como o dia em que você deseja receber as flores.
   - Após fazer o pedido, digite outro tipo de flor em vez de responder "sim" para confirmar o pedido. Em resposta, a função do Lambda atualiza o `Price` no atributo da sessão mantendo um total cumulativo de encomendas de flores.

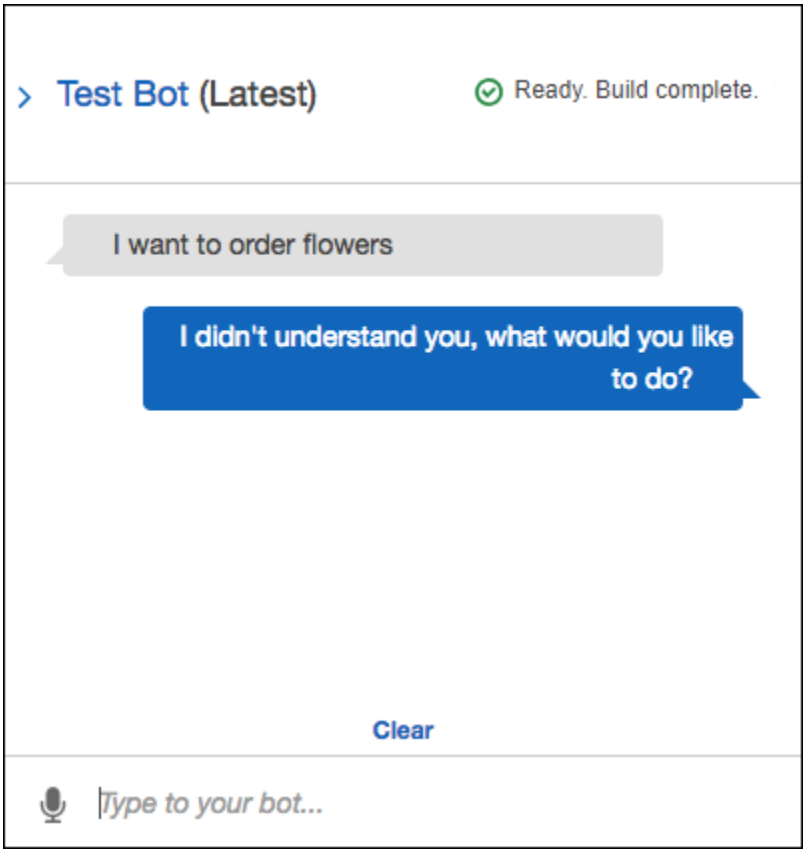
   A função do Lambda também executará a atividade de cumprimento.

Próxima etapa

Etapa 6: Atualize a configuração de intenção para adicionar uma declaração (console)

Etapa 6: Atualize a configuração de intenção para adicionar uma declaração (console)

O bot `OrderFlowers` é configurado com apenas duas declarações. Isso fornece informações limitadas ao Amazon Lex para criar um modelo de Machine Learning que reconhece e responde à intenção do usuário. Tente digitar "Desejo pedir flores" na janela de teste. O Amazon Lex não reconhece o texto e responde com "Eu não entendi, o que você gostaria de fazer?" Você pode melhorar o modelo de Machine Learning adicionando mais declarações.



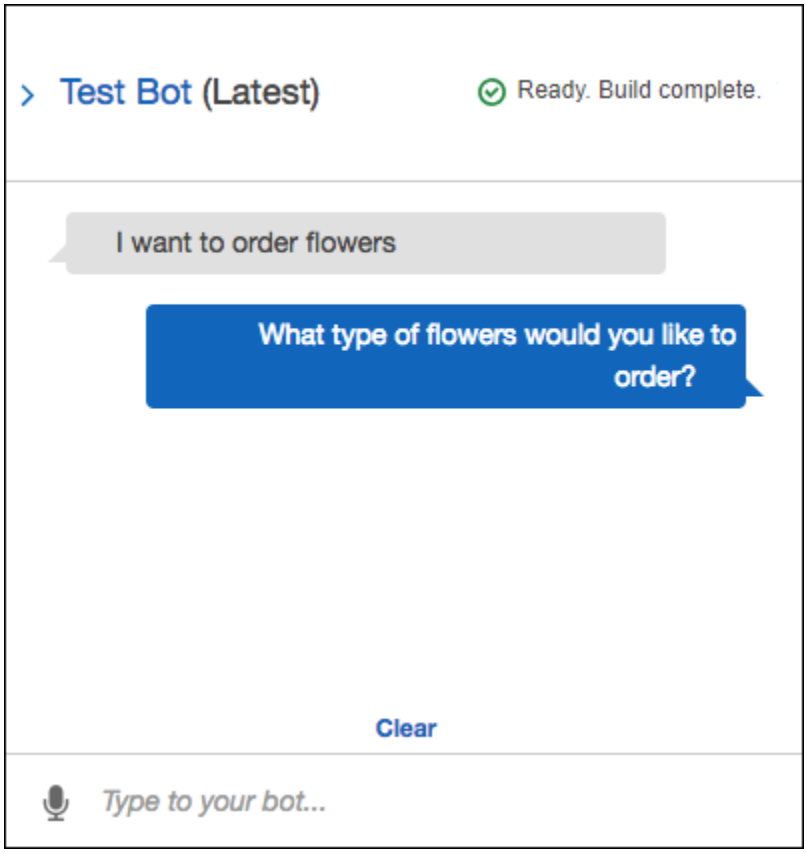
Cada declaração que você adiciona fornece ao Amazon Lex mais informações sobre como responder aos usuários. Você não precisa adicionar uma declaração exata. O Amazon Lex generaliza a partir dos exemplos que você fornece para reconhecer correspondências exatas e entradas semelhantes.

Para adicionar uma declaração (console)

1. Adicione a declaração "Quero flores" à intenção digitando-o na seção Amostra de declarações do editor de intenção e, em seguida, clicando no ícone de mais (+) próximo a nova declaração.

2. Crie seu bot para incorporar a alteração. Escolha Build e, em seguida, escolha Build novamente.

3. Teste o bot para confirmar se ele reconheceu a nova declaração. Na janela de teste, como na imagem a seguir, digite "Quero pedir flores". O Amazon Lex reconhece a frase e responde com "Que tipo de flores você gostaria de pedir?"



Próxima etapa

Etapa 7 (opcional): Limpe (console)

Etapa 7 (opcional): Limpe (console)

Agora, exclua os recursos que você criou e limpe sua conta.

Você só pode excluir os recursos que não estão em uso. Em geral, você deve excluir recursos na seguinte ordem:

- Exclua bots para liberar recursos de intenção.
- Exclua intenções para liberar recursos de tipo de slot.
- Exclua tipos de slot por último.

Para limpar sua conta (console)

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 a caixa de seleção ao lado de OrderFlowers.
3. Para excluir o bot, escolha Delete e, em seguida, escolha Continue na caixa de diálogo de confirmação.
4. No painel à esquerda, selecione Intents.
5. Na lista de intenções, escolha OrderFlowersIntent.
6. Para excluir a intenção, selecione Delete e, em seguida, escolha Continue na caixa de diálogo de confirmação.
7. No painel esquerdo, escolha Slot types.
8. Na lista de tipos de slot, escolha Flowers.
9. Para excluir o tipo de slot, selecione Delete e, em seguida, escolha Continue na caixa de diálogo de confirmação.

Você removeu todos os recursos Amazon Lex que criou e limpou sua conta. Se desejar, você pode usar o console do Lambda para excluir a função do Lambda usada neste exercício.

Exercício 2: criar um bot personalizado do Amazon Lex

Neste exercício, você usa o console do Amazon Lex para criar um bot personalizado para encomendar pizza (`OrderPizzaBot`). Você configura o bot adicionando uma intenção personalizada (`OrderPizza`), definindo tipos de slot personalizados e definindo os slots necessários para um pedido de pizza (massa da pizza, tamanho e assim por diante). Para obter mais informações sobre tipos de slot e slots, consulte Amazon Lex: como funciona.

Tópicos

- Etapa 1: Criar uma função do Lambda
- Etapa 2: criar um bot
- Etapa 3: crie e teste o bot
- Etapa 4 (opcional): Limpeza

Etapa 1: Criar uma função do Lambda

Primeiro, crie uma função do Lambda que atenda a um pedido de pizza. Essa função é específica no bot do Amazon Lex, que será criado na próxima seção.

Criar uma função do Lambda

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função.
3. Na página Create function, selecione Author from scratch.

   Como você está usando um código personalizado fornecido neste exercício para criar uma função do Lambda, selecione essa opção.

   Faça o seguinte:

   a. Digite o nome: (`PizzaOrderProcessor`).
   b. Em Runtime (Tempo de execução), selecione a versão mais recente do Node.js.
   c. Em Role, selecione Create new role from template(s).
   d. Insira um novo nome de função (`PizzaOrderProcessorRole`).
   e. Escolha a opção Criar função.
4. Na página function, faça o seguinte:

   Na seção Function code, selecione Edit code inline e, em seguida, copie o código da função Node.js a seguir e cole-o na janela.

```
'use strict';

// Close dialog with the customer, reporting fulfillmentState of Failed or
 Fulfilled ("Thanks, your pizza will arrive in 20 minutes")
function close(sessionAttributes, fulfillmentState, message) {
    return {
        sessionAttributes,
        dialogAction: {
            type: 'Close',
            fulfillmentState,
            message,
        },
```

```
    };
}

// --------------- Events -----------------------

function dispatch(intentRequest, callback) {
    console.log(`request received for userId=${intentRequest.userId}, intentName=
${intentRequest.currentIntent.name}`);
    const sessionAttributes = intentRequest.sessionAttributes;
    const slots = intentRequest.currentIntent.slots;
    const crust = slots.crust;
    const size = slots.size;
    const pizzaKind = slots.pizzaKind;

    callback(close(sessionAttributes, 'Fulfilled',
    {'contentType': 'PlainText', 'content': `Okay, I have ordered your ${size}
 ${pizzaKind} pizza on ${crust} crust`}));

}

// --------------- Main handler -----------------------

// Route the incoming request based on intent.
// The JSON body of the request is provided in the event slot.
export const handler = (event, context, callback) => {
    try {
        dispatch(event,
            (response) => {
                callback(null, response);
            });
    } catch (err) {
        callback(err);
    }
};
```

5. Escolha Salvar.

Testar a função do Lambda usando dados de eventos de exemplo

No console, teste a função do Lambda usando dados de eventos de exemplo para invocá-la manualmente.

Para testar a função do Lambda:

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Na página da função do Lambda, escolha a função do Lambda (`PizzaOrderProcessor`).
3. Na página da função, na lista de eventos de teste, selecione Configure test events.
4. Na página Configure test event, faça o seguinte:
   a. Selecione Criar evento de teste.
   b. No campo Event name, insira um nome para o evento (`PizzaOrderProcessorTest`).
   c. Copie o evento do Amazon Lex a seguir na janela.

```
{
  "messageVersion": "1.0",
  "invocationSource": "FulfillmentCodeHook",
  "userId": "user-1",
  "sessionAttributes": {},
  "bot": {
    "name": "PizzaOrderingApp",
    "alias": "$LATEST",
    "version": "$LATEST"
  },
  "outputDialogMode": "Text",
  "currentIntent": {
    "name": "OrderPizza",
    "slots": {
      "size": "large",
      "pizzaKind": "meat",
      "crust": "thin"
    },
    "confirmationStatus": "None"
  }
}
```

5. Escolha Criar.

AWS Lambda cria o teste e você volta para a página da função. Escolha Testar para que o Lambda execute a função do Lambda.

Na caixa de resultados, selecione Details. O console exibe a saída a seguir no painel Execution result.

```
{
  "sessionAttributes": {},
  "dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled",
    "message": {
      "contentType": "PlainText",
      "content": "Okay, I have ordered your large meat pizza on thin crust."
    }
}
```

Próxima etapa

Etapa 2: criar um bot

Etapa 2: criar um bot

Nesta etapa, você cria um bot para lidar com os pedidos de pizza.

Tópicos

- Criar o bot
- Criar uma intenção
- Criar tipos de slot
- Configurar a intenção
- Configurar o bot do

Criar o bot

Crie o bot `PizzaOrderingBot` com o mínimo de informações necessárias. Você adiciona uma intenção, uma ação que o usuário deseja executar, para o bot mais tarde.

Para criar o bot

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Crie um bot.

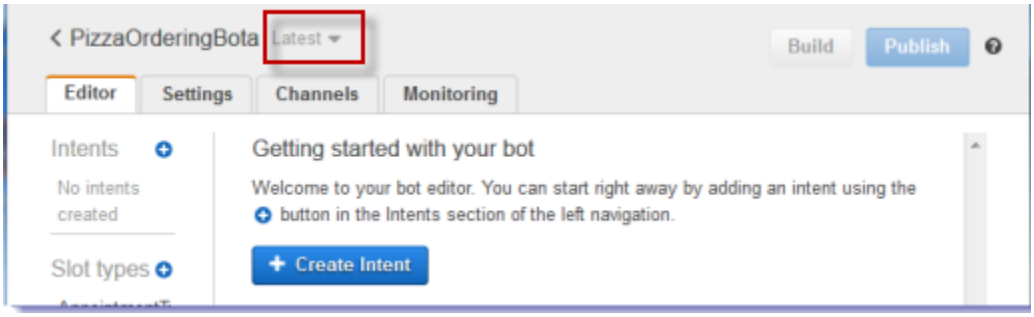
a. Se você estiver criando seu primeiro bot, escolha Get Started. Caso contrário, escolha Bots e, em seguida, Create.

b. Na página Create your Lex bot escolha Custom bot e forneça as seguintes informações:

- Nome do bot: PizzaOrderingBot
- Idioma: escolha o idioma e a localidade do seu bot.
- Output voice: Salli
- Session timeout : 5 minutos.
- COPPA: escolha a resposta apropriada.
- Armazenamento da declaração do usuário: escolha a resposta apropriada.

c. Escolha Criar.

O console envia ao Amazon Lex uma solicitação para criar um novo bot. O Amazon Lex define a versão do bot como `$LATEST`. Depois de criar o bot, o Amazon Lex mostra a guia Editor do bot, como na imagem a seguir:



- A versão Latest do bot encontra-se no console, ao lado do nome do bot. Os novos recursos do Amazon Lex têm `$LATEST` como versão. Para obter mais informações, consulte Versionamento e aliases.
- Como você não criou nenhuma intenção ou tipos de slots, não há nenhum listado.
- Build e Publish são atividades no nível do bot. Depois de configurar todo o bot, você aprenderá mais sobre essas atividades.

Próxima etapa

Criar uma intenção

Criar uma intenção

Agora, crie a intenção `OrderPizza`, uma ação que o usuário deseja executar, com o mínimo de informações necessárias. Você adiciona tipos de slot para a intenção e, mais tarde, configura a intenção.

Para criar uma intenção

1. No console do Amazon Lex escolha o sinal de adição (+) ao lado de Intenções e, em seguida, escolha Criar nova intenção.
2. Na caixa de diálogo Create intent, digite o nome da intenção (`OrderPizza`) e escolha Add.

O console envia uma solicitação ao Amazon Lex para criar a intenção `OrderPizza`. Neste exemplo, você cria slots para a intenção depois de criar tipos de slots.

Próxima etapa

Criar tipos de slot

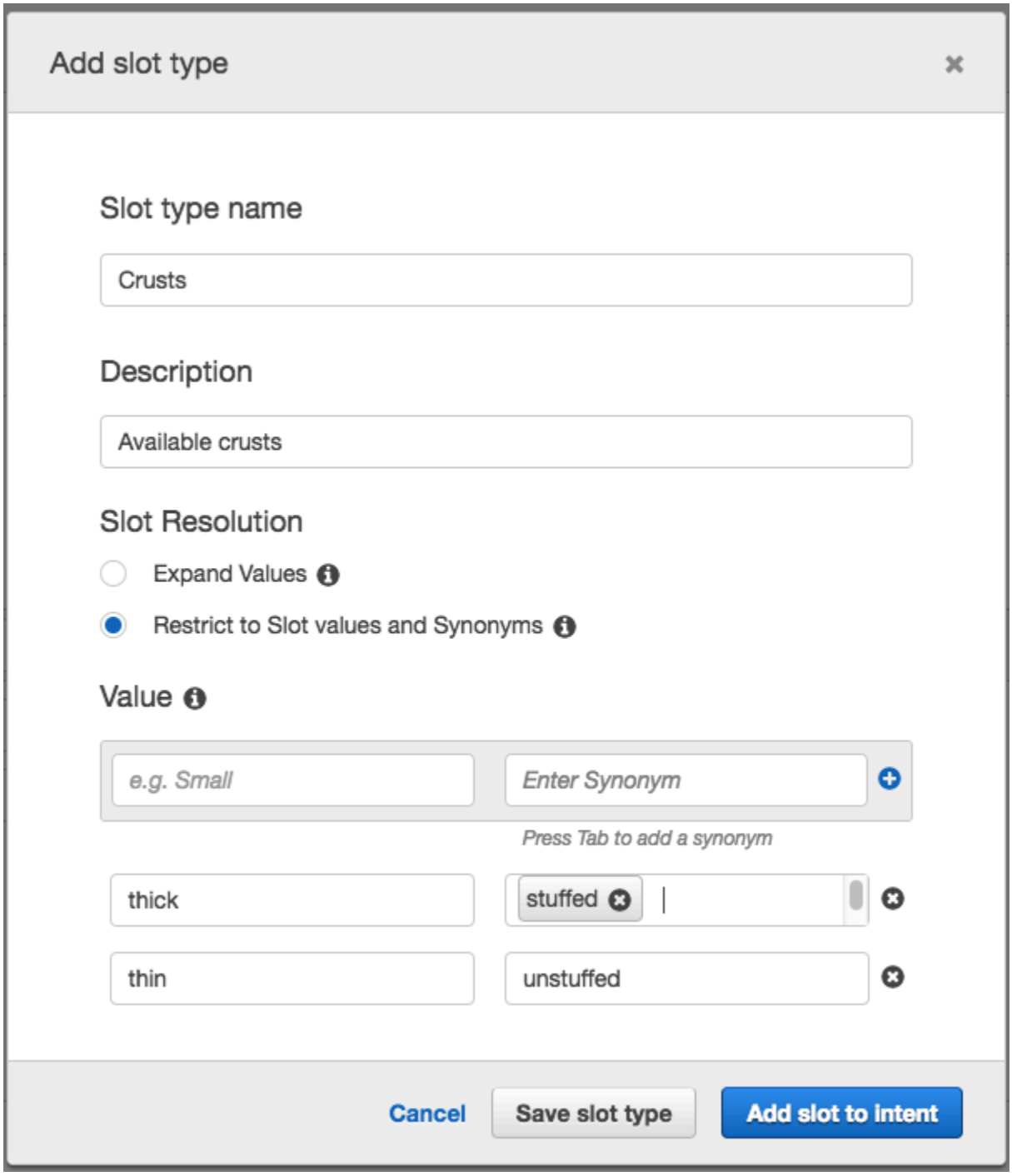
Criar tipos de slot

Crie tipos de slot ou valores de parâmetro que a intenção `OrderPizza` usa.

Para criar tipos de slot

1. No menu esquerdo, escolha o sinal de mais (+) ao lado de Slot types.
2. Na caixa de diálogo Add slot type, adicione o seguinte:
   - Slot type name – Massas
   - Description – Massas disponíveis
   - Escolha Restrict to Slot values and Synonyms
   - Valor: Type **`thick`**. Selecione a guia e, no campo Synonym (Sinônimo), digite **`stuffed`**. Escolha o sinal de adição (+). Digite **`thin`** e, em seguida, escolha o sinal de adição (+) novamente.

   A caixa de diálogo deve ter a seguinte aparência:



3. Escolha Add slot to intent.
4. Na página Intent, escolha Required. Altere o nome do slot de **`slotOne`** para **`crust`**. Mude a solicitação para **`What kind of crust would you like?`**.
5. Repita Step 1 a Step 4 usando os valores da seguinte tabela:

Nome	Descrição	Valores	Nome do slot	Prompt
Sizes	Tamanhos disponíveis	pequena, média e grande	size	qual tamanho de pizza?
PizzaKind	Pizzas disponíveis	vegetariana, queijo	pizzaKind	Quer uma pizza de queijo ou vegetariana?

Próxima etapa

Configurar a intenção

Configurar a intenção

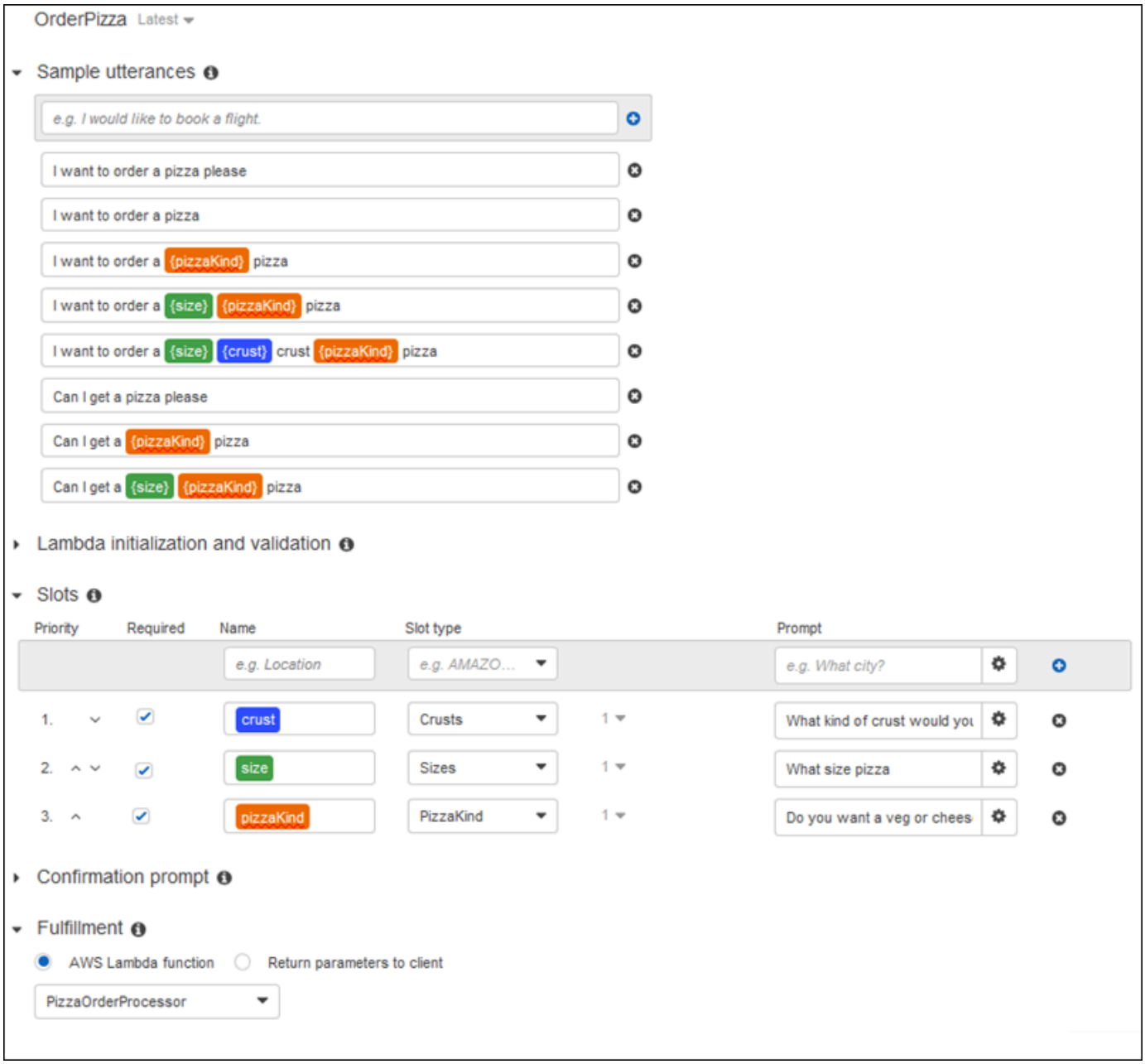
Configure a intenção `OrderPizza` para atender à solicitação do usuário de pedir uma pizza.

Para configurar uma intenção

- Na página OrderPizzade configuração, configure a intenção da seguinte forma:
  - Amostra de declaraçõe: digite as strings a seguir. As chaves {} contém nomes de slot.
    - Quero pedir pizza
    - Quero pedir uma pizza
    - Quero pedir uma pizza de {pizzaKind}
    - Quero pedir uma pizza {size} de {pizzaKind}
    - Quero uma pizza {size} de {pizzaKind} com massa {crust}
    - Posso pedir uma pizza
    - Posso pedir uma pizza de {pizzaKind}
    - Posso pedir uma pizza {size} de {pizzaKind}
  - Lambda initialization and validation – Deixe a configuração padrão.
  - Confirmation prompt – Deixe a configuração padrão.
  - Atendimento: execute as seguintes tarefas:
    - Escolha AWS Lambda function (Função do Lambda).
    - Selecione **`PizzaOrderProcessor`**.

- Se a caixa de diálogo Adicionar permissão à função do Lambda for mostrada, escolha OK para dar à `OrderPizza` intenção a permissão para chamar a `PizzaOrderProcessor` função do Lambda.
- Deixe None selecionado.

A intenção deve ter a seguinte aparência:



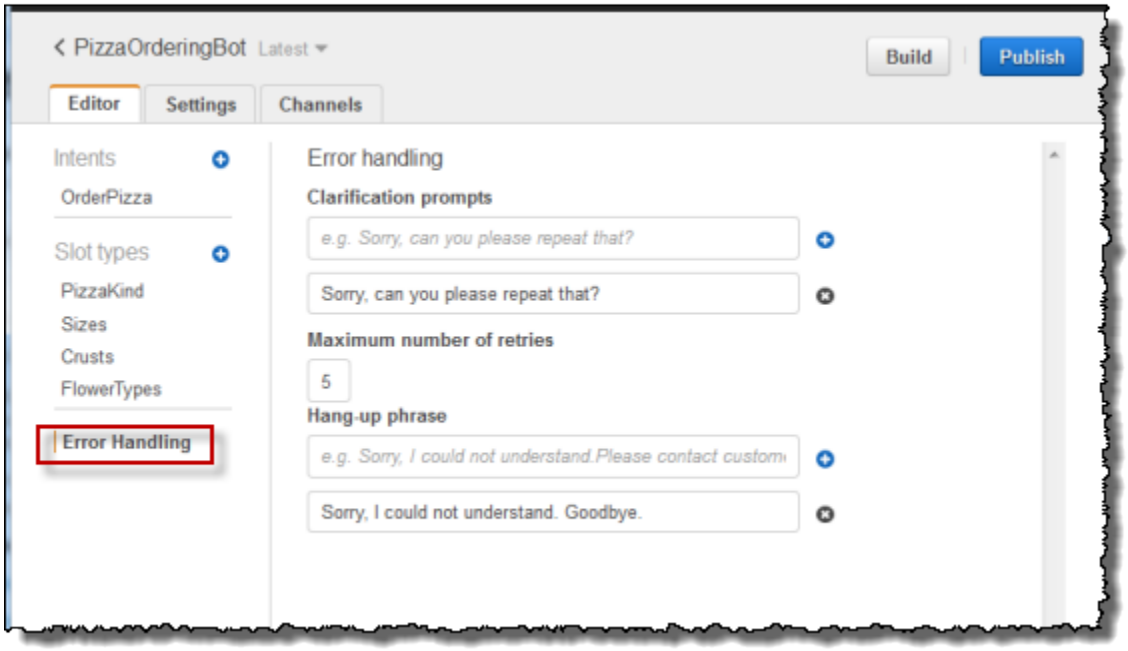
Próxima etapa

Configurar o bot do

Configurar o bot do

Configure a manipulação de erros para o bot `PizzaOrderingBot`.

1. Navegue até o bot `PizzaOrderingBot`. Escolha Editor e, em seguida, escolha Tratamento de erros, como na imagem a seguir:



2. Use a guia Editor para configurar a manipulação de erros do bot.

   - As informações que você fornece em Clarification Prompts (Solicitações de esclarecimento) são mapeadas para a configuração clarificationPrompt do bot.

     Quando o Amazon Lex não consegue determinar a intenção do usuário, o serviço retorna uma resposta com esta mensagem.
   - As informações que você fornece na frase Hang-up (Encerramento) são mapeadas para a configuração abortStatement do bot.

     Se o serviço não puder determinar a intenção do usuário depois de um determinado número de solicitações consecutivas, o Amazon Lex retornará uma resposta com esta mensagem.

   Deixe os valores padrão.

Próxima etapa

Etapa 3: crie e teste o bot

Etapa 3: crie e teste o bot

Crie e teste o bot para garantir que ele funciona.

Para criar e testar o bot

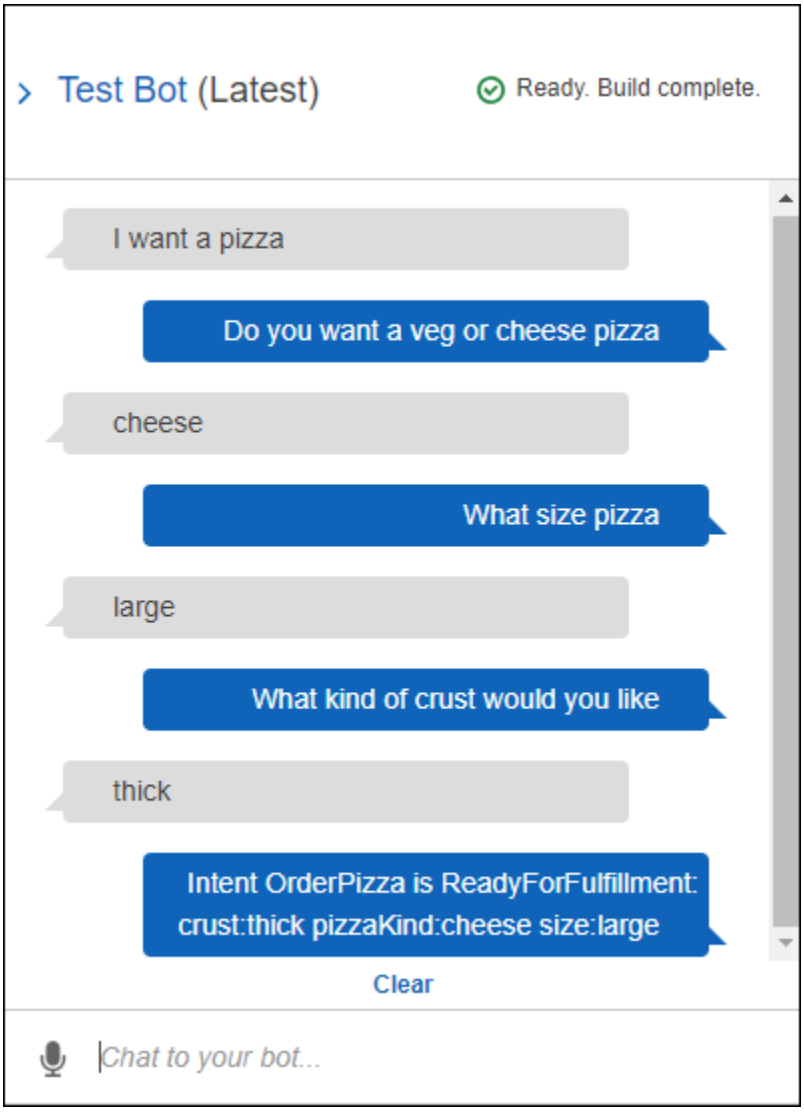
1. Para criar o bot `PizzaOrderingBot`, escolha Build.

   O Amazon Lex compila um modelo de machine learning para o bot. Quando você testa o bot, o console usa a API de runtime para enviar a entrada do usuário de volta para o Amazon Lex. Em seguida, o Amazon Lex usa o modelo de machine learning para interpretar a entrada do usuário.

   Pode levar algum tempo para concluir a criação.

2. Para testar o bot, na janela Testar bot, comece a se comunicar com o bot de do Amazon Lex.

   - Por exemplo, você pode dizer ou digitar:



- Use a amostra de declarações que você configurou na intenção `OrderPizza` para testar o bot. Por exemplo, a amostra a seguir é uma das amostras de declarações que você configurou para a intenção `PizzaOrder`:

  ```
  I want a {size} {crust} crust {pizzaKind} pizza
  ```

  Para testá-la, digite o seguinte:

  ```
  I want a large thin crust cheese pizza
  ```

Quando você digita "Quero encomendar uma pizza", o Amazon Lex detecta a intenção (`OrderPizza`). Em seguida, o Amazon Lex solicita informações de slot.

Depois que você fornece todas as informações do slot, o Amazon Lex invoca a função do Lambda que você configurou para a intenção.

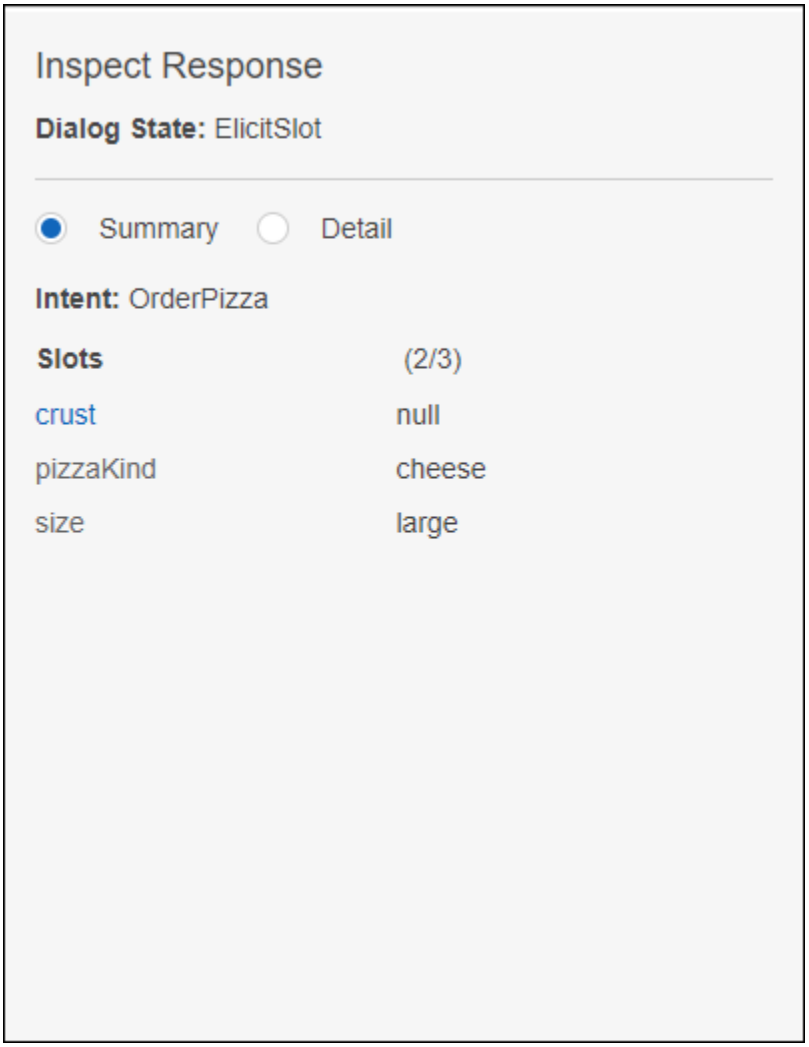
A função do Lambda retorna uma mensagem ("OK, eu pedi sua...") para o Amazon Lex, que o Amazon Lex retorna para você.

Inspeção da resposta

Abaixo da janela de chat, há um painel que permite inspecionar a resposta do Amazon Lex. O painel fornece informações completas sobre o estado do seu bot, que muda conforme você interage com ele. O conteúdo do painel mostra o estado atual da operação.

- Estado do diálogo: o estado atual da conversa com o usuário. Ele pode ser `ElicitIntent`, `ElicitSlot`, `ConfirmIntent` ou `Fulfilled`.

- Resumo: mostra uma visão simplificada da caixa de diálogo, que mostra os valores de slot para que a intenção seja cumprida, permitindo que você controle o fluxo de informações. Ele mostra o nome do método, o número de slots, o número de slots preenchidos e uma lista de slots e seus valores associados. Veja a imagem a seguir:



- Detalhe: mostra a resposta JSON bruta do chatbot para proporcionar uma visão mais detalhada da interação do bot e o estado atual da caixa de diálogo à medida que você testa e depura o chatbot. Se você digitar na janela de bate-papo, o painel de inspeção mostra a resposta JSON da operação PostText. Se você falar com a janela de bate-papo, o painel de inspeção mostra o cabeçalho da resposta da operação PostContent. Veja a imagem a seguir:

Inspect Response

Dialog State: ElicitSlot

Summary Detail

RequestID: 41392c21-97ff-11e7-a10b-5bcc0093a006


```
{
  "dialogState": "ElicitSlot",
  "intentName": "OrderPizza",
  "message": "What kind of crust would you like",
  "responseCard": null,
  "sessionAttributes": {},
  "slotToElicit": "crust",
  "slots": {
    "crust": null,
    "pizzaKind": "cheese",
    "size": "large"
  }
}
```


Próxima etapa

Etapa 4 (opcional): Limpeza

Etapa 4 (opcional): Limpeza

Exclua os recursos que você criou e limpe sua conta para evitar mais cobranças pelos os recursos que você criou.

Você só pode excluir os recursos que não estão em uso. Por exemplo, você não pode excluir um tipo de slot que seja referenciado por uma intenção. Você não pode excluir uma intenção que seja referenciada por um bot.

Exclua recursos na seguinte ordem:

- Exclua bots para liberar recursos de intenção.

- Exclua intenções para liberar recursos de tipo de slot.
- Exclua tipos de slot por último.

Para limpar sua conta

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 PizzaOrderingBot.
3. Para excluir o bot, escolha Delete e, em seguida, escolha Continue.
4. No painel à esquerda, selecione Intents.
5. Na lista de intenções, escolha OrderPizza.
6. Para excluir a intenção, escolha Delete e, em seguida, escolha Continue.
7. No menu esquerdo, escolha Slot types.
8. Na lista de tipos de slot, escolha Crusts.
9. Para excluir o tipo de slot, escolha Delete e, em seguida, escolha Continue.
10. Repita Step 8 e Step 9 para os tipos de slot `Sizes` e `PizzaKind`.

Você removeu todos os recursos que criou e limpou sua conta.

Próximas etapas

- Publique uma versão e crie um alias.
- Crie um bot Amazon Lex com o AWS Command Line Interface

Exercício 3: publique uma versão e crie um alias

Nos Exercícios 1 e 2 de Conceitos básicos, você criou e testou um bot. Neste exercício, você faz o seguinte:

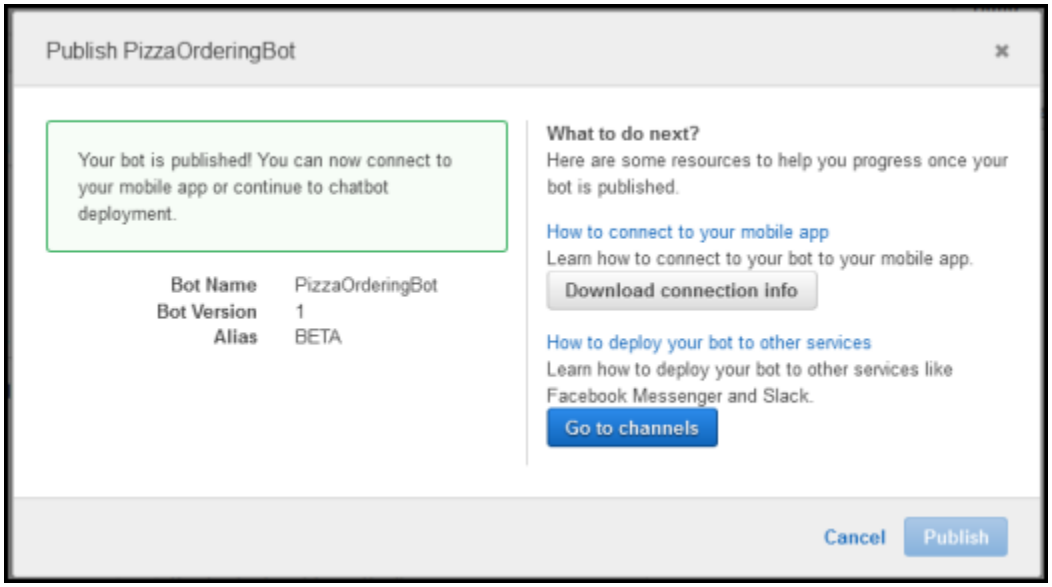
- Publique uma nova versão do bot. O Amazon Lex faz uma cópia de snapshot da versão `$LATEST` para publicar uma nova versão.
- Crie um alias que aponte para a versão nova.

Para obter mais informações sobre versionamento e aliases, consulte Versionamento e aliases.

Faça o seguinte para publicar uma versão de um bot criado neste exercício:

1. No console do Amazon Lex, escolha um dos bots que você criou.

   Verifique se o console mostra o `$LATEST` como a versão do bot ao lado do nome do bot.
2. Selecione Publish.
3. No *`botname`* assistente de publicação, especifique o **`BETA`** alias e escolha Publicar.
4. Verifique se o console do Amazon Lex mostra a nova versão ao lado do nome do bot, como mostrado na imagem a seguir.



Agora que você tem um bot funcionando com versão publicada e um alias, pode implantá-lo (em sua aplicação móvel ou integrar o bot com o Facebook Messenger). Para ver um exemplo, consulte Integração de um bot do Amazon Lex com o Facebook Messenger.

Etapa 4: Conceitos básicos (AWS CLI)

Nesta etapa, você usa o AWS CLI para criar, testar e modificar um bot do Amazon Lex. Para concluir esses exercícios, você precisa estar familiarizado com o uso da CLI e tem um editor de texto. Para ter mais informações, consulte Etapa 2: Configurar o AWS Command Line Interface

- Exercício 1: Criar e testar um bot do Amazon Lex. O exercício fornece todos os objetos JSON de que você precisa para criar um tipo de slot personalizado, uma intenção e um bot. Para ter mais informações, consulte Amazon Lex: como funciona

- Exercício 2: atualize o bot que você criou no Exercício 1 para adicionar mais uma amostra de declaração. Amazon Lex usa amostra de declarações para criar o modelo de Machine Learning para o seu bot.
- Exercício 3: atualize o bot que você criou no exercício 1 para adicionar uma função do Lambda para validar a entrada do usuário e atender à intenção.
- Exercício 4: Publicar uma versão do tipo de slot, intenção e recursos do bot que você criou no Exercício 1. Uma versão é um snapshot de um recurso que não pode ser alterada.
- Exercício 5: Criar um alias para o bot que você criou no Exercício 1.
- Exercício 6: Limpar sua conta excluindo o tipo de slot, a intenção e o bot que você criou no Exercício 1, e o alias que você criou no Exercício 5.

Tópicos

- Exercício 1: Criar um bot do Amazon Lex (AWS CLI)
- Exercício 2: Adicionar uma nova declaração (AWS CLI)
- Exercício 3: Adicione uma função do Lambda (AWS CLI)
- Exercício 4: Publicar uma versão (AWS CLI)
- Exercício 5: Criar um alias (AWS CLI)
- Exercício 6: Limpar (AWS CLI)

Exercício 1: Criar um bot do Amazon Lex (AWS CLI)

Em geral, quando você cria bots, você:

1. Cria tipos de slot para definir as informações com as quais seu bot vai trabalhar.
2. Cria as intenções que definem as ações do usuário a que o bot oferece suporte. Use os tipos de slot personalizados que você criou anteriormente para definir os slots, ou parâmetros, que a sua intenção requer.
3. Crie um bot que usa as intenções que você definiu.

Neste exercício, você cria e testa um novo bot do Amazon Lex usando a CLI. Use as estruturas JSON que fornecemos para criar o bot. Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Tópicos

- Etapa 1: Criar uma função vinculada a serviço (AWS CLI)
- Etapa 2: criar um tipo de slot personalizado (AWS CLI)
- Etapa 3: Criar uma intenção (AWS CLI)
- Etapa 4: Criar um bot (AWS CLI)
- Etapa 5: Testar um bot (AWS CLI)

Etapa 1: Criar uma função vinculada a serviço (AWS CLI)

O Amazon Lex assume funções AWS Identity and Access Management vinculadas a serviços para chamar AWS serviços em nome de seus bots. As funções, que estão em sua conta, são vinculadas aos casos de uso do Amazon Lex e têm permissões predefinidas. Para obter mais informações, consulte Uso de perfis vinculadas ao serviço para o Amazon Lex.

Se você já tiver criado um bot do Amazon Lex usando o console, a função vinculada a serviço terá sido criada automaticamente. Vá para Etapa 2: criar um tipo de slot personalizado (AWS CLI).

Para criar um perfil vinculado ao serviço (AWS CLI)

1. No AWS CLI, digite o seguinte comando:

```
aws iam create-service-linked-role --aws-service-name lex.amazonaws.com
```

2. Verifique a política usando o seguinte comando:

```
aws iam get-role --role-name AWSServiceRoleForLexBots
```

A resposta é:

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
```

```
                        "Service": "lex.amazonaws.com"
                    }
                }
            ]
        },
        "RoleName": "AWSServiceRoleForLexBots",
        "Path": "/aws-service-role/lex.amazonaws.com/",
        "Arn": "arn:aws:iam::account-id:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
}
```

Próxima etapa

Etapa 2: criar um tipo de slot personalizado (AWS CLI)

Etapa 2: criar um tipo de slot personalizado (AWS CLI)

Crie um tipo de slot personalizado com valores de enumeração para as flores que podem ser solicitadas. Você vai usar esse tipo na próxima etapa quando criar a intenção `OrderFlowers`. Um tipo de slot define os valores possíveis para um slot, ou parâmetro, da intenção.

Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Para criar um tipo de slot personalizado (AWS CLI)

1. Crie um arquivo de texto chamado **`FlowerTypes.json`**. Copie o seguinte código JSON de FlowerTypes.json para o arquivo de texto.
2. Chame a PutSlotType operação usando o AWS CLI para criar o tipo de slot. O exemplo é formatado para Unix, Linux e macOS. Para Windows, substitua o caractere de continuação Unix de barra invertida (\) no final de cada linha por um circunflexo (^).

   ```
   aws lex-models put-slot-type \
       --region region \
       --name FlowerTypes \
       --cli-input-json file://FlowerTypes.json
   ```

   A resposta do servidor é:

   ```
   {
   ```

```
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "checksum": "checksum",
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Próxima etapa

Etapa 3: Criar uma intenção (AWS CLI)

FlowerTypes.json

O código a seguir são os dados JSON necessários para criar o tipo de slot personalizado `FlowerTypes`:

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "description": "Types of flowers to pick up"
}
```

Etapa 3: Criar uma intenção (AWS CLI)

Crie uma intenção para o bot `OrderFlowersBot` e forneça três slots, ou parâmetros. Os slots permitem que o bot cumpra a intenção:

- `FlowerType` é um tipo de slot personalizado que especifica quais tipos de flores podem ser solicitados.
- `AMAZON.DATE` e `AMAZON.TIME` são tipos de slot integrados usados para obter a data e a hora para entregar as flores do usuário.

Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Para criar a intenção **`OrderFlowers`** (AWS CLI)

1. Crie um arquivo de texto chamado **`OrderFlowers.json`**. Copie o seguinte código JSON de OrderFlowers.json para o arquivo de texto.
2. No AWS CLI, chame a PutIntent operação para criar a intenção. O exemplo é formatado para Unix, Linux e macOS. Para Windows, substitua o caractere de continuação Unix de barra invertida (\) no final de cada linha por um circunflexo (^).

```
aws lex-models put-intent \
   --region region \
   --name OrderFlowers \
   --cli-input-json file://OrderFlowers.json
```

O servidor responde com o seguinte:

```
{
    "confirmationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "Okay, your {FlowerType} will be ready for pickup by
 {PickupTime} on {PickupDate}.  Does this sound okay?",
                "contentType": "PlainText"
            }
```

```
        ]
    },
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
    "rejectionStatement": {
        "messages": [
            {
                "content": "Okay, I will not place your order.",
                "contentType": "PlainText"
            }
        ]
    },
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "sampleUtterances": [
        "I would like to pick up flowers",
        "I would like to order some flowers"
    ],
    "slots": [
        {
            "slotType": "AMAZON.TIME",
            "name": "PickupTime",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "Pick up the {FlowerType} at what time on
{PickupDate}?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 3,
            "description": "The time to pick up the flowers"
        },
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
```

```
                    {
                        "content": "What type of flowers would you like to
 order?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "$LATEST",
            "sampleUtterances": [
                "I would like to order {FlowerType}"
            ],
            "description": "The type of flowers to pick up"
        },
        {
            "slotType": "AMAZON.DATE",
            "name": "PickupDate",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What day do you want the {FlowerType} to be
 picked up?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 2,
            "description": "The date to pick up the flowers"
        }
    ],
    "fulfillmentActivity": {
        "type": "ReturnIntent"
    },
    "description": "Intent to order a bouquet of flowers for pick up"
}
```

Próxima etapa

[Etapa 4: Criar um bot (AWS CLI)]

OrderFlowers.json

O código a seguir são os dados JSON necessários para criar a intenção `OrderFlowers`:

```
{
    "confirmationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "Okay, your {FlowerType} will be ready for pickup by
 {PickupTime} on {PickupDate}.  Does this sound okay?",
                "contentType": "PlainText"
            }
        ]
    },
    "name": "OrderFlowers",
    "rejectionStatement": {
        "messages": [
            {
                "content": "Okay, I will not place your order.",
                "contentType": "PlainText"
            }
        ]
    },
    "sampleUtterances": [
        "I would like to pick up flowers",
        "I would like to order some flowers"
    ],
    "slots": [
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers would you like to order?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "$LATEST",
```

```
            "sampleUtterances": [
                "I would like to order {FlowerType}"
            ],
            "description": "The type of flowers to pick up"
        },
        {
            "slotType": "AMAZON.DATE",
            "name": "PickupDate",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What day do you want the {FlowerType} to be picked
up?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 2,
            "description": "The date to pick up the flowers"
        },
        {
            "slotType": "AMAZON.TIME",
            "name": "PickupTime",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "Pick up the {FlowerType} at what time on
{PickupDate}?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 3,
            "description": "The time to pick up the flowers"
        }
    ],
    "fulfillmentActivity": {
        "type": "ReturnIntent"
    },
    "description": "Intent to order a bouquet of flowers for pick up"
```

```
}
```

Etapa 4: Criar um bot (AWS CLI)

O bot `OrderFlowersBot` tem uma intenção, a intenção `OrderFlowers` que você criou na etapa anterior. Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST`.

Para criar o bot **`OrderFlowersBot`** (AWS CLI)

1. Crie um arquivo de texto chamado **`OrderFlowersBot.json`**. Copie o seguinte código JSON de OrderFlowersBot.json para o arquivo de texto.
2. No AWS CLI, chame a PutBot operação para criar o bot. O exemplo é formatado para Unix, Linux e macOS. Para Windows, substitua o caractere de continuação Unix de barra invertida (\) no final de cada linha por um circunflexo (^).

   ```
   aws lex-models put-bot \
       --region region \
       --name OrderFlowersBot \
       --cli-input-json file://OrderFlowersBot.json
   ```

   A resposta do servidor é a seguinte. Quando você cria ou atualiza o bot, o campo `status` é definido como `BUILDING`. Isso indica que o bot não está pronto para uso. Para determinar quando o bot está pronto para uso, use a operação GetBot na próxima etapa.

   ```
   {
       "status": "BUILDING",
       "intents": [
           {
               "intentVersion": "$LATEST",
               "intentName": "OrderFlowers"
           }
   ```

```
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "processBehavior": "BUILD",
    "description": "Bot to order flowers on the behalf of a user"
}
```

3. Para determinar se o seu novo bot está pronto para uso, execute o comando a seguir. Repita esse comando até que o campo `status` retorne `READY`. O exemplo é formatado para Unix, Linux e macOS. Para Windows, substitua o caractere de continuação Unix de barra invertida (\) no final de cada linha por um circunflexo (^).

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST"
```

Procure pelo campo `status` na resposta:

```
{
    "status": "READY",

    ...

}
```

Próxima etapa

Etapa 5: Testar um bot (AWS CLI)

OrderFlowersBot.json

O código a seguir fornece os dados JSON necessários para criar o bot do Amazon Lex `OrderFlowers`:

```
{
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
```

```
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Etapa 5: Testar um bot (AWS CLI)

Para testar o bot, você pode usar um arquivo baseado em texto ou um teste com base em voz.

Tópicos

- Testar o bot usando entrada de texto (AWS CLI)
- Testar o bot usando entrada de fala (AWS CLI)

Testar o bot usando entrada de texto (AWS CLI)

Para verificar se o bot funciona corretamente com a entrada de texto, use a operação PostText. Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Service Quotas de runtime.

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST` e substitua o caractere de continuação de barra invertida (\) no final de cada linha por um circunflexo (^).

Para usar texto para testar o bot (AWS CLI)

1. No AWS CLI, inicie uma conversa com o `OrderFlowersBot` bot. O exemplo é formatado para Unix, Linux e macOS. Para Windows, substitua o caractere de continuação Unix de barra invertida (\) no final de cada linha por um circunflexo (^).

   ```
   aws lex-runtime post-text \
       --region region \
       --bot-name OrderFlowersBot \
       --bot-alias "\$LATEST" \
       --user-id UserOne \
   ```

```
    --input-text "i would like to order flowers"
```

O Amazon Lex reconhece a intenção do usuário e inicia uma conversa retornando a seguinte resposta:

```
{
    "slotToElicit": "FlowerType",
    "slots": {
        "PickupDate": null,
        "PickupTime": null,
        "FlowerType": null
    },
    "dialogState": "ElicitSlot",
    "message": "What type of flowers would you like to order?",
    "intentName": "OrderFlowers"
}
```

2. Execute os seguintes comandos para concluir a conversa com o bot.

```
aws lex-runtime post-text \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "roses"
```

```
aws lex-runtime post-text  \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "tuesday"
```

```
aws lex-runtime post-text  \
    --region region \
    --bot-name OrderFlowersBot --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "10:00 a.m."
```

```
aws lex-runtime post-text  \
```

```
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "yes"
```

Após a confirmação do pedido, o Amazon Lex envia uma resposta de atendimento para concluir a conversa:

```
{
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    },
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers"
}
```

Próxima etapa

Testar o bot usando entrada de fala (AWS CLI)

Testar o bot usando entrada de fala (AWS CLI)

Para testar o bot usando arquivos de áudio, use a operação PostContent . Você gera os arquivos de áudio usando as operações do Amazon Polly. text-to-speech

Para executar os comandos neste exercício, você precisa saber em qual região os comandos do Amazon Lex e do Amazon Polly serão executados. Para obter uma lista das regiões para o Amazon Lex, consulte Service Quotas de runtime. Para obter uma lista das regiões da Amazon Polly, consulte AWS Regiões e endpoints na Referência geral da Amazon Web Services.

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST` e substitua o caractere de continuação de barra invertida (\) no final de cada linha por um circunflexo (^).

Para usar uma entrada de fala para testar o bot (AWS CLI)

1. No AWS CLI, crie um arquivo de áudio usando o Amazon Polly. O exemplo é formatado para Unix, Linux e macOS. Para Windows, substitua o caractere de continuação Unix de barra invertida (\) no final de cada linha por um circunflexo (^).

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "i would like to order flowers" \
    --voice-id "Salli" \
    IntentSpeech.mpg
```

2. Para enviar o arquivo de áudio para o Amazon Lex, execute o comando a seguir. O Amazon Lex salva o áudio da resposta no arquivo de saída especificado.

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream IntentSpeech.mpg \
    IntentOutputSpeech.mpg
```

O Amazon Lex responde com uma solicitação para o primeiro slot. Ele salva o áudio da resposta no arquivo de saída especificado.

```
{
    "contentType": "audio/mpeg",
    "slotToElicit": "FlowerType",
    "dialogState": "ElicitSlot",
    "intentName": "OrderFlowers",
    "inputTranscript": "i would like to order some flowers",
    "slots": {
        "PickupDate": null,
        "PickupTime": null,
        "FlowerType": null
    },
    "message": "What type of flowers would you like to order?"
}
```

3. Para pedir rosas, crie o seguinte arquivo de áudio e envie-o para o Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "roses" \
    --voice-id "Salli" \
    FlowerTypeSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream FlowerTypeSpeech.mpg \
    FlowerTypeOutputSpeech.mpg
```

4. Para definir a data de entrega, crie o seguinte arquivo de áudio e envie-o para o Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "tuesday" \
    --voice-id "Salli" \
    DateSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream DateSpeech.mpg \
    DateOutputSpeech.mpg
```

5. Para definir o horário de entrega, crie o seguinte arquivo de áudio e envie-o para o Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
```

```
    --text "10:00 a.m." \
    --voice-id "Salli" \
    TimeSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream TimeSpeech.mpg \
    TimeOutputSpeech.mpg
```

6. Para confirmar a entrega, crie o seguinte arquivo de áudio e envie-o para o Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "yes" \
    --voice-id "Salli" \
    ConfirmSpeech.mpg
```

```
aws lex-runtime post-content \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream ConfirmSpeech.mpg \
    ConfirmOutputSpeech.mpg
```

Depois que você confirmar a entrega, o Amazon Lex envia uma resposta que confirma o atendimento da intenção:

```
{
    "contentType": "text/plain;charset=utf-8",
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers",
    "inputTranscript": "yes",
    "slots": {
        "PickupDate": "2017-05-16",
```

```
        "PickupTime": "10:00",
        "FlowerType": "roses"
    }
}
```

Próxima etapa

Exercício 2: Adicionar uma nova declaração (AWS CLI)

Exercício 2: Adicionar uma nova declaração (AWS CLI)

Para melhorar o modelo de Machine Learning que o Amazon Lex usa para reconhecer solicitações dos usuários, adicione outra amostra de declaração ao bot.

A adição de uma nova declaração é um processo de quatro etapas.

1. Use a operação GetIntent para obter uma intenção do Amazon Lex.
2. Atualize a intenção.
3. Use a operação PutIntent para enviar a intenção atualizada de volta ao Amazon Lex.
4. Use as operações GetBot e PutBot para recriar qualquer bot que use a intenção.

Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

A resposta da operação `GetIntent` contém um campo chamado `checksum` que identifica uma revisão específica da intenção. Você deve fornecer o valor de soma de verificação quando usa a operação PutIntent para atualizar uma intenção. Se não fizer isso, você receberá a seguinte mensagem de erro:

```
An error occurred (PreconditionFailedException) when calling
the PutIntent operation: Intent intent name already exists.
If you are trying to update intent name you must specify the
checksum.
```

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST` e substitua o caractere de continuação de barra invertida (\) no final de cada linha por um circunflexo (^).

Para atualizar a intenção **`OrderFlowers`** (AWS CLI)

1. No AWS CLI, obtenha a intenção do Amazon Lex. O Amazon Lex envia a saída para um arquivo chamado **`OrderFlowers-V2.json.`**

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V2.json
```

2. Abra **`OrderFlowers-V2.json`** em um editor de texto.

   1. Encontre e exclua os campos `createdDate`, `lastUpdatedDatee` `version`.
   2. Adicione o seguinte ao campo `sampleUtterances`:

```
I want to order flowers
```

   3. Salve o arquivo.

3. Envie a intenção atualizada ao Amazon Lex com o seguinte comando:

```
aws lex-models put-intent  \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V2.json
```

   O Amazon Lex envia a seguinte resposta:

```
{
    "confirmationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
```

```
                "content": "Okay, your {FlowerType} will be ready for pickup by
{PickupTime} on {PickupDate}.  Does this sound okay?",
                "contentType": "PlainText"
            }
        ]
    },
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
    "rejectionStatement": {
        "messages": [
            {
                "content": "Okay, I will not place your order.",
                "contentType": "PlainText"
            }
        ]
    },
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "sampleUtterances": [
        "I would like to pick up flowers",
        "I would like to order some flowers",
        "I want to order flowers"
    ],
    "slots": [
        {
            "slotType": "AMAZON.TIME",
            "name": "PickupTime",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "Pick up the {FlowerType} at what time on
{PickupDate}?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 3,
            "description": "The time to pick up the flowers"
        },
        {
            "slotType": "FlowerTypes",
```

```
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers would you like to
 order?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "$LATEST",
            "sampleUtterances": [
                "I would like to order {FlowerType}"
            ],
            "description": "The type of flowers to pick up"
        },
        {
            "slotType": "AMAZON.DATE",
            "name": "PickupDate",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What day do you want the {FlowerType} to be
 picked up?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 2,
            "description": "The date to pick up the flowers"
        }
    ],
    "fulfillmentActivity": {
        "type": "ReturnIntent"
    },
    "description": "Intent to order a bouquet of flowers for pick up"
}
```

Agora que você atualizou a intenção, recrie qualquer bot que a utilize.

Para recriar o bot **`OrderFlowersBot`** (AWS CLI)

1. No AWS CLI, obtenha a definição do `OrderFlowersBot` bot e salve-a em um arquivo com o seguinte comando:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V2.json
```

2. Em um editor de texto, abra **`OrderFlowersBot-V2.json`**. Remova os campos `createdDate`, `lastUpdatedDate`, `status` e `version`.
3. Em um editor de texto, adicione a seguinte linha à definição do bot:

```
"processBehavior": "BUILD",
```

4. No AWS CLI, crie uma nova revisão do bot executando o seguinte comando para:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V2.json
```

A resposta do servidor é:

```
{
    "status": "BUILDING",
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
```

```
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Próxima etapa

Exercício 3: Adicione uma função do Lambda (AWS CLI)

Exercício 3: Adicione uma função do Lambda (AWS CLI)

Adicione uma função do Lambda que valide a entrada do usuário e atenda à intenção do usuário ao bot.

A adição de uma expressão função do Lambda é um processo de cinco etapas.

1. Use a AddPermissionfunção Lambda para ativar a `OrderFlowers` intenção de chamar a operação Lambda Invoke.
2. Use a operação GetIntent para obter a intenção do Amazon Lex.
3. Atualize a intenção para adicionar a função do Lambda.
4. Use a operação PutIntent para enviar a intenção atualizada de volta ao Amazon Lex.
5. Use as operações GetBot e PutBot para recriar qualquer bot que use a intenção.

Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Se você adicionar uma função do Lambda a uma intenção antes de adicionar a permissão `InvokeFunction`, verá a seguinte mensagem de erro:

```
An error occurred (BadRequestException) when calling the
PutIntent operation: Lex is unable to access the Lambda
function Lambda function ARN in the context of intent
intent ARN.  Please check the resource-based policy on
the function.
```

A resposta da operação `GetIntent` contém um campo chamado `checksum` que identifica uma revisão específica da intenção. Quando usa a operação PutIntent para atualizar uma intenção, você deve fornecer o valor de soma de verificação. Se não fizer isso, você receberá a seguinte mensagem de erro:

```
An error occurred (PreconditionFailedException) when calling
the PutIntent operation: Intent intent name already exists.
If you are trying to update intent name you must specify the
checksum.
```

Este exercício usa a função do Lambda do Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console).. Para obter instruções para criar a função do Lambda, consulte Etapa 3: Crie uma função do Lambda (console).

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST`.

Para adicionar uma função do Lambda a uma intenção

1. No AWS CLI, adicione a `InvokeFunction` permissão para a `OrderFlowers` intenção:

```
aws lambda add-permission \
    --region region \
    --function-name OrderFlowersCodeHook \
    --statement-id LexGettingStarted-OrderFlowersBot \
    --action lambda:InvokeFunction \
    --principal lex.amazonaws.com \
    --source-arn "arn:aws:lex:region:account ID:intent:OrderFlowers:*"
    --source-account account ID
```

O Lambda envia a seguinte resposta:

```
{
    "Statement": "{\"Sid\":\"LexGettingStarted-OrderFlowersBot\",
      \"Resource\":\"arn:aws:lambda:region:account ID:function:OrderFlowersCodeHook
\",
      \"Effect\":\"Allow\",
      \"Principal\":{\"Service\":\"lex.amazonaws.com\"},
      \"Action\":[\"lambda:InvokeFunction\"],
      \"Condition\":{\"StringEquals\":
        {\"AWS:SourceAccount\": \"account ID\"},
        {\"AWS:SourceArn\":
          \"arn:aws:lex:region:account ID:intent:OrderFlowers:*\"}}}"
}
```

2. Obtenha a intenção do Amazon Lex. O Amazon Lex envia a saída para um arquivo chamado **`OrderFlowers-V3.json`**.

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V3.json
```

3. Em um editor de texto, abra o **`OrderFlowers-V3.json`**.

   1. Encontre e exclua os campos `createdDate`, `lastUpdatedDatee` `version`.
   2. Atualize o campo `fulfillmentActivity`:

```
    "fulfillmentActivity": {
        "type": "CodeHook",
        "codeHook": {
```

```
            "uri": "arn:aws:lambda:region:account
ID:function:OrderFlowersCodeHook",
            "messageVersion": "1.0"
        }
    }
```

3. Salve o arquivo.

4. Em AWS CLI, envie a intenção atualizada para o Amazon Lex:

```
aws lex-models put-intent \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V3.json
```

Agora que você atualizou a intenção, recrie o bot.

Para recriar o bot **`OrderFlowersBot`**

1. No AWS CLI, obtenha a definição do `OrderFlowersBot` bot e salve-a em um arquivo:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V3.json
```

2. Em um editor de texto, abra **`OrderFlowersBot-V3.json`**. Remova os campos `createdDate`, `lastUpdatedDate`, `status` e `version`.

3. No editor de texto, adicione a seguinte linha à definição do bot:

```
"processBehavior": "BUILD",
```

4. No AWS CLI, crie uma nova revisão do bot:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V3.json
```

A resposta do servidor é:

```
{
    "status": "READY",
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "clarificationPrompt": {
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Próxima etapa

Exercício 4: Publicar uma versão (AWS CLI)

Exercício 4: Publicar uma versão (AWS CLI)

Agora, crie uma versão do bot que você criou no Exercício 1. Uma versão é um snapshot do bot. Depois que criar uma versão, você não poderá alterá-la. A única versão de um bot que você pode atualizar é a versão `$LATEST`. Para obter mais informações sobre versões, consulte Versionamento e aliases.

Antes de poder publicar uma versão de um bot, você deve publicar a intenção que ele usa. Da mesma forma, você deve publicar os tipos de slot a que essas intenções se referem. No geral, para publicar uma versão de um bot, você faz o seguinte:

1. Publica uma versão de um tipo de slot com a operação CreateSlotTypeVersion.
2. Publica uma versão de uma intenção com a operação CreateIntentVersion.
3. Publica uma versão de um bot com a operação CreateBotVersion.

Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Tópicos

- Etapa 1: Publicar o tipo de slot (AWS CLI)
- Etapa 2: publicar a intenção (AWS CLI)
- Etapa 3: Publicar o bot (AWS CLI)

Etapa 1: Publicar o tipo de slot (AWS CLI)

Antes de poder publicar uma versão de qualquer intenção que usa um tipo de slot, você deve publicar uma versão desse tipo de slot. Neste caso, você publica o tipo de slot `FlowerTypes`.

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST` e substitua o caractere de continuação de barra invertida (\) no final de cada linha por um circunflexo (^).

Para publicar o tipo de slot (AWS CLI)

1. No AWS CLI, obtenha a versão mais recente do tipo de slot:

```
aws lex-models get-slot-type \
    --region region \
    --name FlowerTypes \
    --slot-type-version "\$LATEST"
```

A resposta do Amazon Lex vem em seguida. Registre a soma de verificação para a revisão atual da versão `$LATEST`.

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "checksum": "checksum",
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

2. Publique uma nova versão do tipo de slot. Use a soma de verificação que você registrou na etapa anterior.

```
aws lex-models create-slot-type-version \
    --region region \
    --name FlowerTypes \
    --checksum "checksum"
```

A resposta do Amazon Lex vem em seguida. Registre o número da versão para a próxima etapa.

```
{
    "version": "1",
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Próxima etapa

Etapa 2: publicar a intenção (AWS CLI)

Etapa 2: publicar a intenção (AWS CLI)

Antes de poder publicar uma intenção, você tem que publicar todos os tipos de slot mencionados pela intenção. Os tipos de slot devem ser versões numeradas, não a versão `$LATEST`.

Primeiro, atualize a intenção `OrderFlowers` para usar a versão do tipo de slot `FlowerTypes` que você publicou na etapa anterior. Depois, publique uma nova versão da intenção `OrderFlowers`.

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST` e substitua o caractere de continuação de barra invertida (\) no final de cada linha por um circunflexo (^).

Para publicar uma versão de uma intenção (AWS CLI)

1. No AWS CLI, obtenha a `$LATEST` versão da `OrderFlowers` intenção e salve-a em um arquivo:

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4.json
```

2. Em um editor de texto, abra o arquivo **`OrderFlowers_V4.json`**. Exclua os campos `createdDate`, `lastUpdatedDate` e `version`. Encontre o tipo de slot `FlowerTypes` e altere a versão para o número da versão que você registrou na etapa anterior. O seguinte fragmento do arquivo **`OrderFlowers_V4.json`** mostra a localização da alteração:

```
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "version",
            "sampleUtterances": []
        },
```

3. No AWS CLI, salve a revisão da intenção:

```
aws lex-models put-intent \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers_V4.json
```

4. Obtenha a soma de verificação da última revisão da intenção:

```
aws lex-models get-intent \
    --region region \
```

```
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4a.json
```

O seguinte fragmento de resposta mostra a soma de verificação da intenção. Registre isso para a próxima etapa.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
```

5. Publique uma nova versão da intenção:

```
aws lex-models create-intent-version \
    --region region \
    --name OrderFlowers \
    --checksum "checksum"
```

O seguinte fragmento de resposta mostra a nova versão da intenção. Registre o número da versão para a próxima etapa.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "version",
```

Próxima etapa

Etapa 3: Publicar o bot (AWS CLI)

Etapa 3: Publicar o bot (AWS CLI)

Depois que você tiver publicado todos os tipos de slot e intenções que são usados pelo seu bot, você pode publicar o bot.

Atualize o bot `OrderFlowersBot` para usar a intenção `OrderFlowers` que você atualizou na etapa anterior. Depois, publique uma nova versão do bot `OrderFlowersBot`.

> **Note**
>
> O AWS CLI exemplo a seguir está formatado para Unix, Linux e macOS. Para Windows, altere `"\$LATEST"` para `$LATEST` e substitua o caractere de continuação de barra invertida (\) no final de cada linha por um circunflexo (^).

Para publicar uma versão de um bot (AWS CLI)

1. No AWS CLI, obtenha a `$LATEST` versão do `OrderFlowersBot` bot e salve-a em um arquivo:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot_V4.json
```

2. Em um editor de texto, abra o arquivo **`OrderFlowersBot_V4.json`**. Exclua os campos `createdDate`, `lastUpdatedDate`, `status` e `version`. Encontre a intenção `OrderFlowers` e altere a versão para o número da versão que você registrou na etapa anterior. O seguinte fragmento de **`OrderFlowersBot_V4.json`** mostra a localização da alteração.

```
    "intents": [
        {
            "intentVersion": "version",
            "intentName": "OrderFlowers"
        }
```

3. Em AWS CLI, salve a nova revisão do bot. Anote o número de versão retornado pela chamada a `put-bot`.

```
aws lex-models put-bot \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot_V4.json
```

4. Obtenha a soma de verificação da última revisão do bot. Use o número de versão retornado na etapa 3.

```
aws lex-models get-bot \
    --region region \
    --version-or-alias version \
```

```
    --name OrderFlowersBot > OrderFlowersBot_V4a.json
```

O seguinte fragmento de resposta mostra a soma de verificação do bot. Registre isso para a próxima etapa.

```
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
```

5. Publique uma nova versão do bot:

```
aws lex-models create-bot-version \
    --region region \
    --name OrderFlowersBot \
    --checksum "checksum"
```

O seguinte fragmento de resposta mostra a nova versão do bot.

```
    "checksum": "checksum",
    "abortStatement": {
        ...
    },
    "version": "1",
    "lastUpdatedDate": timestamp,
```

Próxima etapa

Exercício 5: Criar um alias (AWS CLI)

Exercício 5: Criar um alias (AWS CLI)

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. Para obter mais informações, consulte Versionamento e aliases. Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Para criar um alias (AWS CLI)

1. No AWS CLI, obtenha a versão do `OrderFlowersBot` bot que você criou em[Exercício 4: Publicar uma versão (AWS CLI)](#).

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias version > OrderFlowersBot_V5.json
```

2. Em um editor de texto, abra **`OrderFlowersBot_v5.json`**. Encontre e registre o número da versão.
3. No AWS CLI, crie o alias do bot:

```
aws lex-models put-bot-alias  \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot \
    --bot-version version
```

A resposta do servidor é a seguinte:

```
{
    "name": "PROD",
    "createdDate": timestamp,
    "checksum": "checksum",
    "lastUpdatedDate": timestamp,
    "botName": "OrderFlowersBot",
    "botVersion": "1"
}}
```

Próxima etapa

[Exercício 6: Limpar (AWS CLI)](#)

Exercício 6: Limpar (AWS CLI)

Exclua os recursos que você criou e limpe sua conta.

Você só pode excluir os recursos que não estão em uso. Em geral, você deve excluir recursos na seguinte ordem.

1. Exclua aliases para liberar recursos de bot.
2. Exclua bots para liberar recursos de intenção.
3. Exclua intenções para liberar recursos de tipo de slot.
4. Exclua tipos de slot.

Para executar os comandos neste exercício, você precisa saber em que região os comandos serão executados. Para obter uma lista de regiões, consulte Cotas de criação de modelos .

Para limpar sua conta (AWS CLI)

1. Na linha de AWS CLI comando, exclua o alias:

```
aws lex-models delete-bot-alias \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot
```

2. Na linha de AWS CLI comando, exclua o bot:

```
aws lex-models delete-bot \
    --region region \
    --name OrderFlowersBot
```

3. Na linha de AWS CLI comando, exclua a intenção:

```
aws lex-models delete-intent \
    --region region \
    --name OrderFlowers
```

4. Na linha de AWS CLI comando, exclua o tipo de slot:

```
aws lex-models delete-slot-type \
    --region region \
    --name FlowerTypes
```

Você removeu todos os recursos que criou e limpou sua conta.

Versionamento e aliases

O Amazon Lex oferece suporte a versões de publicação de bots, intenções e tipos de slot para que você possa controlar a implementação usada por seus aplicativos cliente. Uma 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.

Os bots do Amazon Lex também oferecem suporte a 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 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.

Tópicos

- Versionamento
- Aliases

Versionamento

Quando você versiona um recurso do Amazon Lex, você cria um snapshot do recurso para que possa usar o recurso da forma como existia quando a versão foi criada. Após criar uma versão, ela permanecerá a mesma enquanto você continuar trabalhando no seu aplicativo.

A versão $LATEST

Quando você cria um bot, intenção ou tipo de slot do Amazon Lex, há apenas uma versão, a versão `$LATEST`.

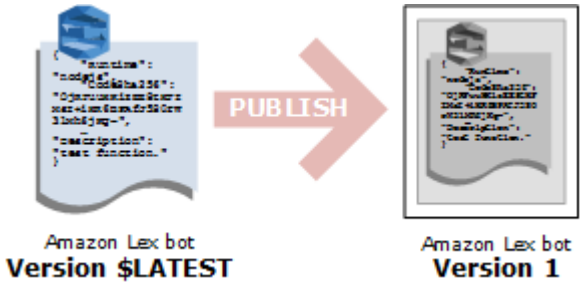
`$LATEST` é a cópia de trabalho do seu recurso. Você pode atualizar apenas a versão `$LATEST`, e até você publicar sua primeira versão, `$LATEST` será a única versão do recurso.

Somente a versão `$LATEST` de um recurso pode usar a versão `$LATEST` do outro recurso. Por exemplo, a versão `$LATEST` de um bot pode usar a versão `$LATEST` de uma intenção, e a versão `$LATEST` de uma intenção pode usar a versão `$LATEST` de um tipo de slot.

A versão `$LATEST` do bot deve ser usada somente para testes manuais. O Amazon Lex limita o número de solicitações de runtime que você pode fazer para a versão `$LATEST` do bot.

Publicação de uma versão de recurso do Amazon Lex

Quando você publica um recurso, o Amazon Lex faz uma cópia da versão `$LATEST` e salva como uma versão numerada. A versão publicada não pode ser alterada.



Crie e publique as versões usando o console do Amazon Lex ou a operação CreateBotVersion. Para obter um exemplo, consulte Exercício 3: publique uma versão e crie um alias.

Quando você modifica a versão `$LATEST` de um recurso, pode publicar a nova versão para disponibilizar as alterações para os aplicativos de seus clientes. Toda vez que você publica uma versão, o Amazon Lex copia a versão `$LATEST` para criar a nova versão e incrementa 1 no número da versão. Os números de versão nunca são reutilizados. Por exemplo, se você remover a versão 10 numerada de um recurso e, em seguida, recriá-la, o próximo número de versão atribuído pelo Amazon Lex será a versão 11.

Para publicar um bot, você deve apontá-lo para uma versão numerada de qualquer intenção que ele use. Se você tentar publicar uma nova versão de um bot que usa a versão $LATEST versão de uma intenção, O Amazon Lex retorna uma exceção HTTP 400: solicitação inválida. Para publicar uma versão numerada da intenção, você deve apontar a intenção de uma versão numerada de qualquer tipo de slot que ele use. Caso contrário, você receberá uma exceção HTTP 400: solicitação inválida.

> **Note**
>
> O Amazon Lex publica uma nova versão somente se a última versão publicada for diferente da versão `$LATEST`. Se você tentar publicar a versão `$LATEST` sem modificá-la, o Amazon Lex não criará nem publicará uma nova versão.

Atualização de um recurso do Amazon Lex

Você pode atualizar apenas a versão `$LATEST` de um bot, uma intenção ou um tipo de slot do Amazon Lex. As versões publicadas não podem ser alteradas. Você pode publicar uma nova versão a qualquer momento após atualizar um recurso no console ou com as operações CreateBotVersion, CreateIntentVersion ou CreateSlotTypeVersion.

Exclusão de um recurso ou uma versão do Amazon Lex.

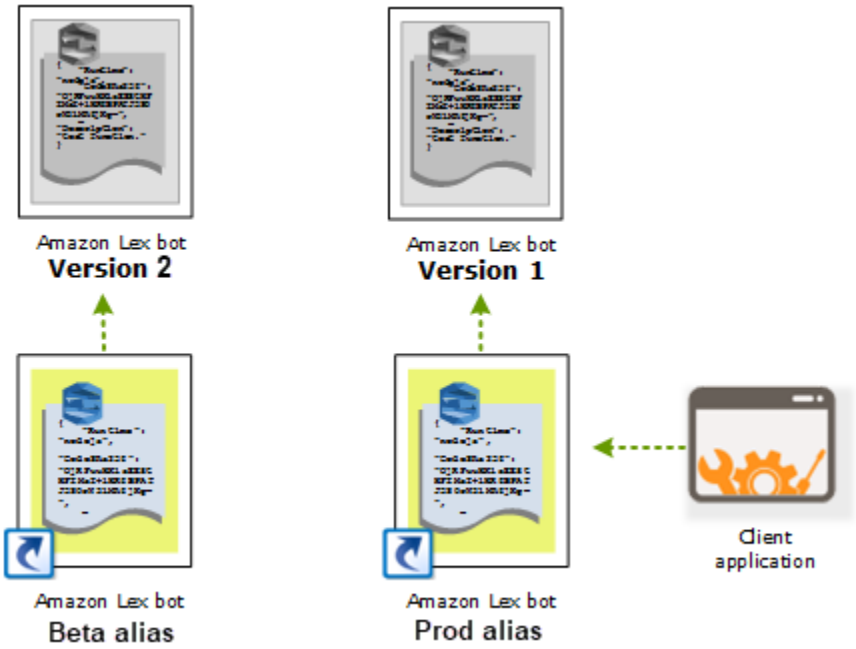
O Amazon Lex oferece suporte à exclusão de um recurso ou versão usando o console ou uma das operações de API:

- DeleteBot
- DeleteBotVersion
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion

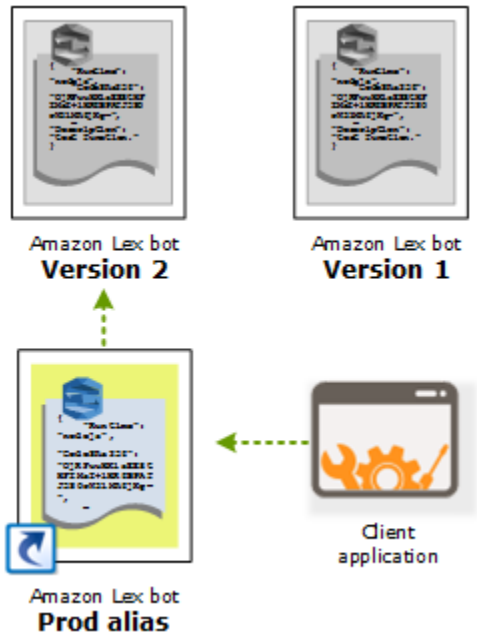
Aliases

Um alias é um indicador para uma versão específica de um bot do Amazon Lex. 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.

O exemplo a seguir mostra duas versões de um bot do Amazon Lex, 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 PutBot. 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.



> **Note**
>
> Embora você possa testar a versão $LATEST de um bot no console, recomendamos que, ao integrar um bot ao aplicativo do cliente, primeiro você publique uma versão e crie um alias que aponta para essa versão. Use o alias na aplicação de seus clientes pelos motivos explicados nesta seção. Quando você atualiza um alias, o Amazon Lex aguardará até que o tempo limite de sessão de todas as sessões atuais expire para começar a usar a nova versão. Para obter mais informações sobre tempo limite da sessão, consulte the section called “Definição do tempo limite da sessão”

Uso de funções do Lambda

Você pode criar AWS Lambda funções para usar como ganchos de código para seu bot Amazon Lex. Você pode identificar funções do Lambda para executar a inicialização e a validação, o atendimento ou ambos na configuração da intenção.

Recomendamos usar uma função do Lambda como um hook de código para seu bot. Sem uma função do Lambda, seu bot retorna as informações da intenção ao aplicativo cliente para atendimento.

Tópicos

- Evento de entrada de função do Lambda e formato de resposta
- Amazon Lex e esquemas AWS Lambda

Evento de entrada de função do Lambda e formato de resposta

Esta seção descreve a estrutura dos dados de eventos que o Amazon Lex fornece para uma função do Lambda. Use essas informações para analisar a entrada em seu código do Lambda. Ela também explica o formato da resposta que o Amazon Lex espera que sua função do Lambda retorne.

Tópicos

- Formato de eventos de entrada
- Formato de resposta

Formato de eventos de entrada

O seguinte é o formato geral de um evento do Amazon Lex que é passado para uma função do Lambda. Use essas informações ao criar sua função do Lambda.

> **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.

```
{
  "currentIntent": {
    "name": "intent-name",
    "nluIntentConfidenceScore": score,
    "slots": {
      "slot name": "value",
      "slot name": "value"
    },
    "slotDetails": {
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      },
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      }
    },
    "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)"
  },
  "alternativeIntents": [
    {
      "name": "intent-name",
      "nluIntentConfidenceScore": score,
      "slots": {
        "slot name": "value",
        "slot name": "value"
      },
      "slotDetails": {
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        },
```

```
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        }
      },
      "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)"
    }
  ],
  "bot": {
    "name": "bot name",
    "alias": "bot alias",
    "version": "bot version"
  },
  "userId": "User ID specified in the POST request to Amazon Lex.",
  "inputTranscript": "Text used to process the request",
  "invocationSource": "FulfillmentCodeHook or DialogCodeHook",
  "outputDialogMode": "Text or Voice, based on ContentType request header in runtime
API request",
  "messageVersion": "1.0",
  "sessionAttributes": {
     "key": "value",
     "key": "value"
  },
  "requestAttributes": {
     "key": "value",
     "key": "value"
  },
  "recentIntentSummaryView": [
    {
        "intentName": "Name",
        "checkpointLabel": Label,
        "slots": {
          "slot name": "value",
          "slot name": "value"
        },
        "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
Close",
        "fulfillmentState": "Fulfilled or Failed",
```

```
        "slotToElicit": "Next slot to elicit"
    }
  ],
   "sentimentResponse": {
      "sentimentLabel": "sentiment",
      "sentimentScore": "score"
   },
   "kendraResponse": {
       Complete query response from Amazon Kendra
   },
   "activeContexts": [
        {
            "timeToLive": {
                "timeToLiveInSeconds": seconds,
                "turnsToLive": turns
            },
            "name": "name",
            "parameters": {
                "key name": "value"
            }
        }
    ]
}
```

Observe as seguintes informações adicionais sobre os campos de evento:

- currentIntent – Fornece os campos `name`, `slots`, `slotDetails` e `confirmationStatus` da intenção.

  `nluIntentConfidenceScore` é a confiança que o Amazon Lex tem de que a intenção atual é a que melhor corresponde à intenção atual do usuário.

  `slots` é um mapa de nomes de slots, configurado para a intenção, com valores de slot que o Amazon Lex reconheceu na conversa do usuário. Um valor de slot permanece nulo até o usuário fornecer um valor.

O valor de slot no evento de entrada pode não corresponder com um dos valores configurados para o slot. Por exemplo, se o usuário responde ao prompt "Você gostaria de um carro de qual cor?" com "pizza", o Amazon Lex retornará "pizza" como valor de slot. Sua função deve validar os valores de forma que façam sentido no contexto.

`slotDetails` fornece informações adicionais sobre um valor de slot. O matriz `resolutions` contém uma lista de valores adicionais reconhecidos para o slot. Cada slot pode ter no máximo cinco valores.

O campo `originalValue` contém o valor que foi inserido pelo usuário para o slot. Quando o tipo de slot é configurado para retornar o primeiro valor de resolução como valor de slot, o `originalValue` pode ser diferente do valor no campo `slots`.

`confirmationStatus` fornece ao usuário resposta a um prompt de confirmação, se houver. Por exemplo, se o Amazon Lex perguntar "Você quer pedir uma pizza grande de queijo?, dependendo da resposta do usuário, o valor desse campo poderá ser `Confirmed` ou `Denied`. Caso contrário, o valor desse campo é `None`.

Se o usuário confirmar a intenção, o Amazon Lex define esse campo como `Confirmed`. Se o usuário negar a intenção, o Amazon Lex define esse valor como `Denied`.

Na resposta de confirmação, a declaração do usuário pode fornecer atualizações de slot. Por exemplo, o usuário pode dizer "sim, mude o tamanho para médio". Nesse caso, o evento de Lambda subsequente tem o valor de slot atualizado, `PizzaSize` definido como `medium`. O Amazon Lex define o `confirmationStatus` como `None`, porque o usuário modificou alguns dados de slot, exigindo que a função do Lambda; realize a validação dos dados do usuário.

- alternativeIntents: se você ativar pontuações de confiança, o Amazon Lex retornará até quatro intenções alternativas. Cada intenção inclui uma pontuação que indica o nível de confiança que o Amazon Lex tem de que a intenção é a intenção correta com base na declaração do usuário.

  O conteúdo das intenções alternativas é igual ao conteúdo do `currentIntent` campo. Para obter mais informações, consulte Usar pontuações de confiança.

- bot – Informações sobre o bot que processou a solicitação.
  - `name` – O nome do bot que processou a solicitação.
  - `alias` – O alias da versão do bot que processou a solicitação.
  - `version` – A versão do bot que processou a solicitação.

- userId: esse valor é fornecido pelo aplicativo cliente. O Amazon Lex transmite para a função do Lambda.

- inputTranscript – O texto usado para processar a solicitação.

  Se a entrada tiver sido um texto, o campo `inputTranscript` conterá o texto que foi inserido pelo usuário.

  Se a entrada tiver sido um fluxo de áudio, o campo `inputTranscript` conterá o texto extraído do fluxo de áudio. Esse é o texto que é processado para reconhecer as intenções e os valores de slot.

- invocationSource: para indicar por que o Amazon Lex está invocando a função do Lambda, isso é definido como um dos seguintes valores:
  - `DialogCodeHook`: o Amazon Lex define esse valor para direcionar a função do Lambda; para inicializar a função e validar a entrada de dados do usuário.

Quando a intenção é configurada para invocar uma função do Lambda como um hook de código de inicialização e validação, o Amazon Lex invoca a função do Lambda especificada em cada entrada do usuário (declaração) após o Amazon Lex entender a intenção.

> **Note**
>
> Se a intenção não estiver clara, o Amazon Lex não conseguirá invocar a função do Lambda.

- `FulfillmentCodeHook`: o Amazon Lex define esse valor para direcionar a função do Lambda para atender uma intenção.

  Se a intenção estiver configurada para invocar uma função do Lambda como um hook de código de atendimento, o Amazon Lex definirá o `invocationSource` como esse valor somente depois de ter todos os dados de slot para atender à intenção.

Em sua configuração de intenção, você pode ter duas funções do Lambda separadas para inicializar e validar os dados do usuário e para cumprir a intenção. Você também pode usar uma função do Lambda para fazer ambos. Neste caso, sua função do Lambda pode usar o valor `invocationSource` para seguir o caminho de código correto.

- outputDialogMode— Para cada entrada do usuário, o cliente envia a solicitação para o Amazon Lex usando uma das operações de API de tempo de execução, PostContent ouPostText. O Amazon Lex usa os parâmetros de solicitação para determinar se a resposta ao cliente é por texto ou voz e define esse campo de acordo.

  A função do Lambda pode usar essas informações para gerar uma mensagem apropriada. Por exemplo, se o cliente espera uma resposta de voz, a função do Lambda pode retornar Linguagem de marcação de síntese de fala (SSML) em vez de texto.

- messageVersion: a versão da mensagem que identifica o formato dos dados de evento indo para 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 a versão de mensagem 1.0 é compatível. Portanto, o console assume o valor padrão de 1.0 e não mostra a versão da mensagem.

- sessionAttributes: atributos de sessão específicos do aplicativo que o cliente envia na solicitação. Se você quiser que o Amazon Lex os inclua na resposta ao cliente, sua função do Lambda deverá enviá-los de volta ao Amazon Lex na resposta. Para ter mais informações, consulte Definição dos atributos da sessão

- requestAttributes: atributos de sessão específicos da 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 ter mais informações, consulte Definição de atributos de solicitação

- recentIntentSummaryExibir — Informações sobre o estado de uma intenção. É possível ver informações sobre as últimas três intenções usadas. É possível usar essas informações para definir valores na intenção ou para retornar a uma intenção anterior. Para obter mais informações, consulte Gerenciamento de sessões com a API do Amazon Lex.

- sentimentResponse: o resultado de uma análise de sentimento do Amazon Comprehend da última declaração. É possível usar essas informações para gerenciar o fluxo de conversa do bot dependendo do sentimento expresso pelo usuário. Para obter mais informações, consulte Análise de sentimento.

- kendraResponse: o resultado de uma consulta a um índice do Amazon Kendra. Presente apenas na entrada para um hook de código de atendimento e somente quando a intenção estende a

intenção interna `AMAZON.KendraSearchIntent`. O campo contém toda a resposta da pesquisa do Amazon Kendra. Para obter mais informações, consulte AMAZON.KendraSearchIntent.

- activeContexts: um ou mais contextos ativos durante esse turno de uma conversa com o usuário.
  - timeToLive— O período de tempo ou o número de turnos na conversa com o usuário em que o contexto permanece ativo.
  - name: o nome do contexto.
  - parâmetros uma lista de pares de chave/valor que contém o nome e o valor de slots da intenção que ativou o contexto.

  Para obter mais informações, consulte Definir o contexto da intenção.

Formato de resposta

O Amazon Lex espera uma resposta de uma função do Lambda no formato a seguir:

```
{
  "sessionAttributes": {
    "key1": "value1",
    "key2": "value2"
    ...
  },
  "recentIntentSummaryView": [
    {
       "intentName": "Name",
       "checkpointLabel": "Label",
       "slots": {
         "slot name": "value",
         "slot name": "value"
        },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
 Close",
        "fulfillmentState": "Fulfilled or Failed",
        "slotToElicit": "Next slot to elicit"
    }
  ],
  "activeContexts": [
```

```
      {
        "timeToLive": {
           "timeToLiveInSeconds": seconds,
           "turnsToLive": turns
       },
       "name": "name",
       "parameters": {
         "key name": "value"
       }
     }
   ],
   "dialogAction": {
     "type": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or Close",
     Full structure based on the type field. See below for details.
   }
}
```

A resposta consiste em quatro campos. Os campos `sessionAttributes`, `recentIntentSummaryView` e `activeContexts` são opcionais, mas o campo `dialogAction` é obrigatório. O conteúdo do campo `dialogAction` depende do valor do campo `type`. Para obter detalhes, consulte dialogAction.

sessionAttributes

Opcional. Se você incluir o campo `sessionAttributes`, ele pode ficar vazio. Se a função do Lambda não retornar os atributos da sessão, o último `sessionAttributes` conhecido passado pela API ou pela função do Lambda permanecerá. Para obter mais informações, consulte as operações PostContent e PostText.

```
   "sessionAttributes": {
      "key1": "value1",
      "key2": "value2"
   }
```

recentIntentSummaryVisualizar

Opcional. Se incluído, define valores para uma ou mais intenções recentes. É possível incluir informações para até três intenções. Por exemplo, é possível definir valores para intenções anteriores com base nas informações coletadas pela intenção atual. As informações no resumo devem ser válidas para a intenção. Por exemplo, o nome da intenção deve ser uma intenção no

bot. Se você incluir um valor de slot na visão resumida, o slot deverá existir na intenção. Se você não incluir o `recentIntentSummaryView` em sua resposta, todos os valores para as intenções recentes permanecerão inalterados. Para obter mais informações, consulte a operação PutSession ou o tipo de dados IntentSummary.

```
"recentIntentSummaryView": [
    {
       "intentName": "Name",
       "checkpointLabel": "Label",
       "slots": {
         "slot name": "value",
         "slot name": "value"
        },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
 Close",
        "fulfillmentState": "Fulfilled or Failed",
        "slotToElicit": "Next slot to elicit"
    }
  ]
```

activeContexts

Opcional. Se incluído, define o valor para um ou mais contextos. Por exemplo, você pode incluir um contexto para tornar uma ou mais intenções que tenham esse contexto como uma entrada elegíveis para reconhecimento no próximo turno da conversa.

Todos os contextos ativos que não estão incluídos na resposta têm seus time-to-live valores diminuídos e ainda podem estar ativos na próxima solicitação.

Se você especificar um time-to-live de 0 para um contexto que foi incluído no evento de entrada, ele ficará inativo na próxima solicitação.

Para obter mais informações, consulte Definir o contexto da intenção.

dialogAction

Obrigatório. O campo `dialogAction` direciona o Amazon Lex para a próxima ação e descreve o que esperar do usuário depois que o Amazon Lex retornar uma resposta ao cliente.

O campo `type` indica a próxima ação. Ele também determina quais outros campos a função do Lambda precisa fornecer como parte do valor `dialogAction`.

- `Close`: informa ao Amazon Lex para não esperar uma resposta do usuário. Por exemplo, "Seu pedido de pizza foi feito" não requer uma resposta.

  O campo `fulfillmentState` é obrigatório. O Amazon Lex usa esse valor para definir o campo `dialogState` em PostContent ou PostText na resposta ao aplicativo cliente. Os campos `message` e `responseCard` são opcionais. Se você não especificar uma mensagem, o Amazon Lex usará a mensagem de despedida ou a mensagem de acompanhamento configurada para a intenção.

```
"dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled or Failed",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, Thanks, your pizza has
 been ordered."
    },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
 card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
       ]
     }
  }
```

- `ConfirmIntent`: informa ao Amazon Lex que o usuário deve fornecer uma resposta Sim ou Não para confirmar ou negar a intenção atual.

Você deve incluir os campos `intentName` e `slots`. O campo `slots` deve conter uma entrada para cada um dos slots preenchidos para a intenção especificada. Não é necessário incluir uma entrada no campo `slots` para slots que não estão preenchidos. Você deve incluir o campo `message` se o campo `confirmationPrompt` de intenção for nulo. O conteúdo do campo `message` retornado pela função do Lambda tem precedência sobre o `confirmationPrompt` especificado na intenção. O campo `responseCard` é opcional.

```
"dialogAction": {
    "type": "ConfirmIntent",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, Are you sure you want a
 large pizza?"
    },
   "intentName": "intent-name",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
 card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
```

```
        ]
      }
  }
```

- `Delegate`: direciona o Amazon Lex para escolher a próxima ação de acordo com a configuração do bot. Se a resposta não incluir nenhum atributo de sessão, o Amazon Lex manterá os atributos existentes. Se você deseja que o valor de um slot seja nulo, não é necessário incluir o campo do slot na solicitação. Uma exceção `DependencyFailedException` será gerada se a função de cumprimento retornar a ação de diálogo `Delegate` sem remover nenhum slot.

  Os campos `kendraQueryRequestPayload` e `kendraQueryFilterString` são opcionais e usados somente quando a intenção é derivada da intenção interna `AMAZON.KendraSearchIntent`. Para obter mais informações, consulte AMAZON.KendraSearchIntent.

```
  "dialogAction": {
   "type": "Delegate",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "kendraQueryRequestPayload": "Amazon Kendra query",
   "kendraQueryFilterString": "Amazon Kendra attribute filters"
  }
```

- `ElicitIntent`: informa ao Amazon Lex que o usuário deve responder com uma declaração que inclua uma intenção. Por exemplo, "Eu quero uma pizza grande", que indica a `OrderPizzaIntent`. Por outro lado, a declaração "grande" não é suficiente para que o Amazon Lex infira a intenção do usuário.

  Os campos `message` e `responseCard` são opcionais. Se você não fornecer uma mensagem, o Amazon Lex usará um dos prompts de esclarecimento do bot. Se não houver um prompt de esclarecimento definido, o Amazon Lex retornará uma exceção 400 Solicitação inválida.

```
{
  "dialogAction": {
    "type": "ElicitIntent",
```

```
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, What can I help you
with?"
    },
    "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
       ]
    }
}
```

- `ElicitSlot`: informa ao Amazon Lex que o usuário deve fornecer um valor de slot na resposta.

Os campos `intentName`, `slotToElicit` e `slots` são obrigatórios. Os campos `message` e `responseCard` são opcionais. Se você não especificar uma mensagem, o Amazon Lex usará uma das solicitações de seleção de valor de slot configuradas para o slot.

```
  "dialogAction": {
    "type": "ElicitSlot",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, What size pizza would
you like?"
    },
   "intentName": "intent-name",
```

```
  "slots": {
     "slot-name": "value",
     "slot-name": "value",
     "slot-name": "value"
  },
  "slotToElicit" : "slot-name",
  "responseCard": {
     "version": integer-value,
     "contentType": "application/vnd.amazonaws.card.generic",
     "genericAttachments": [
         {
            "title":"card-title",
            "subTitle":"card-sub-title",
            "imageUrl":"URL of the image to be shown",
            "attachmentLinkUrl":"URL of the attachment to be associated with the
card",
            "buttons":[
                {
                   "text":"button-text",
                   "value":"Value sent to server on button click"
                }
             ]
          }
      ]
    }
 }
```

Amazon Lex e esquemas AWS Lambda

O console do Amazon Lex fornece bots de exemplo (chamados de esquemas de bot) pré-configurados para que você possa criar e testar rapidamente um bot no console. Para cada um desses esquemas de bot, também são fornecidos esquemas da função do Lambda. Esses esquemas fornecem um código de exemplo que funciona com os bots correspondentes. Você pode usar esses esquemas para criar rapidamente um bot configurado com uma função Lambda como gancho de código e testar end-to-end a configuração sem precisar escrever código.

Você pode usar os seguintes esquemas de bots do Amazon Lex e os esquemas de AWS Lambda função correspondentes como ganchos de código para bots:

- Esquema do Amazon Lex: `OrderFlowers`
  - AWS Lambda planta — `lex-order-flowers-python`

- Esquema do Amazon Lex: `ScheduleAppointment`
  - AWS Lambda planta — `lex-make-appointment-python`
- Esquema do Amazon Lex: `BookTrip`
  - AWS Lambda planta — `lex-book-trip-python`

Para criar um bot usando um esquema e configurá-lo para usar uma função do Lambda como um hook de código, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console).. Para obter um exemplo do uso de outros esquemas, consulte Exemplos adicionais: criação de bots do Amazon Lex.

Atualização de um esquema para uma localidade específica

Se estiver usando um esquema em uma localidade diferente do inglês (EUA) (en-US), você precisará atualizar o nome de todas as intenções para incluir a localidade. Por exemplo, se estiver usando o esquema `OrderFlowers`, você precisará fazer o seguinte.

- Encontre a função `dispatch` perto do final do código da função do Lambda.
- Na função `dispatch`, atualize o nome da intenção para incluir a localidade que você está usando. Por exemplo, se você estiver usando a localidade inglesa (australiana) (en-AU), altere a linha:

  ```
  if intent_name == 'OrderFlowers':
  ```

  para

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

Outros esquemas usam outros nomes de intenção. Eles devem ser atualizados conforme descrito acima antes de serem usados.

Implantação de bots do Amazon Lex

Esta seção fornece exemplos de implantação de bots do Amazon Lex em várias plataformas de sistema de mensagens e em aplicativos para dispositivos móveis.

Tópicos

- Implantação de um bot do Amazon Lex em uma plataforma de sistema de mensagens
- Implantação de um bot do Amazon Lex em aplicativos móveis

Implantação de um bot do Amazon Lex em uma plataforma de sistema de mensagens

Esta seção explica como implantar bots do Amazon Lex nas plataformas de sistema de mensagens Facebook, Slack e Twilio.

> **Note**
>
> Ao armazenar as configurações do Facebook, Slack ou Twilio, o Amazon Lex usa uma CMK do AWS Key Management Service para criptografar as informações. Na primeira vez que você cria um canal para uma das seguintes plataformas de sistema de mensagens, o Amazon Lex cria uma CMK padrão (`aws/lex`). Como alternativa, você pode criar sua própria chave gerenciada pelo cliente com AWS KMS. Isso lhe dá mais flexibilidade, incluindo a capacidade de criar, girar e desabilitar as chaves. Você também pode definir controles de acesso e auditar as chaves de criptografia usadas para proteger seus dados. 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, informações específicas à plataforma são incluídas como um atributo de solicitação para a função do Lambda. Use esses atributos para personalizar a forma como o seu bot se comporta. Para obter mais informações, consulte Definição de atributos de solicitação.

Todos os atributos levam o namespace `x-amz-lex:` como prefixo. Por exemplo, o atributo `user-id` é chamado `x-amz-lex:user-id`. Existem atributos comuns que são enviados por todas as plataformas de mensagens, além de outros que são específicos de uma determinada plataforma. A

tabela a seguir lista os atributos de solicitação que as plataformas de sistema de mensagens enviam à função do Lambda do bot.

Atributos de solicitação comuns

Atributo	Descrição
`channel-id`	O identificador do endpoint do canal do Amazon Lex.
`channel-name`	O nome do canal do Amazon Lex.
`channel-type`	Um dos seguintes valores: • `Facebook` • `Kik` • `Slack` • `Twilio-SMS`
`webhook-endpoint-url`	O endpoint do Amazon Lex para o canal.

Atributos de solicitação do Facebook

Atributo	Descrição
`user-id`	O identificador do Facebook do remetente. Veja https://developers.facebook.com/docs/messenger- platform/webhook-reference/message -received.
`facebook-page-id`	O identificador da página do Facebook do destinatário. Veja https://developers.facebook.com/docs/messenger- platform/webhook-reference/message -received.

Atributos de solicitação do Kik

Atributo	Descrição
`kik-chat-id`	O identificador para a conversa no qual o bot está envolvido. Para obter mais informações, consulte https://dev.kik.com/#/docs/messaging #message -formats.
`kik-chat-type`	O tipo de conversa do qual a mensagem foi originada. Para obter mais informações, consulte https://dev.kik.com/#/docs/messaging #message -formats.
`kik-message-id`	Um UUID que identifica a mensagem. Para obter mais informações, consulte https://dev.kik.com/#/docs/messaging #message -formats.
`kik-message-type`	O tipo de mensagem. Para obter mais informações, consulte https://dev.kik.com/#/docs/messaging #message -types.

Atributos de solicitação do Twilio

Atributo	Descrição
`user-id`	O número de telefone do remetente ("De"). Consulte https://www.twilio.com/docs/api/rest/message.
`twilio-target-phone-number`	O número de telefone do destinatário ("Para"). Consulte https://www.twilio.com/docs/api/rest/message.

Atributos de solicitação do Slack

Atributo	Descrição
`user-id`	O identificador do usuário do Slack. Veja https://api.slack.com/types/o usuário.
`slack-team-id`	O identificador da equipe que enviou a mensagem. Consulte https://api.slack.com/methods/team.info.

Atributo	Descrição
`slack-bot-token`	O token do desenvolvedor que dá ao bot acesso ao Slack APIs. Veja os https://api.slack.com/docs/tipos de tokens.

Integração de um bot do Amazon Lex com o Facebook Messenger

Este exercício mostra como integrar o Facebook Messenger ao bot do Amazon Lex. Execute as seguintes etapas:

1. Criar um bot do Amazon Lex
2. Criação de um aplicativo do Facebook
3. Integrar o Facebook Messenger ao seu bot do Amazon Lex
4. Validação da integração

Tópicos

- Etapa 1: criar um bot do Amazon Lex
- Etapa 2: criar um aplicativo do Facebook
- Etapa 3: integrar o Facebook Messenger ao bot do Amazon Lex
- Etapa 4: teste a integração

Etapa 1: criar um bot do Amazon Lex

Se você ainda não tem um bot do Amazon Lex, crie e implante um. Neste tópico, pressupomos que você esteja usando o bot que criou no Exercício 1 dos Conceitos básicos. No entanto, você pode usar qualquer um dos bots de exemplo fornecidos neste guia. Para o Exercício 1 dos Conceitos básicos, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..

1. Crie um bot do Amazon Lex. Para obter instruções, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..
2. Implante o bot e crie um alias. Para obter instruções, consulte Exercício 3: publique uma versão e crie um alias.

Etapa 2: criar um aplicativo do Facebook

No portal de desenvolvedor do Facebook, crie um aplicativo do Facebook e uma página do Facebook. Para obter instruções, consulte Início rápido na documentação da plataforma Facebook Messenger. Anote o seguinte:

- O App Secret para o aplicativo do Facebook
- O Page Access Token para a página do Facebook

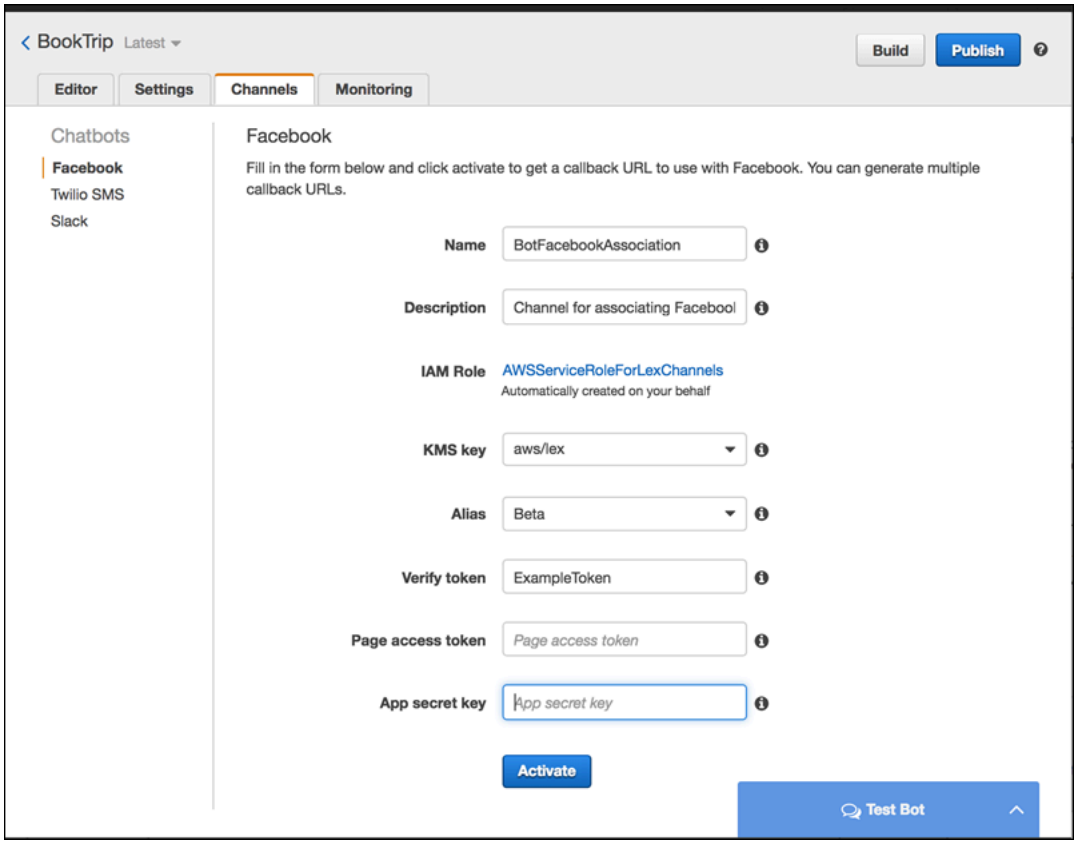
Etapa 3: integrar o Facebook Messenger ao bot do Amazon Lex

Nesta seção, você integrará o Facebook Messenger ao seu bot do Amazon Lex.

Depois de concluir essa etapa, o console fornece um URL de retorno de chamada. Anote esse URL.

Para integrar o Facebook Messenger ao seu bot

1. a. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
   b. Escolha seu bot do Amazon Lex.
   c. Escolha Canais.
   d. Escolha Facebook em Chatbots. O console exibe a página de integração do Facebook.
   e. Na página de integração do Facebook:
      - Digite este nome: `BotFacebookAssociation`.
      - Para KMS key, escolha aws/lex.
      - Para Alias, escolha o alias bot.
      - Para Verify token, digite um token. Pode ser qualquer string que você escolher (por exemplo, `ExampleToken`). Use esse token posteriormente no portal de desenvolvedor do Facebook ao configurar o webhook.
      - Para acessar Page access token, digite o token que você obteve do Facebook na Etapa 2.
      - Para App secret key, digite a chave que você obteve do Facebook na Etapa 2.



f. Selecione Ativar.

O console cria a associação de canal de bot e retorna um URL de retorno de chamada. Anote esse URL.

2. No portal de desenvolvedor do Facebook, escolha seu aplicativo.
3. Escolha o produto Messenger e selecione Setup webhooks na seção Webhook da página.

   Para obter instruções, consulte Início rápido na documentação da plataforma Facebook Messenger.
4. Na página webhook do assistente de assinatura:
   - Para Callback URL, digite o URL de retorno de chamada no console do Amazon Lex anteriormente no procedimento.
   - Para Verify Token, digite o mesmo token usado no Amazon Lex.
   - Escolha Subscription Fields (messages, messaging_postbacks e messaging_optins).
   - Escolha Verificar e salvar. Isso inicia um handshake entre o Facebook e o Amazon Lex.

5. Ative a integração do Webhooks. Escolha a página que você criou e, em seguida, escolha subscribe.

> **Note**
>
> Se você atualizar ou recriar um webhook, deverá cancelar a assinatura e, em seguida, assinar a página novamente.

Etapa 4: teste a integração

Agora, você pode iniciar uma conversa no Facebook Messenger com seu bot do Amazon Lex.

1. Abra a página do Facebook e escolha Message.
2. Na janela do Messenger, use os mesmas declarações de teste fornecidos no Etapa 1: criar um bot Amazon Lex (console).

Integrar um bot do Amazon Lex com o Kik

Este exercício fornece instruções para integrar um bot do Amazon Lex com o aplicativo de sistema de mensagens Kik. Execute as seguintes etapas:

1. Crie um bot do Amazon Lex.
2. Crie um bot do Kik usando o aplicativo e o site do sistema.
3. Integre o bot do Amazon Lex com o do Kik usando o console do Amazon Lex.
4. Inicie uma conversa com o bot do Amazon Lex usando o Kik para testar a associação entre o bot do Amazon Lex e o Kik.

Tópicos

- Etapa 1: criar um bot do Amazon Lex
- Etapa 2: criar um bot do Kik
- Etapa 3: integrar o bot do Kik com o do Amazon Lex
- Etapa 4: teste a integração

Etapa 1: criar um bot do Amazon Lex

Se você ainda não tem um bot do Amazon Lex, crie e implante um. Neste tópico, pressupomos que você esteja usando o bot que criou no Exercício 1 dos Conceitos básicos. No entanto, você pode usar qualquer um dos bots de exemplo fornecidos neste guia. Para o Exercício 1 dos Conceitos básicos, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console).

1. Crie um bot do Amazon Lex. Para obter instruções, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..
2. Implante o bot e crie um alias. Para obter instruções, consulte Exercício 3: publique uma versão e crie um alias.

Próxima etapa

Etapa 2: criar um bot do Kik

Etapa 2: criar um bot do Kik

Nesta etapa, use a interface de usuário do Kik para criar um bot. Use as informações geradas ao criar o bot para conectá-lo ao bot do Amazon Lex.

1. Se você ainda não tiver feito isso, baixe e instale o aplicativo do Kik e cadastre-se para criar uma conta. Se você já tem uma conta, faça login.
2. Abra o site da Kik em https://dev.kik.com/. Deixe a janela do navegador aberta.
3. No aplicativo do Kik, selecione o ícone de engrenagem para abrir as configurações e, em seguida, selecione Seu código Kid.
4. Digitalize o código no site do Kik para abrir o chatbot do Botsworth. Selecione Sim para abrir o painel de bot.
5. No aplicativo do Kik, selecione Criat um Bot. Siga as solicitações para criar o bot do Kik.
6. Assim que o bot for criado, selecione Configuração no seu navegador. Verifique se o novo bot está selecionado.
7. Anote o nome do bot e a chave de API para a próxima seção.

Próxima etapa

Etapa 3: integrar o bot do Kik com o do Amazon Lex

Etapa 3: integrar o bot do Kik com o do Amazon Lex

Agora que criou os bots do Amazon Lex e do Kik, você está pronto para criar um associação de canal entre eles no Amazon Lex. Quando a associação for ativada, o Amazon Lex configurará automaticamente uma URL de retorno de chamada com o Kik.

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Selecione o bot do Amazon Lex que você criou na Etapa 1.
3. Escolha a guia Canais.
4. Na seção Canais, selecione Kik.
5. Na página do Kik, insira as informações a seguir:
   - Digite um nome. Por exemplo, `BotKikIntegration`.
   - Digite uma descrição.
   - No menu suspenso Chave do KMS, escolha "aws/lex".
   - Em Alias, selecione um alias da lista suspensa.
   - Em Nome de usuário do bot do Kik, digite o nome que você deu ao bot no Kik.
   - Em Chave da API do Kik, digite a chave de API que foi atribuída ao bot no Kik.
   - Em Saudação do usuário, digite a saudação que você deseja que o bot envie na primeira vez em que um usuário iniciar um bate-papo com ele.
   - Em Mensagem de erro, insira uma mensagem de erro que é exibida ao usuário quando parte da conversa não é compreendida.
   - Em Comportamento do bate-papo em grupo, selecione uma das opções:
     - Habilitar: permite que todo o grupo de bate-papo interaja com o bot em uma única conversa.
     - Desabilitar: restringe a conversa a um usuário do grupo de bate-papo.
   - Selecione Ativar para criar a associação e vinculá-la ao bot do Kik.

Kik

Fill in the form below and click activate to get a callback URL to use with Kik. You can generate multiple callback URLs. Learn more on steps to integrate with Kik.

Channel Name*	KikBotIntegration
Channel Description	Integrate an Amazon Lex bot with Kik
IAM Role	AWSServiceRoleForLexChannels Automatically created on your behalf
KMS key	aws/lex
Alias*	BETA
Kik Bot User Name*	XXXXXXXX
Kik API Key*	XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXX
User Greeting*	Welcome to my first Amazon Lex bot on Kik

Advanced configuration

Error Message*	There seems to be a problem.
Group Chat Behavior	○ Enable ◉ Disable

* Required Field

Activate

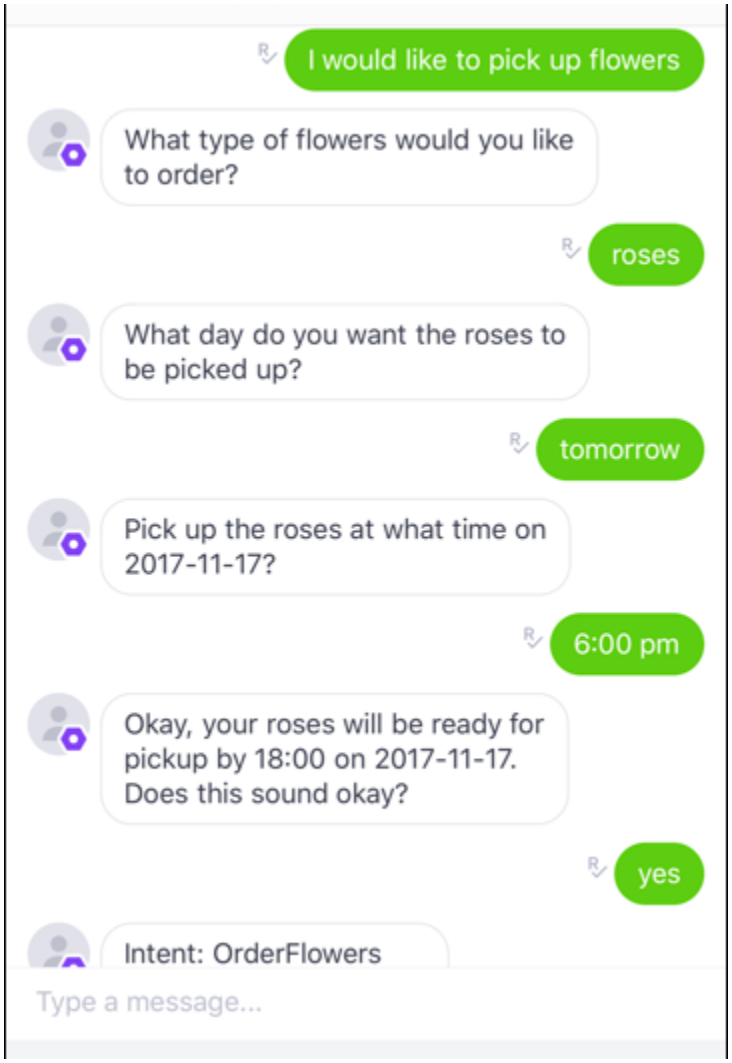
Próxima etapa

Etapa 4: teste a integração

Etapa 4: teste a integração

Agora que você criou uma associação entre o bot do Amazon Lex e o do Kik, use o aplicativo do Kik para testá-la.

1. Inicie o aplicativo do Kik e faça login. Selecione o bot que você criou.
2. Teste o bot da seguinte forma:



À medida que você inserir cada frase, o bot do Amazon Lex responderá por meio do Kik com um prompt que você criou para cada slot.

Integração de um bot de Amazon Lex com o Slack

Este exercício fornece instruções para integrar um bot do Amazon Lex com o aplicativo de sistema de mensagens Slack. Execute as seguintes etapas:

1. Crie um bot do Amazon Lex.
2. Crie um aplicativo de mensagem do Slack
3. Integre o aplicativo do Slack ao bot do Amazon Lex.

4. Teste a integração iniciando uma conversa com seu bot do Amazon Lex. Você envia mensagens com o aplicativo do Slack e testa em uma janela do navegador.

Tópicos

- Etapa 1: criar um bot do Amazon Lex
- Etapa 2: cadastre-se no Slack e crie uma equipe do Slack
- Etapa 3: crie uma aplicação do Slack
- Etapa 4: Integrar o aplicativo do Slack com o bot do Amazon Lex
- Etapa 5: Completar a integração do Slack
- Etapa 6: teste a integração

Etapa 1: criar um bot do Amazon Lex

Se você ainda não tem um bot do Amazon Lex, crie e implante um. Neste tópico, pressupomos que você esteja usando o bot que criou no Exercício 1 dos Conceitos básicos. No entanto, você pode usar qualquer um dos bots de exemplo fornecidos neste guia. Para o Exercício 1 dos Conceitos básicos, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console).

1. Crie um bot do Amazon Lex. Para obter instruções, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..
2. Implante o bot e crie um alias. Para obter instruções, consulte Exercício 3: publique uma versão e crie um alias.

Próxima etapa

Etapa 2: cadastre-se no Slack e crie uma equipe do Slack

Etapa 2: 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 3: crie uma aplicação do Slack

Etapa 3: crie uma aplicação do Slack

Nesta seção, faça o seguinte:

1. Crie uma aplicação do Slack no console da API do Slack
2. Configure o aplicativo para adicionar mensagens interativas ao seu bot:

No final desta seção, você obterá as credenciais do aplicativo (ID do cliente, segredo do cliente e token de verificação). Na próxima seção, você usará essas informações para configurar a associação do canal do bot no console do Amazon Lex.

1. Faça login no console da API do Slack em http://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 da aplicação 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 válido. Por exemplo, você poderá 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 4: Integrar o aplicativo do Slack com o bot do Amazon Lex

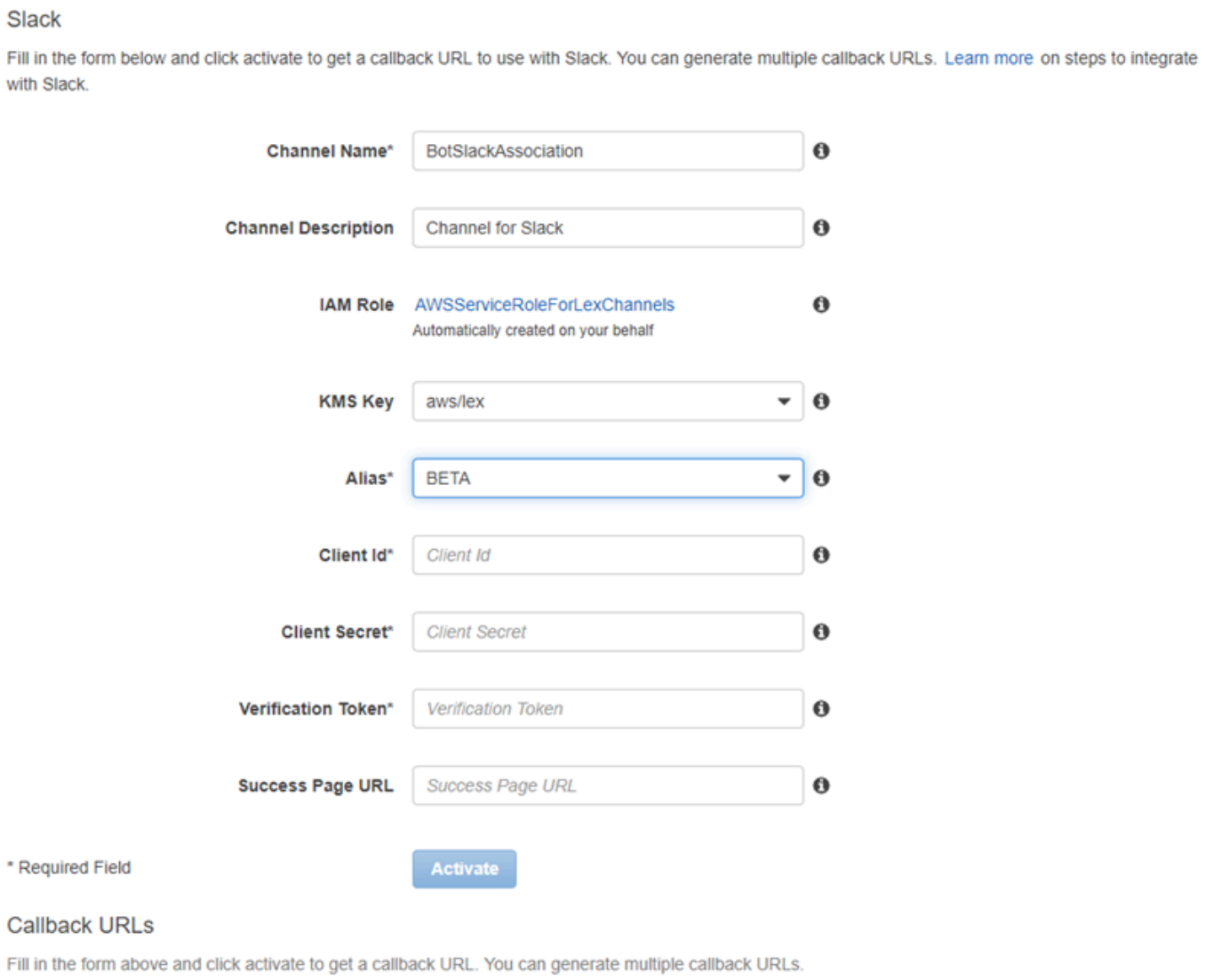
Etapa 4: Integrar o aplicativo do Slack com o bot do Amazon Lex

Agora que tem as credenciais do aplicativo do Slack, você pode integrar o aplicativo com o bot do Amazon Lex. Para associar o aplicativo do Slack a seu bot, adicione uma associação de canal do bot no Amazon Lex.

No console do Amazon Lex, ative uma associação de canal do bot para associar o bot ao aplicativo do Slack. Quando a associação do canal do bot é ativada, o Amazon Lex retorna dois URLs (URL do Postback e OAuthURL). Grave-os URLs porque você precisará deles mais tarde.

Para integrar o aplicativo do Slack ao bot do Amazon Lex

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Selecione o bot do Amazon Lex que você criou na Etapa 1.
3. Escolha a guia Canais.
4. No menu esquerdo, selecione Slack.
5. Na página Slack, forneça os valores a seguir:
   - Digite um nome. Por exemplo, `BotSlackIntegration`.
   - No menu suspenso Chave do KMS, escolha "aws/lex".
   - Para Alias, escolha o alias bot.
   - Digite o ID do cliente, o Segredo do clientee o Token de verificação, que você registrou na etapa anterior. Essas são as credenciais da aplicação do Slack.



6. Selecione Ativar.

   O console cria a associação do canal do bot e retorna duas URLs (URL do postback e OAuth URL). Anote-os. Na próxima seção, você atualizará a configuração da sua aplicação do Slack para usar esses endpoints conforme o seguinte:

   - A URL do Postback é o endpoint do bot do Amazon Lex que ouve aos eventos do Slack. Você usa este URL:
     - Como o URL de solicitação no atributo Assinaturas de eventos do aplicativo do Slack.
     - Para substituir o valor do espaço reservado para o URL solicitado no atributo Mensagens interativas do aplicativo do Slack.

- A OAuth URL é o endpoint do seu bot Amazon Lex para um OAuth handshake com o Slack.

Próxima etapa

Etapa 5: Completar a integração do Slack

Etapa 5: Completar a integração do Slack

Nesta seção, use o console da API do Slack para completar a integração do aplicativo do Slack.

1. Faça login no console da API do Slack em http://api.slack.com API. Selecione o aplicativo que você criou em Etapa 3: crie uma aplicação do Slack.
2. Atualize o recurso OAuth & Permissões da seguinte forma:
   a. No menu à esquerda, escolha OAuth & Permissões.
   b. Na URLs seção Redirecionar, adicione a OAuth URL que o Amazon Lex forneceu na etapa anterior. Escolha Adicionar um novo URL de redirecionamento e, em seguida, escolha Salvar URLs.
   c. Na seção Escopos do Bot Token, adicione duas permissões com o botão Adicionar um OAuth escopo. 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 URL do Postback que o Amazon Lex forneceu na etapa anterior. Insira o URL Postback que você salvou na etapa 4 e selecione Salvar alterações.
4. Assine o atributo Assinaturas de eventos da seguinte forma:
   - Habilite eventos escolhendo a opção Ligado.
   - Defina o valor da URL da solicitação como URL de Postback que o Amazon Lex forneceu na etapa anterior.
   - Na seção Inscreva-se nos eventos do bot, inscreva-se no evento de 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 a partir da guia de mensagens da seguinte maneira:
   - No menu à esquerda, escolha Página inicial do aplicativo.

- Na seção Mostrar guias, escolha Permitir que os usuários enviem comandos e mensagens com barras a partir da guia de mensagens.

Próxima etapa

Etapa 6: teste a integração

Etapa 6: teste a integração

Agora, use uma janela do navegador para testar a integração do Slack com o bot do Amazon Lex.

1. Escolha Gerenciar distribuição em Configurações. Escolha Adicionar ao Slack para instalar o aplicativo. Autorize o bot a responder a mensagens.
2. Você é redirecionado para sua equipe do Slack. No menu esquerdo, na seção Mensagens diretas, escolha o bot. Se você não vir o bot, escolha o ícone de mais (+) ao lado de Mensagens diretas para procurá-lo.
3. Inicie um bate-papo com o aplicativo do Slack, que está vinculado ao bot do Amazon Lex. Seu bot agora responde a mensagens.

   Se você criou o bot usando o Exercício 1 dos Conceitos básicos, pode usar as conversas de exemplo fornecidas no exercício. Para obter mais informações, consulte Etapa 4: Adicione a função do Lambda como hook de código (console).

Integração de um bot do Amazon Lex com o SMS programável do Twilio

Este exercício fornece instruções para integrar um bot do Amazon Lex com o serviço de mensagens simples (SMS) do Twilio. Execute as seguintes etapas:

1. Criar um bot do Amazon Lex
2. Integre o SMS programável do Twilio com seu bot de Amazon Lex
3. Iniciar uma interação com o bot do Amazon Lex e testar a configuração usando o serviço SMS no celular
4. Teste da integração

Tópicos

- Etapa 1: criar um bot do Amazon Lex
- Etapa 2: criar uma conta de SMS do Twilio
- Etapa 3: Integrar o endpoint do serviço de sistema de mensagens do Twilio com o bot do Amazon Lex
- Etapa 4: teste a integração

Etapa 1: criar um bot do Amazon Lex

Se você ainda não tem um bot do Amazon Lex, crie e implante um. Neste tópico, pressupomos que você esteja usando o bot que criou no Exercício 1 dos Conceitos básicos. No entanto, você pode usar qualquer um dos bots de exemplo fornecidos neste guia. Para o Exercício 1 dos Conceitos básicos, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..

1. Crie um bot do Amazon Lex. Para obter instruções, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..
2. Implante o bot e crie um alias. Para obter instruções, consulte Exercício 3: publique uma versão e crie um alias.

Etapa 2: criar 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 inscrição, consulte https://www.twilio.com/console.

Etapa 3: Integrar o endpoint do serviço de sistema de mensagens do Twilio com o bot do Amazon Lex

Como integrar o Twilio com seu bot de Amazon Lex

1. Para associar o bot do Amazon Lex com o endpoint de SMS programável do Twilio, ative a associação de canal do bot no console do Amazon Lex. Quando a associação de canal do bot estiver ativada, o Amazon Lex retornará uma URL de retorno de chamada. Anote o URL de retorno de chamada, pois você precisará dele mais tarde.

a. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.

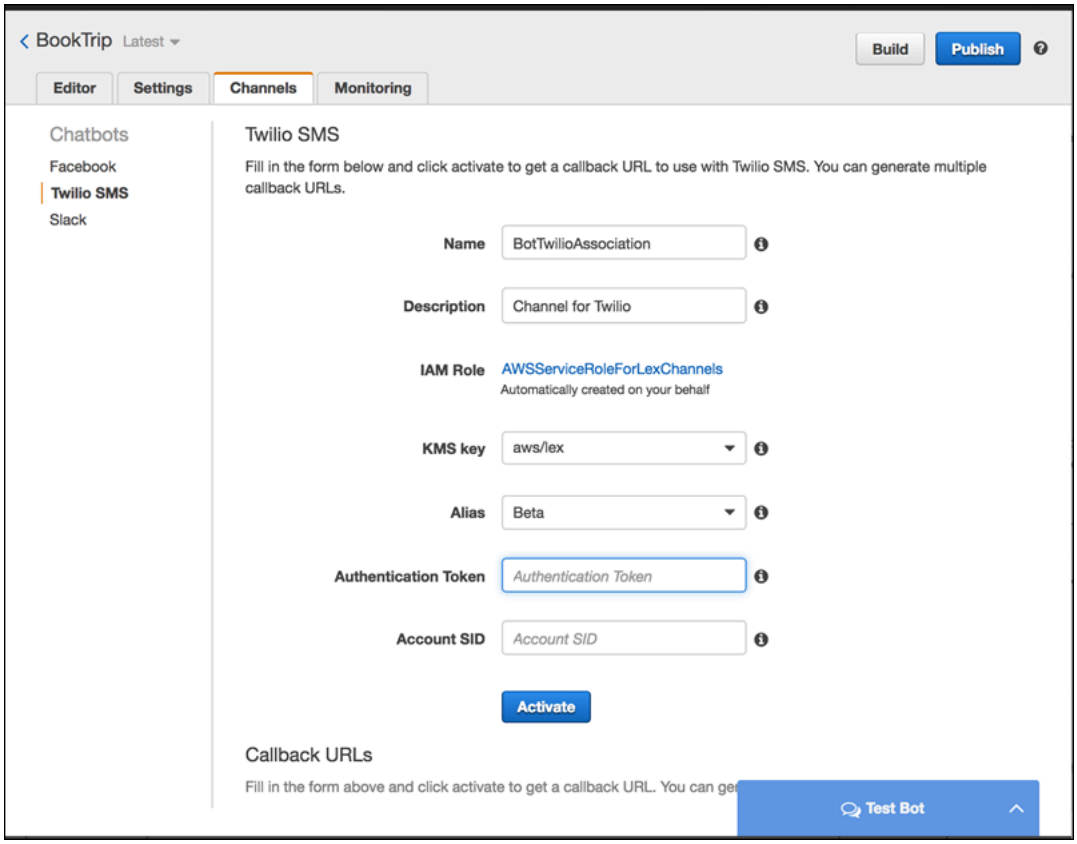
b. Selecione o bot do Amazon Lex que você criou na Etapa 1.

c. Escolha a guia Canais.

d. Na seção Chatbots, escolha Twilio SMS.

e. Na página Twilio SMS, forneça as seguintes informações:

- Digite um nome. Por exemplo, `BotTwilioAssociation`.
- Escolha "aws/lex" em KMS key.
- Para Alias, escolha o alias bot.
- Para Authentication Token, digite o TOKEN DE AUTORIZAÇÃO para sua conta do Twilio.
- Em Account SID, digite o SID DA CONTA do Twilio.



f. Selecione Ativar.

O console cria a associação de canal de bot e retorna um URL de retorno de chamada. Anote o URL.

2. No console do Twilio, conecte o endpoint de SMS do Twilio ao bot do Amazon Lex.

   a. Faça login no console Twilio em https://www.twilio.com /console.

   b. Se você não tiver um endpoint de SMS do Twilio, crie um.

   c. Atualize a configuração Configurações de entrada do serviço de mensagens definindo o valor URL DA SOLICITAÇÃO para o URL de retorno de chamada que o Amazon Lex forneceu na etapa anterior.

Etapa 4: teste a integração

Use seu celular para testar a integração entre o SMS do Twilio e o bot.

Como testar a integração

1. Entre no console do Twilio em https://www.twilio.com/console e faça o seguinte:

   a. Verifique se você tem um número do Twilio associado ao serviço de mensagens em Manage Numbers.

      Envie mensagens para esse número e inicie uma interação por SMS com o bot do Amazon Lex no celular.

   b. Verifique se seu telefone celular está na lista branca como Verified Caller ID.

      Se não estiver, siga as instruções no console do Twilio para permitir o celular que você planeja usar para testes.

      Agora, você pode usar o celular para enviar mensagens ao endpoint de SMS do Twilio, que é mapeado para o bot do Amazon Lex.

2. Usando seu celular, envie mensagens para o número do Twilio.

   O bot do Amazon Lex responde. Se você criou o bot usando o Exercício 1 dos Conceitos básicos, pode usar as conversas de exemplo fornecidas no exercício. Para obter mais informações, consulte Etapa 4: Adicione a função do Lambda como hook de código (console).

Implantação de um bot do Amazon Lex em aplicativos móveis

Usando AWS Amplify, você pode integrar seus bots do Amazon Lex com aplicativos móveis ou web. Para obter mais informações, consulte Interações: introdução nos documentos do AWS Amplify .

Importação e exportação de bots, intenções e tipos de slot do Amazon Lex

Você pode importar ou exportar um bot, uma intenção ou um tipo de slot. Por exemplo, se quiser compartilhar um bot com um colega em uma conta da AWS diferente, você poderá exportá-lo e, em seguida, enviá-lo para ele. Se quiser adicionar várias declarações a um bot, você pode exportá-lo, adicionar as declarações e reimportá-lo para a conta.

Você pode exportar bots, intenções e tipos de slot no Amazon Lex (para compartilhar ou modificá-los) ou em um formato de habilidade do Alexa. Você só pode importar em formato do Amazon Lex.

Ao exportar um recurso, você precisa exportá-lo em um formato que seja compatível com o serviço para o qual está exportando, o Amazon Lex ou o Alexa Skills Kit. Se exportar um bot no formato do Amazon Lex, você poderá reimportá-lo para a conta, ou um usuário do Amazon Lex em outra conta poderá importá-lo para a respectiva conta. Você também pode exportar um bot em um formato compatível com uma habilidade do Alexa. Em seguida, você poderá importar o bot usando o Alexa Skills Kit para disponibilizá-lo com o Alexa. Para obter mais informações, consulte Exportar para uma habilidade do Alexa.

Quando você exporta um bot, uma intenção ou um tipo de slot, os recursos são gravados em um arquivo JSON. Para exportar um bot, uma intenção ou um tipo de slot, você pode usar o console do Amazon Lex ou a operação GetExport. Importe um bot, uma intenção ou um tipo de slot usando a StartImport.

Tópicos

- Exportar e importar em formato do Amazon Lex
- Exportar para uma habilidade do Alexa

Exportar e importar em formato do Amazon Lex

Para exportar bots, intenções e tipos de slot do Amazon Lex com a intenção de reimportá-los no Amazon Lex, você cria um arquivo JSON no formato do Amazon Lex. Você pode editar os recursos neste arquivo e reimportá-los no Amazon Lex. Por exemplo, você pode adicionar declarações a uma intenção e reimportar a intenção alterada para a conta. Você também pode usar o formato JSON para compartilhar um recurso. Por exemplo, você pode exportar um bot de uma região da AWS e,

em seguida, importá-lo para outra região. Ou você pode enviar o arquivo JSON a um colega para compartilhar um bot.

Tópicos

- Exportar em formato do Amazon Lex
- Importar em formato do Amazon Lex
- Formato JSON para importação e exportação

Exportar em formato do Amazon Lex

Exporte seus bots, intenções e tipos de slots do Amazon Lex para um formato que você possa importar para uma AWS conta. Você pode exportar os seguintes recursos:

- Um bot, inclusive todas as intenções e os tipos de slot personalizados usados pelo bot
- Uma intenção, inclusive todos os tipos de slot personalizados usados pela intenção
- Um tipo de slot personalizado, inclusive todos os valores do tipo de slot

Você só pode exportar uma versão numerada de um recurso. Você não pode exportar uma versão `$LATEST` do recurso.

A exportação é um processo assíncrono. Quando a exportação é concluída, você obtém um URL pré-assinado do Amazon S3. O URL fornece o local de um arquivo .zip que contém o recurso exportado em formato JSON.

Você usa o console ou a operação GetExport para exportar bots, intenções e tipos de slot personalizados.

O processo para exportar um bot, uma intenção ou um tipo de slot é o mesmo. Nos procedimentos a seguir, substitua a intenção ou o tipo de slot do bot.

Exportar um bot

Para exportar 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 Bots e o bot a ser exportado.

3. No menu Ações, escolha Export (Exportar).
4. Na caixa de diálogo Export Bot (Exportar bot), escolha a versão do bot a ser exportada. Em Plataforma, escolha Amazon Lex.
5. Escolha Exportar.
6. Faça download e salve o arquivo .zip.

O Amazon Lex exporta o bot para um arquivo JSON contido no arquivo .zip. Para atualizar o bot, modifique o texto JSON e o reimporte para o Amazon Lex.

Próxima etapa

Importar em formato do Amazon Lex

Importar em formato do Amazon Lex

Depois de exportar um recurso para um arquivo JSON no formato Amazon Lex, você pode importar o arquivo JSON contendo o recurso em uma ou mais contas. AWS Por exemplo, você pode exportar um bot e, em seguida, importá-lo para outra região da AWS. Ou você pode enviar o bot para uma colega, de maneira que ela possa importá-lo para a respectiva conta.

Ao importar um bot, uma intenção ou um tipo de slot, você deve decidir se deseja substituir a versão `$LATEST` de um recurso, como uma intenção ou um tipo de slot, durante a importação, ou se deseja que a importação falhe caso queira preservar o recurso que está na conta. Por exemplo, se estivesse fazendo upload de uma versão editada de um recurso para a conta, você optaria por substituir a versão `$LATEST`. Se estiver fazendo upload de um recurso enviado por um colega, você poderá optar por fazer a importação falhar se houver conflitos de recurso. Dessa maneira, os próprios recursos não são substituídos.

Durante a importação de um recurso, as permissões atribuídas ao usuário que está fazendo a solicitação de importação se aplicam. O usuário deve ter permissões para todos os recursos na conta afetada pela importação. O usuário também deve ter permissão para as operações GetBot, PutBot, GetIntent PutIntent, GetSlotType, PutSlotType. Para obter mais informações sobre permissões, consulte Como o Amazon Lex funciona com o IAM.

A importação relata erros ocorridos durante o processamento. Alguns erros são relatados antes do início da importação, outros são relatados durante o processo de importação. Por exemplo, se a conta que estiver importando uma intenção não tiver permissão para chamar uma função do Lambda usada pela intenção, a importação falhará antes das alterações serem feitas nos tipos de slot ou

nas intenções. Se uma importação falhar durante o processo de importação, a versão `$LATEST` de qualquer intenção ou tipo de slot importado antes do processo ter falhado será modificada. Você não pode reverter alterações feitas na versão `$LATEST`.

Quando você importa um recurso, todos os recursos dependentes são importados para a versão `$LATEST` do recurso e, em seguida, recebem uma versão numerada. Por exemplo, se um bot usar uma intenção, ela receberá uma versão numerada. Se uma intenção usar um tipo de slot personalizado, ele receberá uma versão numerada.

Um recurso é importado somente uma vez. Por exemplo, se o bot contiver uma intenção `OrderPizza` e uma intenção `OrderDrink` em que ambas dependam do tipo de slot personalizado `Size`, o tipo de slot `Size` será importado uma vez e usado em ambas as intenções.

> **Note**
>
> Se você exportou seu bot com o parâmetro `enableModelImprovements` definido como `false`, deverá abrir o arquivo .zip contendo a definição do bot e alterar o parâmetro `enableModelImprovements` para `true` nas seguintes regiões:
>
> - Ásia-Pacífico (Singapura) (ap-southeast-1)
> - Ásia Pacific (Tóquio) (ap-northeast-1)
> - Europa (Frankfurt) (eu-central-1)
> - Europa (Londres) (eu-west-2)

O processo para importar um bot, uma intenção ou um tipo de slot personalizado é o mesmo. Nos procedimentos a seguir, substitua a intenção ou o tipo de slot, conforme apropriado.

Importar um bot

Para importar 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 Bots e o bot a ser importado. Para importar um novo bot, ignore esta etapa.
3. Em Ações, escolha Import (Importar).
4. Em Import Bot (Importar bot), escolha o arquivo .zip que contém o arquivo JSON que contém o bot a ser importado. Se você quiser ver conflitos de mesclagem antes de mesclar, escolha Notify

me of merge conflicts (Notificar-me de conflitos de mesclagem). Se você desativar a verificação de conflitos, a versão `$LATEST` de todos os recursos usados pelo bot será substituída.

5. Escolha Importar. Se você tiver optado por ser notificado sobre conflitos de mesclagem e houver conflitos, será exibida uma caixa de diálogo os listando. Para substituir a versão `$LATEST` de todos os recursos conflitantes, escolha Overwrite and continue (Substituir e continuar). Para interromper a importação, escolha Cancelar.

Você já pode testar o bot na conta.

Formato JSON para importação e exportação

Os exemplos a seguir mostram a estrutura JSON para exportar e importar tipos de slot, intenções e bots em formato do Amazon Lex.

Estrutura do tipo de slot

Esta é a estrutura JSON para tipos de slot personalizados. Use essa estrutura ao importar ou exportar tipos de slot, além de exportar intenções que dependam de tipos de slot personalizados.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "name": "slot type name",
    "version": "version number",
    "enumerationValues": [
      {
        "value": "enumeration value",
        "synonyms": []
      },
      {
        "value": "enumeration value",
        "synonyms": []
      }
    ],
    "valueSelectionStrategy": "ORIGINAL_VALUE or TOP_RESOLUTION"
  }
```

```
}
```

Estrutura da intenção

Esta é a estrutura JSON para intenções. Use essa estrutura ao importar ou exportar intenções e bots que dependam de uma intenção.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "description": "intent description",
    "rejectionStatement": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ]
    },
    "name": "intent name",
    "version": "version number",
    "fulfillmentActivity": {
      "type": "ReturnIntent or CodeHook"
    },
    "sampleUtterances": [
      "string",
      "string"
    ],
    "slots": [
      {
        "name": "slot name",
        "description": "slot description",
        "slotConstraint": "Required or Optional",
        "slotType": "slot type",
        "valueElicitationPrompt": {
          "messages": [
            {
              "contentType": "PlainText or SSML or CustomPayload",
              "content": "string"
```

```
            }
          ],
          "maxAttempts": value
        },
        "priority": value,
        "sampleUtterances": []
      }
    ],
    "confirmationPrompt": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        },
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ],
      "maxAttempts": value
    },
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Estrutura do tipo de slot.
    ]
  }
}
```

Estrutura do bot

Esta é a estrutura JSON para bots. Use essa estrutura ao importar ou exportar bots.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "name": "bot name",
    "version": "version number",,
    "nluIntentConfidenceThreshold": 0.00-1.00,
    "enableModelImprovements": true | false,
```

```
    "intents": [
        List of intent JSON structures.
        For more information, see Estrutura da intenção.
    ],
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Estrutura do tipo de slot.
    ],
    "voiceId": "output voice ID",
    "childDirected": boolean,
    "locale": "en-US",
    "idleSessionTTLInSeconds": timeout,
    "description": "bot description",
    "clarificationPrompt": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ],
      "maxAttempts": value
    },
    "abortStatement": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ]
    }
  }
}
```

Exportar para uma habilidade do Alexa

Exporte seu esquema de bot em um formato compatível com uma habilidade do Alexa. Depois de exportar o bot para um arquivo JSON, você poderá fazer upload dele para o Alexa usando o criador de habilidades.

Para exportar um bot e o esquema (modelo de interação)

1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Selecione o bot que você deseja exportar.
3. Em Ações, escolha Export (Exportar).
4. Escolha a versão do bot que você deseja exportar. Para o formato, escolha Alexa Skills Kit e Export (Exportar).
5. Se a caixa de diálogo de download for exibida, escolha um local para salvar o arquivo e selecione Save (Salvar).

O arquivo obtido por download é um arquivo .zip contendo um arquivo com o nome do bot exportado. Ele contém as informações necessárias para importar o bot como uma habilidade do Alexa.

> **Note**
>
> O Amazon Lex e o Alexa Skills Kit são diferentes nos seguintes aspectos:
>
> - Os atributos de sessão, indicados por colchetes ([]), não são compatíveis com o Alexa Skills Kit. Você precisa atualizar solicitações que usem atributos de sessão.
> - Sinais de pontuação não são compatíveis com o Alexa Skills Kit. Você precisa atualizar declarações que usem pontuação.

Para fazer upload do bot para uma habilidade do Alexa

1. Faça login no portal do desenvolvedor em https://developer.amazon.com/.
2. Na página Alexa Skills (Skills da Alexa), selecione Create Skill (Criar skill).
3. Na página Criar uma skill, insira um nome de skill e a linguagem padrão para a skill. Verifique se a opção Custom (Personalizar) está selecionada para o modelo de skill e selecione Create skill (Criar skill).
4. Verifique se a opção Start from scratch (Começar do zero) está selecione e selecione Choose (Escolher).
5. No menu à esquerda, selecione JSON Editor (Editor JSON). Arraste o arquivo JSON exportado do Amazon Lex para o editor JSON.
6. Selecione Save Model (Salvar modelo) para salvar o modelo de interação.

Depois de fazer upload do esquema para a habilidade do Alexa, faça alterações necessárias para executar a habilidade com o Alexa. Para obter mais informações sobre como criar um recurso do Alexa, consulte Usar o Skill Builder (Beta) no Alexa Skills Kit.

Exemplos adicionais: criação de bots do Amazon Lex

As seções a seguir fornecem exercícios adicionais do Amazon Lex com step-by-step instruções.

Tópicos

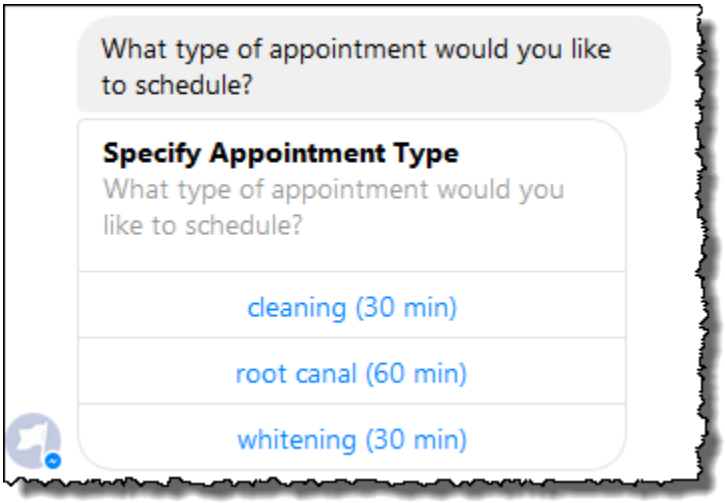
- Agendar uma consulta
- Reservar uma viagem
- Exemplo: uso de um cartão de resposta
- Atualizar declarações
- Exemplo: Integração com um site
- Assistente de atendente do call center

Agendar uma consulta

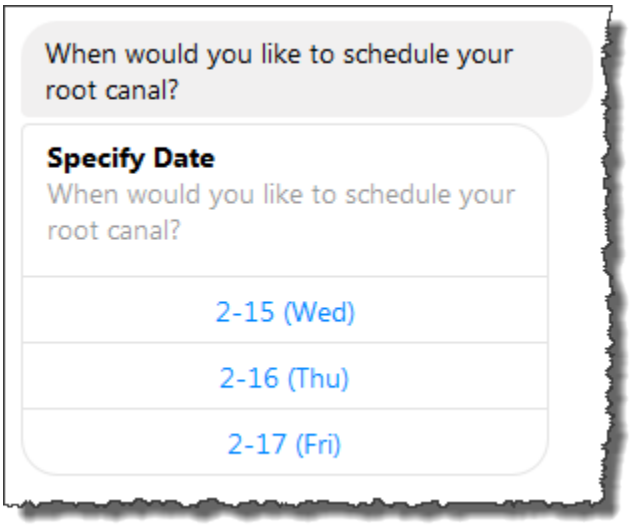
O exemplo de bot neste exercício marca consultas em uma clínica odontológica. O exemplo também ilustra o uso de cartões de resposta para obter entradas do usuário com botões. Especificamente, o exemplo ilustra a geração dinâmica de cartões de resposta em runtime.

Você pode configurar cartões de resposta no momento da construção (também chamados de cartões de resposta estáticos) ou gerá-los dinamicamente em uma AWS Lambda função. Neste exemplo, o bot usa os seguintes cartões de resposta:

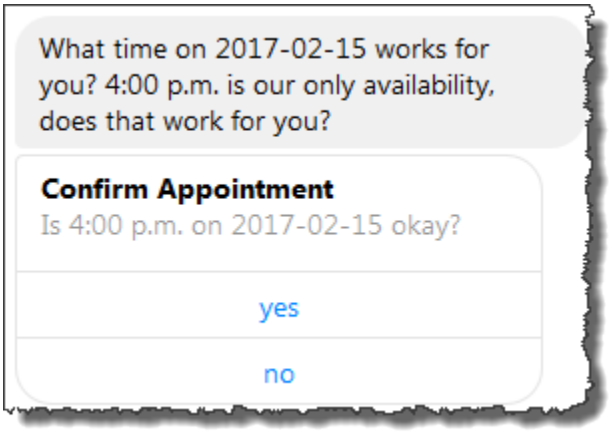
- Um cartão de resposta que lista botões para o tipo de consulta. Para obter um exemplo, veja a imagem a seguir.



- Um cartão de resposta que lista botões para a data de consulta. Para obter um exemplo, veja a imagem a seguir.



- Um cartão de resposta que lista botões para confirmar um horário de consulta sugerido. Para obter um exemplo, veja a imagem a seguir.



As datas e os horários de consulta disponíveis variam, o que exige que você gere cartões de resposta em runtime. Você usa uma AWS Lambda função para gerar esses cartões de resposta dinamicamente. A função do Lambda retorna cartões de resposta em sua resposta a Amazon Lex. Amazon Lex inclui o cartão de resposta em sua resposta ao cliente.

Se um cliente (por exemplo, Facebook Messenger) oferecer suporte a cartões de resposta, o usuário poderá escolher na lista de botões ou digitar a resposta. Caso contrário, o usuário simplesmente digitará a resposta.

Além do botão mostrado no exemplo anterior, você também pode incluir imagens, anexos e outras informações úteis a serem exibidas nos cartões de resposta. Para obter mais informações sobre cartões de resposta, consulte Cartões de resposta.

Neste exercício, você faz o seguinte:

- Crie e teste um bot (usando o ScheduleAppointment blueprint). Para este exercício, você usará um esquema de bot para configurar e testar o bot rapidamente. Para obter uma lista de esquemas disponíveis, consulte Amazon Lex e esquemas AWS Lambda.Este bot está pré-configurado com uma intenção (`MakeAppointment`).

- Crie e teste uma função Lambda (usando o lex-make-appointment-python esquema fornecido pelo Lambda). Configure a intenção `MakeAppointment` para usar essa função do Lambda como um hook de código para executar a inicialização, a validação e o cumprimento de tarefas.

> **Note**
>
> A função do Lambda de exemplo fornecida exibe uma conversa dinâmica baseada na disponibilidade inventada de uma consulta odontológica. Em uma aplicação real, você pode usar um calendário real para definir uma consulta.

- Atualize a configuração de intenção `MakeAppointment` para usar a função do Lambda como hook de código. Em seguida, teste a end-to-end experiência.
- Publique o bot de agendamento de consultas no Facebook Messenger para você possa ver os cartões de resposta em ação (atualmente, o cliente no console do Amazon Lex não oferece suporte a cartões de resposta).

As seções a seguir fornecem informações resumidas sobre os esquemas usados por você neste exercício.

Tópicos

- Visão geral do Bot Blueprint () ScheduleAppointment
- Visão geral da função Lambda Blueprint () lex-make-appointment-python
- Etapa 1: criar um bot do Amazon Lex
- Etapa 2: criar uma função do Lambda
- Etapa 3: atualizar a intenção - configuração de um hook de código

- Etapa 4: implantar o bot na plataforma do Facebook Messenger
- Detalhes do fluxo de informações

Visão geral do Bot Blueprint () ScheduleAppointment

O ScheduleAppointment esquema que você usa para criar um bot para este exercício é pré-configurado com o seguinte:

- Tipos de slot – um tipo de slot personalizado chamado `AppointmentTypeValue`, com os valores de enumeração `root canal`, `cleaning` e `whitening`.
- Intenção – uma intenção (`MakeAppointment`), que está pré-configurada da seguinte forma:
  - Slots – a intenção está configurado com os seguintes slots:
    - Slot `AppointmentType`, do tipo de slot personalizado `AppointmentTypes`.
    - Slot `Date`, do tipo de slot integrado `AMAZON.DATE`.
    - Slot `Time`, do tipo de slot integrado `AMAZON.TIME`.
  - Declarações: a intenção está pré-configurada com as seguintes declarações:
    - "Eu gostaria de marcar uma consulta"
    - "Marcar uma consulta"
    - “Reserve um {AppointmentType}”

    Se o usuário disser qualquer um desses, Amazon Lex determinará que `MakeAppointment` é a intenção e, em seguida, usará os prompts para escolher os dados de slot.
  - Prompts – a intenção está pré-configurada com os seguintes prompts:
    - Prompt do slot `AppointmentType` – "Que tipo de consulta você deseja marcar?"
    - Solicitar a `Date` vaga — “Quando devo agendar seu {AppointmentType}?”
    - Solicitar o `Time` horário — “A que horas você deseja agendar o {AppointmentType}?” and

      "Em que horário em {Date}?"
    - Prompt de confirmação – "{Time} está disponível. Posso prosseguir e marcar sua consulta?"
    - Mensagem de cancelamento – "OK, não marcarei a consulta."

Visão geral da função Lambda Blueprint () lex-make-appointment-python

A função Lambda blueprint (lex-make-appointment-python) é um gancho de código para bots que você cria usando o blueprint do ScheduleAppointment bot.

Esse código de esquema da função do Lambda pode executar tarefas de inicialização/validação e de atendimento.

- A função do Lambda exibe uma conversa dinâmica baseada em um exemplo de disponibilidade para uma consulta odontológica (em aplicativos reais, você pode usar um calendário). Para o dia ou a data que o usuário especificar, o código será configurado da seguinte forma:
  - Se não houver consultas disponíveis, a função do Lambda retornará uma resposta direcionando o Amazon Lex a solicitar outro dia ou data ao usuário (definindo o tipo `dialogAction` como `ElicitSlot`). Para obter mais informações, consulte Formato de resposta.
  - Se houver apenas uma consulta disponível no dia ou data especificada, a função do Lambda sugerirá o horário disponível na resposta e direcionará o Amazon Lex a obter a confirmação do usuário definindo a `dialogAction` na resposta como `ConfirmIntent`. Isso ilustra como você pode melhorar a experiência do usuário sugerindo proativamente o horário disponível para uma consulta.
  - Se houver várias consultas disponíveis, a função do Lambda retornará uma lista de horários disponíveis na resposta ao Amazon Lex. Amazon Lex retorna uma resposta ao cliente com a mensagem da função do Lambda.
- Como o hook de código de atendimento, a função do Lambda retorna um mensagem de resumo indicando que uma consulta está agendada (ou seja, a intenção está atendida).

> **Note**
>
> Neste exemplo, mostramos como usar cartões de resposta. A função do Lambda cria e retorna um cartão de resposta ao Amazon Lex. As listas de cartão de resposta lista dias e horários disponíveis como botões para o usuário escolher. Ao testar o bot usando o cliente fornecido pelo console do Amazon Lex, não é possível ver o cartão de resposta. Para vê-lo, você deve integrar o bot com uma plataforma de mensagens, como Facebook Messenger. Para obter instruções, consulte Integração de um bot do Amazon Lex com o Facebook Messenger. Para obter mais informações sobre cartões de resposta, consulte Gerenciamento de mensagens .

Quando o Amazon Lex invoca a função do Lambda, ela passa os dados do evento como entrada. Um dos campos do evento é `invocationSource`, que a função do Lambda usa para escolher entre uma validação de entrada e uma atividade de atendimento. Para obter mais informações, consulte Formato de eventos de entrada.

Próxima etapa

Etapa 1: criar um bot do Amazon Lex

Etapa 1: criar um bot do Amazon Lex

Nesta seção, você cria um bot do Amazon Lex usando o ScheduleAppointment blueprint, que é fornecido no console do Amazon Lex.

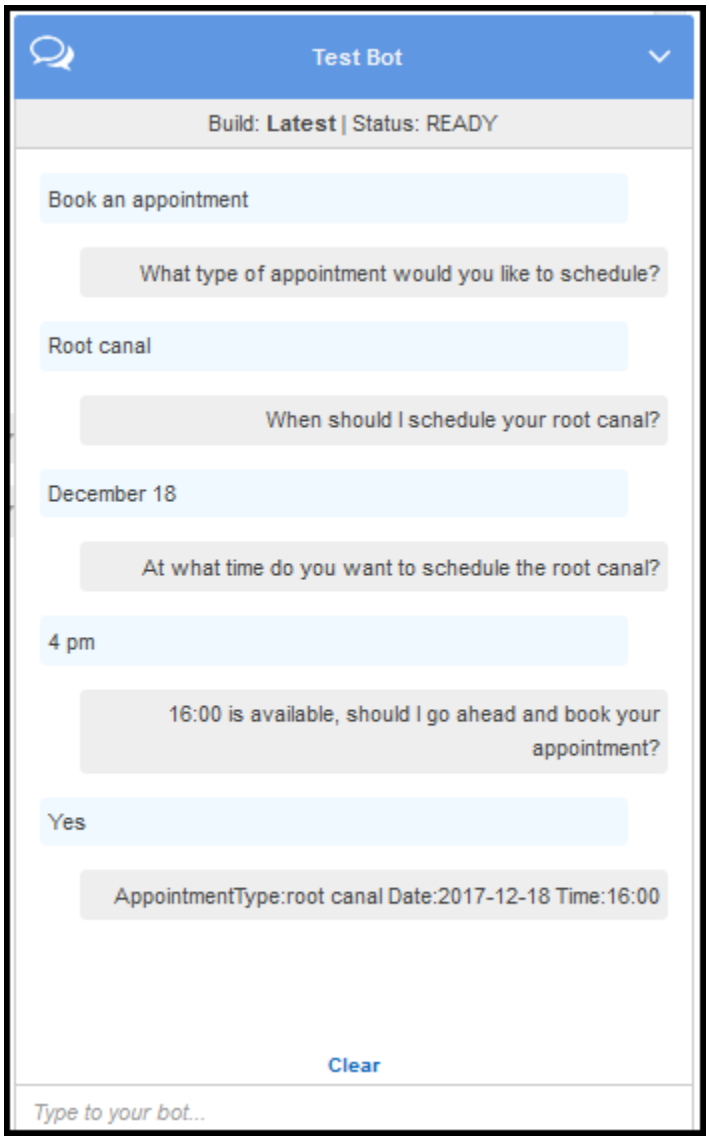
1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na página Bots, selecione Create.
3. Na página Create your Lex bot:
   - Escolha o esquema ScheduleAppointment.
   - Deixe o nome padrão do bot (ScheduleAppointment).
4. Escolha Criar.

   Esta etapa salva e cria o bot. O console envia as seguintes solicitações para Amazon Lex durante o processo de criação:

   - Crie uma nova versão dos tipos de slot (da versão $LATEST). Para obter mais informações sobre tipos de slot neste esquema de bot, consulte Visão geral do Bot Blueprint () ScheduleAppointment.
   - Crie uma versão da intenção `MakeAppointment` (da versão $LATEST). Em alguns casos, o console envia uma solicitação para a operação da API `update` antes de criar uma nova versão.
   - Atualize a versão $LATEST do bot.

     Neste ponto, o Amazon Lex cria um modelo de machine learning para o bot. Quando você testa o bot no console, o console usa a API de runtime para enviar entradas do usuário de volta para o Amazon Lex. Em seguida, o Amazon Lex usa o modelo de machine learning para interpretar a entrada do usuário.

5. O console mostra o ScheduleAppointment bot. Na guia Editor, analise os detalhes da intenção pré-configurada (`MakeAppointment`).
6. Teste o bot na janela de teste. Use a seguinte captura de tela para ter uma conversa de teste com o bot:



Observe o seguinte:

- Na entrada inicial do usuário ("Marcar uma consulta"), o bot infere a intenção (`MakeAppointment`).
- Em seguida, ele usa os prompts configurados para obter dados de slot do usuário.
- O esquema de bot tem a intenção `MakeAppointment` configurada com o seguinte prompt de confirmação:

```
{Time} is available, should I go ahead and book your appointment?
```

Depois que o usuário fornece todos os dados do slot, o Amazon Lex retorna uma resposta ao cliente com um prompt de confirmação como a mensagem. O cliente exibe a mensagem para o usuário:

```
16:00 is available, should I go ahead and book your appointment?
```

Observe que o bot aceita quaisquer data e horário de consulta, pois você não tem nenhum código para inicializar ou validar os dados do usuário. Na próxima seção, adicione uma função do Lambda para fazer isso.

Próxima etapa

Etapa 2: criar uma função do Lambda

Etapa 2: criar uma função do Lambda

Nesta seção, você cria uma função Lambda usando um blueprint (lex-make-appointment-python) fornecido no console Lambda. Você também testa a função do Lambda invocando-a usando dados do evento de exemplo do Amazon Lex fornecido pelo console.

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha Create a Lambda function.
3. Em Selecionar blueprint, digite **`lex`** para encontrar o blueprint e, em seguida, escolha o lex-make-appointment-pythonblueprint.
4. Configure a função do Lambda da seguinte forma.
   - Digite o nome da função do Lambda (`MakeAppointmentCodeHook`).
   - Para a função, escolha Create a new role from template(s) e, em seguida, digite um nome de função.
   - Deixe outros valores padrão.
5. Escolha Criar função.

6. Se você estiver usando uma localidade diferente do inglês (EUA) (en-US), atualize os nomes das intenções conforme descrito em Atualização de um esquema para uma localidade específica.
7. Testar a função do Lambda.
   a. Escolha Actions e, em seguida, escolha Configure test event.
   b. Na lista Sample event template, escolha Lex-Make Appointment (preview). Esse exemplo de evento usa o request/response model, with values set to match a request from your Amazon Lex bot. For information about the Amazon Lex request/response modelo Amazon Lex, consulteUso de funções do Lambda.
   c. Escolha Save and test.
   d. Verifique se a função do Lambda foi executada com êxito. A resposta, neste caso, corresponde ao modelo de resposta do Amazon Lex.

Próxima etapa

Etapa 3: atualizar a intenção - configuração de um hook de código

Etapa 3: atualizar a intenção - configuração de um hook de código

Nesta seção, você atualiza a configuração da intenção `MakeAppointment` para usar a função do Lambda como hook de código para a validação e as atividades de cumprimento.

1. No console do Amazon Lex, selecione o ScheduleAppointment bot. O console mostra a MakeAppointmentintenção. Modifique a configuração de intenção da seguinte forma.

   > **Note**
   > Você pode atualizar apenas as versões $LATEST de qualquer um dos recursos do Amazon Lex incluindo as intenções. Verifique se a versão da intenção está definida como $LATEST. Você ainda não publicou uma versão de seu bot, então a versão ainda deve ser $LATEST no console.

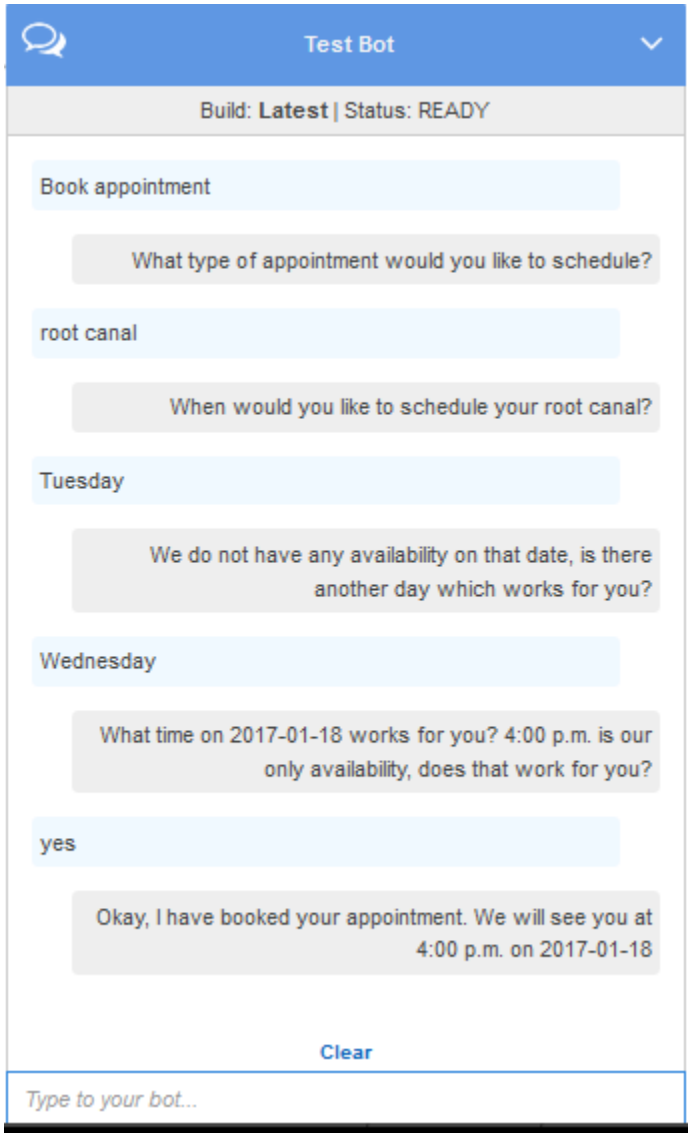
   a. Na seção Options, escolha Initialization and validation code hook e, em seguida, escolha a função do Lambda na lista.

b. Na seção Fulfillment, escolha AWS Lambda function e, em seguida, escolha a função do Lambda na lista.

c. Escolha Goodbye message e digite uma mensagem.

2. Escolha Save e, em seguida, Build.

3. Teste o bot, como na imagem a seguir:



Próxima etapa

Etapa 4: implantar o bot na plataforma do Facebook Messenger

Etapa 4: implantar o bot na plataforma do Facebook Messenger

Na seção anterior, você testou o ScheduleAppointment bot usando o cliente no console do Amazon Lex. No momento, o console Amazon Lex não oferece suporte a cartões de resposta. Para testar os cartões de resposta gerados dinamicamente aos quais o bot oferece suporte, implante o bot na plataforma do Facebook Messenger e teste-o.

Para obter instruções, consulte Integração de um bot do Amazon Lex com o Facebook Messenger.

Próxima etapa

Detalhes do fluxo de informações

Detalhes do fluxo de informações

O esquema de bot `ScheduleAppointment` exibe, principalmente, o uso de cartões de resposta gerados dinamicamente. A função do Lambda neste exercício inclui cartões de resposta em sua resposta ao Amazon Lex. Amazon Lex inclui os cartões de resposta em sua resposta ao cliente. Esta seção explica o seguinte:

- Fluxo de dados entre o cliente e Amazon Lex.

  A seção supõe que o cliente envia solicitações ao Amazon Lex usando a API de runtime `PostText` e exibe os detalhes de solicitação/resposta adequadamente. Para obter mais informações sobre a API de runtime `PostText`, consulte PostText.

  > **Note**
  >
  > Para obter um exemplo do fluxo de informações entre o cliente e Amazon Lex em que o cliente usa a API `PostContent`, consulte Etapa 2a (opcional): revisar os detalhes do fluxo de informações falado (console) .

- Fluxo de dados entre Amazon Lex e a função do Lambda. Para obter mais informações, consulte Evento de entrada de função do Lambda e formato de resposta.

> **Note**
>
> O exemplo supõe que você esteja usando o cliente Facebook Messenger, que não passa atributos de sessão na solicitação para Amazon Lex. Dessa forma, os exemplos de solicitações desta seção mostram `sessionAttributes` vazio. Se você testar o bot usando o cliente fornecido no console Amazon Lex, o cliente incluirá os atributos de sessão.

Esta seção descreve o que acontece após cada entrada do usuário.

1. Usuário: tipos **`Book an appointment`**

   a. O cliente (console) envia a seguinte solicitação PostContent para o Amazon Lex:

   ```
   POST /bot/ScheduleAppointment/alias/$LATEST/
   user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book appointment",
      "sessionAttributes":{}
   }
   ```

   O URI de solicitação e o corpo fornecem informações ao Amazon Lex:

   - URI de solicitação — fornece o nome do bot (ScheduleAppointment), o alias do bot ($LATEST) e o ID do nome do usuário. O `text` final indica que esta é uma solicitação de API `PostText` (não `PostContent`).
   - Corpo da solicitação: inclui a entrada do usuário (`inputText`) e `sessionAttributes` vazio.

   b. A partir de `inputText`, o Amazon Lex detecta a intenção (`MakeAppointment`). O serviço invoca a função do Lambda, que está configurada como um hook de código, para executar a inicialização e a validação passando o seguinte evento. Para obter detalhes, consulte Formato de eventos de entrada.

   ```
   {
       "currentIntent": {
           "slots": {
   ```

```
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "bijt6rovckwecnzesbthrr1d7lv3ja3n",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Além das informações enviadas pelo cliente, o Amazon Lex também inclui os seguintes dados:

- `currentIntent` - Fornece informações de intenção atuais.
- `invocationSource`: indica o objetivo da invocação da função do Lambda. Nesse caso, o objetivo é executar os dados de inicialização e validação do usuário. (Amazon Lex sabe que o usuário não forneceu todos os dados de slot para cumprir a intenção.)
- `messageVersion`: atualmente, o Amazon Lex oferece suporte apenas à versão 1.0.

c. No momento, todos os valores de slot são nulos (não há nada para validar). A função do Lambda retorna a seguinte resposta para o Amazon Lex direcionando o serviço a obter informações para o slot `AppointmentType`. Para obter mais informações sobre o formato de resposta, consulte Formato de resposta.

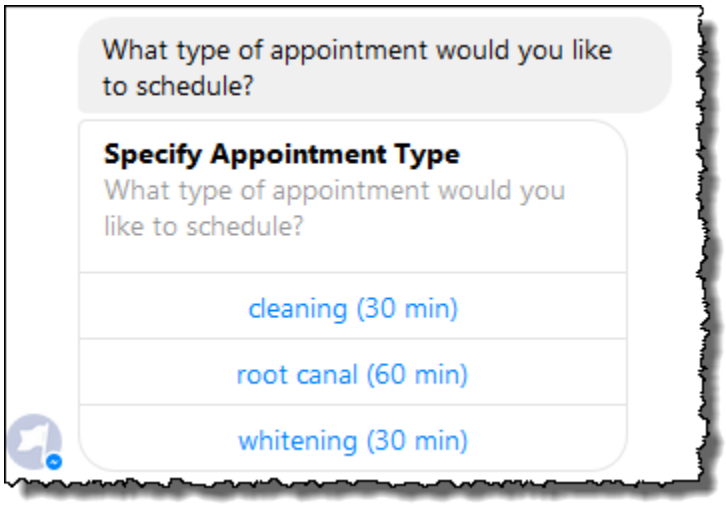
```
{
    "dialogAction": {
        "slotToElicit": "AppointmentType",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
```

```
                            "text": "cleaning (30 min)",
                            "value": "cleaning"
                        },
                        {
                            "text": "root canal (60 min)",
                            "value": "root canal"
                        },
                        {
                            "text": "whitening (30 min)",
                            "value": "whitening"
                        }
                    ],
                    "subTitle": "What type of appointment would you like to
 schedule?",
                    "title": "Specify Appointment Type"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

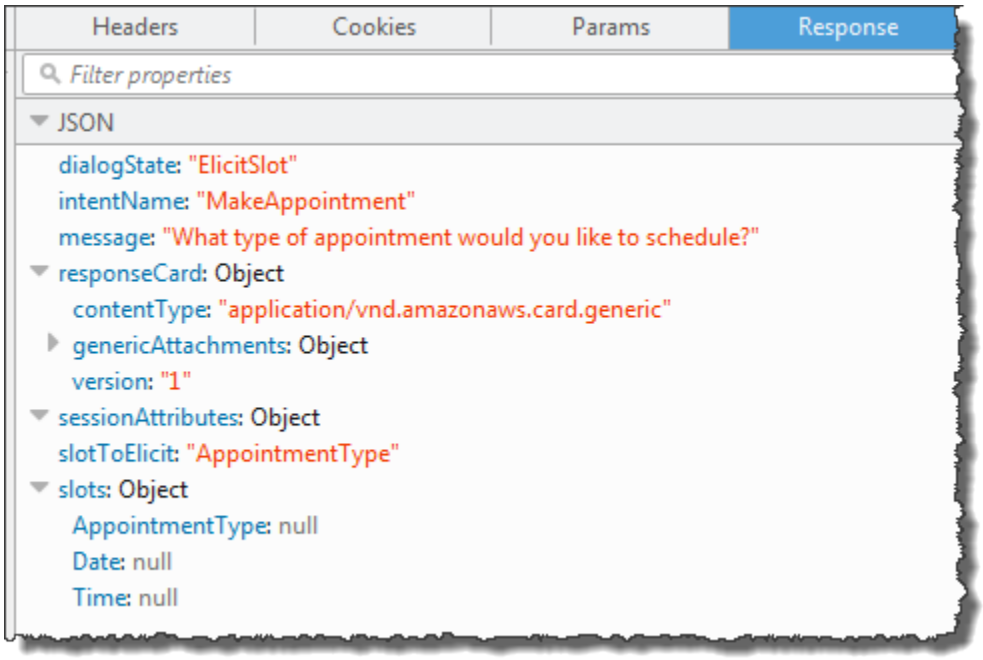
A resposta inclui os campos `dialogAction` e `sessionAttributes`. Dentre outras coisas, o campo `dialogAction` retorna os seguintes campos:

- `type`: ao definir esse campo como `ElicitSlot`, a função do Lambda direciona o Amazon Lex a escolher o valor para o slot especificado no campo `slotToElicit`. A função do Lambda também fornece uma `message` para transmitir ao usuário.
- `responseCard`: identifica uma lista de valores possíveis para o slot `AppointmentType`. Um cliente que oferece suporte a cartões de resposta (por exemplo, Facebook

Messenger) exibe um cartão de resposta para permitir que o usuário escolha um tipo de consulta da seguinte forma:



d. Como indicado pelo `dialogAction.type` na resposta da função do Lambda, Amazon Lex envia a seguinte resposta de volta para o cliente:



O cliente lê a resposta e, em seguida, exibe a mensagem: "Que tipo de consulta você deseja marcar?" e o cartão de resposta (se o cliente oferece suporte a cartões de resposta).

2. Usuário: dependendo do cliente, o usuário tem duas opções:

- Se o cartão de resposta for exibido, escolha root canal (60 min) (canal de raiz (60 min)) ou digite **`root canal`**.

- Se o cliente não oferecer suporte a cartões de resposta, digite **`root canal`**.

a. O cliente envia a seguinte solicitação `PostText` ao Amazon Lex (quebras de linha foram adicionadas para legibilidade):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "root canal",
    "sessionAttributes": {}
}
```

b. O Amazon Lex invoca a função do Lambda para validação dos dados do usuário enviando o seguinte evento como parâmetro:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "bijt6rovckwecnzesbthrr1d7lv3ja3n",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Nos dados de evento, observe o seguinte:

- `invocationSource` continua sendo `DialogCodeHook`. Nesta etapa, estamos apenas validando os dados do usuário.
- O Amazon Lex define o campo `AppointmentType` no slot `currentIntent.slots` como `root canal`.
- Amazon Lex simplesmente passa o campo `sessionAttributes` entre o cliente e a função do Lambda.

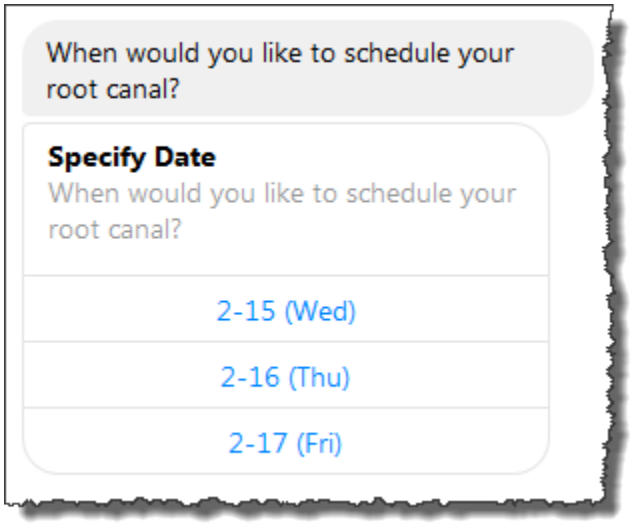
c. A função do Lambda valida a entrada do usuário e retorna a seguinte resposta ao Amazon Lex, direcionando o serviço a escolher um valor para a data da consulta.

```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
                            "value": "Wednesday, February 15, 2017"
                        },
                        {
                            "text": "2-16 (Thu)",
                            "value": "Thursday, February 16, 2017"
                        },
                        {
                            "text": "2-17 (Fri)",
                            "value": "Friday, February 17, 2017"
                        },
                        {
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "subTitle": "When would you like to schedule your root
 canal?",
```

```
                    "title": "Specify Date"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "When would you like to schedule your root canal?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

Novamente, a resposta inclui os campos `dialogAction` e `sessionAttributes`. Dentre outras coisas, o campo `dialogAction` retorna os seguintes campos:

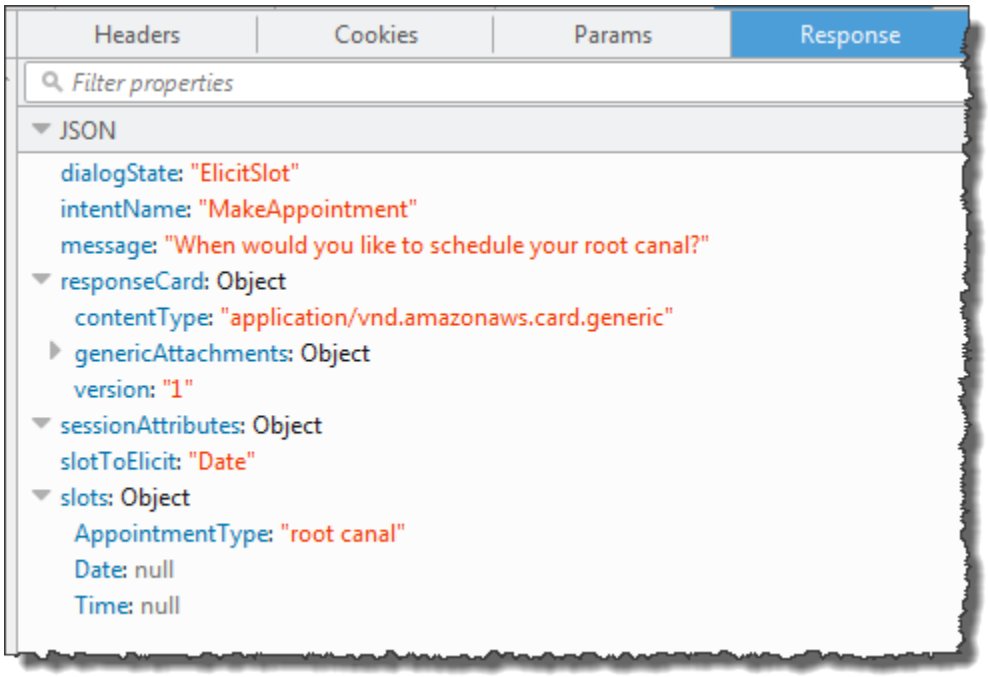
- `type`: ao definir esse campo como `ElicitSlot`, a função do Lambda direciona o Amazon Lex a escolher o valor para o slot especificado no campo `slotToElicit`. A função do Lambda também fornece uma `message` para transmitir ao usuário.
- `responseCard`: identifica uma lista de valores possíveis para o slot `Date`. Um cliente que oferece suporte a cartões de resposta (por exemplo, Facebook Messenger) exibe um cartão de resposta que permite que o usuário escolha uma data de consulta:



Apesar de a função do Lambda ter retornado cinco datas, o cliente (Facebook Messenger) tem um limite de três botões por cartão de resposta. Portanto, você verá apenas os primeiros três valores na captura de tela.

Essas datas são codificadas na função do Lambda. Em uma aplicação de produção, você pode usar um calendário para obter datas disponíveis em tempo real. Como as datas são dinâmicas, você deve gerar o cartão de resposta dinamicamente na função do Lambda.

d. O Amazon Lex observa o `dialogAction.type` e retorna a resposta a seguir ao cliente que inclui informações da reposta da função do Lambda.



O cliente exibe a mensagem: When would you like to schedule your root canal? (Quando você deseja agendar seu canal de raiz?) e o cartão de resposta (se o cliente oferecer suporte a cartões de resposta).

3. Usuário: tipos **`Thursday`**

   a. O cliente envia a seguinte solicitação `PostText` ao Amazon Lex (quebras de linha foram adicionadas para legibilidade):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   b. O Amazon Lex invoca a função do Lambda para validação dos dados do usuário enviando o seguinte evento como parâmetro:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-16",
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Nos dados de evento, observe o seguinte:

- `invocationSource` continua sendo `DialogCodeHook`. Nesta etapa, estamos apenas validando os dados do usuário.
- O Amazon Lex define o campo `Date` no slot `currentIntent.slots` como `2017-02-16`.
- Amazon Lex simplesmente passa o campo `sessionAttributes` entre o cliente e a função do Lambda.

c. A função do Lambda valida a entrada do usuário. Desta vez, a função do Lambda determina que não há consultas disponíveis na data especificada. Ela retorna a seguinte resposta ao Amazon Lex direcionando o serviço a escolher novamente um valor para a data da consulta.

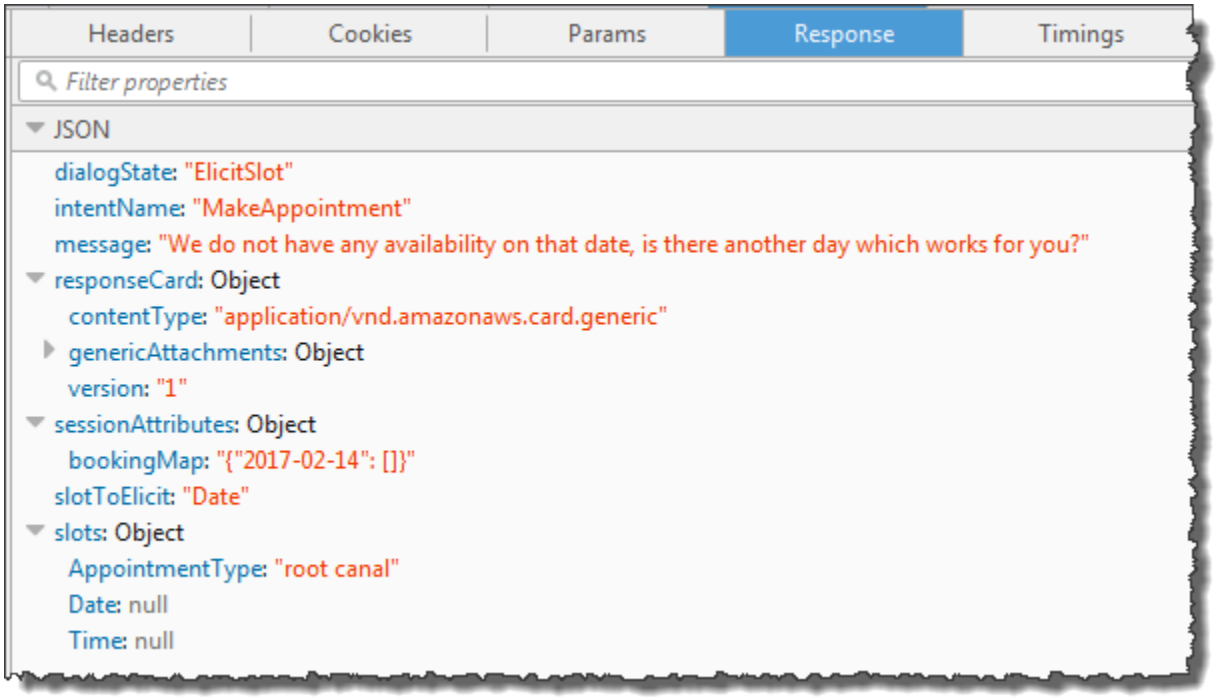
```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
                            "value": "Wednesday, February 15, 2017"
                        },
                        {
                            "text": "2-17 (Fri)",
                            "value": "Friday, February 17, 2017"
                        },
                        {
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "subTitle": "When would you like to schedule your root
 canal?",
```

```
                    "title": "Specify Date"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "We do not have any availability on that date, is there
 another day which works for you?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {
     "bookingMap": "{\"2017-02-16\": []}"
   }
}
```

Novamente, a resposta inclui os campos `dialogAction` e `sessionAttributes`. Dentre outras coisas, o `dialogAction` retorna os seguintes campos:

- campo `dialogAction`:
  - `type` - A função do Lambda define esse valor como `ElicitSlot` e redefine o campo `slotToElicit` como `Date`. A função do Lambda também fornece um `message` adequado para transmitir ao usuário.
  - `responseCard` - Retorna uma lista de valores para o slot `Date`.
- `sessionAttributes`: dessa vez, a função do Lambda inclui o atributo de sessão `bookingMap`. Seu valor é a data solicitada da consulta e das consultas disponíveis (um objeto vazio indica que não há consultas disponíveis).

d. O Amazon Lex observa o `dialogAction.type` e retorna a resposta a seguir ao cliente que inclui informações da reposta da função do Lambda.



O cliente exibe a mensagem: We do not have any availability on that date, is there another day which works for you? (Não temos nenhuma disponibilidade nessa data, há outro dia que funcione para você?) e o cartão de resposta (se o cliente oferecer suporte a cartões de resposta).

4. Usuário: dependendo do cliente, o usuário tem duas opções:

   - Se o cartão de resposta for exibido, escolha 2-15 (Wed) (15/2 (quarta)) ou digite **`Wednesday`**.
   - Se o cliente não oferecer suporte a cartões de resposta, digite **`Wednesday`**.

   a. O cliente envia a seguinte solicitação `PostText` para o Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

> **Note**
>
> O cliente Facebook Messenger não define atributos de sessão. Se desejar manter estados de sessão entre as solicitações, você deverá fazer isso na função do Lambda. Em uma aplicação real, talvez você precise manter esses atributos de sessão em banco de dados de back-end.

b. O Amazon Lex invoca a função do Lambda para validação dos dados do usuário enviando o seguinte evento como parâmetro:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {
    }
}
```

Amazon Lex atualizou `currentIntent.slots` definindo o slot `Date` como `2017-02-15`.

c. A função do Lambda valida a entrada do usuário e retorna a seguinte resposta ao Amazon Lex, direcionando-o a escolher o valor para o horário da consulta.

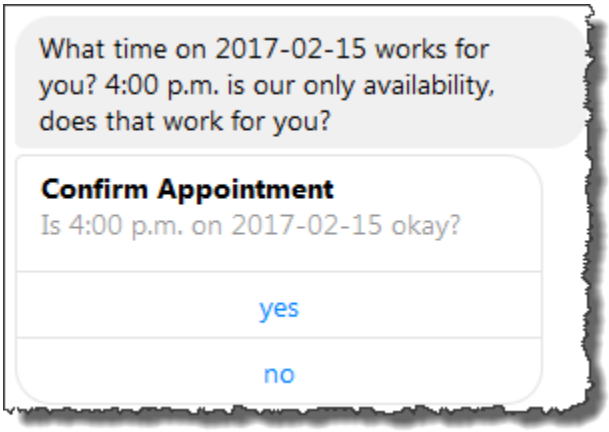
```
{
    "dialogAction": {
        "slots": {
```

```
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": "16:00"
        },
        "message": {
            "content": "What time on 2017-02-15 works for you? 4:00 p.m. is our
 only availability, does that work for you?",
            "contentType": "PlainText"
        },
        "type": "ConfirmIntent",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "yes",
                            "value": "yes"
                        },
                        {
                            "text": "no",
                            "value": "no"
                        }
                    ],
                    "subTitle": "Is 4:00 p.m. on 2017-02-15 okay?",
                    "title": "Confirm Appointment"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        }
    },
    "sessionAttributes": {
        "bookingMap": "{\"2017-02-15\": [\"10:00\", \"16:00\", \"16:30\"]}"
    }
}
```

Novamente, a resposta inclui os campos `dialogAction` e `sessionAttributes`. Dentre outras coisas, o `dialogAction` retorna os seguintes campos:

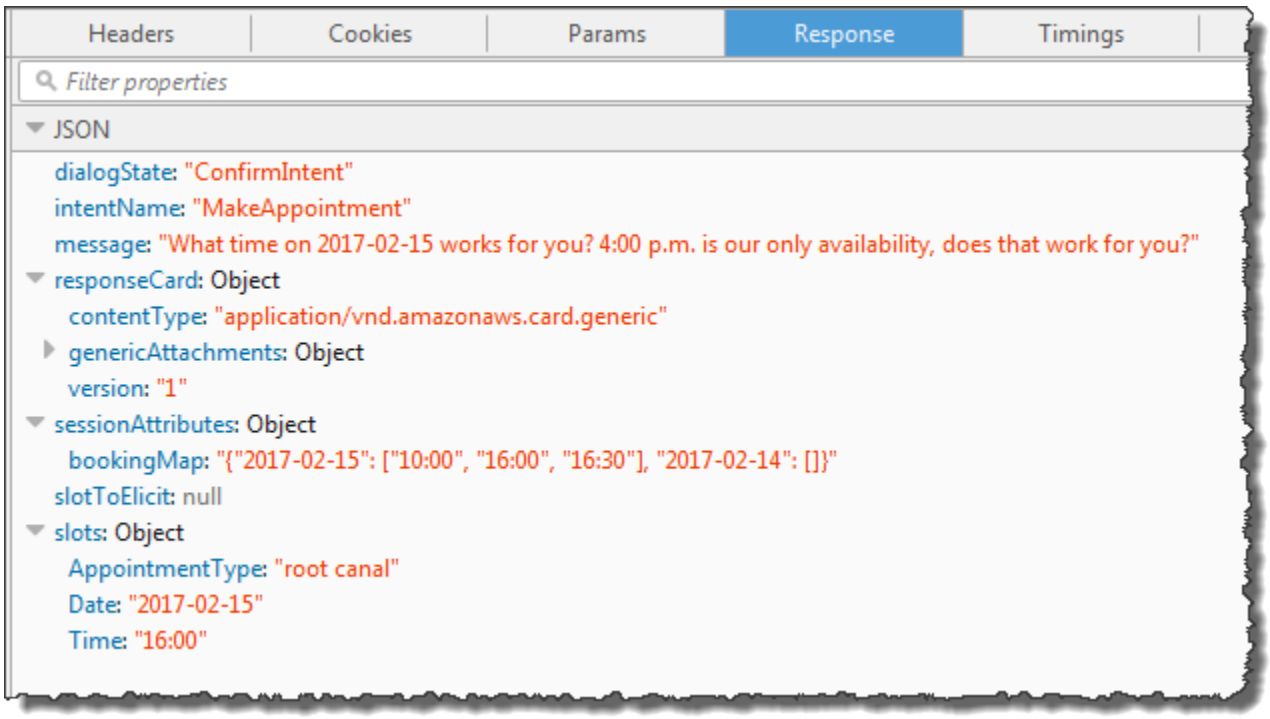
- campo `dialogAction`:

- `type`: a função do `Lambda` define esse valor como `ConfirmIntent`, direcionando o Amazon Lex a obter a confirmação do usuário do horário da consulta sugerido na `message`.
- `responseCard` - Retorna uma lista de valores sim/não para o usuário escolher. Se o cliente for compatível com cartões de resposta, ele exibirá o cartão de resposta, como mostrado no exemplo a seguir:



- `sessionAttributes`: a função do Lambda define o atributo de sessão `bookingMap` com o valor definido como a data da consulta e as consultas disponíveis naquela data. Neste exemplo, são consultas de 30 minutos. Para um canal de raiz que leva uma hora, apenas 16h pode ser marcada.

d. Como indicado no `dialogAction.type` na resposta da função do Lambda, o Amazon Lex retorna a seguinte resposta ao cliente:



O cliente exibe a mensagem: Qual é o melhor horário para você em 15/02/2017? Nossa única disponibilidade é às 16h. Tudo bem para você?

5. Usuário: selecione **`yes`**.

   Amazon Lex invoca a função do Lambda com os seguintes dados de evento: Como o usuário respondeu **`yes`**, o Amazon Lex define `confirmationStatus` como `Confirmed` e define o campo `Time` no `currentIntent.slots` como `4 p.m`.

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": "16:00"
        },
        "name": "MakeAppointment",
        "confirmationStatus": "Confirmed"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
```

```
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "FulfillmentCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {
   }
}
```

Como o `confirmationStatus` é confirmado, a função do Lambda processa a intenção (marca uma consulta odontológica) e retorna a seguinte resposta ao Amazon Lex:

```
{
    "dialogAction": {
        "message": {
            "content": "Okay, I have booked your appointment.  We will see you at
 4:00 p.m. on 2017-02-15",
            "contentType": "PlainText"
        },
        "type": "Close",
        "fulfillmentState": "Fulfilled"
    },
    "sessionAttributes": {
        "formattedTime": "4:00 p.m.",
        "bookingMap": "{\"2017-02-15\": [\"10:00\"]}"
    }
}
```

Observe o seguinte:

- A função do Lambda atualizou o `sessionAttributes`.
- `dialogAction.type` é definido como `Close`, que direciona Amazon Lex a não esperar uma resposta do usuário.
- `dialogAction.fulfillmentState` é definido como `Fulfilled`, indicando que a intenção foi cumprida com êxito.

O cliente exibe a mensagem: Ok, já agendei sua consulta. Nos vemos às 16h do dia 15/02/2017.

Reservar uma viagem

Este exemplo mostra como criar um bot configurado para oferecer suporte a várias intenções. O exemplo também ilustra como é possível usar atributos de sessão para o compartilhamento de informações entre intenções. Depois de criar o bot, você usa um cliente de teste no console do Amazon Lex para testar o bot (BookTrip). O cliente usa a operação da API runtime do PostText para enviar solicitações ao Amazon Lex para cada entrada do usuário.

O BookTrip bot neste exemplo está configurado com duas intenções (BookHotel e BookCar). Por exemplo, suponha que um usuário reserve um hotel primeiro. Durante a interação, o usuário fornece informações como datas de check-in, local e número de diárias. Depois de a intenção ser cumprida, o cliente pode manter essas informações usando atributos de sessão. Para mais informações sobre atributos de sessão, consulte PostText.

Agora, suponha que o usuário prossiga para reservar um carro. Usando informações fornecidas pelo usuário na BookHotel intenção anterior (ou seja, cidade de destino e datas de check-in e check-out), o cohook (função Lambda) que você configurou para inicializar e validar a intenção inicializa os dados BookCar do slot da intenção (ou seja, destino, cidade de coleta, data de coleta e data de devolução). BookCar Isso mostra como o compartilhamento de informações entre intenções permite que você crie bots que podem participar de conversas dinâmicas com o usuário.

Neste exemplo, usamos os seguintes atributos de sessão. Somente o cliente e a função do Lambda podem definir e atualizar os atributos de sessão. Amazon Lex apenas os passa entre o cliente e a função do Lambda. O Amazon Lex não mantem nem modifica atributos de sessão.

- `currentReservation`: contém dados de slot para uma reserva em andamento e outras informações relevantes. A seguir, veja um exemplo de solicitação do cliente para o Amazon Lex. Ele mostra o atributo de sessão `currentReservation` no corpo da solicitação.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Chicago",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Moscow\",
                              \"RoomType\":null,
```

```
                                \"CheckInDate\":null,
                                \"Nights\":null}"
    }
}
```

- `lastConfirmedReservation`: contém informações semelhantes para uma intenção anterior, se houver. Por exemplo, se o usuário reservou um hotel e está em processo de reserva de um carro, esse atributo de sessão armazena os dados do slot para a BookHotel intenção anterior.

- `confirmationContext`: a função do Lambda define isso como `AutoPopulate` quando preenche automaticamente alguns dos dados de slot com base nos dados de slot da reserva anterior (se houver). Isso permite o compartilhamento de informações entre intenções. Por exemplo, se o usuário tiver reservado um hotel anteriormente e agora desejar reservar um carro, o Amazon Lex poderá solicitar que o usuário confirme (ou negue) se o carro está sendo reservado para a mesma cidade e as mesmas datas que a reserva de hotel

Neste exercício, você usa esquemas para criar um bot do Amazon Lex e uma função do Lambda. Para obter mais informações sobre esquemas, consulte Amazon Lex e esquemas AWS Lambda.

Próxima etapa

Etapa 1: revisão dos esquemas usados neste exercício

Etapa 1: revisão dos esquemas usados neste exercício

Tópicos

- Visão geral do Bot Blueprint () BookTrip
- Visão geral da função Lambda Blueprint () lex-book-trip-python

Visão geral do Bot Blueprint () BookTrip

O esquema (BookTrip) que você usa para criar um bot fornece as seguintes pré-configurações:

- Tipos de slot: dois tipos de slot personalizados:

- `RoomTypes` com valores de enumeração: `king`, `queen` e `deluxe`, para uso na intenção `BookHotel`.
- `CarTypes` com valores de enumeração: `economy`, `standard`, `midsize`, `full size`, `luxury` e `minivan`, para uso na intenção `BookCar`.

- Intenção 1 (BookHotel) — É pré-configurado da seguinte forma:
  - Slots pré-configurados
    - `RoomType`, do tipo de slot personalizado `RoomTypes`
    - `Location`, do tipo de slot integrado `AMAZON.US_CITY`
    - `CheckInDate`, do tipo de slot integrado `AMAZON.DATE`
    - `Nights`, do tipo de slot integrado `AMAZON.NUMBER`
  - Declarações pré-configuradas
    - "Reservar um hotel"
    - "Eu gostaria de fazer reservas de hotéis"
    - "Reservar uma estadia de {Nights} em {Location}"

    Se o usuário disser qualquer um desses, o Amazon Lex determinará que `BookHotel` é a intenção e, em seguida, solicitará ao usuário dados de slot.
  - Prompts pré-configurados
    - Prompt do slot `Location`: "Em que cidade você se hospedará?"
    - Prompt do slot `CheckInDate`: "Em que dia você deseja fazer check-in?"
    - Prompt do slot `Nights`: "Por quantas diárias você se hospedará?"
    - Prompt do slot `RoomType`: "Que tipo de quarto você deseja: queen, king ou deluxe?"
    - Declaração de confirmação — “Ok, eu tenho você pronto para uma estadia de {noites} em {Location} a partir de {CheckInDate}. Posso fazer a reserva?"
    - Negação: "OK, cancelei sua reserva em andamento."

- Intenção 2 (BookCar) — É pré-configurado da seguinte forma:
  - Slots pré-configurados
    - `PickUpCity`, do tipo integrado `AMAZON.US_CITY`

    - `PickUpDate`, do tipo integrado `AMAZON.DATE`

  - `ReturnDate`, do tipo integrado `AMAZON.DATE`
  - `DriverAge`, do tipo integrado `AMAZON.NUMBER`
  - `CarType`, do tipo personalizado `CarTypes`
- Declarações pré-configuradas
  - "Reservar um carro"
  - "Reservar um carro"
  - "Fazer uma reserva de carro"

  Se o usuário proferir alguma dessas palavras, o Amazon Lex determina qual BookCar é a intenção e, em seguida, solicita que o usuário forneça os dados do slot.
- Prompts pré-configurados
  - Prompt do slot `PickUpCity`: "Em que cidade você precisa alugar um carro?"
  - Prompt do slot `PickUpDate`: "Em que dia você deseja iniciar seu aluguel?"
  - Prompt do slot `ReturnDate`: "Em que dia você deseja devolver o carro?"
  - Prompt do slot `DriverAge` – "Quantos anos tem o motorista deste aluguel de carro?"
  - Prompt do slot `CarType`: "Que tipo de carro você deseja alugar?" Nossas opções mais populares são econômico, médio e luxo"
  - Declaração de confirmação — “Ok, eu tenho você pronto para alugar {CarType} em {PickUpCity} de {PickUpDate} a {ReturnDate}. Devo fazer a reserva?"
  - Negação: "OK, cancelei sua reserva em andamento."

Visão geral da função Lambda Blueprint () lex-book-trip-python

Além do blueprint do bot, AWS Lambda fornece um blueprint (lex-book-trip-python) que você pode usar como um gancho de código com o blueprint do bot. Para obter uma lista de esquemas de bot e esquemas da função do Lambda correspondentes, consulte Amazon Lex e esquemas AWS Lambda.

Ao criar um bot usando o BookTrip blueprint, você atualiza a configuração das intenções (BookCar e BookHotel) adicionando essa função Lambda como um gancho de código para inicialização/validação da entrada de dados do usuário e cumprimento das intenções.

O código de função do Lambda fornecido demonstra uma conversa dinâmica usando informações conhecidas anteriormente (mantidas nos atributos de sessão) sobre um usuário para inicializar os

valores de slot para uma intenção. Para obter mais informações, consulte Gerenciar contexto da conversa.

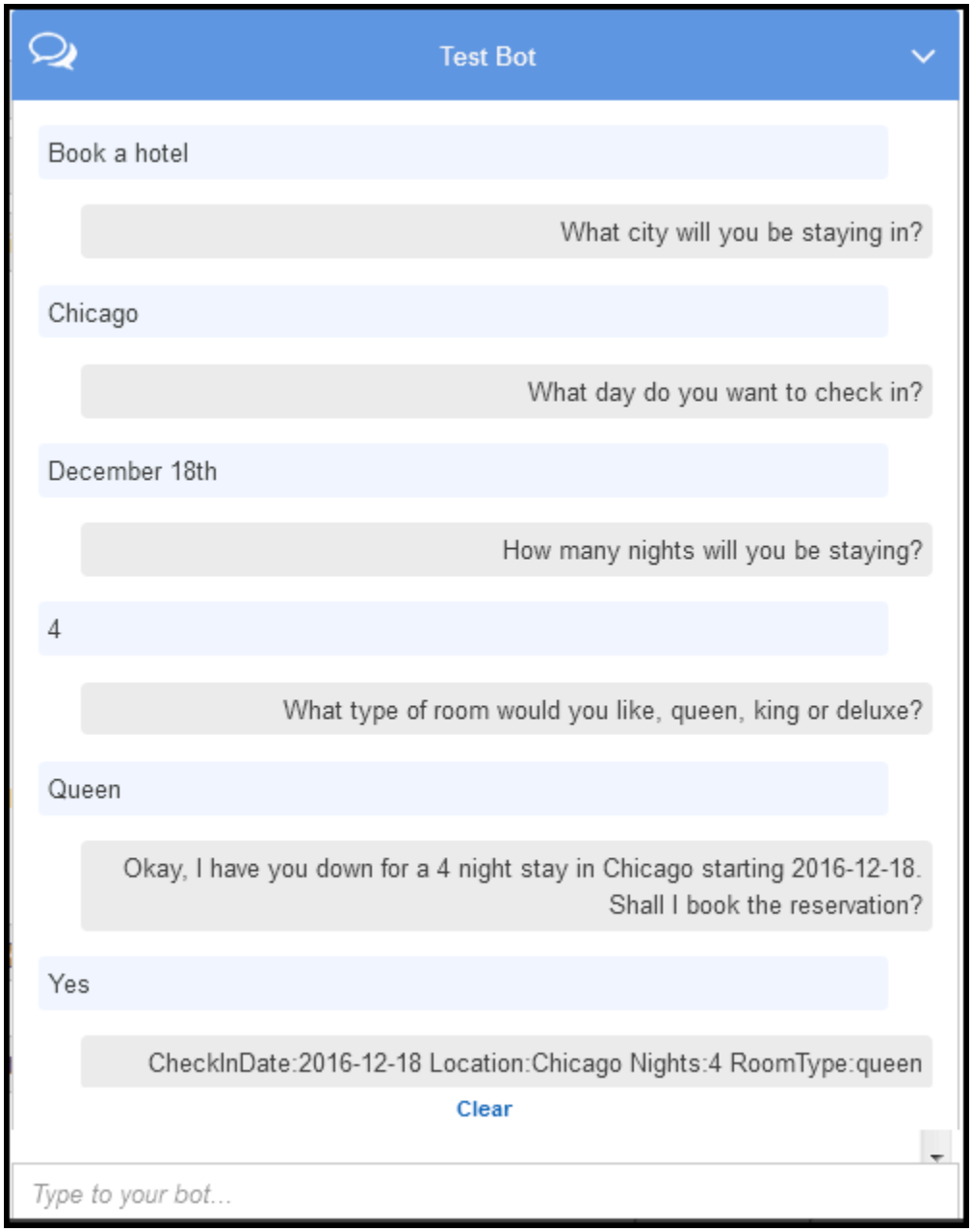
Próxima etapa

Etapa 2: criar um bot do Amazon Lex

Etapa 2: criar um bot do Amazon Lex

Nesta seção, você cria um bot do Amazon Lex (BookTrip).

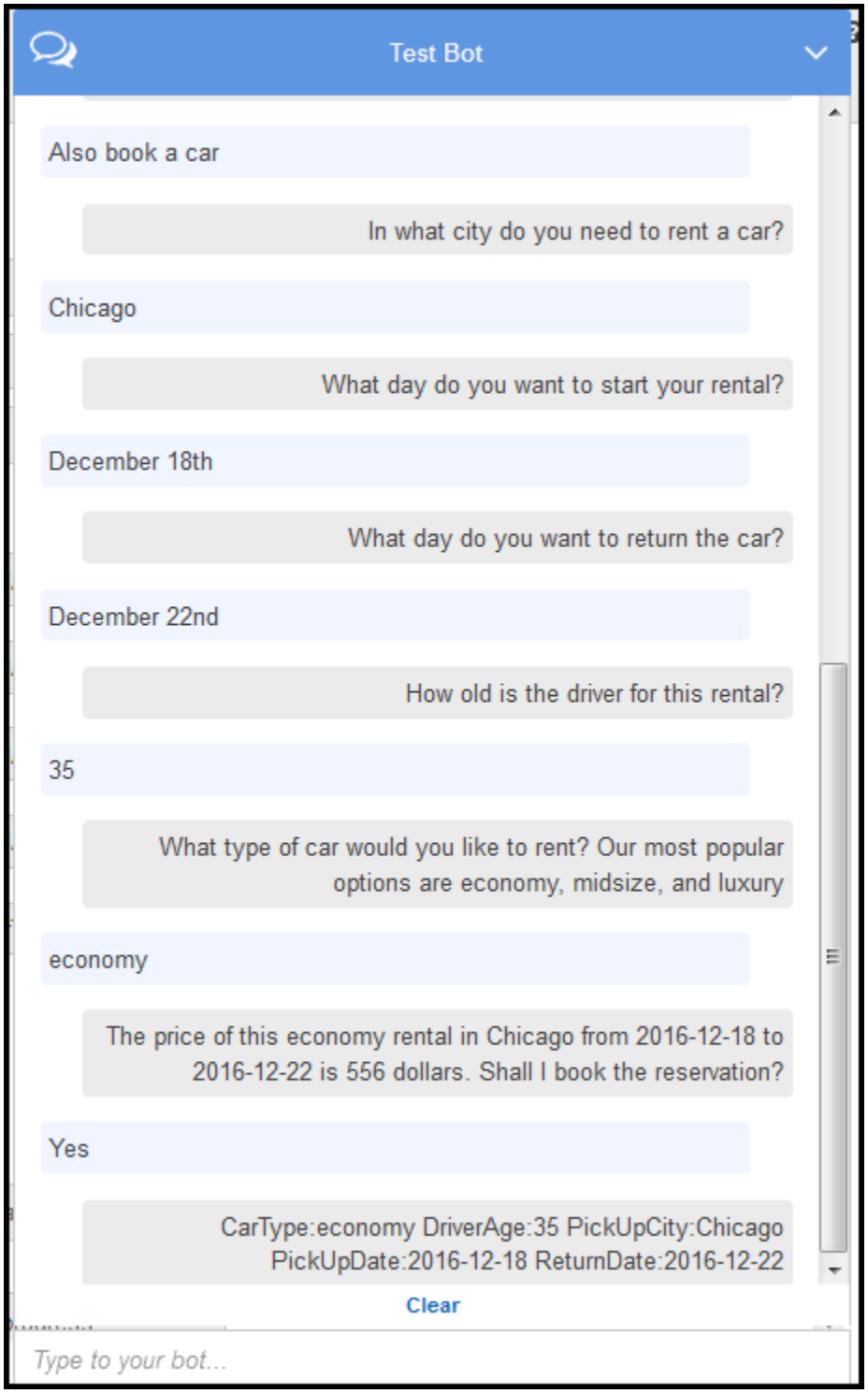
1. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
2. Na página Bots, selecione Create.
3. Na página Create your Lex bot,
   - Escolha o esquema BookTrip.
   - Deixe o nome padrão do bot (BookTrip).
4. Escolha Criar. O console envia uma série de solicitações ao Amazon Lex para criar o bot. Observe o seguinte:
5. O console mostra o BookTrip bot. Na guia Editor, revise os detalhes das intenções pré-configuradas (BookCar e BookHotel).
6. Teste o bot na janela de teste. Use o seguinte para ter uma conversa de teste com o bot:



A partir da entrada inicial do usuário (“Reservar um hotel”), o Amazon Lex deduz a intenção ()BookHotel. Em seguida, o bot usa os prompts pré-configurados nessa intenção para escolher os dados de slot do usuário. Depois de o usuário fornecer todos os dados de slot, o Amazon Lex retorna uma resposta ao cliente com uma mensagem que inclui todas as entradas do usuário como uma mensagem. O cliente exibe a mensagem na resposta, conforme exibido.

```
CheckInDate:2016-12-18 Location:Chicago Nights:5 RoomType:queen
```

Agora, continue a conversa e tente reservar um carro.



Observe que,

- Não há validação de dados do usuário dessa vez. Por exemplo, você pode fornecer qualquer cidade para reservar um hotel.

- Você está fornecendo algumas das mesmas informações novamente (destino, cidade de retirada, data de retirada e data de devolução) para reservar um carro. Em uma conversa dinâmica, seu bot deve inicializar algumas dessas informações com base na entrada anterior que o usuário forneceu para reservar o hotel.

Na próxima seção, você criará uma função do Lambda para executar a validação dos dados do usuário e a inicialização usando o compartilhamento de informações entre intenções por meio de atributos de sessão. Em seguida, use a configuração de intenção, adicionando a função do Lambda como hook de código para executar a inicialização/validação de entradas do usuário e cumprir intenções.

Próxima etapa

Etapa 3: criar uma função do Lambda

Etapa 3: criar uma função do Lambda

Nesta seção, você cria uma função Lambda usando um blueprint (lex-book-trip-python) fornecido no console. AWS Lambda Você também testa a função do Lambda ao invocá-la usando dados de eventos de exemplo fornecidos pelo console.

Essa função do Lambda é escrita em Python.

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função.
3. Selecione Use o esquema. Digite **`lex`** para localizar o esquema e escolha o esquema `lex-book-trip-python`.
4. Escolha Configurar a função do Lambda da seguinte forma.
   - Digite o nome de uma função do Lambda (`BookTripCodeHook`).
   - Para a função, escolha Create a new role from template(s) e, em seguida, digite um nome de função.
   - Deixe os outros valores padrão.
5. Escolha a opção Criar função.

6. Se você estiver usando uma localidade diferente do inglês (EUA) (en-US), atualize os nomes das intenções conforme descrito em Atualização de um esquema para uma localidade específica.
7. Testar a função do Lambda. Invoque a função do Lambda duas vezes usando dados de exemplo para reservar um carro e um hotel.

    a. Escolha Configure test event no menu suspenso Select a test event.
    b. Escolha Amazon Lex Book Hotel na lista Sample event template.

    Esse evento de exemplo corresponde ao modelo de solicitação/resposta do Amazon Lex. Para obter mais informações, consulte Uso de funções do Lambda.

    c. Escolha Save and test.
    d. Verifique se a função do Lambda foi executada com êxito. A resposta, neste caso, corresponde ao modelo de resposta do Amazon Lex.
    e. Repita a etapa. Desta vez, você escolhe Amazon Lex Book Car na lista Sample event template (Modelo de evento de exemplo). A função do Lambda processa a reserva do carro.

Próxima etapa

Etapa 4: adicionar a função do Lambda como hook de código

Etapa 4: adicionar a função do Lambda como hook de código

Nesta seção, você atualiza as configurações do BookCar e das BookHotel intenções adicionando a função Lambda como um gancho de código para atividades de inicialização/validação e atendimento. Verifique se você escolheu a versão $LATEST das intenções, pois só é possível atualizar a versão $LATEST de seus recursos do Amazon Lex.

1. No console do Amazon Lex, escolha o BookTripbot.
2. Na guia Editor, escolha a BookHotelintenção. Atualize a configuração de intenção da seguinte forma:

    a. Verifique se a versão da intenção (ao lado do nome da intenção) é $LATEST.
    b. Adicione a função do Lambda como hook de código de inicialização e validação da seguinte forma:

- Em Opções, escolha Initialization and validation code hook.
- Escolha sua função do Lambda na lista.

c. Adicione a função do Lambda como hook de código de cumprimento da seguinte forma:

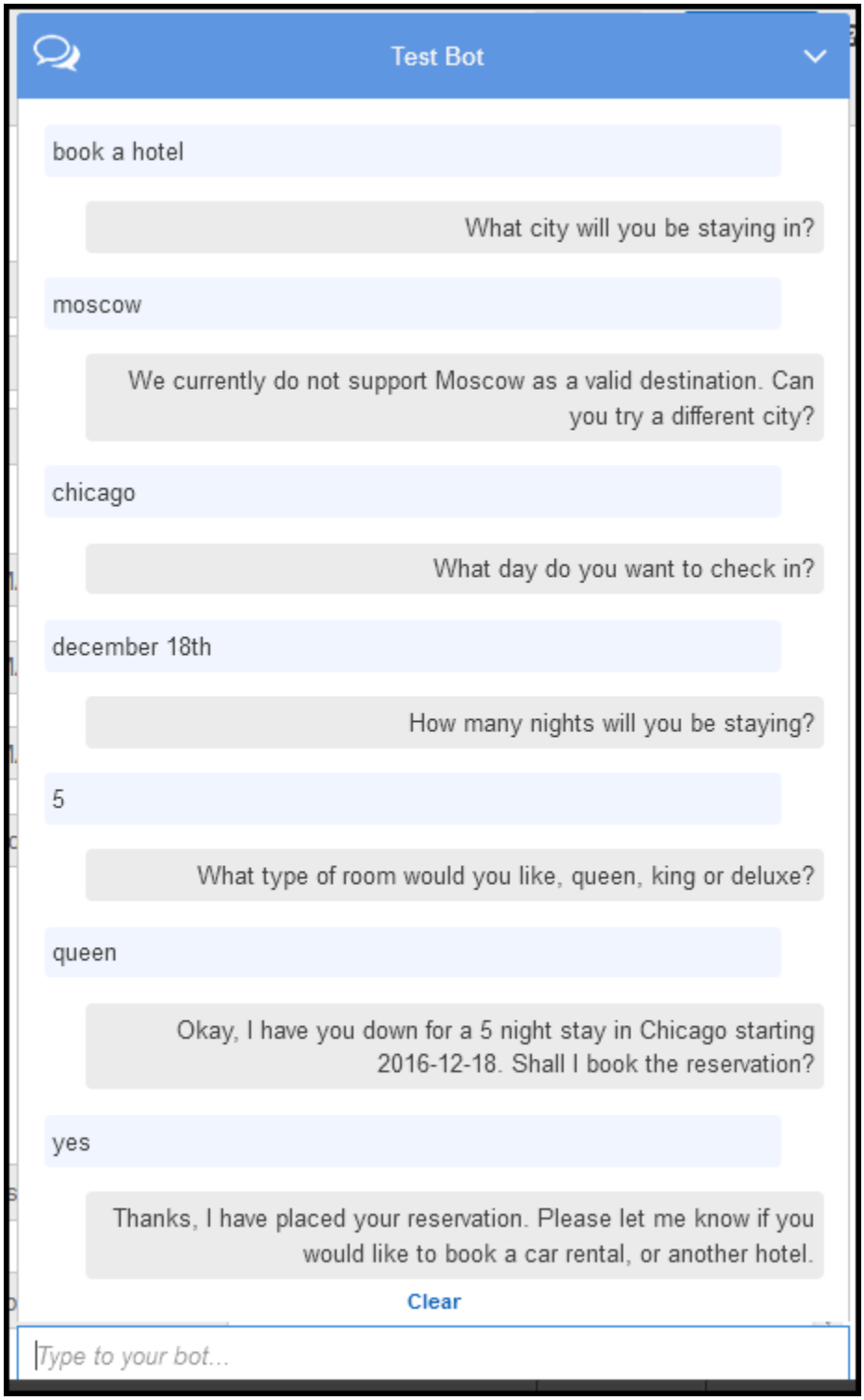
- Em Fulfillment, escolha AWS Lambda function.
- Escolha sua função do Lambda na lista.
- Escolha Goodbye message e digite uma mensagem.

d. Escolha Salvar.

3. Na guia Editor, escolha a BookCar intenção. Siga a etapa anterior para adicionar sua função do Lambda como hook de código de validação e cumprimento.

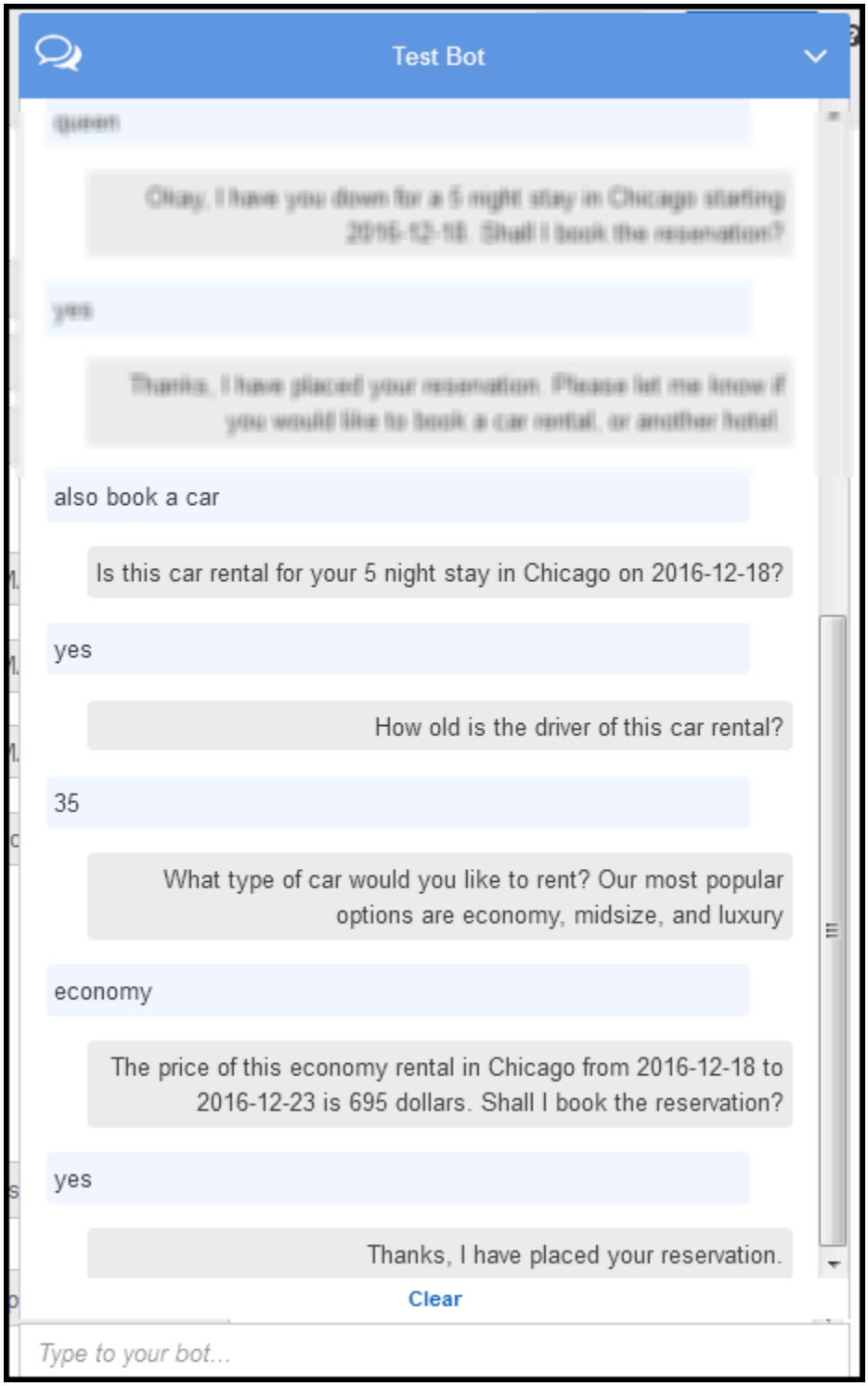
4. Escolha Criar. O console envia uma série de solicitações ao Amazon Lex para salvar as configurações.

5. Teste o bot. Agora que você tem uma função do Lambda executando a inicialização, a validação dos dados do usuário e o atendimento, verá a diferença na interação do usuário.



Para obter mais informações sobre o fluxo de dados do cliente (console) para o Amazon Lex e do Amazon Lex para a função do Lambda, consulte Fluxo de dados: intenção Book Hotel.

6. Continue a conversa e reserve um carro conforme mostrado a seguir:



Quando você escolhe reservar um carro, o cliente (console) envia uma solicitação para o Amazon Lex que inclui os atributos da sessão (da conversa anterior BookHotel). O Amazon Lex passa essas informações para a função Lambda, que então inicializa (ou seja, pré-preenche) alguns dos dados do BookCar slot (ou seja,, PickUpDate e). ReturnDate PickUpCity

> **Note**
>
> Isso ilustra como atributos de sessão podem ser usados para manter o contexto nas intenções. O cliente do console fornece o link Clear na janela de teste que um usuário pode usar para limpar atributos de sessão anterior.

Para obter mais informações sobre o fluxo de dados do cliente (console) para o Amazon Lex e do Amazon Lex para a função do Lambda, consulte Fluxo de dados: intenção Book Car.

Detalhes do fluxo de informações

Neste exercício, você se envolveu em uma conversa com o BookTrip bot do Amazon Lex usando o cliente de janela de teste fornecido no console do Amazon Lex. Esta seção explica o seguinte:

- O fluxo de dados entre o cliente e o Amazon Lex..

  A seção supõe que o cliente envia solicitações ao Amazon Lex usando a API de runtime `PostText` e mostra os detalhes da solicitação e da resposta adequadamente. Para obter mais informações sobre a API de runtime `PostText`, consulte PostText.

  > **Note**
  >
  > Para obter um exemplo do fluxo de informações entre o cliente e o Amazon Lex no qual o cliente usa a API `PostContent`, consulte Etapa 2a (opcional): revisar os detalhes do fluxo de informações falado (console) .

- O fluxo de dados entre Amazon Lex e a função do Lambda. Para obter mais informações, consulte Evento de entrada de função do Lambda e formato de resposta.

Tópicos

- Fluxo de dados: intenção Book Hotel

- Fluxo de dados: intenção Book Car

Fluxo de dados: intenção Book Hotel

Esta seção explica o que acontece após cada entrada do usuário.

1. Usuário: "reservar um hotel"

   a. O cliente (console) envia a seguinte solicitação PostText para o Amazon Lex:

   ```
   POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book a hotel",
      "sessionAttributes":{}
   }
   ```

   Tanto o URI de solicitação como o corpo fornecem informações para Amazon Lex:

   - URI de solicitação — Fornece o nome do bot (BookTrip), o alias do bot ($LATEST) e o nome do usuário. O `text` final indica que esta é uma solicitação de API `PostText` (e não `PostContent`).
   - Corpo da solicitação: inclui a entrada do usuário (`inputText`) e `sessionAttributes` vazio. Inicialmente, esse é um objeto vazio e a função do Lambda primeiro define os atributos de sessão.

   b. A partir de `inputText`, o Amazon Lex detecta a intenção (BookHotel). Essa intenção é configurado com uma função do Lambda como um hook de código para inicialização/validação de dados do usuário. Portanto, o Amazon Lex invoca essa função do Lambda passando as seguintes informações como o parâmetro do evento (consulte Formato de eventos de entrada):

   ```
   {
      "messageVersion":"1.0",
      "invocationSource":"DialogCodeHook",
      "userId":"wch89kjqcpkds8seny7dly5x3otq68j3",
      "sessionAttributes":{
   ```

```
   },
   "bot":{
      "name":"BookTrip",
      "alias":null,
      "version":"$LATEST"
   },
   "outputDialogMode":"Text",
   "currentIntent":{
      "name":"BookHotel",
      "slots":{
         "RoomType":null,
         "CheckInDate":null,
         "Nights":null,
         "Location":null
      },
      "confirmationStatus":"None"
   }
}
```

Além das informações enviadas pelo cliente, o Amazon Lex também inclui os seguintes dados adicionais:

- `messageVersion`: atualmente, o Amazon Lex oferece suporte apenas à versão 1.0.
- `invocationSource`: indica o objetivo da invocação da função do Lambda. Nesse caso, o objetivo é executar os dados de inicialização e validação do usuário (nesse momento, o Amazon Lex sabe que o usuário não forneceu todos os dados de slot para atender à intenção).
- `currentIntent` - Todos os valores de slot são definidos como nulos.

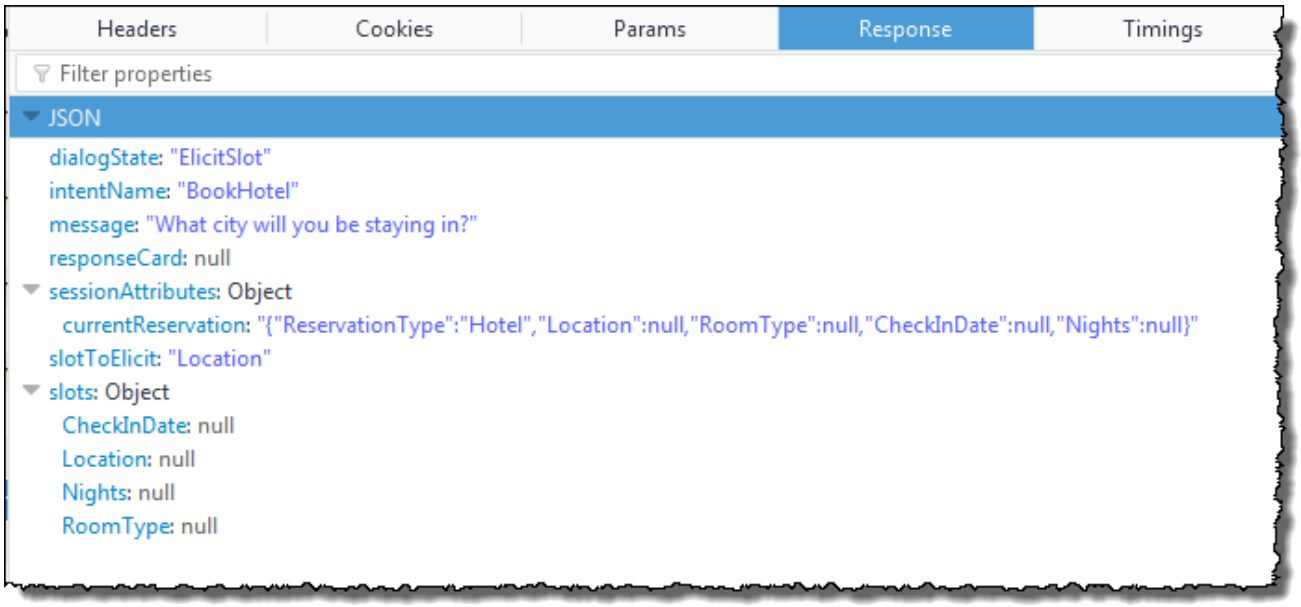
c. No momento, todos os valores de slot são nulos. Não há nada para a função do Lambda validar. A função do Lambda retorna a seguinte resposta para Amazon Lex. Para obter mais informações sobre o formato de resposta, consulte Formato de resposta.

```
{
   "sessionAttributes":{
      "currentReservation":"{\"ReservationType\":\"Hotel\",\"Location\":null,
\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
   },
   "dialogAction":{
      "type":"Delegate",
      "slots":{
         "RoomType":null,
```

```
            "CheckInDate":null,
            "Nights":null,
            "Location":null
        }
    }
}
```

> **Note**
>
> - `currentReservation`: a função do Lambda inclui este atributo de sessão. Seu valor é uma cópia das informações do slot atual e do tipo de reserva.
>
> Somente o cliente e a função do Lambda podem definir e atualizar esses atributos de sessão. Amazon Lex simplesmente passa esses valores.
> - `dialogAction.type`: ao definir esse valor como `Delegate`, a função do Lambda delega a responsabilidade da próxima ação para Amazon Lex.
>
> Se a função do Lambda tiver detectado algo na validação dos dados do usuário, ela instruirá o Amazon Lex sobre o que fazer a seguir.

d. De acordo com o `dialogAction.type`, o Amazon Lex decide a próxima ação: obter dados do usuário para o slot `Location`. Ele seleciona um dos prompts de mensagens ("Em que cidade você se hospedará?") para esse slot, de acordo com a configuração da intenção, e envia a seguinte resposta de volta ao usuário:



Os atributos de sessão são passados para o cliente.

O cliente lê a resposta e, em seguida, exibe a mensagem: "Em qual cidade você se hospedará?"

2. Usuário: "Moscou"

   a. O cliente envia a seguinte solicitação `PostText` para Amazon Lex (quebras de linha adicionadas para a capacidade de leitura):

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Moscow",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":null,
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

   Além do `inputText`, o cliente inclui os mesmos atributos de sessão `currentReservation` que recebeu.

   b. Primeiro, o Amazon Lex interpreta o `inputText` no contexto da intenção atual (o serviço lembra que pediu ao usuário específico informações sobre o slot `Location`). Ele atualiza o valor de slot para a intenção atual e invoca a função do Lambda usando o evento a seguir:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":null,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
```

```
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Moscow"
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `invocationSource` continua sendo `DialogCodeHook`. Nesta etapa, estamos apenas validando os dados do usuário.
> - Amazon Lex está apenas passando o atributo de sessão para a função do Lambda.
> - Para `currentIntent.slots`, Amazon Lex atualizou o slot `Location` para `Moscow`.

c. A função do Lambda executa a validação dos dados do usuário e determina que `Moscow` é um local inválido.

> **Note**
>
> A função do Lambda neste exercício tem uma lista simples de cidades válidas e `Moscow` não está na lista. Em uma aplicação de produção, você pode usar um banco de dados de back-end para obter essas informações.

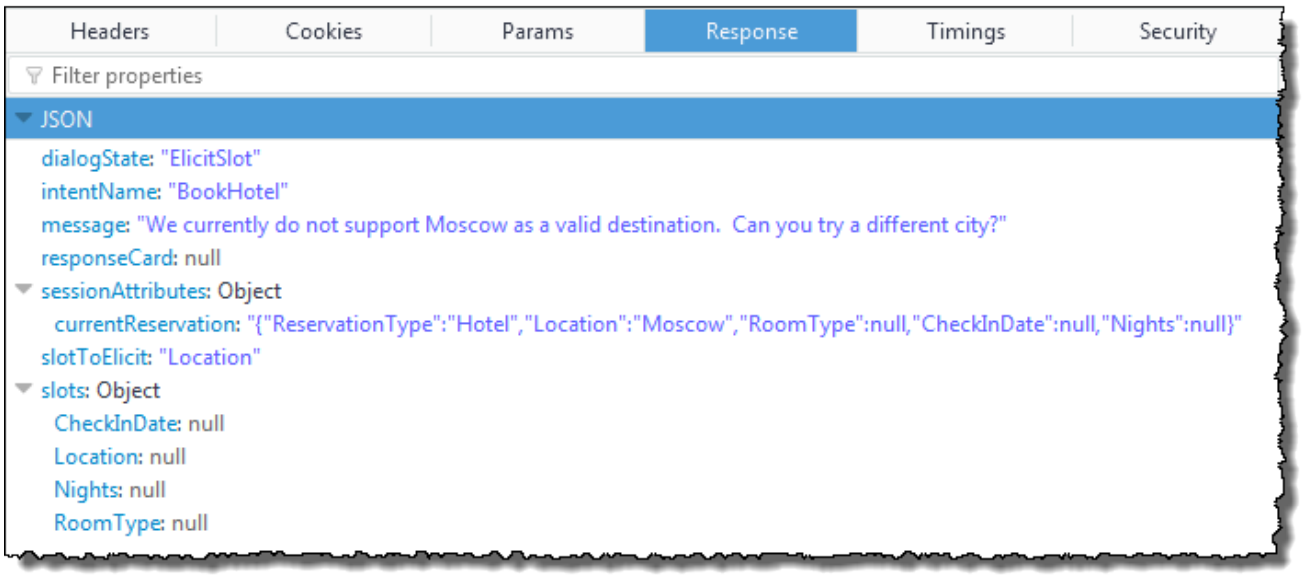
Ele redefine o valor de slot novamente como nulo e direciona o Amazon Lex a solicitar outro valor ao usuário enviando a seguinte resposta:

```
{
```

```
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Moscow\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "ElicitSlot",
        "intentName": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": null
        },
        "slotToElicit": "Location",
        "message": {
            "contentType": "PlainText",
            "content": "We currently do not support Moscow as a valid
 destination.  Can you try a different city?"
        }
    }
}
```

> **Note**
>
> - `currentIntent.slots.Location` é redefinido como nulo.
> - `dialogAction.type` é definido como `ElicitSlot`, que direciona Amazon Lex a solicitar o usuário novamente, fornecendo o seguinte:
> - `dialogAction.slotToElicit`: slot para onde obter dados do usuário.
> - `dialogAction.message`: um `message` para transmitir ao usuário.

d. Amazon Lex avisa o `dialogAction.type` e passa as informações para o cliente na resposta a seguir:



O cliente simplesmente exibe a mensagem: "No momento, não oferecemos suporte a Moscou como um destino válido. Poderia tentar outra cidade?"

3. Usuário: "Chicago"

   a. O cliente envia a seguinte solicitação `PostText` para o Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Chicago",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Moscow\",
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

   b. Amazon Lex sabe o contexto de onde estava escolhendo dados para o slot `Location`. Nesse contexto, ele sabe que o valor `inputText` é para o slot `Location`. Em seguida, ele invoca a função do Lambda enviando o seguinte evento:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":Moscow,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        },
        "confirmationStatus": "None"
    }
}
```

Amazon Lex atualizou o `currentIntent.slots` ao definir o slot `Location` como `Chicago`.

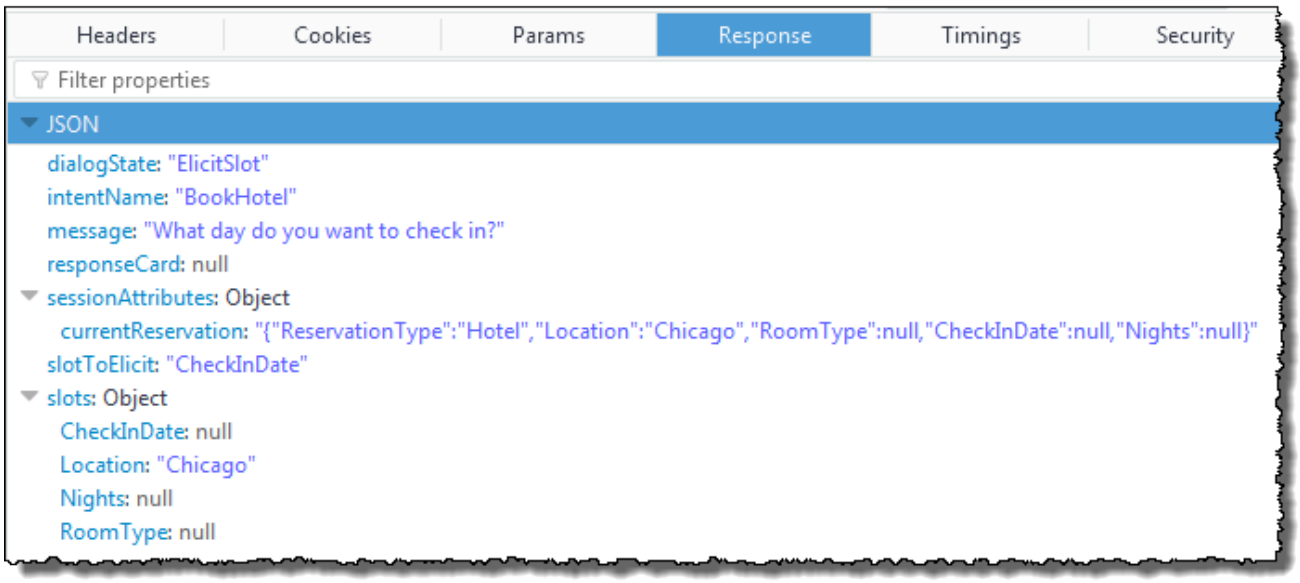
c. De acordo com o valor `invocationSource` de `DialogCodeHook`, a função do Lambda executa a validação dos dados do usuário. Ela reconhece `Chicago` como um valor de slot válido, atualiza o atributo de sessão adequadamente e, em seguida, retorna a seguinte resposta ao Amazon Lex.

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
```

```
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        }
    }
}
```

> **Note**
>
> - `currentReservation`: a função do Lambda atualiza esse atributo de sessão definindo o `Location` como `Chicago`.
> - `dialogAction.type`: é definido como `Delegate`. Os dados do usuário são válidos e a função do Lambda direciona Amazon Lex a escolher a próxima ação.

d. De acordo com o `dialogAction.type`, Amazon Lex escolhe a próxima ação. Amazon Lex sabe que precisa de mais dados de slot e escolhe o próximo slot não preenchido (`CheckInDate`) com a maior prioridade de acordo com a configuração de intenção. Ele seleciona um dos prompts de mensagens ("Em que dia você deseja fazer check-in?") para esse slot, de acordo com a configuração da intenção, e envia a seguinte resposta de volta ao cliente:

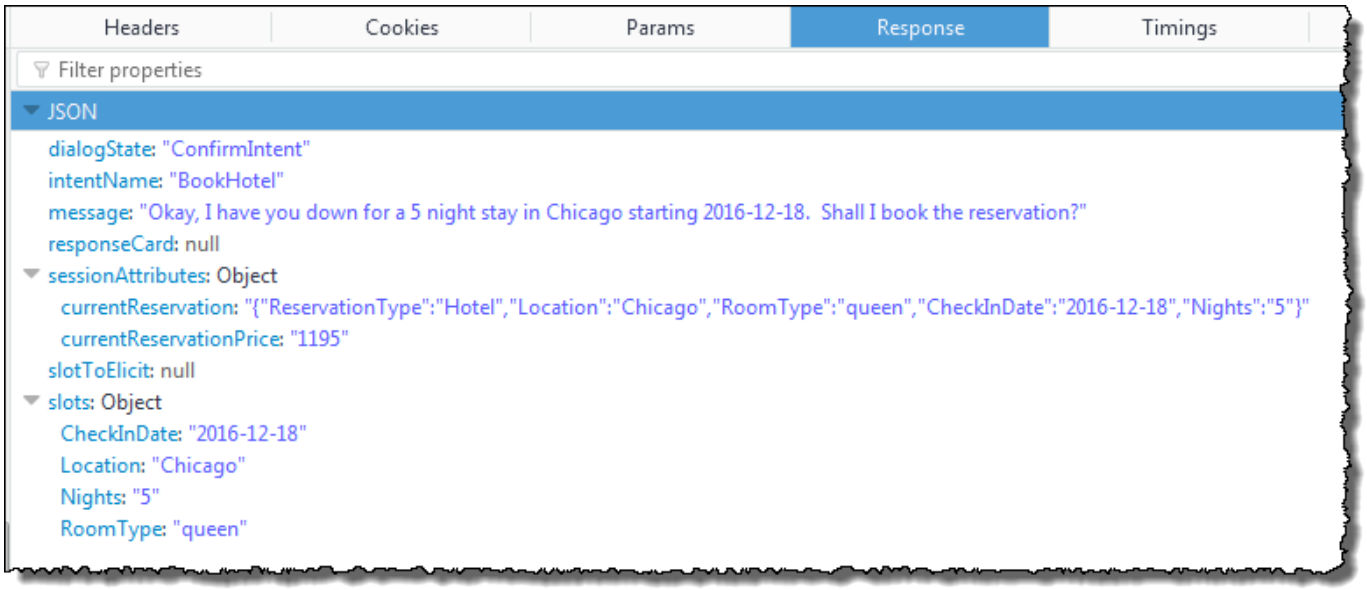


O cliente exibe a mensagem: "Em que dia você deseja fazer check-in?"

4. A interação do usuário continua. O usuário fornece os dados, a função do Lambda valida os dados e, em seguida, delega a próxima ação para o Amazon Lex. Por fim, o usuário fornece todos os dados do slot, a função do Lambda valida todas as entradas do usuário e, em seguida, o Amazon Lex reconhece que tem todos os dados do slot.

> **Note**
>
> Neste exercício, depois que o usuário fornece todos os dados do slot, a função do Lambda calcula o preço da reserva de hotel e o retorna como outro atributo de sessão (`currentReservationPrice`).

Nesse momento, a intenção está pronta para ser cumprida, mas a BookHotel intenção é configurada com um aviso de confirmação que exige a confirmação do usuário antes que o Amazon Lex possa cumprir a intenção. Portanto, o Amazon Lex envia a seguinte mensagem ao cliente solicitando a confirmação antes de reservar o hotel:



O cliente exibe a mensagem: "OK, reservarei para você 5 diárias em Chicago a partir de 18/12/2016. Posso fazer a reserva?"

5. Usuário: "sim"

a. O cliente envia a seguinte solicitação `PostText` para o Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Yes",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Chicago\",
                              \"RoomType\":\"queen\",
                              \"CheckInDate\":\"2016-12-18\",
                              \"Nights\":\"5\"}",
      "currentReservationPrice":"1195"
   }
}
```

b. O Amazon Lex interpreta o `inputText` no contexto de confirmação da intenção atual. Amazon Lex entende que o usuário deseja prosseguir com a reserva. Dessa vez, o Amazon Lex invoca a função do Lambda para atender à intenção enviando o seguinte evento. Ao definir o `invocationSource` como `FulfillmentCodeHook` no evento, ele envia para a função do Lambda. O Amazon Lex também define o `confirmationStatus` como `Confirmed`.

```
{
    "messageVersion": "1.0",
    "invocationSource": "FulfillmentCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights\":
\"5\"}",
        "currentReservationPrice": "956"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
```

```
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": "queen",
            "CheckInDate": "2016-12-18",
            "Nights": "5",
            "Location": "Chicago"
        },
        "confirmationStatus": "Confirmed"
    }
}
```

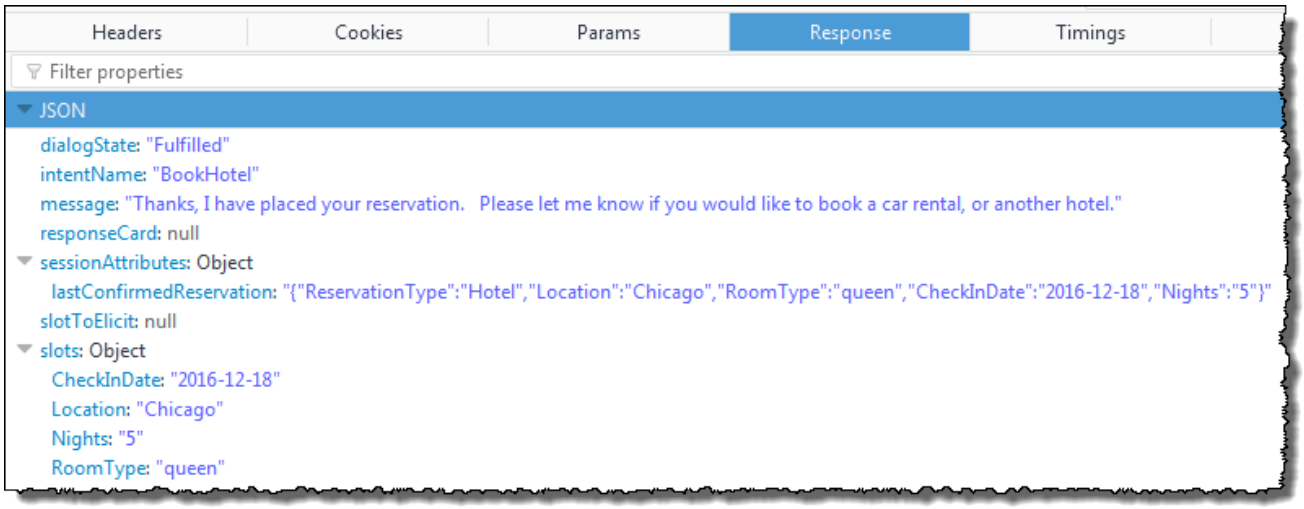
> **Note**
>
> - `invocationSource`: dessa vez, o Amazon Lex define esse valor como `FulfillmentCodeHook`, direcionando a função do Lambda a atender à intenção.
> - `confirmationStatus`: é definido como `Confirmed`.

c. Dessa vez, a função Lambda cumpre a BookHotel intenção, o Amazon Lex conclui a reserva e, em seguida, retorna a seguinte resposta:

```
{
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, I have placed your reservation.   Please let me
 know if you would like to book a car rental, or another hotel."
        }
    }
}
```

> **Note**
>
> - `lastConfirmedReservation`: é um novo atributo de sessão que a função do Lambda adicionou (em vez do `currentReservation`, `currentReservationPrice`).
> - `dialogAction.type`: a função do Lambda define esse valor como `Close`, indicando a Amazon Lex que não aguarde uma resposta do usuário.
> - `dialogAction.fulfillmentState`: é definido como `Fulfilled` e inclui uma `message` adequada para transmitir para o usuário.

d. Amazon Lex analisa o `fulfillmentState` e envia a resposta seguinte de volta para o cliente:



> **Note**
>
> - `dialogState`: o Amazon Lex define esse valor como `Fulfilled`.
> - `message`: é a mesma mensagem que a função do Lambda forneceu.

O cliente exibe a mensagem.

Fluxo de dados: intenção Book Car

O BookTrip bot neste exercício apoia duas intenções (BookHotel e BookCar). Depois de reservar um hotel, o usuário pode continuar a conversa para reservar um carro. Desde que a sessão não tenha expirado, em cada solicitação subsequente o cliente continua a enviar os atributos de sessão (neste exemplo, o `lastConfirmedReservation`). A função Lambda pode usar essas informações para inicializar os dados do slot para a intenção. BookCar Isso mostra como é possível usar atributos de sessão no compartilhamento de informações entre intenções.

Especificamente, quando o usuário escolhe a BookCar intenção, a função Lambda usa informações relevantes no atributo de sessão para preencher previamente os slots (PickUpDate, ReturnDate, e) para a intenção. PickUpCity BookCar

> **Note**
>
> O console do Amazon Lex fornece o link Limpar que você pode usar para limpar os atributos da sessão anterior.

Siga as etapas neste procedimento para continuar a conversa.

1. Usuário: "também reservar um carro"

   a. O cliente envia a seguinte solicitação `PostText` para o Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"also book a car",
   "sessionAttributes":{
       "lastConfirmedReservation":""{\"ReservationType\":\"Hotel\",
                                     \"Location\":\"Chicago\",
                                     \"RoomType\":\"queen\",
                                     \"CheckInDate\":\"2016-12-18\",
                                     \"Nights\":\"5\"}"
   }
}
```

   O cliente inclui o atributo de sessão `lastConfirmedReservation`.

b. O Amazon Lex detecta a intenção (BookCar) do. `inputText` Essa intenção também está configurada para invocar a função do Lambda para executar a inicialização e a validação dos dados do usuário. Amazon Lex invoca a função do Lambda com os seguintes eventos:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookCar",
        "slots": {
            "PickUpDate": null,
            "ReturnDate": null,
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": null
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `messageVersion`: atualmente, Amazon Lex oferece suporte apenas à versão 1.0.
> - `invocationSource`: indica que a finalidade de invocação é executar a inicialização e a validação dos dados do usuário.

- `currentIntent`: inclui o nome da intenção e os slots. No momento, todos os valores de slot são nulos.

c. A função do Lambda observa todos os valores de slot nulos com nada para validar. No entanto, ela usa atributos de sessão para inicializar alguns dos valores de slot (`PickUpDate`, `ReturnDate` e `PickUpCity`) e, em seguida, retorna a seguinte resposta:

```
{
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
        "confirmationContext": "AutoPopulate"
    },
    "dialogAction": {
        "type": "ConfirmIntent",
        "intentName": "BookCar",
        "slots": {
            "PickUpCity": "Chicago",
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "CarType": null,
            "DriverAge": null
        },
        "message": {
            "contentType": "PlainText",
            "content": "Is this car rental for your 5 night stay in Chicago on
 2016-12-18?"
        }
    }
}
```

Note

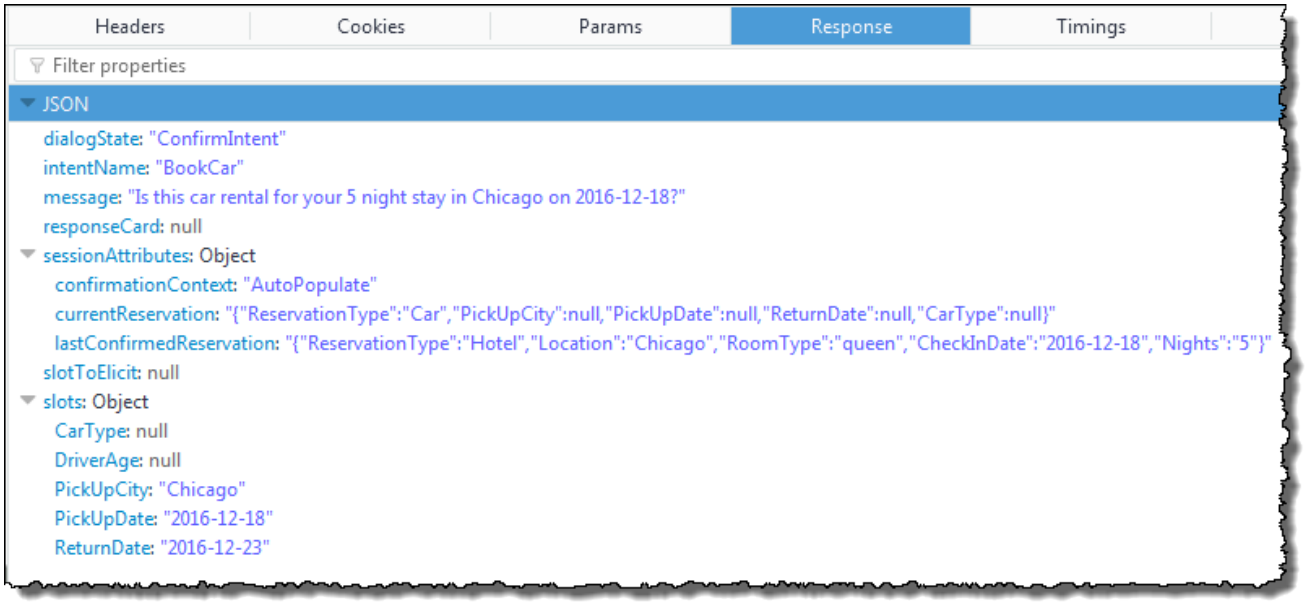
- Além do `lastConfirmedReservation`, a função do Lambda inclui mais atributos de sessão (`currentReservation` e `confirmationContext`).

- `dialogAction.type`é definido como`ConfirmIntent`, o que informa ao Amazon Lex que uma resposta sim, não é esperada do usuário (o ConfirmationContext definido como, a função Lambda sabe que AutoPopulate a resposta do usuário sim/não é para obter a confirmação do usuário da inicialização realizada pela função Lambda (dados de slot preenchidos automaticamente).

  A função do Lambda também inclui uma mensagem informativa na resposta no `dialogAction.message` para Amazon Lex retornar ao cliente.

  > **Note**
  >
  > O termo `ConfirmIntent` (valor do `dialogAction.type`) não é relacionado a nenhuma intenção de bot. No exemplo, a função do Lambda usa esse termo para direcionar o Amazon Lex para obter uma resposta sim/não do usuário.

d. De acordo com o `dialogAction.type`, Amazon Lex retorna a seguinte resposta ao cliente:



O cliente exibe a mensagem: "Este aluguel de carro é para sua estadia de 5 diárias em Chicago em 18/12/2016?"

2. Usuário: "sim"

   a. O cliente envia a seguinte solicitação `PostText` para o Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"yes",
   "sessionAttributes":{
      "confirmationContext":"AutoPopulate",
      "currentReservation":"{\"ReservationType\":\"Car\",
                            \"PickUpCity\":null,
                            \"PickUpDate\":null,
                            \"ReturnDate\":null,
                            \"CarType\":null}",
      "lastConfirmedReservation":"{\"ReservationType\":\"Hotel\",
                                   \"Location\":\"Chicago\",
                                   \"RoomType\":\"queen\",
                                   \"CheckInDate\":\"2016-12-18\",
                                   \"Nights\":\"5\"}"
   }
}
```

   b. O Amazon Lex lê o `inputText` e sabe o contexto (solicitou que o usuário confirme o preenchimento automático). O Amazon Lex invoca a função do Lambda enviando o seguinte evento:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "confirmationContext": "AutoPopulate",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "bot": {
        "name": "BookTrip",
```

```
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookCar",
        "slots": {
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": "Chicago"
        },
        "confirmationStatus": "Confirmed"
    }
}
```

Como o usuário respondeu Sim, Amazon Lex define o `confirmationStatus` como `Confirmed`.

c. A partir do `confirmationStatus`, a função do Lambda sabe que os valores preenchidos automaticamente estão corretos. A função do Lambda faz o seguinte:

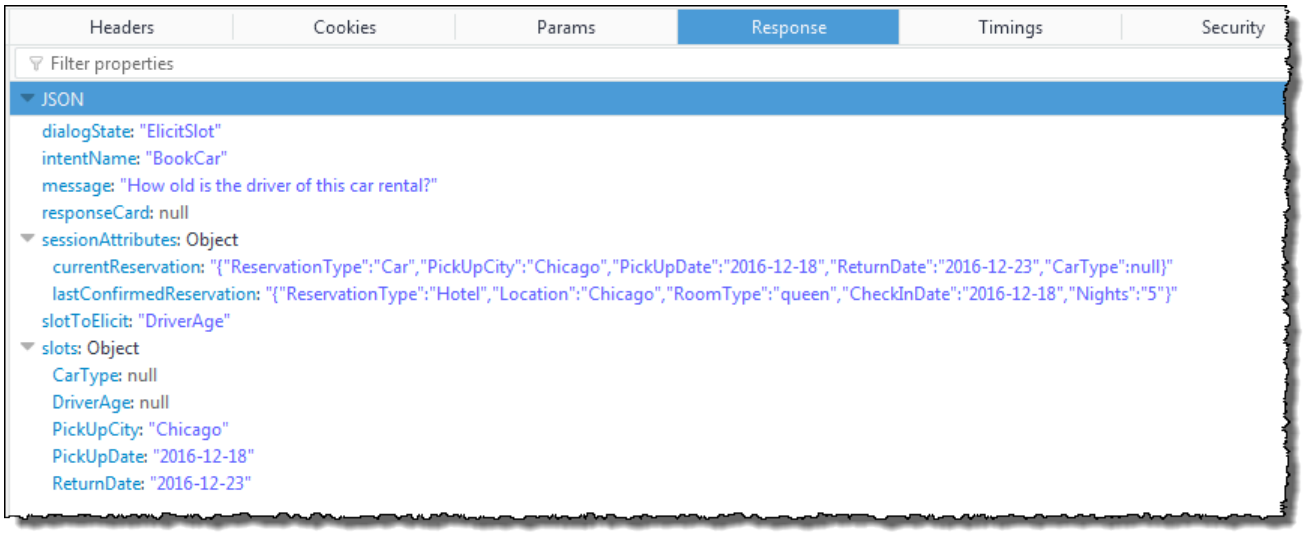
- Atualiza o atributo de sessão `currentReservation` para o valor de slot que tinha preenchido automaticamente.
- Define o `dialogAction.type` como `ElicitSlot`
- Define o valor `slotToElicit` como `DriverAge`.

A resposta seguinte é enviada:

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":\"Chicago\",\"PickUpDate\":\"2016-12-18\",\"ReturnDate\":\"2016-12-22\",
\"CarType\":null}",
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "dialogAction": {
        "type": "ElicitSlot",
```

```
        "intentName": "BookCar",
        "slots": {
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": "Chicago"
        },
        "slotToElicit": "DriverAge",
        "message": {
            "contentType": "PlainText",
            "content": "How old is the driver of this car rental?"
        }
    }
}
```

d. Amazon Lex retorna a seguinte resposta:



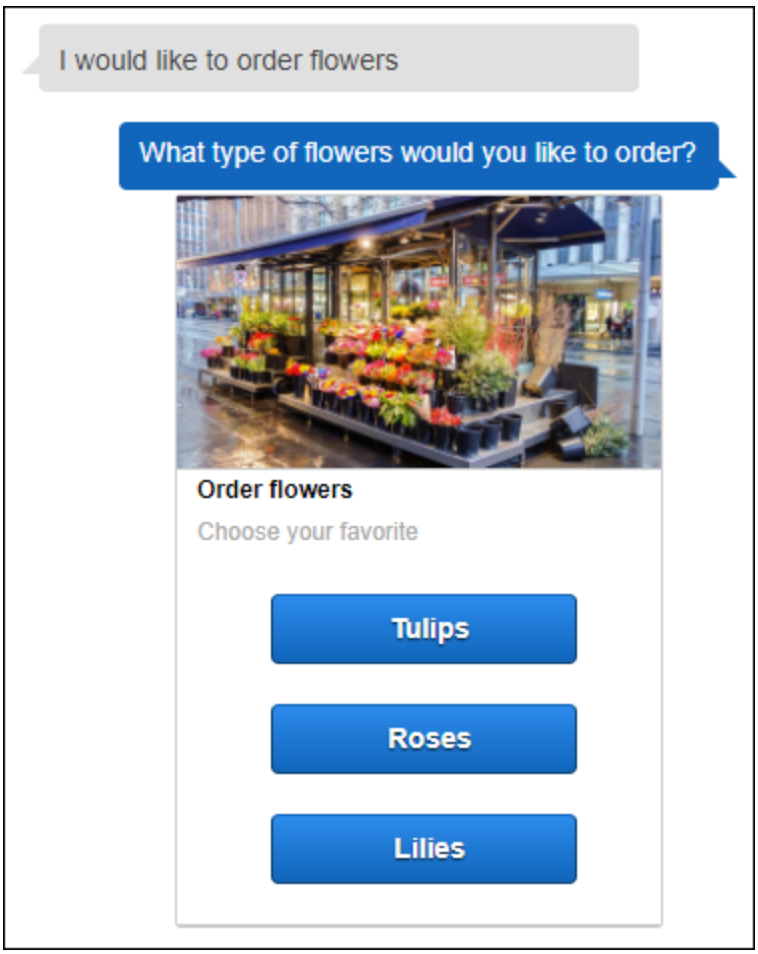
O cliente exibe a mensagem "Quantos anos tem o motorista deste aluguel de carro?" e a conversa continua.

Exemplo: uso de um cartão de resposta

Neste exercício, o Exercício 1 dos Conceitos básicos é estendido adicionando um cartão de resposta. Você cria um bot que suporta a OrderFlowers intenção e, em seguida, atualiza a intenção adicionando um cartão de resposta ao `FlowerType` slot. Além do seguinte prompt para o slot `FlowerType`, o usuário pode escolher os tipos de flor do cartão de resposta:

```
What type of flowers would you like to order?
```

O cartão de resposta está a seguir:



O usuário do bot pode digitar o texto ou escolher a partir de uma lista de tipos de flor. Este cartão de resposta é configurado com uma imagem, que aparece no cliente como mostrado. Para obter mais informações sobre cartões de resposta, consulte Cartões de resposta.

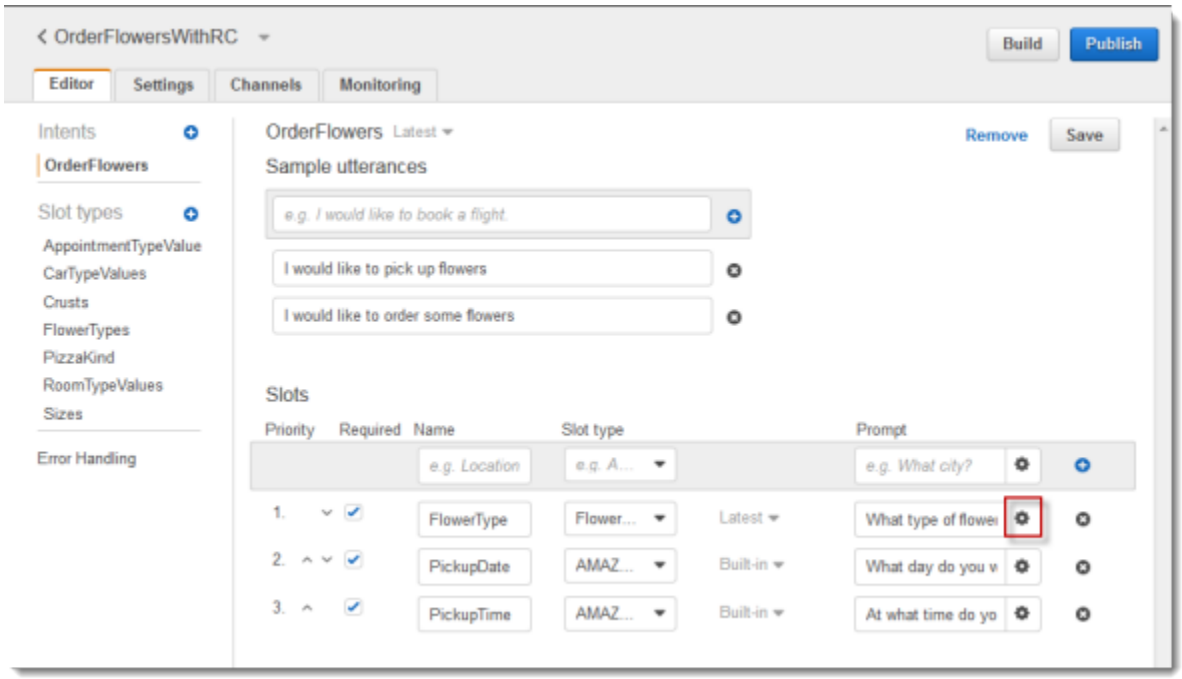
Como criar e testar um bot com um cartão de resposta:

1. Siga o Exercício de Introdução 1 para criar e testar um OrderFlowers bot. Você deve concluir as etapas 1, 2 e 3. Não é necessário adicionar uma função do Lambda para testar o cartão de resposta. Para obter instruções, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..
2. Atualize o bot adicionando o cartão de resposta e, em seguida, publique uma versão. Quando for publicar uma versão, especifique um alias (BETA) para apontar para ela.

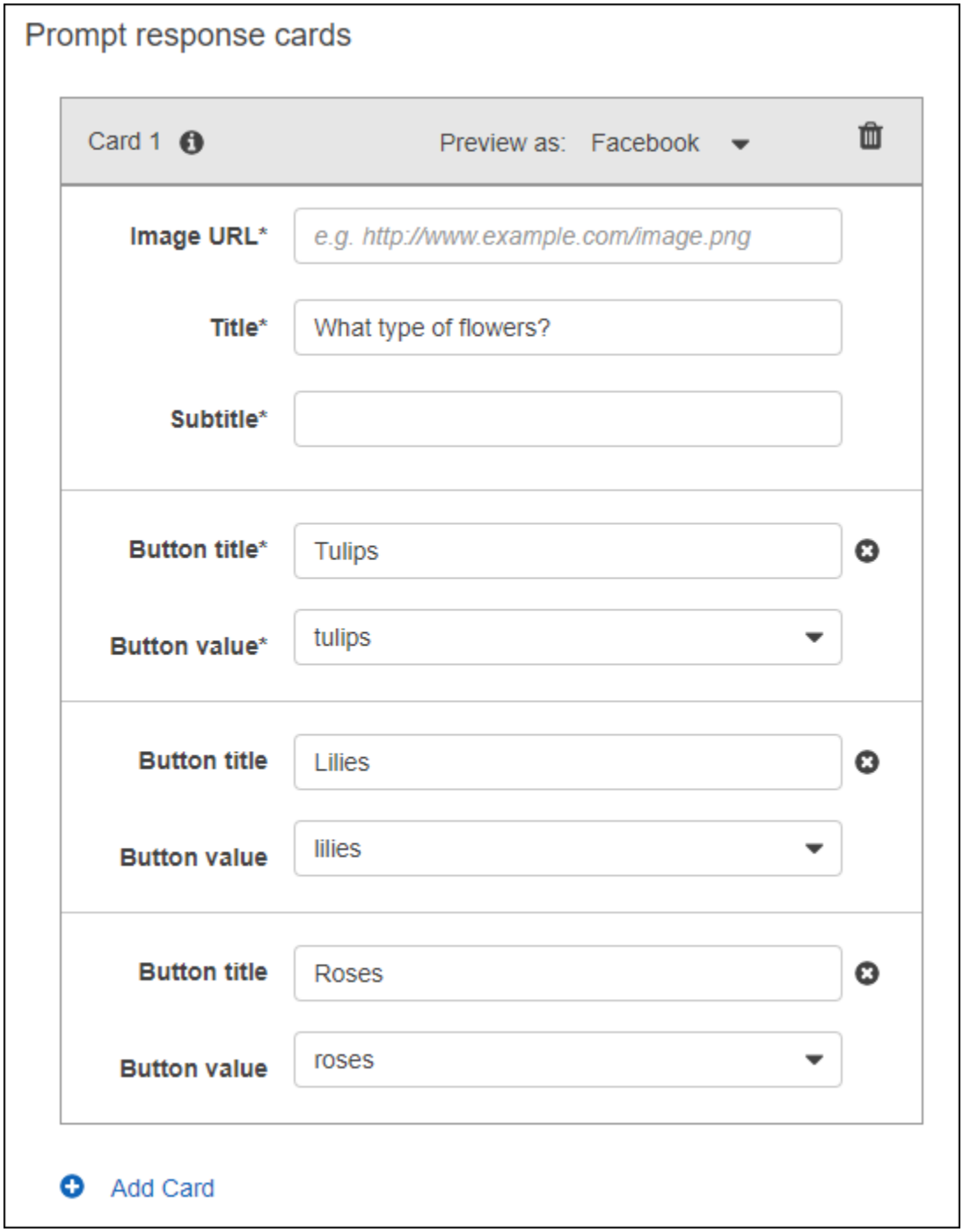
a. No console do Amazon Lex, escolha seu bot.

b. Escolha a intenção `OrderFlowers`.

c. Escolha o ícone de engrenagem de configuração próximo ao Prompt "Que tipo de flores" para configurar um cartão de resposta para o `FlowerType`.



d. Forneça um título ao cartão e configure três botões conforme mostrado na captura de tela a seguir. Opcionalmente, você pode adicionar uma imagem ao cartão de resposta, desde que você tenha um URL da imagem. Se estiver implantando um bot usando SMS do Twilio, deverá fornecer um URL da imagem.



e. Escolha Save (Salvar) para salvar o cartão de resposta.

f. Escolha Save intent (Salvar intenção) para salvar a configuração da intenção.

g. Para criar o bot, escolha Build.

h. Para publicar uma versão do bot, escolha Publish. Especifique BETA como um alias que aponta para a versão do bot. Para obter informações sobre versionamento, consulte Versionamento e aliases.

3. Implante o bot em uma plataforma de mensagens:

   - Implante o bot na plataforma do Facebook Messenger e teste a integração. Para obter instruções, consulte Integração de um bot do Amazon Lex com o Facebook Messenger. Quando você encomenda flores, a janela de mensagem mostra o cartão de resposta para que você possa escolher um tipo de flor.
   - Implante o bot na plataforma Slack e teste a integração. Para obter instruções, consulte Integração de um bot de Amazon Lex com o Slack . Quando você encomenda flores, a janela de mensagem mostra o cartão de resposta para que você possa escolher um tipo de flor.
   - Implante o bot na plataforma de SMS Twilio. Para obter instruções, consulte Integração de um bot do Amazon Lex com o SMS programável do Twilio . Ao encomendar flores, a mensagem do Twilio mostra a imagem do cartão de resposta. O SMS do Twilio não comporta botões na resposta.

Atualizar declarações

Neste exercício, adicione declarações adicionais aos que você criou no Exercício 1 de Conceitos básicos. Use a guia Monitoramento no console do Amazon Lex para visualizar as declarações que seu bot não reconheceu. Para melhorar a experiência dos usuários, adicione essas declarações ao bot.

As estatísticas de declaração não são geradas nas seguintes condições:

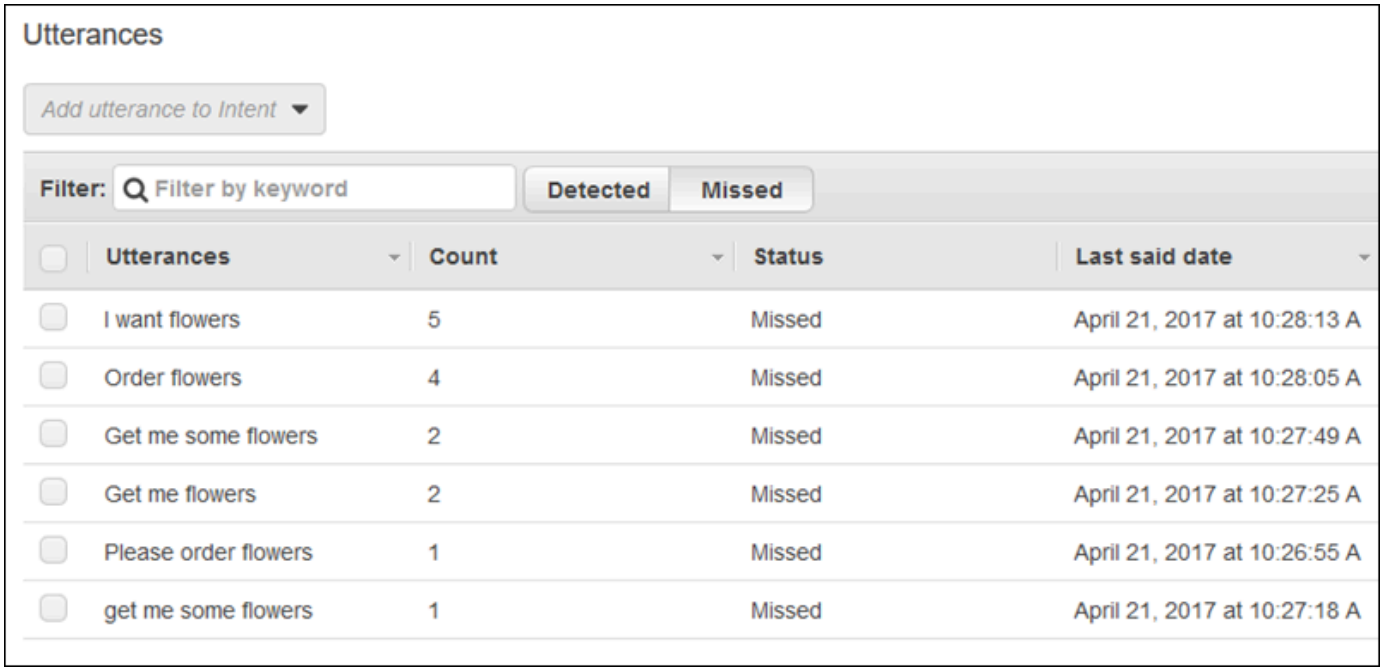
- O campo `childDirected` foi definido como verdadeiro quando o bot foi criado.
- Você está usando a ofuscação de slots com um ou mais slots.
- Você optou por não participar da melhoria do Amazon Lex.

> **Note**
>
> As estatísticas de declaração são geradas uma vez por dia. Você pode ver a declaração que não foi reconhecida, quantas vezes ela foi ouvido, e a última data e hora em que foi ouvida. Pode levar até 24 horas para as declarações perdidas aparecerem no console.

Você pode ver declarações para diferentes versões do bot. Para alterar a versão do bot para a qual você está vendo as declarações, escolha uma versão diferente na lista vertical ao lado do nome do bot.

Para visualizar e adicionar as declarações perdidas a um bot:

1. Siga a primeira etapa do Exercício 1 de Conceitos básicos para criar e testar um bot `OrderFlowers`. Para obter instruções, consulte Exercício 1: Criar um bot do Amazon Lex usando um esquema (Console)..
2. Teste o bot digitando a seguinte declaração na janela Test Bot. Digite cada declaração várias vezes. O bot de exemplo não reconhece os seguintes declarações:
   - Pedir flores
   - Obtenha flores para mim
   - Peça flores
   - Obtenha algumas flores para mim
3. Aguarde o Amazon Lex coletar os dados de uso sobre as declarações perdidas. Os dados da declaração são gerados uma vez por dia, geralmente durante a noite.
4. Faça login no AWS Management Console e abra o console do Amazon Lex em https://console.aws.amazon.com/lex/.
5. Escolha o bot `OrderFlowers`.
6. Escolha a guia Monitoramento, selecione Declarações no menu esquerdo e, em seguida, escolha o botão Perdida. O painel mostra um máximo de 100 declarações perdidas.



7. Para escolher as declarações perdidss que você deseja adicionar ao bot, selecione a caixa de seleção próxima a eles. Para adicionar a declaração à versão `$LATEST` da intenção, escolha a seta para baixo ao lado da lista suspensa Add utterance to intent e, em seguida, escolha a intenção.
8. Para recriar o bot, escolha Build e, em seguida, Build novamente.
9. Para verificar se o seu bot reconhece a nova declaração, use o painel Test Bot.

Exemplo: Integração com um site

Neste exemplo, você integra um bot com um site usando texto e voz. Você usa JavaScript os AWS serviços para criar uma experiência interativa para os visitantes do seu site. Você pode escolher entre esses exemplos documentados no Blog de IA da AWS:

- Implantar uma interface de usuário da Web para seu Chatbot: demonstra uma interface de usuário da Web completa que fornece um cliente da Web para chatbots do Amazon Lex. Você pode usar isso para saber mais sobre clientes da Web, ou como um bloco de criação para sua própria aplicação.

- “Saudações, visitante!” —Envolva seus usuários da Web com o Amazon Lex —Demonstra o uso do Amazon Lex, o SDK da AWS para o navegador, e do Amazon Cognito para criar uma experiência de conversação JavaScript em seu site.

- Capturar a entrada de voz em um navegador e enviá-la para o Amazon Lex — demonstra a incorporação de um chatbot baseado em voz em um site usando o SDK para o navegador. JavaScript O aplicativo grava o áudio, envia o áudio para o Amazon Lex e, em seguida, reproduz a resposta.

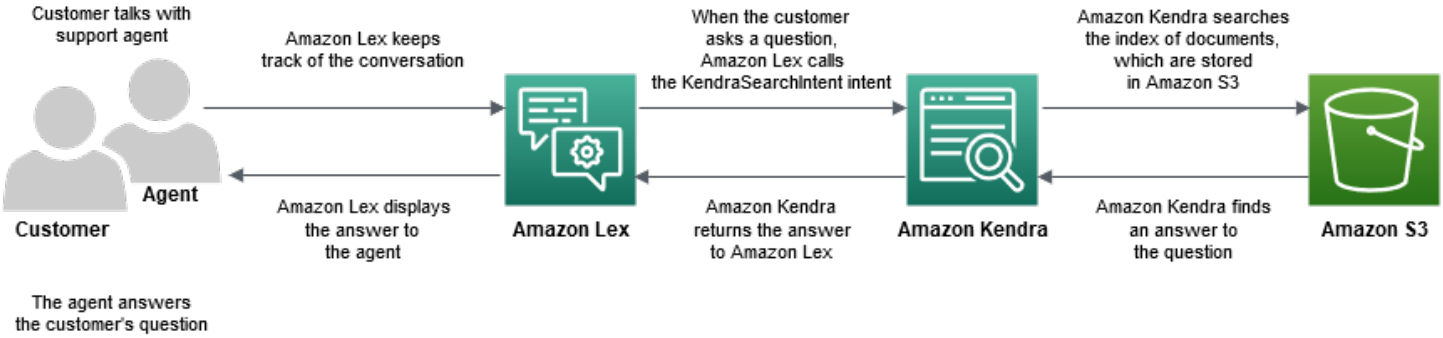
Assistente de atendente do call center

Neste tutorial, você usará o Amazon Lex com o Amazon Kendra para criar um bot de assistência que ajude os atendentes do suporte ao cliente e publicá-lo como um aplicativo Web. O Amazon Kendra é um serviço de pesquisa empresarial que usa machine learning para pesquisar documentos e encontrar respostas. Para obter mais informações sobre o Amazon Kendra, consulte o Guia do desenvolvedor do Amazon Kendra.

Os bots Amazon Lex são amplamente usados em call centers como o primeiro ponto de contato dos clientes. Frequentemente, um bot consegue resolver as dúvidas dos clientes. Quando um bot não consegue responder a uma pergunta, ele transfere a conversa para um funcionário do suporte ao cliente.

Neste tutorial, criamos um bot Amazon Lex que os atendentes usam para responder às consultas dos clientes em tempo real. Ao ler as respostas do bot, o atendente não precisa pesquisar as respostas manualmente.

O bot e o aplicativo Web que você cria neste tutorial ajudam os atendentes a responder aos clientes com eficiência e precisão, fornecendo rapidamente os recursos certos. O diagrama a seguir mostra como o aplicativo Web funciona.



Como mostra o diagrama, o índice de documentos do Amazon Kendra é armazenado em um bucket do Amazon Simple Storage Service (Amazon S3). Se ainda não tem um bucket do S3, você pode configurá-lo ao criar o índice do Amazon Kendra. Além do Amazon S3, você usa o Amazon Cognito neste tutorial. O Amazon Cognito gerencia as permissões para implantar o bot como um aplicativo Web.

Neste tutorial, você cria um índice do Amazon Kendra que responde às perguntas dos clientes, cria o bot e adiciona intenções que permitem que ele sugira respostas com base na conversa com o cliente, configura o Amazon Cognito para gerenciar permissões de acesso e implanta o bot como um aplicativo Web.

Tempo estimado: 75 minutos

Custo estimado: USD 2,50 por hora para um índice do Amazon Kendra e USD 0,75 para mil solicitações do Amazon Lex. Seu índice do Amazon Kendra continuará em execução depois que você terminar este exercício. Exclua-o para evitar custos desnecessários.

Observação: escolha a mesma região da AWS para todos os serviços usados neste tutorial.

Tópicos

- Etapa 1: criar um índice do Amazon Kendra
- Etapa 2: criar um bot do Amazon Lex
- Etapa 3: adicionar uma intenção personalizada e integrada
- Etapa 4: configurar o Amazon Cognito
- Etapa 5: implantar seu bot como um aplicativo Web
- Etapa 6: usar o bot

Etapa 1: criar um índice do Amazon Kendra

Comece criando um índice de documentos do Amazon Kendra que respondam às perguntas dos clientes. O índice fornece uma API de pesquisa para consultas dos clientes. Você cria o índice a partir dos documentos de origem. O Amazon Kendra retorna as respostas que encontra em documentos indexados para o bot, que as exibe para o atendente.

A qualidade e a precisão das respostas sugeridas pelo Amazon Kendra dependem dos documentos que você indexa. Os documentos devem incluir arquivos acessados com frequência pelo atendente e ser armazenados em um bucket do S3. Você pode indexar dados não estruturados e semiestruturados nos formatos .html, Microsoft Office (.doc, .ppt), PDF e texto.

Para criar um índice do Amazon Kendra, consulte Conceitos básicos do bucket do S3 (console) no Guia do desenvolvedor do Amazon Kendra.

Para adicionar perguntas e respostas (FAQs) que ajudem a responder às dúvidas dos clientes, consulte Adicionar perguntas e respostas no Guia do desenvolvedor do Amazon Kendra. Para este tutorial, use o arquivo ML_FAQ.csv em GitHub.

Próxima etapa

Etapa 2: criar um bot do Amazon Lex

Etapa 2: criar um bot do Amazon Lex

O Amazon Lex fornece uma interface entre o atendente do call center e o índice do Amazon Kendra. Ele acompanha a conversa entre o atendente e o cliente e chama a intenção de `AMAZON.KendraSearchIntent` com base nas perguntas que o cliente faz. A intenção é uma ação que o usuário quer realizar.

O Amazon Kendra pesquisa os documentos indexados e retorna uma resposta para o Amazon Lex, que ele exibe no bot. Esta resposta é visível somente para o atendente.

Para criar um bot de assistente do atendente

1. Faça login no 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.
4. Escolha Bot personalizado e configure o bot.
   a. Nome do bot: insira um nome que indique a finalidade do bot, como **`AgentAssistBot`**.
   b. Voz de saída: escolha Nenhuma.
   c. Tempo limite da sessão: insira **`5`**.
   d. COPPA: escolha Não.
5. Escolha Criar. Depois de criar o bot, o Amazon Lex exibe a guia do editor de bots.

Próxima etapa

Etapa 3: adicionar uma intenção personalizada e integrada

Etapa 3: adicionar uma intenção personalizada e integrada

A intenção representa uma ação que o atendente do call center quer que o bot realize. Nesse caso, o atendente quer que o bot sugira respostas e recursos úteis com base na conversa do atendente com o cliente.

O Amazon Lex tem dois tipos de intenção: intenções personalizadas e intenções integradas. `AMAZON.KendraSearchIntent` é uma intenção integrada. O bot usa a intenção `AMAZON.KendraSearchIntent` para consultar o índice e exibir as respostas sugeridas pelo Amazon Kendra.

O bot neste exemplo não precisa de uma intenção personalizada. No entanto, para criar o bot, é necessário criar pelo menos uma intenção personalizada com pelo menos uma amostra de declaração. Essa intenção é necessária apenas para criar o bot de assistente do atendente. Ela não executa nenhuma outra função. a declaração para a intenção não deve responder a nenhuma das perguntas feitas pelo cliente. Isso garante que a intenção `AMAZON.KendraSearchIntent` seja chamada para responder às dúvidas dos clientes. Para obter mais informações, consulte AMAZON.KendraSearchIntent.

Para criar a intenção personalizada necessária

1. Na página Começar a usar o bot, escolha Criar intenção.
2. Em Adicionar intenção, escolha Criar intenção.
3. Na caixa de diálogo Criar intenção, insira um nome descritivo para a intenção, como **`RequiredIntent`**.
4. Em Amostra de declarações, insira uma declaração descritiva, como **`Required utterance`**.
5. Selecione Salvar intenção.

Para adicionar o AMAZON. KendraSearchIntent mensagem de intenção e resposta

1. No painel de navegação, escolha o sinal de adição (+) ao lado de Intenções.
2. Escolha Pesquisar intenções existentes.
3. Na caixa Pesquisar intenções, insira **`AMAZON.KendraSearchIntent`** e escolha-a na lista.
4. Insira um nome descritivo para a intenção, como **`AgentAssistSearchIntent`**, e escolha Adicionar.
5. No editor de intenções, escolha Consulta do Amazon Kendra para abrir as opções de consulta.

6. Escolha o índice que você quer que a intenção pesquise,
7. Na seção Resposta, adicione as três mensagens a seguir a um grupo de mensagens.

```
I found an answer for the customer query: ((x-amz-lex:kendra-search-response-
question_answer-question-1)) and the answer is ((x-amz-lex:kendra-search-response-
question_answer-answer-1)).
I found an excerpt from a helpful document: ((x-amz-lex:kendra-search-response-
document-1)).
I think this answer will help the customer: ((x-amz-lex:kendra-search-response-
answer-1)).
```

8. Selecione Salvar intenção.
9. Escolha Criar para criar o bot.

Próxima etapa

Etapa 4: configurar o Amazon Cognito

Etapa 4: configurar o Amazon Cognito

Para gerenciar permissões e usuários do aplicativo Web, você precisa configurar o Amazon Cognito. O Amazon Cognito garante que o aplicativo Web seja seguro e tenha controle de acesso. O Amazon Cognito usa grupos de identidade para fornecer AWS credenciais que concedem aos usuários acesso a outros serviços. AWS Para este tutorial, ele fornece acesso ao Amazon Lex.

Ao criar um grupo de identidades, o Amazon Cognito fornece funções AWS Identity and Access Management (IAM) para usuários autenticados e não autenticados. Modifique os perfis do IAM adicionando políticas que concedem acesso ao Amazon Lex.

Para configurar o Amazon Cognito

1. Faça login AWS Management Console e abra o console do Amazon Cognito em. https://console.aws.amazon.com/cognito/
2. Escolha Gerenciar grupos de identidades.
3. Escolha Criar novo grupo de identidades.
4. Configure o banco de identidades.
   a. Nome do banco de identidades: insira um nome que indique a finalidade do banco, como **`BotPool`**.

   b. Na seção Identidades não autenticadas, escolha Permitir acesso a identidades não autenticadas.

5. Selecione Criar grupo.
6. Na página Identificar os perfis do IAM a serem usados com seu novo banco de identidades, escolha Exibir detalhes.
7. Registre os nomes dos perfis do IAM. Você os modificará posteriormente.
8. Selecione Permitir.
9. Na página Getting Started with Amazon Cognito, em Plataforma, escolha. JavaScript
10. Na seção Obter AWS credenciais, localize e registre o ID do grupo de identidades.
11. Para permitir acesso ao Amazon Lex, modifique os perfis do IAM autenticados e não autenticados.

    a. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
    b. No painel de navegação, em Gerenciamento de acesso, escolha Perfis.
    c. Na caixa de pesquisa, insira o nome do perfil do IAM autenticado e marque a caixa de seleção ao lado dele.

       i. Escolha Anexar políticas.
       ii. Na caixa de pesquisa, insira **`AmazonLexRunBotsOnly`** e marque a caixa de seleção ao lado dele.
       iii. Escolha Anexar política.

    d. Insira o nome do perfil do IAM não autenticado na caixa de pesquisa e marque a caixa de seleção ao lado dele.

       i. Escolha Anexar políticas.
       ii. Na caixa de pesquisa, insira **`AmazonLexRunBotsOnly`** e marque a caixa de seleção ao lado dele.
       iii. Escolha Anexar política.

Próxima etapa

Etapa 5: implantar seu bot como um aplicativo Web

Etapa 5: implantar seu bot como um aplicativo Web

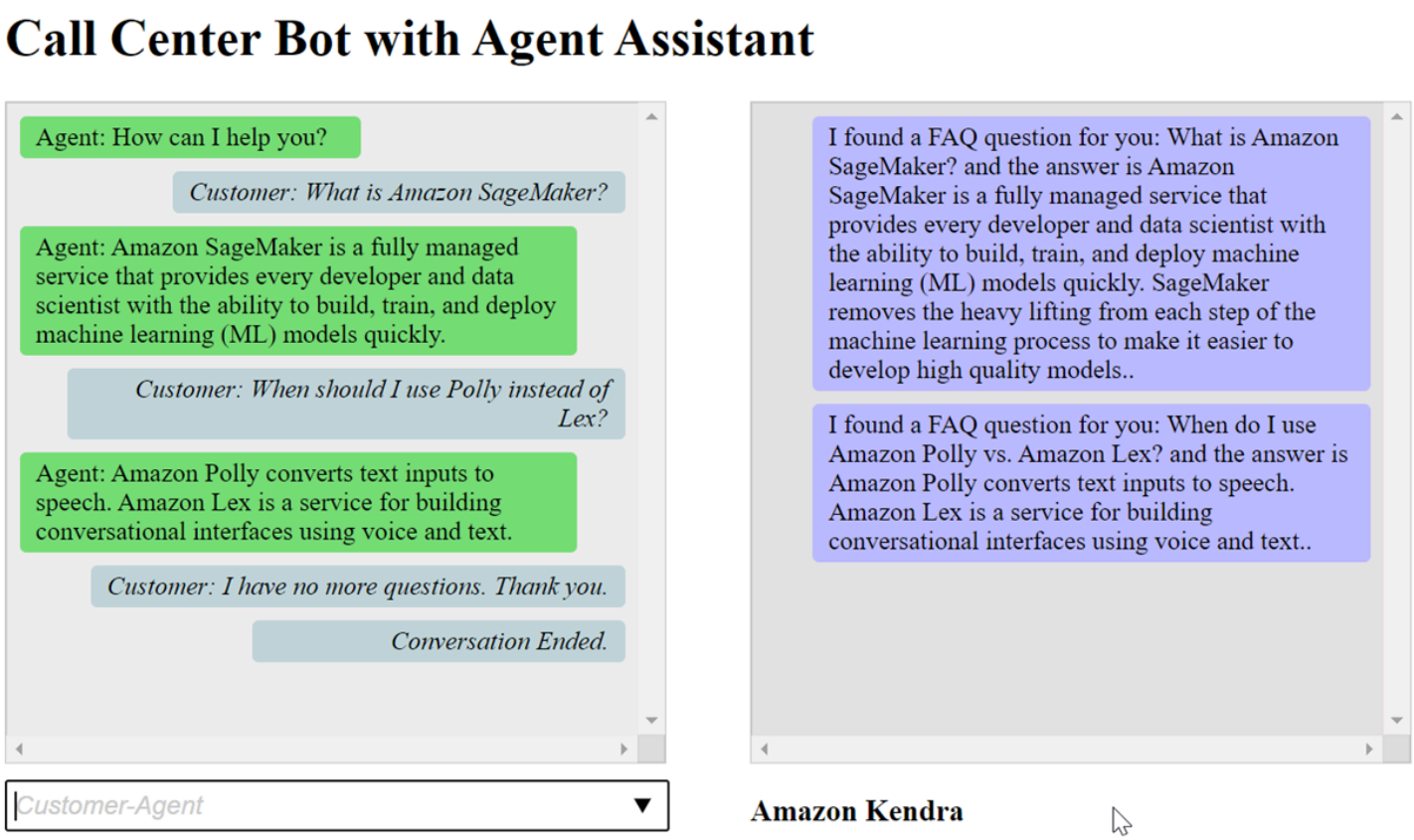
Para implantar seu bot como um aplicativo Web

1. Baixe o repositório em https://github.com/awsdocs/amazon-lex-developer-guide/blob/master/example_apps/agent_assistance_bot/ para o seu computador.
2. Navegue até o repositório baixado e abra o arquivo index.html em um editor.
3. Faça as alterações a seguir.
   a. Na seção `AWS.config.credentials`, insira o nome da sua região e o ID do seu banco de identidades.
   b. Na seção `Amazon Lex runtime parameters`, insira o nome do bot.
   c. Salve o arquivo.

Etapa 6: usar o bot

Para fins de demonstração, você fornece informações ao bot como o cliente e o atendente. Para diferenciar os dois, as perguntas feitas pelo cliente começam com “Cliente:” e as respostas fornecidas pelo atendente começam com “Atendente:”. Você pode escolher uma opção em um menu de entradas sugeridas.

Execute seu aplicativo Web abrindo `index.html` para iniciar uma conversa semelhante à imagem a seguir com seu bot:



A função `pushChat()` no arquivo index.html é explicada abaixo.

```
            var endConversationStatement = "Customer: I have no more questions. Thank
you."
            // If the agent has to send a message, start the message with 'Agent'
            var inputText = document.getElementById('input');
            if (inputText && inputText.value && inputText.value.trim().length > 0 &&
inputText.value[0]=='Agent') {
                showMessage(inputText.value, 'agentRequest','conversation');
                inputText.value = "";
            }
            // If the customer has to send a message, start the message with 'Customer'
            if(inputText && inputText.value && inputText.value.trim().length > 0 &&
inputText.value[0]=='Customer') {
                // disable input to show we're sending it
                var input = inputText.value.trim();
                inputText.value = '...';
                inputText.locked = true;
                customerInput = input.substring(2);
```

```
                // Send it to the Lex runtime
                var params = {
                    botAlias: '$LATEST',
                    botName: 'KendraTestBot',
                    inputText: customerInput,
                    userId: lexUserId,
                    sessionAttributes: sessionAttributes
                };

                showMessage(input, 'customerRequest', 'conversation');
                if(input== endConversationStatement){
                    showMessage('Conversation
 Ended.','conversationEndRequest','conversation');
                }
                lexruntime.postText(params, function(err, data) {
                    if (err) {
                        console.log(err, err.stack);
                        showMessage('Error:  ' + err.message + ' (see console for
 details)', 'lexError', 'conversation1')
                    }

                    if (data &&input!=endConversationStatement) {
                        // capture the sessionAttributes for the next cycle
                        sessionAttributes = data.sessionAttributes;

                            showMessage(data, 'lexResponse', 'conversation1');
                    }
                    // re-enable input
                    inputText.value = '';
                    inputText.locked = false;
                });
            }
            // we always cancel form submission
            return false;
```

Quando você fornece informações como cliente, a API de runtime do Amazon Lex as envia para o Amazon Lex.

A função `showMessage(daText, senderRequest, displayWindow)` exibe a conversa entre o atendente e o cliente na janela de chat. As respostas sugeridas pelo Amazon Kendra são mostradas em uma janela adjacente. A conversa termina quando o cliente diz **`“I have no more questions. Thank you.”`**

Observação: exclua o índice do Amazon Kendra quando ele não estiver em uso.

Migração de um bot

A API do Amazon Lex V2 usa uma arquitetura de informações atualizada que permite o versionamento simplificado de recursos e o suporte a várias linguagens em um bot. Para obter mais informações, consulte o Guia de migração no Guia do desenvolvedor do Amazon Lex V2.

Para usar esses novos atributos, você precisa migrar seu bot. Quando você migra um bot, o Amazon Lex fornece o seguinte:

- A migração copia suas intenções personalizadas e tipos de slots para o bot do Amazon Lex V2.
- É possível adicionar várias linguagens ao mesmo bot do Amazon Lex V2. No Amazon Lex V1, você cria um bot separado para cada linguagem. Você pode migrar vários bots do Amazon Lex V1, cada um usando uma linguagem diferente, para um bot do Amazon Lex V2.
- O Amazon Lex mapeia os tipos e as intenções de slots integrados do Amazon Lex V1 para os tipos e as intenções de slots integrados do Amazon Lex V2. Se não for possível migrar um integrado, o Amazon Lex retornará uma mensagem informando o que fazer a seguir.

O processo de migração não migra o seguinte:

- Aliases
- Índices do Amazon Kendra
- AWS Lambda funções
- Configurações do log de conversação
- Canais de mensagens, como o Slack
- Tags

Para migrar um bot, seu usuário ou função deve ter permissão do IAM para as operações de API do Amazon Lex e do Amazon Lex V2. Para ver as permissões necessárias, consulte Permitir que um usuário migre um bot para o Amazon Lex V2 APIs.

Migração de um bot (Console)

Use o console do Amazon Lex V1 para migrar a estrutura de um bot para um bot do Amazon Lex V2.

Para usar o console para migrar um bot para a API 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. No menu à esquerda, escolha Ferramenta de migração.
3. Na lista de bots, escolha o bot que você deseja migrar e, em seguida, selecione Migrar.
4. Escolha a versão do bot que você deseja migrar e digite o nome do bot para o qual fará a migração. Se você digitar o nome de um bot do Amazon Lex V2 existente, o bot do Amazon Lex V1 será migrado para a linguagem mostrada nos detalhes e substituirá a versão de rascunho da linguagem.
5. Selecione Avançar.
6. Escolha o perfil do IAM que o Amazon Lex usa para executar a versão da API do Amazon Lex V2 do bot. Você pode optar por criar uma função com as permissões mínimas necessárias para executar o bot ou usar um perfil do IAM existente.
7. Escolha Próximo.
8. Revise as configurações de migração. Se elas estiverem corretas, selecione Iniciar migração.

Depois de iniciar o processo de migração, você retornará à página inicial da ferramenta de migração. Monitore o andamento da migração na tabela Histórico. Quando a coluna Status da migração indicar Concluído, a migração estará concluída.

O Amazon Lex usa a operação do `StartImport` na API do Amazon Lex V2 para importar o bot migrado. Você verá uma entrada na tabela do histórico de importação do console do Amazon Lex V2 para cada migração.

Durante a migração, o Amazon Lex pode encontrar recursos no bot que não podem ser migrados. Você recebe uma mensagem de erro ou aviso para cada recurso que não pode ser migrado. Cada mensagem inclui um link para a documentação que explica como resolver o problema.

Migração de uma função do Lambda

O Amazon Lex V2 muda a forma como as funções do Lambda são definidas para um bot. Ele só permite uma função do Lambda em um alias para cada linguagem em um bot. Para obter mais informações sobre como migrar suas funções do Lambda, consulte Migração de uma função do Lambda do Amazon Lex V1 para o Amazon Lex V2.

Mensagens de migração

Durante a migração, o Amazon Lex pode encontrar recursos, como tipos de slots integrados, que não podem ser migrados para o recurso equivalente do Amazon Lex V2. Quando isso acontece, o Amazon Lex retorna uma mensagem de migração que descreve o que aconteceu e fornece um link para a documentação que explica como corrigir o problema de migração. As seções a seguir descrevem os problemas que podem surgir durante a migração de um bot e como corrigi-los.

Tópicos

- Intenção integrada
- Tipo de slot integrado
- Logs de conversa
- Grupos de mensagens
- Prompts e frases
- Outros atributos do Amazon Lex V1

Intenção integrada

Quando você usa uma intenção incorporada que não é compatível com o Amazon Lex V2, a intenção é mapeada para uma intenção personalizada em seu bot do Amazon Lex V2. A intenção personalizada não contém declarações. Para continuar usando a intenção, adicione amostra de declarações.

Tipo de slot integrado

Qualquer slot migrado que use um tipo de slot que não tenha suporte no Amazon Lex V2 não receberá um valor de tipo de slot. Para usar esse slot:

- Crie um tipo de slot personalizado
- Adicione valores do tipo de slot que sejam esperados para o tipo de slot
- Atualize o slot para usar o novo tipo de slot personalizado

Logs de conversa

A migração não atualiza as configurações do log de conversas do bot do Amazon Lex V2.

Para configurar logs de conversa

1. Abra o console do Amazon Lex V2 em https://console.aws.amazon.com/lexv2.
2. Na lista de bots, escolha o bot cujos logs de conversas você deseja configurar.
3. No menu à esquerda, selecione Aliases e, em seguida, escolha um alias na lista.
4. Na seção Logs de conversas, selecione Gerenciar logs de conversas para configurar logs de conversas para o alias do bot.

Grupos de mensagens

O Amazon Lex V2 oferece suporte a apenas uma mensagem e duas mensagens alternativas por grupo de mensagens. Se você tiver mais de três mensagens por grupo de mensagens em um bot do Amazon Lex V1, somente as três primeiras mensagens serão migradas. Para usar mais mensagens em um grupo de mensagens, use uma função do Lambda para gerar várias mensagens.

Prompts e frases

O Amazon Lex V2 usa um mecanismo diferente para prompts de acompanhamento, esclarecimento e desligamento.

Para prompts de acompanhamento, use a transferência de contexto para mudar para uma intenção diferente após o atendimento.

Por exemplo, suponhamos que você tenha a intenção de reservar um aluguel de carro configurado para retornar um contexto de saída chamado `book_car_fulfilled`. Quando a intenção é atendida, o Amazon Lex define a variável de contexto de saída como `book_car_fulfilled`. Como `book_car_fulfilled` é um contexto ativo, uma intenção com `book_car_fulfilled` como contexto de entrada é considerada para reconhecimento, desde que a declaraçãoo do usuário seja reconhecida como uma tentativa de obter essa intenção. Você pode usar isso para intenções que só façam sentido depois de reservar um carro, como enviar um recibo por e-mail ou modificar uma reserva.

O Amazon Lex V2 não oferece suporte a prompts de esclarecimento e frases suspensas (instruções de cancelamento). Os bots do Amazon Lex V2 contêm uma intenção de fallback padrão que é invocada se não houver correspondência com nenhuma intenção. Para enviar um prompt de esclarecimento com novas tentativas, configure uma função do Lambda e ative o hook do código de diálogo na intenção de fallback. A função do Lambda pode gerar um prompt de esclarecimento como resposta e o valor da nova tentativa em um atributo de sessão. Se o valor da nova tentativa

exceder o número máximo de tentativas, você poderá gerar uma frase de desligamento e encerrar a conversa.

Outros atributos do Amazon Lex V1

A ferramenta de migração oferece suporte somente a migração dos bots do Amazon Lex V1 e suas intenções, tipos de slots e slots subjacentes. Para obter outros atributos, consulte os tópicos a seguir na documentação do Amazon Lex V2.

- Aliases de bots: aliases
- Canais de bots: implantação de um bot do Amazon Lex V2 em uma plataforma de mensagens
- Configurações do log de conversas: monitoramento com logs de conversas
- Índices Amazon Kendra: AMAZON. KendraSearchIntent
- Funções do Lambda: uso de uma função do AWS Lambda
- Tags: Marcação de recursos

Migração de uma função do Lambda do Amazon Lex V1 para o Amazon Lex V2

O Amazon Lex V2 permite somente uma função do Lambda para cada linguagem em um bot. A função do Lambda e suas configurações são definidas para o alias de bot que você usa no runtime.

A função do Lambda é invocada para todas as intenções nessa linguagem se os hooks de diálogo e código de atendimento estiverem habilitados para a intenção.

As funções do Lambda do Amazon Lex V2 têm um formato de mensagem de entrada e saída diferente do Amazon Lex V1. Essas são as diferenças no formato de entrada da função do Lambda.

- O Amazon Lex V2 substitui as estruturas `currentIntent` e `alternativeIntents` pela estrutura `interpretations`. Cada interpretação contém uma intenção, a pontuação de confiança da NLU para a intenção e uma análise de sentimento opcional.
- O Amazon Lex V2 move o `activeContexts`, `sessionAttributes` no Amazon Lex V1 para a estrutura unificada `sessionState`. Essa estrutura fornece informações sobre o estado atual da conversa, incluindo o ID da solicitação de origem.
- O Amazon Lex V2 não retorna o `recentIntentSummaryView`. Em vez disso, use as informações na estrutura `sessionState`.

- A entrada do Amazon Lex V2 fornece o `botId` e o `localeId` no atributo `bot`.
- A estrutura de entrada contém um atributo `inputMode` que fornece informações sobre o tipo de entrada: texto, fala ou DTMF.

Essas são as diferenças no formato de entrada da função do Lambda.

- As estruturas `activeContexts` e `sessionAttributes` no Amazon Lex V1 são substituídas pela estrutura `sessionState` no Amazon Lex V2.
- A `recentIntentSummaryView` não é incluída na saída:
- A estrutura `dialogAction` do Amazon Lex V1 é dividida em duas estruturas, `dialogAction` que faz parte da estrutura `sessionState`, e `messages` que é necessária quando `dialogAction.type` é `ElicitIntent`. O Amazon Lex escolhe mensagens dessa estrutura para mostrar ao usuário.

Quando você cria um bot com o Amazon Lex V2 APIs, há somente uma função Lambda por alias de bot por linguagem, em vez de uma função Lambda para cada intenção. Se quiser continuar usando funções separadas, você pode criar uma função de roteamento que ativa uma função separada para cada intenção. A seguir há uma função de roteamento que você pode usar ou modificar para seu aplicativo.

```
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
```

Lista de campos atualizados

As tabelas a seguir fornecem informações detalhadas sobre os campos atualizados na solicitação e na resposta do Amazon Lex V2. Você pode usar essas tabelas para mapear campos entre as versões.

Solicitação

Os campos a seguir foram atualizados no formato de solicitação da função do Lambda.

Contextos ativos

A estrutura `activeContexts` agora faz parte da estrutura `sessionState`.

Estrutura V1	Estrutura V2
activeContexts	sessionState.activeContexts
Contextos ativos [*]. timeToLive	SessionState.ActiveContexts [*]. timeToLive
Contextos ativos [*]. timeToLive. timeToLiveInSeconds	SessionState.ActiveContexts [*]. timeToLive. timeToLiveInSeconds
Contextos ativos [*]. timeToLive. turnsToLive	SessionState.ActiveContexts [*]. timeToLive. turnsToLive
activeContexts[*].name	sessionState.activeContexts[*].name
activeContexts[*].parameters	sessionState.activeContexts[*].contextAttributes

Intenções alternativas

A lista de interpretações do índice 1 ao N contém a lista de intenções alternativas previstas pelo Amazon Lex V2, junto com suas pontuações de confiança. O `recentIntentSummaryView`

é removido da estrutura de solicitações no Amazon Lex V2. Para ver os detalhes do `recentIntentSummaryView`, use a operação GetSession.

Estrutura V1	Estrutura V2
alternativeIntents	interpretations[1:*]
recentIntentSummaryExibir	N/D

Bot

No Amazon Lex V2, bots e aliases têm identificadores. O ID do bot faz parte da entrada do codehook. O ID do alias está incluído, mas não o nome do alias. O Amazon Lex V2 oferece suporte a várias localidades para o mesmo bot, portanto, a ID de localidade é incluída.

Estrutura V1	Estrutura V2
bot	bot
bot.name	bot.name
N/D	bot.id
bot.alias	N/D
N/D	bot.aliasId
bot.version	bot.version
N/D	bot.localeId

Intenção atual

A `sessionState.intent` estrutura contém os detalhes da intenção ativa. O Amazon Lex V2 também retorna uma lista de todas as intenções, incluindo intenções alternativas, na estrutura `interpretations`. O primeiro elemento na lista de interpretações é sempre o mesmo que `sessionState.intent`.

Estrutura V1	Estrutura V2
currentIntent	sessionState.intent OU interpretations[0].intent
currentIntent.name	sessionState.intent.name OU interpretations[0] .intent.name
CurrentIntent. nluConfidenceScore	interpretations[0].nluConfidence.score

Ação do diálogo

O campo `confirmationStatus` agora faz parte da estrutura `sessionState`.

Estrutura V1	Estrutura V2
currentIntent.confirmationStatus	sessionState.intent.confirmationState OU interpretations[0].intent.confirmationState
N/D	sessionState.intent.state OU interpretations[*] .intent.state

Amazon Kendra

O campo `kendraResponse` agora faz parte das estruturas `sessionState` e `interpretations`.

Estrutura V1	Estrutura V2
kendraResponse	sessionState.intent.kendraResponse OU interpretations[0].intent.kendraResponse

Sentimento

A estrutura `sentimentResponse` é movida para a nova estrutura `interpretations`.

Estrutura V1	Estrutura V2
sentimentResponse	interpretations[0].sentimentResponse

Estrutura V1	Estrutura V2
sentimentResponse.sentimentLabel	interpretations[0].sentimentResponse.sentiment
sentimentResponse.sentimentScore	interpretations[0].sentimentResponse.sentimentScore

Slots

O Amazon Lex V2 fornece um único objeto `slots` dentro da estrutura `sessionState.intent` que contém os valores resolvidos, o valor interpretado e o valor original do que o usuário disse. O Amazon Lex V2 também oferece suporte a slots de vários valores definindo o `slotShape` como `List` e definindo a lista `values`. Os slots de valor único são compatíveis pelo campo `value`, presume-se que sua forma seja `Scalar`.

Estrutura V1	Estrutura V2
currentIntent.slots	sessionState.intent.slots OU interpretations[0].intent.slots
currentIntent.slots[*].value	sessionState.intent.slots[*].value.interpretedValue OU interpretations[0].intent.slots[*].value.interpretedValue
N/D	sessionState.intent.slots[*].value.shape OU interpretations[0].intent.slots[*].shape
N/D	sessionState.intent.slots[*].values OU interpretations[0].intent.slots[*].values
currentIntent.slotDetails	sessionState.intent.slots OU interpretations[0].intent.slots
currentIntent.slotDetails[*].resolutions	sessionState.intent.slots[*].resolvedValues OU interpretations[0].intent.slots[*].resolvedValues
currentIntent.slotDetails[*].originalValue	sessionState.intent.slots[*].originalValue OU interpretations[0].intent.slots[*].originalValue

Outros

O campo `sessionId` do Amazon Lex V2 é igual ao campo `userId` no Amazon Lex V1. O Amazon Lex V2 também envia o nome `inputMode` do chamador: texto, DTMF ou fala.

Estrutura V1	Estrutura V2
userId	sessionId
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion
sessionAttributes	sessionState.sessionAttributes
requestAttributes	requestAttributes
N/D	inputMode
N/D	originatingRequestId

Resposta

Os campos a seguir foram atualizados no formato de solicitação da função do Lambda.

Contextos ativos

A estrutura `activeContexts` é movida para a estrutura `sessionState`.

Estrutura V1	Estrutura V2
activeContexts	sessionState.activeContexts
Contextos ativos [*]. timeToLive	SessionState.ActiveContexts [*]. timeToLive

Estrutura V1	Estrutura V2
Contextos ativos [*]. timeToLive. timeToLiveInSeconds	SessionState.ActiveContexts [*]. timeToLive. timeToLiveInSeconds
Contextos ativos [*]. timeToLive. turnsToLive	SessionState.ActiveContexts [*]. timeToLive. turnsToLive
activeContexts[*].name	sessionState.activeContexts[*].name
activeContexts[*].parameters	sessionState.activeContexts[*].contextAttributes

Ação do diálogo

A estrutura `dialogAction` é movida para a estrutura `sessionState`. Agora você pode especificar múltiplas mensagens em uma ação de diálogo, e a estrutura `genericAttachments` agora é a estrutura `imageResponseCard`.

Estrutura V1	Estrutura V2
dialogAction	sessionState.dialogAction
dialogAction.type	sessionState.dialogAction.type
DialogAction. slotToElicit	SessionState.Intent.DialogAction. slotToElicit
dialogAction.type.fulfillmentState	sessionState.intent.state
dialogAction.message	messages
dialogAction.message.contentType	messages[*].contentType
dialogAction.message.content	messages[*].content
dialogAction.responseCard	mensagens [*]. imageResponseCard
dialogAction.responseCard.version	N/D
dialogAction.responseCard.contentType	messages[*].contentType

Estrutura V1	Estrutura V2
dialogAction.responseCard.genericAttachments	N/D
dialogAction.responseCard.genericAttachments[*].title	mensagens [*]. imageResponseCard.título
dialogAction.responseCard.genericAttachments[*].subTitle	mensagens [*]. imageResponseCard.subtítulo
dialogAction.responseCard.genericAttachments[*].imageUrl	mensagens [*]. imageResponseCard.URL da imagem
dialogAction.responseCard.genericAttachments[*].buttons	mensagens [*]. imageResponseCard.botões
dialogAction.responseCard.genericAttachments[*].buttons[*].value	mensagens [*]. imageResponseCard.botões [*] .valor
dialogAction.responseCard.genericAttachments[*].buttons[*].text	mensagens [*]. imageResponseCard.botões [*] .texto
DialogAction. kendraQueryRequestCarga útil	DialogAction. kendraQueryRequestCarga útil
DialogAction. kendraQueryFilterCadeia	DialogAction. kendraQueryFilterCadeia

Intenções e slots

Os campos de intenção e slots que faziam parte da estrutura `dialogAction` agora fazem parte da estrutura `sessionState`.

Estrutura V1	Estrutura V2
dialogAction.intentName	sessionState.intent.name
dialogAction.slots	sessionState.intent.slots
dialogAction.slots[*].key	sessionState.intent.slots[*].key

Estrutura V1	Estrutura V2
dialogAction.slots[*].value	sessionState.intent.slots[*].value.interpretedValue
N/D	sessionState.intent.slots[*].value.shape
N/D	sessionState.intent.slots[*].values

Outros

A estrutura `sessionAttributes` agora faz parte da estrutura `sessionState`. A estrutura `recentIntentSummaryReview` não existe mais.

Estrutura V1	Estrutura V2
sessionAttributes	sessionState.sessionAttributes
recentIntentSummaryExibir	N/D

Segurança no Amazon Lex

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada 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 isto 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. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon Lex, consulte Serviços da AWS no escopo pelo programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Também existe a responsabilidade por outros fatores, incluindo a confidencialidade de dados, os requisitos da organização e as leis e regulamentos aplicáveis.

Essa documentação ajudará você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon Lex. Os tópicos a seguir mostram como configurar o Amazon Lex para atender aos seus objetivos de segurança e compatibilidade. Você também aprenderá como usar outros serviços da AWS que podem ajudar a monitorar e proteger seus recursos do Amazon Lex.

Tópicos

- Proteção de dados no Amazon Lex
- Gerenciamento de identidade e acesso para o Amazon Lex
- Monitoramento no Amazon Lex
- Validação de conformidade para o Amazon Lex
- Resiliência no Amazon Lex
- Segurança da infraestrutura no Amazon Lex

Proteção de dados no Amazon Lex

O Amazon Lex coleta o conteúdo do cliente para solução de problemas e para ajudar a melhorar o serviço. O conteúdo do cliente é protegido por padrão. Você pode excluir o conteúdo para clientes individuais usando a API do Amazon Lex.

O Amazon Lex armazena quatro tipos de conteúdo:

- Amostra de declarações, que é usada para criar e treinar um bot
- Declarações do cliente de usuários que interagem com o bot
- Atributos de sessão, que fornecem informações específicas do aplicativo para a duração de uma interação do usuário com um bot
- Atributos de solicitação, que contêm informações que se aplicam a uma única solicitação a um bot

Qualquer bot do Amazon Lex criado para uso por crianças é regido pelo Children's Online Privacy Protection Act (COPPA). Você informa o Amazon Lex que o bot está sujeito ao COPPA usando o console ou a API do Amazon Lex para definir o campo `childDirected` como `true`. Quando o campo `childDirected` é definido como `true`, nenhuma declaração de usuário é armazenada.

Tópicos

- Criptografia em repouso
- Criptografia em trânsito
- Gerenciamento de chaves

Criptografia em repouso

O Amazon Lex criptografa as declarações de usuário que armazena.

Tópicos

- Amostra de declarações
- Declarações do cliente
- Atributos da sessão
- Atributos de solicitação

Amostra de declarações

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.

Declarações do cliente

O Amazon Lex criptografa declarações que os usuários enviam para seu bot, a menos que o campo `childDirected` seja definido como `true`.

Quando o campo `childDirected` é definido como `true`, nenhuma declaração de usuário é armazenada.

Quando o campo `childDirected` é definido como `false` (o padrão), declarações do usuário são criptografadas e armazenadas por 15 dias para uso com a operação GetUtterancesView. Para excluir declarações armazenadas para um usuário específico, use a operação DeleteUtterances.

Quando seu bot aceita entrada de voz, a entrada é armazenada indefinidamente. O Amazon Lex o usa para melhorar a capacidade do seu bot de responder a uma entrada do usuário.

Use a operação DeleteUtterances para excluir declarações armazenadas por um usuário específico.

Atributos da sessão

Atributos da sessão contêm informações específicas do aplicativo que são passadas entre o Amazon Lex e os aplicativos clientes. O Amazon Lex transmite atributos de sessão para todas AWS Lambda as funções 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 o aplicativo 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 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 usa o protocolo HTTPS para se comunicar com o aplicativo cliente. Ele usa assinaturas HTTPS e AWS para se comunicar com outros serviços, como o Amazon Polly AWS Lambda e em nome do seu aplicativo.

Gerenciamento de chaves

O Amazon Lex protege o conteúdo de uso não autorizado com chaves internas.

Gerenciamento de identidade e acesso para o Amazon Lex

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. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticar com identidades
- Gerenciar o acesso usando políticas
- Como o Amazon Lex funciona com o IAM
- Exemplos de políticas baseadas em identidade para o Amazon Lex
- AWS políticas gerenciadas para o Amazon Lex
- Uso de perfis vinculadas ao serviço para o Amazon Lex
- Solução de problemas de identidade e acesso da Amazon Lex

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Amazon Lex.

Usuário do serviço: se você usar o serviço do Amazon Lex para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões necessárias. À medida que mais atributos do Amazon Lex forem usados para realizar o trabalho, talvez sejam necessárias permissões adicionais. Compreenda como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não puder acessar um atributo no Amazon Lex, consulte Solução de problemas de identidade e acesso da Amazon Lex.

Administrador do serviço: se você for o responsável pelos recursos do Amazon Lex em sua empresa, provavelmente terá acesso total ao Amazon Lex. Cabe a você determinar quais atributos e recursos do Amazon Lex os usuários do serviço deverão acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como a empresa pode usar o IAM com o Amazon Lex, consulte Como o Amazon Lex 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. Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex.

Autenticar 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 designar solicitações por conta própria, consulte Versão 4 do AWS Signature para solicitações de API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser necessário fornecer 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 do usuário do AWS IAM Identity Center e Usar a autenticação multifator da AWS no IAM 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, é recomendável 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 do 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, é recomendável 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, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte Alternar 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 IAMAdminse 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 Casos de uso para usuários do IAM 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. Para assumir temporariamente uma função do IAM no AWS Management Console, você pode alternar de um usuário para uma função do IAM (console). 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 usar perfis, consulte Métodos para assumir um perfil no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, é possível 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 por ele. Para ter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidade de terceiros (federação) no Guia do usuário do IAM. Se 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 do 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 conhecer a diferença entre perfis e políticas baseadas em recurso 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 na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado o 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 Sessões de acesso direto.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para executar 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 o perfil 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 perfis vinculados ao serviço.

- Aplicativos em execução na Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo solicitações AWS CLI de AWS API. Isso é preferível ao armazenamento de chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia do usuário do IAM.

Gerenciar o 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 perfis não têm permissões. Para conceder permissão aos usuários 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 assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente 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, 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 Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas 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 compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo 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 diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a 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 recurso 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 recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas 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 Organizations e SCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.
- Políticas de controle de recursos (RCPs) — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo a Usuário raiz da conta da AWS, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- 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 recursos. 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 funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon Lex, entenda que atributos do IAM estão disponíveis para uso com o Amazon Lex.

Atributos do IAM que você pode usar com o Amazon Lex

Atributo do IAM	Suporte ao Amazon Lex
Políticas baseadas em identidade	Sim
Políticas baseadas em recurso	Não
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Parcial
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Sim

Para ter uma visão de alto nível de como o Amazon Lex 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

Compatível com 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 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 Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e 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 elemento de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para o Amazon Lex

Para visualizar exemplos de políticas baseadas em identidade do Amazon Lex, consulte Exemplos de políticas baseadas em identidade para o Amazon Lex.

Políticas baseadas em recursos no Amazon Lex

Compatibilidade com políticas baseadas em recursos: não

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 compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo 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

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um administrador do IAM na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações de políticas para o Amazon Lex

Compatível 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 podem ser usadas 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, consulte Ações definidas pelo Amazon Lex na Referência de autorização do serviço.

As ações de política no Amazon Lex 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"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "lex:Describe*"
```

Recursos de políticas para o Amazon Lex

Compatível 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 suporte a permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

O ARN de um recurso de bot do Amazon Lex tem o seguinte formato.

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

Para obter mais informações sobre o formato de ARNs, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces.

Por exemplo, para especificar o bot `OrderFlowers` em sua instrução, use o seguinte ARN.

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:OrderFlowers"
```

Para especificar todos os bots que pertencem a uma conta específica, use o caractere curinga (*):

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:*"
```

Algumas ações do Amazon Lex, como as que servem para a criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, é necessário usar o caractere curinga (*).

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon Lex e seus ARNs, consulte Recursos definidos pelo Amazon Lex na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Lex.

Chaves de condição de política para o Amazon Lex

Compatível com chaves de condição de política específicas de serviço: 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 `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 de `Condition` em uma declaração ou várias chaves em um único elemento de `Condition`, 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 recurso 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, consulte Chaves de condição do Amazon Lex na Referência de autorização do serviço. Para saber com quais ações e recursos é possível usar a chave de condição, consulte Ações definidas pelo Amazon Lex.

A tabela a seguir mostra as chaves de condição do Amazon Lex aplicáveis aos recursos do Amazon Lex. Você pode incluir essas chaves em elementos `Condition` de uma política de permissões do IAM.

ACLs no Amazon Lex

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com o Amazon Lex

Compatível com ABAC (tags em políticas): parcial

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define as 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. Marcar 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 em que 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 for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobre o ABAC, consulte Definir permissões com autorização do 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.

É possível associar tags a determinados tipos de recursos do Amazon Lex para autorização. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de

uma política usando as chaves de condição `lex:ResourceTag/${TagKey}`, `aws:RequestTag/${TagKey}` ou `aws:TagKeys`.

Para ter mais informações sobre recursos de marcação do Amazon Lex, consulte Marcação de seus recursos do Amazon Lex.

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Uso de uma tag para acessar um recurso.

A tabela a seguir lista as ações e os tipos de recursos correspondentes para o controle de acesso baseado em tags. Cada ação é autorizada com base nas tags associadas ao tipo de recurso correspondente.

Ação	Tipo de recurso	Chaves de condição	Observações
CreateBotVersion	bot	`lex:ResourceTag`	
DeleteBot	bot	`lex:ResourceTag`	
DeleteBotAlias	alias	`lex:ResourceTag`	
DeleteBotChannelAssociation	channel	`lex:ResourceTag`	
DeleteBotVersion	bot	`lex:ResourceTag`	
DeleteSession	bot ou alias	`lex:ResourceTag`	Usa as tags associadas ao bot quando o alias é definido como `$LATEST`. Usa as tags associadas ao alias especificado quando usado com outros aliases.
DeleteUtterances	bot	`lex:ResourceTag`	
GetBot	bot ou alias	`lex:ResourceTag`	Usa as tags associadas ao bot

Ação	Tipo de recurso	Chaves de condição	Observações
			quando `versionOrAlias` é definido como `$LATEST` ou versão numérica. Usa as tags associadas ao alias especificado quando usado com aliases
GetBotAlias	alias	`lex:ResourceTag`	
GetBotChannelAssociation	canal	`lex:ResourceTag`	
GetBotChannelAssociations	canal	`lex:ResourceTag`	Usa as tags associadas ao bot quando o alias é definido como “-”. Usa as tags associadas ao alias especificado quando um alias de bot é especificado
GetBotVersions	bot	`lex:ResourceTag`	
GetExport	bot	`lex:ResourceTag`	
GetSession	bot ou alias	`lex:ResourceTag`	Usa as tags associadas ao bot quando o alias é definido como `$LATEST`. Usa as tags associadas ao alias especificado quando usado com outros aliases.

Ação	Tipo de recurso	Chaves de condição	Observações
GetUtterancesView	bot	`lex:ResourceTag`	
ListTagsForResource	bot, alias ou canal	`lex:ResourceTag`	
PostContent	bot ou alias	`lex:ResourceTag`	Usa as tags associadas ao bot quando o alias é definido como `$LATEST`. Usa as tags associadas ao alias especificado quando usado com outros aliases.
PostText	bot ou alias	`lex:ResourceTag`	Usa as tags associadas ao bot quando o alias é definido como `$LATEST`. Usa as tags associadas ao alias especificado quando usado com outros aliases.
PutBot	bot	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutBotAlias	alias	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	

Ação	Tipo de recurso	Chaves de condição	Observações
PutSession	bot ou alias	`lex:ResourceTag`	Usa as tags associadas ao bot quando o alias é definido como `$LATEST`. Usa as tags associadas ao alias especificado quando usado com outros aliases.
StartImport	bot	`lex:ResourceTag`	Baseia-se na política de acesso para a operação `PutBot`. As tags e permissões específicas para a operação `StartImport` são ignoradas.
TagResource	bot, alias ou canal	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
UntagResource	bot, alias ou canal	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	

Usar credenciais temporárias com o Amazon Lex

Compatível com credenciais temporárias: sim

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 do IAM (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 obter mais informações, consulte Credenciais de segurança temporárias no IAM.

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de AWS STS API, como AssumeRoleou GetFederationToken.

Permissões de entidades principais entre serviços para o Amazon Lex

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

Quando você usa um usuário ou uma função do IAM para realizar ações em AWS, você é considerado um 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 Sessões de acesso direto.

Perfis de serviço para o Amazon Lex

Compatível com perfis de serviço: sim

O perfil de serviço é um perfil do IAM que um serviço assume para executar 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.

> **Warning**
>
> A alteração das permissões de um perfil de serviço pode interromper a funcionalidade do Amazon Lex. Edite perfis de serviço somente quando o Amazon Lex fornecer orientação para isso.

Escolher uma perfil do IAM no Amazon Lex

O Amazon Lex usa funções vinculadas a serviços para chamar o Amazon Comprehend e o Amazon Polly. Ele usa permissões em nível de recurso em suas AWS Lambda funções para invocá-las.

Você deve fornecer uma perfil do IAM para habilitar a marcação de conversa. Para obter mais informações, consulte Criar um perfil e políticas do IAM para logs de conversa.

Perfis vinculados ao serviço para o Amazon Lex

Compatibilidade com perfis vinculados a serviços: sim

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir o perfil 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 perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar funções vinculadas ao serviço do Amazon Lex, consulte Uso de perfis vinculadas ao serviço para o Amazon Lex.

Exemplos de políticas baseadas em identidade para o Amazon Lex

Por padrão, usuários e perfis não têm permissão para criar ou modificar recursos do Amazon Lex. Eles também não podem realizar tarefas usando a AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) ou. Para conceder permissão aos usuários 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 assumir os perfis.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Amazon Lex, incluindo o formato de cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição para o Amazon Lex na Referência de autorização de serviço. ARNs

Tópicos

- Práticas recomendadas de política
- Usar o console do Amazon Lex
- Permitir que os usuários visualizem suas próprias permissões
- Excluir todos os bots do Amazon Lex
- Permitir que um usuário migre um bot para o Amazon Lex V2 APIs
- Uso de uma tag para acessar um recurso

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon Lex em sua conta. Essas ações podem incorrer em custos para sua 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 recursos 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 escrever 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 cem verificações de política e recomendações práticas para ajudar 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 obter 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

Para acessar o console da Amazon Lex, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon Lex em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva 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 à operação de API que estiverem tentando executar.

AWS aborda muitos casos de uso comuns fornecendo políticas autônomas do IAM que são criadas e administradas pela AWS. Essas políticas são chamadas de políticas gerenciadas pela AWS. As políticas gerenciadas pela AWS facilitam a atribuição de permissões apropriadas a usuários, grupos e funções em comparação com a elaboração de suas próprias políticas. Para obter mais informações, consulte Políticas gerenciadas pela AWS no Guia do usuário do IAM.

As seguintes políticas AWS gerenciadas, que você pode anexar a grupos e funções em sua conta, são específicas do Amazon Lex:

- AmazonLexReadOnly— Concede acesso somente para leitura aos recursos do Amazon Lex.
- AmazonLexRunBotsOnly— Concede acesso para executar bots conversacionais do Amazon Lex.
- AmazonLexFullAccess— Concede acesso total para criar, ler, atualizar, excluir e executar todos os recursos do Amazon Lex. Também concede a capacidade de associar funções do Lambda com nomes que começam com `AmazonLex` com intenções do Amazon Lex.

> **Note**
>
> Você pode analisar essas políticas de permissões fazendo login no console do IAM; e procurando políticas específicas.

A AmazonLexFullAccesspolítica não concede ao usuário permissão para usar a `KendraSearchIntent` intenção para consultar um índice da Amazon Kendra. Para consultar um índice, você deve adicionar permissões adicionais à política. Para ver as permissões necessárias, consulte Política do IAM para pesquisa Amazon Kendra.

Além disso, você pode criar políticas do IAM personalizadas para conceder permissões para ações de API do Amazon Lex. É possível anexar essas políticas personalizadas a grupos ou perfis do IAM que exijam essas permissões.

Para obter detalhes sobre políticas gerenciadas pela AWS para o Amazon Lex, consulte AWS políticas gerenciadas para o Amazon Lex.

Permitir que os 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": "*"
        }
    ]
}
```

Excluir todos os bots do Amazon Lex

Este exemplo de política concede a um usuário em sua conta da AWS permissão para excluir qualquer bot em sua conta.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
            "Resource": [
                "*"
            ]
        }
```

```
    ]
}
```

Permitir que um usuário migre um bot para o Amazon Lex V2 APIs

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 APIs e veja a lista de migrações e seu progresso.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "startMigration",
            "Effect": "Allow",
            "Action": "lex:StartMigration",
            "Resource": "arn:aws:lex:<Region>:<123456789012>:bot:*"
        },
        {
            "Sid": "passRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<123456789012>:role/<v2 bot role>"
        },
        {
            "Sid": "allowOperations",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBot",
                "lex:CreateIntent",
                "lex:UpdateSlot",
                "lex:DescribeBotLocale",
                "lex:UpdateBotAlias",
                "lex:CreateSlotType",
                "lex:DeleteBotLocale",
                "lex:DescribeBot",
                "lex:UpdateBotLocale",
                "lex:CreateSlot",
                "lex:DeleteSlot",
                "lex:UpdateBot",
                "lex:DeleteSlotType",
                "lex:DescribeBotAlias",
                "lex:CreateBotLocale",
                "lex:DeleteIntent",
```

```
                "lex:StartImport",
                "lex:UpdateSlotType",
                "lex:UpdateIntent",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "lex:DescribeCustomVocabulary",
                "lex:DescribeCustomVocabularyMetadata"
            ],
            "Resource": [
                "arn:aws:lex:<Region>:<123456789012>:bot/*",
                "arn:aws:lex:<Region>:<123456789012>:bot-alias/*/*"
            ]
        },
        {
            "Sid": "showBots",
            "Effect": "Allow",
            "Action": [
                "lex:CreateUploadUrl",
                "lex:ListBots"
            ],
            "Resource": "*"
        },
        {
            "Sid": "showMigrations",
            "Effect": "Allow",
            "Action": [
                "lex:GetMigration",
                "lex:GetMigrations"
            ],
            "Resource": "*"
        }
    ]
}
```

Uso de uma tag para acessar um recurso

Este exemplo de política concede a um usuário ou uma função em sua conta da AWS a permissão para usar a operação `PostText` com qualquer recurso marcado com a chave **`Department`** e o valor **`Support`**.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lex:PostText",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "lex:ResourceTag/Department": "Support"
                }
            }
        }
    ]
}
```

AWS políticas gerenciadas para o Amazon Lex

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 elas estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são 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) for lançada ou novas operações de API forem 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` às 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 e Amazon Lex V2.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `lex`: acesso somente para leitura aos recursos do Amazon Lex e do Amazon Lex V2 no serviço de criação de modelos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:GetBot",
                "lex:GetBotAlias",
                "lex:GetBotAliases",
                "lex:GetBots",
                "lex:GetBotChannelAssociation",
                "lex:GetBotChannelAssociations",
                "lex:GetBotVersions",
                "lex:GetBuiltinIntent",
                "lex:GetBuiltinIntents",
                "lex:GetBuiltinSlotTypes",
                "lex:GetIntent",
                "lex:GetIntents",
                "lex:GetIntentVersions",
                "lex:GetSlotType",
                "lex:GetSlotTypes",
                "lex:GetSlotTypeVersions",
                "lex:GetUtterancesView",
                "lex:DescribeBot",
                "lex:DescribeBotAlias",
                "lex:DescribeBotChannel",
                "lex:DescribeBotLocale",
                "lex:DescribeBotVersion",
                "lex:DescribeExport",
```

```
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBots",
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotChannels",
                "lex:ListBotVersions",
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gerenciada pela AWS: AmazonLexRunBotsOnly

É possível anexar a política `AmazonLexRunBotsOnly` às identidades do IAM.

Essa política concede permissões somente de leitura que permitem o acesso para executar bots conversacionais do Amazon Lex e do Amazon Lex V2.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `lex`: acesso somente para leitura a todas as ações no runtime do Amazon Lex e do Amazon Lex V2.

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

```
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gerenciada pela AWS: AmazonLexFullAccess

É possível anexar a política `AmazonLexFullAccess` às identidades do IAM.

Essa política concede permissões administrativas que permitem ao usuário criar, ler, atualizar e excluir recursos do Amazon Lex e do Amazon Lex V2 e executar bots de conversação do Amazon Lex e do Amazon Lex V2.

Detalhes de permissões

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 e do Amazon Lex V2.
- `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 os diretores descrevam AWS KMS chaves e aliases.
- `lambda`— Permite que os diretores listem AWS Lambda funções 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": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
```

```
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Atualizações do Amazon Lex para políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Amazon Lex desde que esse serviço começou a monitorar essas mudanças. Para obter alertas automáticos sobre alterações feitas nesta página, inscreva-se no feed RSS na página Histórico da documentação do Amazon Lex do Amazon Lex.

Alteração	Descrição	Data
AmazonLexFullAccess: atualizar para uma política existente	O Amazon Lex 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 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
AmazonLexRunBotsOnly: atualizar para uma política existente	O Amazon Lex adicionou novas permissões para permitir acesso somente	18 de agosto de 2021

Alteração	Descrição	Data
	para leitura às operações do serviço de runtime do Amazon Lex V2.	
O Amazon Lex passou a monitorar alterações	O Amazon Lex passou a monitorar as alterações para as políticas gerenciadas pela AWS .	18 de agosto de 2021

Uso de perfis vinculadas ao serviço para o Amazon Lex

O Amazon Lex usa AWS Identity and Access Management funções vinculadas a serviços (IAM). Um perfil vinculado ao serviço é um tipo especial de perfil do IAM vinculado diretamente ao Amazon Lex. As funções vinculadas ao serviço são predefinidas pelo Amazon Lex 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 porque você não precisa adicionar as permissões necessárias manualmente. O Amazon Lex define as permissões dos perfis vinculados ao serviço e, a não ser que esteja definido de outra forma, somente o Amazon Lex poderá assumir os perfis. As permissões definidas incluem as políticas de confiança e de permissões, e essa política de permissões não pode ser anexada a nenhuma outra entidade do IAM.

Você só pode excluir um perfil vinculado a serviço depois de excluir os recursos relacionados. Isso protege seus recursos do Amazon Lex, pois você não pode remover por engano as permissões para acessar os recursos.

Permissões de perfil vinculado ao serviço para o Amazon Lex

O Amazon Lex usa dois perfis vinculados ao serviço:

- AWSServiceRoleForLexBots— O Amazon Lex usa essa função vinculada ao serviço para invocar o Amazon Polly para sintetizar respostas de fala para seu bot, para chamar o Amazon Comprehend para análise de sentimentos e, opcionalmente, para o Amazon Kendra para pesquisar índices.
- AWSServiceRoleForLexChannels— O Amazon Lex usa essa função vinculada ao serviço para publicar texto no seu bot ao gerenciar canais.

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criação de um perfil vinculado ao serviço para Amazon Lex

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria um bot, um canal de bots ou uma intenção de pesquisa do Amazon Kendra no, o AWS Management Console Amazon Lex cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando um bot, uma associação de canal ou uma intenção de pesquisa do Amazon Kendra é criada, o Amazon Lex cria novamente o perfil vinculado ao serviço.

Você também pode usar o AWS CLI para criar uma função vinculada ao serviço com o caso de AWSServiceRoleForLexBotsuso. Em, AWS CLI crie uma função vinculada ao serviço com o nome do serviço Amazon Lex. `lex.amazonaws.com` Para obter mais informações, consulte Etapa 1: Criar uma função vinculada a serviço (AWS CLI). Se você excluir essa função vinculada ao serviço, será possível usar esse mesmo processo para criar a função novamente.

Edição de um perfil vinculado ao serviço do Amazon Lex

O Amazon Lex não permite que você edite os perfis vinculados ao serviço do Amazon Lex. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Exclusão de um perfil vinculado ao serviço do Amazon Lex

Se você não precisar mais usar um atributo ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar os recursos de seu perfil vinculado ao serviço antes de excluí-lo manualmente.

> **Note**
>
> Se o serviço do Amazon Lex 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 excluir recursos do Amazon Lex usados por funções vinculadas ao serviço:

1. Exclua todos os canais de bot que você estiver usando.
2. Exclua todos os bots na conta.

Como excluir manualmente o perfil vinculado ao serviço usando o IAM

Use o console do IAM AWS CLI, o ou a AWS API para excluir as funções vinculadas ao serviço Amazon Lex. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a perfis vinculados ao serviço do Amazon Lex

O Amazon Lex é compatível com perfis vinculados ao serviço em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon Lex.

Solução de problemas de identidade e acesso da Amazon Lex

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon Lex e o IAM.

Tópicos

- Não tenho autorização para executar uma ação no Amazon Lex
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas de fora da minha tenham acesso Conta da AWS aos meus recursos do Amazon Lex

Não tenho autorização para executar uma ação no Amazon Lex

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `lex:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `lex:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

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.

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. 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.

Quero permitir que pessoas de fora da minha tenham acesso Conta da AWS aos meus recursos do Amazon Lex

É possível criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a 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 é compatível com esses atributos, consulte Como o Amazon Lex 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 conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Monitoramento no Amazon Lex

O monitoramento é importante para manter a confiabilidade, a disponibilidade e o desempenho dos chatbots do Amazon Lex. Este tópico descreve como usar o Amazon CloudWatch Logs e AWS CloudTrail monitorar o Amazon Lex e descreve o tempo de execução do Amazon Lex e as métricas de associação de canais.

Tópicos

- Monitorando o Amazon Lex com a Amazon CloudWatch
- Monitoramento de chamadas de API do Amazon Lex com AWS CloudTrail registros

Monitorando o Amazon Lex com a Amazon CloudWatch

Para monitorar a saúde dos seus bots Amazon Lex, use a Amazon CloudWatch. Com CloudWatch, você pode obter métricas para operações individuais do Amazon Lex ou para operações globais do Amazon Lex para sua conta. Você também pode configurar CloudWatch alarmes para serem notificados quando uma ou mais métricas excederem um limite definido por você. Por exemplo, você pode monitorar o número de solicitações feitas a um bot durante um determinado período de tempo, visualizar a latência de solicitações bem-sucedidas ou gerar um alarme quando erros excederem um limite.

CloudWatch Métricas para o Amazon Lex

Para obter métricas para as operações do Amazon Lex, você deverá especificar as seguintes informações:

- A dimensão da métrica. Uma dimensão é um conjunto de pares de nome-valor que você usa para identificar uma métrica. O Amazon Lex tem três dimensões:
  - `BotAlias, BotName, Operation`
  - `BotAlias, BotName, InputMode, Operation`
  - `BotName, BotVersion, InputMode, Operation`
- O nome da métrica, como `MissedUtteranceCount` ou `RuntimeRequestCount`.

Você pode obter métricas para o Amazon Lex com a AWS Management Console AWS CLI, a ou a CloudWatch API. Você pode usar a CloudWatch API por meio de um dos kits de desenvolvimento de software da Amazon AWS (SDKs) ou das ferramentas de CloudWatch API. O console do Amazon Lex exibe gráficos com base nos dados brutos da CloudWatch API.

Você deve ter as CloudWatch permissões apropriadas para monitorar o Amazon Lex com CloudWatch . Para obter mais informações, consulte Autenticação e controle de acesso para a Amazon CloudWatch no Guia CloudWatch do usuário da Amazon.

Visualizar métricas do Amazon Lex

Veja as métricas do Amazon Lex usando o console do Amazon Lex ou o CloudWatch console.

Para visualizar métricas (console do Amazon Lex)

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 aqueles cujas métricas você deseja ver.
3. Escolha Monitoramento. As métricas serão exibidos em gráficos.

Para visualizar métricas (CloudWatch console)

1. Faça login no AWS Management Console e abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Escolha Métricas, Todas as métricas e, em seguida, selecione AWS/Lex.
3. Escolha a dimensão, informe um nome de métrica e selecione Adicionar ao gráfico.
4. Escolha um valor para o intervalo de datas. A contagem da métrica para o intervalo de datas selecionado é exibida no gráfico.

Criar um alarme

Um CloudWatch alarme monitora uma única métrica durante um período de tempo especificado e executa uma ou mais ações: enviar uma notificação para um tópico do Amazon Simple Notification Service (Amazon SNS) ou política de Auto Scaling. A ação ou ações são baseadas no valor da métrica em relação a um determinado limite em vários períodos que você especifica. CloudWatchtambém pode enviar uma mensagem do Amazon SNS quando o alarme muda de estado.

CloudWatch os alarmes invocam ações somente quando o estado muda e persiste durante o período especificado.

Para definir um alarme

1. Faça login no AWS Management Console e abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Escolha Alarmes e, em seguida, Criar alarme.
3. Escolha Métricas do AWS/Lex e, em seguida, escolha uma métrica.
4. Para Intervalo de tempo, escolha um intervalo de tempo para monitorar e, em seguida, selecione Avançar.

5. Preencha os campos Nome e Descrição.
6. Para Sempre que, escolha **>=** e digite um valor máximo.
7. Se você quiser CloudWatch enviar um e-mail quando o estado do alarme for atingido, na seção Ações, para Sempre que este alarme for atingido, escolha Estado é ALARME. Em Enviar notificação para, selecione uma lista de correspondência ou selecione Nova lista e crie uma nova.
8. Visualize o alarme na seção Prévia do alarme. Se você estiver satisfeito com o alarme, selecione Criar alarme.

CloudWatchMétricas para o Amazon Lex Runtime

A tabela a seguir descreve as métricas de runtime para o Amazon Lex.

Métrica	Descrição
`KendraIndexAccessError`	O número de vezes que o Amazon Lex não conseguiu acessar seu índice Amazon Kendra. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation` Unidade: contagem
`KendraLatency`	A quantidade de tempo que a Amazon Kendra leva para responder a uma solicitação do `AMAZON.KendraSearchIntent` . Dimensões válidas para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode`

Métrica	Descrição
	Dimensões válidas para a operação `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unidade: milissegundos
`KendraSuccess`	O número de solicitações bem-sucedidas do índice do `AMAZON.KendraSearchIntent` Amazon Kendra. Dimensões válidas para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensões válidas para a operação `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unidade: contagem

Métrica	Descrição
`KendraSystemErrors`	O número de vezes que o Amazon Lex não conseguiu acessar seu índice Amazon Kendra. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation` Unidade: contagem
`KendraThrottledEvents`	O número de vezes que o Amazon Kendra controlou a utilização das solicitações do `AMAZON.KendraSearchIntent` . Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation` Unidade: contagem

Métrica	Descrição
`MissedUtteranceCount`	O número de declarações que não foram reconhecidas no período especificado. Dimensões válidas para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensões válidas para a operação `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation`
`RuntimeConcurrency`	O número de conexões simultâneas no período especificado. `RuntimeConcurrency` é relatado como um `StatisticSet` . Dimensões válidas para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • Operação, BotName, BotVersion, InputMode • Operação, BotName, BotAlias, InputMode Dimensões válidas para outras operações: • Operação, BotName, BotVersion • Operação, BotName, BotAlias Unidade: contagem

Métrica	Descrição
`RuntimeInvalidLambdaResponses`	O número de respostas inválidas AWS Lambda (Lambda) no período especificado. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation`
`RuntimeLambdaErrors`	O número de erros de runtime do Lambda no período especificado. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation`
`RuntimePollyErrors`	O número de respostas inválidas do Amazon Polly no período especificado. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation`

Métrica	Descrição
`RuntimeRequestCount`	O número de solicitações de runtime no período especificado. Dimensões válidas para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensões válidas para a operação `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unidade: contagem
`RuntimeSucessfulRequestLatency` Important Essa métrica é `RuntimeSucessfulRequestLatency` 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 a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensões válidas para a operação `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` 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. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation` Unidade: contagem
`RuntimeThrottledEvents`	O número de solicitações limitadas. O Amazon Lex controla a utilização de uma solicitação 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. Dimensão válida para a operação `PostContent` com o `Text` ou `Speech` `InputMode` : • BotName, BotAlias, Operação, InputMode Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation` 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. Dimensão válida para a operação `PostContent` com `Text` ou `Speech` `InputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensão válida para a operação `PostText`: • `BotName, BotAlias, Operation` Unidade: contagem

As métricas de runtime do Amazon Lex usam o namespace do `AWS/Lex` e fornecem métricas nas dimensões a seguir. Você pode agrupar métricas por dimensões no CloudWatch console:

Dimensão	Descrição
`BotName, BotAlias, Operation, InputMode`	Agrupa as métricas por alias do bot, nome do bot, operação (`PostContent`) e se a entrada foi de texto ou fala.
`BotName, BotVersion, Operation, InputMode`	Agrupa as métricas por nome do bot, versão do bot, operação (`PostContent`) e se a entrada foi de texto ou fala.
`BotName, BotVersion, Operation`	Agrupa as métricas por nome do bot, versão do bot e operação, `PostText`.
`BotName, BotAlias, Operation`	Agrupa as métricas por nome do bot, alias do bot e operação, `PostText`.

CloudWatch Métricas para associações de canais do Amazon Lex

Uma associação de canal é aquela entre o Amazon Lex e um canal de mensagens, como o Facebook. A tabela a seguir descreve as métricas de associação de canal do Amazon Lex.

Métrica	Descrição
`BotChannelAuthErrors`	O número de erros de autenticação retornado pelo canal de mensagens no período especificado. Um erro de autenticação indica que o token secreto fornecido durante a criação de canal é inválido ou expirou.
`BotChannelConfigurationErrors`	O número de erros de configuração no período especificado. Um erro de configuração indica que uma ou mais entradas de configuração para o canal são inválidas.
`BotChannelInboundThrottledEvents`	O número de vezes que as mensagens que foram enviadas pelo canal de mensagens foram limitadas pelo Amazon Lex no período especificado.
`BotChannelOutboundThrottledEvents`	O número de vezes que os eventos de saída do Amazon Lex para o canal de mensagens foram limitados no período de tempo especificado.
`BotChannelRequestCount`	O número de solicitações feitas em um canal no período especificado.
`BotChannelResponseCardErrors`	O número de vezes que o Amazon Lex não conseguiu publicar cartões de resposta no período especificado.
`BotChannelSystemErrors`	O número de erros internos que ocorreram no Amazon Lex para um canal no período especificado.

As métricas de associação de canal do Amazon Lex usam o namespace do `AWS/Lex` e fornecem métricas para a dimensão a seguir. Você pode agrupar métricas por dimensões no CloudWatch console:

Dimensão	Descrição
`BotAlias, BotChannelName, BotName, Source`	Agrupar as métricas por alias do bot, nome do canal, nome do bot e origem do tráfego.

CloudWatch Métricas para registros de conversas

O Amazon Lex usa as seguintes métricas para logs de conversa:

Métrica	Descrição
`ConversationLogsAudioDeliverySuccess`	O número de logs de áudio entregues com êxito ao bucket do S3 no período especificado. Unidades: contagem
`ConversationLogsAudioDeliveryFailure`	O número de logs de áudio que não foram entregues ao bucket do S3 no período especificado. Uma falha de entrega indica um erro com os recursos configurados para logs de conversa. Os erros podem incluir permissões insuficientes do IAM, uma AWS KMS chave inacessível ou um bucket S3 inacessível. Unidades: contagem
`ConversationLogsTextDeliverySuccess`	O número de registros de texto entregues com sucesso ao CloudWatch Logs no período de tempo especificado. Unidades: contagem
`ConversationLogsTextDeliveryFailure`	O número de registros de texto que não foram entregues aos CloudWatch Registros no período de tempo especificado. Uma falha de entrega indica um erro com os recursos configurados para logs de conversa. Os erros

Métrica	Descrição
	podem incluir permissões insuficientes do IAM, uma AWS KMS chave inacessível ou um grupo de registros de CloudWatch registros inacessível. Unidades: contagem

As métricas de log de conversa do Amazon Lex usam o namespace `AWS/Lex` e fornecem métricas para as dimensões a seguir. Você pode agrupar métricas por dimensão no CloudWatch console.

Dimensão	Descrição
`BotAlias`	Agrupe métricas pelo alias do bot.
`BotName`	Agrupe métricas pelo nome do bot.
`BotVersion`	Agrupe métricas pela versão do bot.

Monitoramento de chamadas de API do Amazon Lex com AWS CloudTrail registros

O Amazon Lex é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Amazon Lex. CloudTrail captura um subconjunto de chamadas de API para o Amazon Lex como eventos, incluindo chamadas do console do Amazon Lex e de chamadas de código para o Amazon Lex. APIs 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. 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, 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 CloudTrail, inclusive como configurá-lo e ativá-lo, consulte o Guia AWS CloudTrail do usuário.

Informações sobre o Amazon Lex em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando uma atividade de evento suportada ocorre no Amazon Lex, essa atividade é registrada em um CloudTrail evento junto com outros eventos AWS de 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, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon Simple Storage Service (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 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:

- Visão geral da criação de uma trilha
- CloudTrail Serviços e integrações compatíveis
- Configuração de 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 oferece suporte ao registro das seguintes operações como eventos em arquivos de CloudTrail log:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType

- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots
- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações ajudam a identificar:

- Se a solicitação foi feita com credenciais de usuário raiz ou do
- 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 serviço da AWS

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

Para obter informações sobre as ações do Amazon Lex que são registradas nos CloudTrail registros, consulte Amazon Lex Model Building Service. Por exemplo, chamadas para as DeleteBot operações

PutBotGetBot, e geram entradas no CloudTrail registro. As ações documentadas no Serviço de runtime do Amazon Lex, PostContent e PostText, não são registradas.

Exemplo: entradas de arquivo de log do Amazon Lex

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do 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 de entrada de CloudTrail registro a seguir mostra o resultado de uma chamada para a `PutBot` operação.

```
   {
               "eventVersion": "1.05",
               "userIdentity": {
                   "type": "AssumedRole | FederatedUser | IAMUser | Root | SAMLUser |
WebIdentityUser",
                   "principalId": "principal ID",
                   "arn": "ARN",
                   "accountId": "account ID",
                   "accessKeyId": "access key ID",
                   "userName": "user name"
               },
               "eventTime": "timestamp",
               "eventSource": "lex.amazonaws.com",
               "eventName": "PutBot",
               "awsRegion": "region",
               "sourceIPAddress": "source IP address",
               "userAgent": "user agent",
               "requestParameters": {
                   "name": "CloudTrailBot",
                   "intents": [
                       {
                           "intentVersion": "11",
                           "intentName": "TestCloudTrail"
                       }
                   ],
                   "voiceId": "Salli",
                   "childDirected": false,
```

```
                    "locale": "en-US",
                    "idleSessionTTLInSeconds": 500,
                    "processBehavior": "BUILD",
                    "description": "CloudTrail test bot",
                    "clarificationPrompt": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "I didn't understand you. What would you
like to do?"
                            }
                        ],
                        "maxAttempts": 2
                    },
                    "abortStatement": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "Sorry. I'm not able to assist at this
time."
                            }
                        ]
                    }
                },
                "responseElements": {
                    "voiceId": "Salli",
                    "locale": "en-US",
                    "childDirected": false,
                    "abortStatement": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "Sorry. I'm not able to assist at this
time."
                            }
                        ]
                    },
                    "status": "BUILDING",
                    "createdDate": "timestamp",
                    "lastUpdatedDate": "timestamp",
                    "idleSessionTTLInSeconds": 500,
                    "intents": [
                        {
                            "intentVersion": "11",
```

```
                            "intentName": "TestCloudTrail"
                        }
                    ],
                    "clarificationPrompt": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "I didn't understand you. What would you
like to do?"
                            }
                        ],
                        "maxAttempts": 2
                    },
                    "version": "$LATEST",
                    "description": "CloudTrail test bot",
                    "checksum": "checksum",
                    "name": "CloudTrailBot"
                },
                "requestID": "request ID",
                "eventID": "event ID",
                "eventType": "AwsApiCall",
                "recipientAccountId": "account ID"
            }
        }
```

Validação de conformidade para o Amazon Lex

Auditores terceirizados avaliam a segurança e a conformidade do Amazon Lex como parte de vários programas de AWS conformidade. O Amazon Lex é um serviço qualificado pela HIPAA. Ele é compatível com PCI, SOC e ISO. Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Download de relatórios no AWS Artifact.

Sua responsabilidade de conformidade ao usar o Amazon Lex é determinada pela confidencialidade de seus dados, pelas metas de conformidade da sua organização e pelas regulamentações e leis aplicáveis. Se o seu uso do Amazon Lex estiver sujeito à conformidade com padrões, como o PCI, a AWS fornecerá os seguintes recursos para ajudar:

- Guias de início rápido sobre segurança e conformidade — guias de implantação que discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos focados em segurança e conformidade em AWS

- Whitepaper Arquitetura para segurança e conformidade com HIPAA: esse whitepaper descreve como as empresas podem usar a AWS para criar aplicativos em conformidade com a HIPAA.
- Recursos de compatibilidade da AWS: uma coleção de manuais e guias que pode ser aplicada ao seu setor e local.
- AWS Config: um serviço que avalia até que ponto suas configurações de recursos estão compatíveis com práticas internas, diretrizes do setor e regulamentações.
- AWS Security Hub— Uma visão abrangente do seu estado de segurança interno AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança

Para obter uma lista de AWS serviços no escopo de programas de conformidade específicos, consulte AWS Services in Scope by Compliance Program. Para obter informações gerais, consulte Programas de conformidade da AWS.

Resiliência no Amazon Lex

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 aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, o Amazon Lex oferece vários recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

Segurança da infraestrutura no Amazon Lex

Como um serviço gerenciado, o Amazon Lex 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 AWS API publicadas para acessar o Amazon Lex pela rede. Os clientes devem ter suporte ao TLS (Transport Layer Security) 1.0. Recomendamos usar o TLS 1.2 ou

posterior. Os clientes também devem oferecer suporte a pacotes de criptografia com Perfect Forward Secrecy (PFS – Sigilo de encaminhamento perfeito), como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos, como o Java 7 e versões posteriores, oferece suporte a 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 é possível usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Você pode chamar essas operações de API de qualquer local da rede, mas o Amazon Lex oferece suporte a políticas de acesso com base em recursos, que podem incluir restrições com base no endereço IP de origem. Você também pode usar as políticas do Amazon Lex para controlar o acesso de endpoints específicos da Amazon Virtual Private Cloud (Amazon VPC) ou específicos. VPCs Efetivamente, isso isola o acesso à rede a um determinado recurso do Amazon Lex somente da VPC específica dentro da AWS rede.

Diretrizes e cotas no Amazon Lex

As seções a seguir fornecem diretrizes e cotas ao usar o Amazon Lex

Tópicos

- Regiões do compatíveis
- Diretrizes gerais
- Cotas

Regiões do compatíveis

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

Diretrizes gerais

Esta seção descreve as diretrizes gerais ao usar o Amazon Lex.

- Assinatura de solicitações: todas as operações de API de runtime e de criação de modelo do Amazon Lex no Referência da API usam Signature V4 em solicitações de autenticação. Para obter mais informações sobre a autenticação de solicitações, consulte Processo de cadastramento do Signature versão 4 na Referência geral da Amazon Web Services.

  Para PostContent, o Amazon Lex usa a opção de carga não assinada descrita em Cálculos de assinatura para o cabeçalho de autorização: transferência de carga em um único bloco (AWS Signature versão 4) na Referência de API do Amazon Simple Storage Service (S3).

  Quando você usar a opção de carga não assinada, não inclua o hash da carga na solicitação canônica. Em vez disso, use a string literal "UNSIGNED-PAYLOAD" como o hash de carga. Inclua também um cabeçalho com o nome `x-amz-content-sha256` e o valor `UNSIGNED-PAYLOAD` na solicitação `PostContent`.

- Observe o seguinte sobre como o Amazon Lex captura valores de slot de declarações de usuário:

  O Amazon Lex 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 a seguinte amostra de declaração:

  ```
  "Tell me the prediction for {Sign}"
  ```

  Onde `{Sign}` é um slot de `ZodiacSign` do tipo personalizado. Ele tem 12 valores de enumeração, de `Aries` a `Pisces`. Da declaração do usuário "Informe a previsão para ...", o Amazon Lex entende que o que vem a seguir é um signo do zodíaco.

  Quando o campo `valueSelectionStrategy` é definido como `ORIGINAL_VALUE` usando a operação PutSlotType, ou se Expandir valores for selecionado no console, se o usuário disser "Informe-me a previsão da terra", o Amazon Lex inferirá que "terra" é um `ZodiacSign` e passará isso para seu aplicativo cliente ou para funções do Lambda. Você deve verificar quais valores do slot são válidos antes de usá-los em sua atividade de cumprimento.

  Se você define o campo `valueSelectionStrategy` para `TOP_RESOLUTION` usando a operação PutSlotType, ou se Restrict to slot values and synonyms está selecionada no console, os valores retornados são limitados aos valores que você definiu para o tipo de slot. Por exemplo, se o usuário diz "Fale a previsão para terra", o valor não seria reconhecido, porque não é um dos valores definidos para o tipo de slot. Ao definir sinônimos para valores de slot, eles são reconhecidos da mesma forma que um valor de slot. No entanto, o valor de slot é retornado ao invés do sinônimo.

  Quando o Amazon Lex chama uma função do Lambda ou retorna o resultado de uma interação de fala com o aplicativo cliente, a capitalização dos valores do slot não é garantida. Por exemplo, se você estiver inferindo valores para o tipo de slot integrado AMAZON.Movie e um usuário disser ou digitar "E o vento levou", o Amazon Lex poderá retornar "E o vento levou", "e o vento levou" ou "E

o Vento Levou". 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`.

- Ao definir valores de slot que contêm acrônimos, use os seguintes padrões:
  - Letras maiúsculas separadas por pontos (D.V.D.)
  - Letras maiúsculas separadas por espaços (D V D)

- O Amazon Lex não oferece suporte ao tipo de slot integrado AMAZON.LITERAL, diferentemente do Alexa Skills Kit. No entanto, o Amazon Lex oferece suporte à criação de tipos de slot personalizados que você pode usar para implementar essa funcionalidade. Como mencionado na observação anterior, você pode capturar valores fora da definição de tipo de slot personalizado. Adicione mais e diversificados valores de enumeração para aumentar a precisão de reconhecimento automático de voz (ASR) e compreensão de linguagem natural (NLU).

- Os tipos de slot integrado AMAZON.DATE e AMAZON.TIME capturam as datas e horas absolutas e relativas. As datas e horas relativas são resolvidas na região em que o Amazon Lex está processando 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 e o Amazon Lex perguntará ao usuário novamente. 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".

- O fornecimento de dados de treinamento confusos em seu bot reduz a capacidade do Amazon Lex de compreender a entrada do usuário. Considere estes exemplos:

  Suponha que você tenha duas intenções (`OrderPizza` e `OrderDrink`) no seu bot e ambas estejam configuradas com uma declaração "Quero pedir". Essa declaração não é mapeada para uma intenção específica da qual o Amazon Lex possa aprender ao criar o modelo de linguagem

para o bot no tempo de criação. Como resultado, quando um usuário insere essa declaração em runtime, o Amazon Lex não pode escolher uma intenção com alto grau de confiança.

Considere outro exemplo, em que você define uma intenção personalizada para obter uma confirmação do usuário (por exemplo, `MyCustomConfirmationIntent`) e configura a intenção com as declarações "Sim" e "Não". Observe que o Amazon Lex também tem um modelo de linguagem para entender confirmações de usuário. Isso pode criar uma situação de conflito. Quando o usuário responde com um "Sim", isso significa que é uma confirmação da intenção em andamento ou que o usuário está solicitando a intenção personalizada que você criou?

Em geral, a amostra de declarações fornecida deve ser mapeada para uma intenção específica e, opcionalmente, para valores de slot específicos.

- As operações de API de runtime PostContent e PostText reconhecem um ID de usuário como o 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 exclusivamente a conversa com um bot (pode haver vários usuários pedindo pizza).

- Se o aplicativo cliente usar o Amazon Cognito para autenticação, você poderá usar o ID de usuário do Amazon Cognito como o ID de usuário do Amazon Lex. Observe que qualquer função do Lambda configurada para o bot deve ter seu próprio mecanismo de autenticação para identificar o usuário em cujo nome o Amazon Lex está invocando a função do Lambda.

- Recomendamos que você defina uma intenção que capte a intenção de um usuário de interromper a conversa. Por exemplo, você pode definir uma intenção (`NothingIntent`) com amostra de declarações ("Eu não quero nada", "sair", "tchau tchau"), nenhum slot e nenhuma função do Lambda configurada como um hook de código. Isso permite que os usuários fechem uma conversa com tranquilidade.

Cotas

Esta seção descreve as cotas atuais no Amazon Lex. Essas cotas são agrupadas por categorias.

As Service Quotas podem ser ajustadas ou aumentadas. Ente em contato com o Suporte ao cliente do AWS para solicitar um aumento de cota. Pode demorar alguns dias para aumentar a Service Quota. Se você estiver aumentando sua cota como parte de um projeto maior, não se esqueça de adicionar esse tempo ao seu plano.

Tópicos

- Service Quotas de runtime
- Cotas de criação de modelos

Service Quotas de runtime

Além das cotas descritas na referência de API, observe o seguinte:

Cotas de API

- A entrada de voz na operação PostContent pode ter até 15 segundos.
- Em ambas as operações da API de runtime, PostContent e PostText, o tamanho do texto inserido pode ser de até 1.024 caracteres Unicode.
- O tamanho máximo de cabeçalhos `PostContent` é 16 KB. O tamanho máximo combinado de solicitações e cabeçalhos de sessão é 12 KB.
- Ao usar as operações `PostContent` ou `PostText` no modo de texto, o número máximo de conversas simultâneas com um bot é 2 para o alias `$LATEST` e 50 para todos os outros aliases. A cota aplica-se separadamente para cada API.
- Ao usar a `PostContent` operação no modo de voz, o número máximo de conversas simultâneas em modo de texto com um bot é 2 para o alias `$LATEST` e 125 para todos os outros aliases. A cota aplica-se separadamente para cada API.

- O número máximo de chamadas simultâneas de gerenciamento de sessão (PutSession, GetSession, e DeleteSession) é 2 para o alias `$LATEST` de um bot e 50 para todos os outros aliases.

- O tamanho máximo de entradas para uma função do Lambda é 12 KB. O tamanho máximo de saída é 25 KB, dos quais 12 KB podem ser atributos de sessão.

Usar a versão **`$LATEST`**

- A versão `$LATEST` do bot deve ser usada somente para testes manuais. O Amazon Lex limita o número de solicitações de runtime que você pode fazer para a versão `$LATEST` do bot.

- Quando você atualizar a versão `$LATEST` do bot, o Amazon Lex encerra todas as conversas em andamento para qualquer aplicativo cliente usando a versão `$LATEST` do bot. Em geral, você não deve usar a versão `$LATEST` de um bot em produção, pois a versão `$LATEST` pode ser atualizada. Em vez disso, você deve publicar uma versão e usá-la.

- Quando você atualiza um alias, o Amazon Lex leva alguns minutos para incorporar a alteração. Quando você modificar a `$LATEST` versão do bot, a alteração será capturada imediatamente.

Tempo limite de sessão

- Esse tempo limite de sessão definido quando o bot foi criado determina por quanto tempo o bot retém o contexto de conversa, como a intenção e os dados de slot do usuário atual.

- Depois que um usuário inicia a conversa com o bot e até que a sessão expire, o Amazon Lex usa a mesma versão do bot, mesmo que você atualize o alias do bot para apontar para outra versão.

Cotas de criação de modelos

A criação de modelos se refere à criação e ao gerenciamento de bots. Isso inclui a criação e o gerenciamento de bots, intenções, tipos de slot, slots e associações de canal de bot.

Tópicos

- Cotas de bot
- Cotas de intenção
- Cotas de tipo de slot

Cotas de bot

- Os prompts e as instruções são configurados em toda a API de criação de modelos. Cada um desses prompts ou instruções pode ter até cinco mensagens, sendo que cada mensagem pode conter de 1 a 1.000 caracteres UTF-8.

- Ao usar grupos de mensagens, você pode definir até cinco grupos de mensagens para cada mensagem. Cada grupo de mensagens pode conter no máximo cinco mensagens e existe um limite de 15 mensagens em todos os grupos de mensagens.

- Você pode definir amostra de declarações para intenções e slots. É possível usar um máximo de 200.000 caracteres para todos as declarações.

- Cada tipo de slot pode definir um máximo de 10.000 valores e sinônimos. Cada bot pode conter um máximo de 50.000 valores e sinônimos do tipo de slots.

- Os nomes de bots, aliases e associações de canal de bot não diferenciam maiúsculas de minúsculas no momento da criação. Se você criar `PizzaBot` e, em seguida, tentar criar `pizzaBot`, você receberá um erro. No entanto, ao acessar um recurso, os nomes dos recursos

diferenciam maiúsculas de minúsculas (você deve especificar que é `PizzaBot`, e não `pizzaBot`). Esses nomes devem ter entre 2 e 50 caracteres ASCII.

- O número máximo de versões que você pode publicar para todos os tipos de recurso é 100. Observe que não há versionamento para aliases.

- Em um bot, os nomes de intenção e de slot devem ser exclusivos, ou seja, você não pode ter uma intenção e um slot com o mesmo nome.

- Você pode criar um bot configurado para oferecer suporte a várias intenções. Se duas intenções tiverem um slot com o mesmo nome, o tipo de slot correspondente deve ser o mesmo.

  Por exemplo, suponha que você crie um bot para oferecer suporte a duas intenções (`OrderPizza` e `OrderDrink`). Se essas duas intenções tiverem o slot `size`, então o tipo de slot deve ser o mesmo em ambos os locais.

  Além disso, a amostra de declarações fornecida para um slot em uma das intenções se aplica a um slot com mesmo nome em outras intenções.

- Você pode associar um máximo de 250 intenções a um bot.

- Ao criar um bot, especifique um tempo limite de sessão. O tempo limite de sessão pode ser entre um minuto e um dia. O padrão é cinco minutos.

- É possível criar até cinco aliases para um bot.

- Você pode criar até 250 bots por conta da AWS.

- Não é possível criar várias intenções que se estendam a partir da mesma intenção integrada.

Cotas de intenção

- Os nomes de slot e de intenção não diferenciam maiúsculas de minúsculas no momento da criação. Ou seja, se você criar a intenção `OrderPizza` e, em seguida, tentar criar outra intenção `orderPizza`, receberá uma mensagem de erro. No entanto, ao acessar esses recursos, os nomes dos recursos diferenciam maiúsculas de minúsculas; especifique `OrderPizza`, e não `orderPizza`. Esses nomes devem ter entre 1 e 100 caracteres ASCII.

- Uma intenção pode ter até 1.500 amostras de declarações. É necessário no mínimo uma amostra de declaração. Cada amotra de declaração pode ter até 200 caracteres UTF-8. Você pode usar até 200.000 caracteres para todas as intenções e declarações de slot em um bot. Uma amostra de declaração para uma intenção:
  - Pode fazer referência a zero ou mais nomes de slot.
  - Pode fazer referência a um nome de slot apenas uma vez.

  Por exemplo:

  ```
  I want a pizza
  I want a {pizzaSize} pizza
  I want a {pizzaSize} {pizzaTopping} pizza
  ```

- Embora cada intenção ofereça suporte a até 1.500 declarações, se você usar menos declarações, o Amazon Lex poderá ter uma capacidade melhor de reconhecer entradas fora do conjunto fornecido.

- Você pode criar até cinco grupos de mensagens para cada mensagem em uma intenção. Pode haver um total de 15 mensagens em todos os grupos de mensagens para uma mensagem.

- O console só pode criar grupos de mensagens para as mensagens `conclusionStatement` e `followUpPrompt`. Você pode criar grupos de mensagens para qualquer outra mensagem usando a API do Amazon Lex.

- Cada slot pode ter até 10 amostras de declarações. Cada amostra de declaração deve fazer referência ao nome do slot exatamente uma vez. Por exemplo:

```
{pizzaSize} please
```

- Cada bot pode ter, no máximo, 200.000 caracteres no total para declarações de intenção e de slot.

- Você não pode fornecer declarações para intenções que se estendam a partir de intenções integradas. Para todas as outras intenções, você deve fornecer pelo menos uma amostra de declaração. As intenções contêm slots, mas a amostra de declarações a nível de slot são opcionais.

- Intenções integradas
  - No momento, o Amazon Lex não oferece suporte à inferência de slots para intenções integradas. Não é possível criar funções do Lambda para retornar a diretiva `ElicitSlot` na resposta com uma intenção derivada de intenções integradas. Para obter mais informações, consulte Formato de resposta.
  - O serviço não é compatível com a adição de amostra de declarações a intenções integradas. Da mesma forma, você não pode adicionar nem remover slots de intenções integradas.

- Você pode criar até 1.000 intenções por conta da AWS. Você pode criar até 100 slots em uma intenção.

Cotas de tipo de slot

- Os nomes de tipo de slot não diferenciam maiúsculas de minúsculas no momento da criação. Se você criar o tipo de slot `PizzaSize` e, em seguida, tentar criar o tipo de slot `pizzaSize`, receberá uma mensagem de erro. No entanto, ao acessar esses recursos, os nomes dos recursos diferenciam maiúsculas de minúsculas (você deve especificar que é `PizzaSize`, e não `pizzaSize`). Os nomes devem ter entre 1 e 100 caracteres ASCII.

- Um tipo de slot personalizado que você criar pode ter no máximo 10.000 valores de enumeração e sinônimos. Cada valor pode ter até 140 caracteres UTF-8. Os valores de enumeração e sinônimos não contêm duplicados.

- Para um valor de tipo de slot, especifique tanto as letras maiúsculas quanto as minúsculas, onde for apropriado. Por exemplo, para um tipo de slot chamado `Procedure`, se o valor for `MRI`, especifique "MRI" e "mri" como valores.

- Tipos de slot integrado: no momento, o Amazon Lex não oferece suporte à adição de valores de enumeração ou sinônimos para os tipos de slots integrados.

Referência da API

Esta seção fornece a documentação das operações de API do Amazon Lex. Para obter uma lista de regiões da AWS onde o Amazon Lex está disponível, consulte Regiões e endpoints da AWS na Referência geral do Amazon Web Services.

Tópicos

- Ações
- Tipos de dados

Ações

As seguintes ações são compatíveis com o Serviço de criação de modelos do Amazon Lex:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots

- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents
- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

As seguintes ações são compatíveis com o Serviço de runtime do Amazon Lex:

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Serviço de criação de modelos do Amazon Lex

As seguintes ações são compatíveis com o Serviço de criação de modelos do Amazon Lex:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots
- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents

- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

CreateBotVersion

Serviço: Amazon Lex Model Building Service

Cria uma nova versão do bot com base na versão do `$LATEST`. Se a versão `$LATEST` deste recurso não tiver sido alterada desde que você criou a última versão, o Amazon Lex não criará uma nova versão e retorna a última versão criada. Ele retorna a última versão criada.

> **Note**
>
> Você pode atualizar apenas a versão `$LATEST` do bot. Você não pode atualizar as versões numeradas que você cria com a operação `CreateBotVersion`.

Quando você cria a primeira versão de um bot, o Amazon Lex define essa versão como 1. As versões subsequentes são incrementadas em 1. Para obter mais informações, consulte Versionamento.

Essa operação exige permissão para a ação `lex:CreateBotVersion`.

Sintaxe da Solicitação

```
POST /bots/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do bot do qual você deseja criar uma nova versão. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

checksum

Identifica uma revisão específica da versão `$LATEST` do bot. Se você especificar uma soma de verificação e a versão `$LATEST` do bot tiver uma soma de verificação diferente, uma exceção `PreconditionFailedException` será retornada e o Amazon Lex não publicará uma nova versão. Se você não especificar uma soma de verificação, o Amazon Lex publicará a versão `$LATEST`.

Tipo: String

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 201
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
```

```
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "description": "string",
   "detectSentiment": boolean,
   "enableModelImprovements": boolean,
   "failureReason": "string",
   "idleSessionTTLInSeconds": number,
   "intents": [
      {
         "intentName": "string",
         "intentVersion": "string"
      }
   ],
   "lastUpdatedDate": number,
   "locale": "string",
   "name": "string",
   "status": "string",
   "version": "string",
   "voiceId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 201.

Os dados a seguir são retornados no formato JSON pelo serviço.

abortStatement

A mensagem que o Amazon Lex usa para cancelar uma conversa. Para obter mais informações, consulte PutBot.

Tipo: objeto Statement

checksum

Soma de verificação que identifica a versão do bot que foi criado.

Tipo: String

childDirected

Para cada bot do Amazon Lex criado com o Serviço de criação de modelo do Amazon Lex, você deve especificar se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à Lei de Proteção à Privacidade Online Infantil (COPPA), especificando `true` ou `false` no `childDirected` campo. Ao especificar `true` no campo `childDirected`, você confirma que seu uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Ao especificar `false` no campo `childDirected`, você confirma que seu uso do Amazon Lex não está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Você não pode especificar um valor padrão para o campo `childDirected` que não reflita com precisão se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA.

Se o uso do Amazon Lex estiver relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos, você deverá obter qualquer consentimento parental verificável exigido pela COPPA. Para obter informações sobre o uso do Amazon Lex em conexão com sites, programas ou outros aplicativos direcionados, total ou parcialmente, a crianças menores de 13 anos, consulte as perguntas frequentes do Amazon Lex.

Tipo: booliano

clarificationPrompt

A mensagem que o Amazon Lex usa quando não entende a solicitação do usuário. Para obter mais informações, consulte PutBot.

Tipo: objeto Prompt

createdDate

A data em que a versão do bot foi criada.

Tipo: Timestamp

description

Uma descrição do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

detectSentiment

Indica se as declarações inseridas pelo usuário devem ser enviadas ao Amazon Comprehend para análise de sentimento.

Tipo: booliano

enableModelImprovements

Indica se o bot usa melhorias de precisão. `true` indica que o bot está usando as melhorias, caso contrário, é `false`.

Tipo: booliano

failureReason

Se `status` for `FAILED`, o Amazon Lex fornece o motivo da falha na criação do bot.

Tipo: String

idleSessionTTLInSeconds

O tempo máximo em segundos que o Amazon Lex retém os dados coletados em uma conversa. Para obter mais informações, consulte PutBot.

Tipo: inteiro

Faixa válida: valor mínimo de 60. Valor máximo de 86.400.

intents

Uma matriz de objetos `Intent`. Para obter mais informações, consulte PutBot.

Tipo: matriz de objetos Intent

lastUpdatedDate

A data em que a versão `$LATEST` deste bot foi atualizada.

Tipo: Timestamp

locale

Especifica a localidade de destino deste bot.

Tipo: String

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

O nome do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

status

Ao enviar uma solicitação para criar ou atualizar um bot, o Amazon Lex define o elemento de resposta `status` como `BUILDING`. Depois que o Amazon Lex cria o bot, ele define `status` como `READY`. Se o Amazon Lex não puder criar o bot, ele define `status` como `FAILED`. O Amazon Lex retorna o motivo da falha no elemento de resposta `failureReason`.

Tipo: String

Valores Válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

A versão do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

voiceId

O ID de voz do Amazon Polly que o Amazon Lex usa para interações de voz com o usuário.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3

- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

CreateIntentVersion

Serviço: Amazon Lex Model Building Service

Cria uma nova versão de uma intenção com base na versão `$LATEST` da intenção. Se a versão `$LATEST` desta intenção não tiver sido alterada desde que você a criou, o Amazon Lex não criará uma nova versão. Ele retorna a última versão criada.

> **Note**
>
> Você pode atualizar apenas a versão `$LATEST` da intenção. Você não pode atualizar as versões numeradas que você cria com a operação `CreateIntentVersion`.

Quando você cria uma versão de uma intenção, o Amazon Lex define a versão como 1. As versões subsequentes são incrementadas em 1. Para obter mais informações, consulte Versionamento.

Essa operação exige permissões para executar a ação `lex:CreateIntentVersion`.

Sintaxe da Solicitação

```
POST /intents/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome da intenção da qual você deseja criar uma nova versão. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

checksum

Soma de verificação da versão `$LATEST` da intenção que deve ser usada para criar a nova versão. Se você especificar uma soma de verificação e a versão `$LATEST` da intenção tiver uma soma de verificação diferente, o Amazon Lex retornará uma exceção `PreconditionFailedException` e não publicará uma nova versão. Se você não especificar uma soma de verificação, o Amazon Lex publicará a versão `$LATEST`.

Tipo: String

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 201
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
```

```
    },
    "createdDate": number,
    "description": "string",
    "dialogCodeHook": {
       "messageVersion": "string",
       "uri": "string"
    },
    "followUpPrompt": {
       "prompt": {
          "maxAttempts": number,
          "messages": [
             {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
             }
          ],
          "responseCard": "string"
       },
       "rejectionStatement": {
          "messages": [
             {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
             }
          ],
          "responseCard": "string"
       }
    },
    "fulfillmentActivity": {
       "codeHook": {
          "messageVersion": "string",
          "uri": "string"
       },
       "type": "string"
    },
    "inputContexts": [
       {
          "name": "string"
       }
    ],
    "kendraConfiguration": {
       "kendraIndex": "string",
```

```
        "queryFilterString": "string",
        "role": "string"
    },
    "lastUpdatedDate": number,
    "name": "string",
    "outputContexts": [
        {
            "name": "string",
            "timeToLiveInSeconds": number,
            "turnsToLive": number
        }
    ],
    "parentIntentSignature": "string",
    "rejectionStatement": {
        "messages": [
            {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
    },
    "sampleUtterances": [ "string" ],
    "slots": [
        {
            "defaultValueSpec": {
                "defaultValueList": [
                    {
                        "defaultValue": "string"
                    }
                ]
            },
            "description": "string",
            "name": "string",
            "obfuscationSetting": "string",
            "priority": number,
            "responseCard": "string",
            "sampleUtterances": [ "string" ],
            "slotConstraint": "string",
            "slotType": "string",
            "slotTypeVersion": "string",
            "valueElicitationPrompt": {
                "maxAttempts": number,
```

```
            "messages": [
               {
                  "content": "string",
                  "contentType": "string",
                  "groupNumber": number
               }
            ],
            "responseCard": "string"
         }
      }
   ],
   "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 201.

Os dados a seguir são retornados no formato JSON pelo serviço.

checksum

Soma de verificação da versão de intenção criada.

Tipo: String

conclusionStatement

Depois que a função do Lambda especificada no campo `fulfillmentActivity` cumpre a intenção, o Amazon Lex transmite essa declaração ao usuário.

Tipo: objeto Statement

confirmationPrompt

Se definido, o prompt que o Amazon Lex usa para confirmar a intenção do usuário antes de atendê-la.

Tipo: objeto Prompt

createdDate

A data em que a intenção foi criada.

Tipo: Timestamp

description

Uma descrição da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

dialogCodeHook

Se definido, o Amazon Lex invoca essa função do Lambda para cada entrada do usuário.

Tipo: objeto CodeHook

followUpPrompt

Se definido, o Amazon Lex usa esse prompt para solicitar atividades adicionais do usuário depois que a intenção for atendida.

Tipo: objeto FollowUpPrompt

fulfillmentActivity

Descreve como a intenção é atendida.

Tipo: objeto FulfillmentActivity

inputContexts

Uma matriz de objetos `InputContext` que lista os contextos que devem estar ativos para que o Amazon Lex escolha a intenção em uma conversa com o usuário.

Tipo: matriz de InputContext objetos

Membros da Matriz: número mínimo de 0 itens. Número máximo de 5 itens.

kendraConfiguration

Informações de configuração, se houver, para conectar um índice do Amazon Kendra com a intenção `AMAZON.KendraSearchIntent`.

Tipo: objeto KendraConfiguration

lastUpdatedDate

A data em que a intenção foi atualizada.

Tipo: Timestamp

name

O nome da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

outputContexts

Uma matriz de objetos `OutputContext` que lista os contextos que a intenção ativa quando a intenção é atendida.

Tipo: matriz de objetos OutputContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

parentIntentSignature

Um identificador exclusivo de uma intenção integrada.

Tipo: String

rejectionStatement

Quando o usuário responde “não” à pergunta definida em `confirmationPrompt`, o Amazon Lex responde com essa mensagem para confirmar que a intenção foi cancelada.

Tipo: objeto Statement

sampleUtterances

Uma matriz de amostra de declarações configurada para a intenção.

Tipo: matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 1.500 itens.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 200.

slots

Uma variedade de tipos de slots que define as informações necessárias para atender a intenção.

Tipo: matriz de objetos Slot

Membros da Matriz: número mínimo de 0 itens. Número máximo de 100 itens.

version

O número da versão atribuída à nova versão da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

CreateSlotTypeVersion

Serviço: Amazon Lex Model Building Service

Cria uma nova versão com base na versão `$LATEST` do tipo de slot especificado. Se a versão `$LATEST` deste recurso não tiver sido alterada desde que você criou a última versão, o Amazon Lex não criará uma nova versão e retorna a última versão criada. Ele retorna a última versão criada.

> **Note**
>
> Você pode atualizar apenas a versão `$LATEST` de um tipo de slot. Não é possível atualizar as versões numeradas que você cria com a operação `CreateSlotTypeVersion`.

Quando você cria uma versão de um tipo de slot, o Amazon Lex define a versão como 1. As versões subsequentes são incrementadas em 1. Para obter mais informações, consulte Versionamento.

Essa operação exige permissões para a ação `lex:CreateSlotTypeVersion`.

Sintaxe da Solicitação

```
POST /slottypes/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do tipo de slot do qual você deseja criar uma nova versão. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

checksum

Soma de verificação da versão `$LATEST` do tipo de slot que você deseja publicar. Se você especificar uma soma de verificação e a versão `$LATEST` da intenção tiver uma soma de verificação diferente, o Amazon Lex retornará uma exceção `PreconditionFailedException` e não publicará a nova versão. Se você não especificar uma soma de verificação, o Amazon Lex publicará a versão `$LATEST`.

Tipo: String

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 201
Content-type: application/json

{
   "checksum": "string",
   "createdDate": number,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
```

```
    "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 201.

Os dados a seguir são retornados no formato JSON pelo serviço.

checksum

Soma de verificação da versão `$LATEST` do tipo de slot.

Tipo: String

createdDate

A data em que o tipo de slot foi criado.

Tipo: Timestamp

description

Uma descrição do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

enumerationValues

Uma lista de objetos `EnumerationValue` que define os valores que o tipo de slot pode ter.

Tipo: matriz de objetos EnumerationValue

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10.000 itens.

lastUpdatedDate

A data em que o tipo de slot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

O tipo de slot integrado usado como pai do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informações de configuração que estendem o tipo de slot integrado principal.

Tipo: matriz de objetos SlotTypeConfiguration

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

valueSelectionStrategy

A estratégia que o Amazon Lex usa para determinar o valor de slot. Para obter mais informações, consulte PutSlotType.

Tipo: String

Valores Válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

A versão atribuída à nova versão do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface

- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBot

Serviço: Amazon Lex Model Building Service

Exclui todas as versões do bot, incluindo a versão `$LATEST`. Para excluir uma versão específica do bot, use a operação DeleteBotVersion. A operação `DeleteBot` não remove imediatamente o esquema do bot. Em vez disso, ele é marcado para exclusão e removido posteriormente.

O Amazon Lex armazena declarações indefinidamente para melhorar a capacidade do seu bot de responder às entradas do usuário. Essas declarações não são removidas quando o bot é excluído. Para remover as declarações, use a operação DeleteUtterances.

Se um bot tiver um alias, você não conseguirá excluí-lo. Em vez disso, a operação `DeleteBot` retorna uma exceção `ResourceInUseException` que inclui uma referência ao alias que se refere ao bot. Para remover a referência ao bot, exclua o alias. Se você receber a mesma exceção novamente, exclua o alias de referência até que a operação `DeleteBot` seja bem-sucedida.

Essa operação exige permissões para a ação `lex:DeleteBot`.

Sintaxe da Solicitação

```
DELETE /bots/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do bot. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBotAlias

Serviço: Amazon Lex Model Building Service

Exclui um alias de um bot especificado.

Não é possível excluir um alias usado na associação entre um bot e um canal de mensagens. Se um alias for usado em uma associação de canal, a operação `DeleteBot` retornará uma exceção `ResourceInUseException` que inclui uma referência à associação de canal referente ao bot. Você pode remover a referência ao alias excluindo a associação do canal. Se você receber a mesma exceção novamente, exclua a associação de referência até que a operação `DeleteBotAlias` seja bem-sucedida.

Sintaxe da Solicitação

```
DELETE /bots/botName/aliases/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botName

O nome do bot para o qual o alias aponta.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

name

O nome do alias a ser excluído. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBotChannelAssociation

Serviço: Amazon Lex Model Building Service

Exclui a associação entre um bot do Amazon Lex e uma plataforma de mensagens.

Essa operação exige permissão para a ação `lex:DeleteBotChannelAssociation`.

Sintaxe da Solicitação

```
DELETE /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

aliasName

Um alias que aponta para a versão específica do bot do Amazon Lex à qual essa associação está sendo feita.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

botName

O nome do bot do Amazon Lex.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

name

O nome da associação. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBotVersion

Serviço: Amazon Lex Model Building Service

Exclui uma versão específica de um bot. Para excluir todas as versão de um bot, use a operação DeleteBot.

Essa operação exige permissões para a ação `lex:DeleteBotVersion`.

Sintaxe da Solicitação

```
DELETE /bots/name/versions/version HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do bot.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

version

A versão do bot a excluir. Você não pode excluir a versão `$LATEST` do bot. Para excluir a versão `$LATEST`, use a operação DeleteBot.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[0-9]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteIntent

Serviço: Amazon Lex Model Building Service

Exclui todas as versões da intenção, incluindo a versão `$LATEST`. Para excluir uma versão específica da intenção, use a operação DeleteIntentVersion.

Você só pode excluir uma versão de uma intenção se ela não for referenciada. Para excluir uma intenção mencionada em um ou mais bots (consulteAmazon Lex: como funciona), primeiro você deve remover essas referências.

> **Note**
>
> Se você receber a exceção `ResourceInUseException`, ela fornecerá um exemplo de referência que mostra onde a intenção é referenciada. Para remover a referência à intenção, atualize o bot ou exclua-o. Se você receber a mesma exceção ao tentar excluir a intenção novamente, repita até que a intenção não tenha referências e a chamada para `DeleteIntent` seja bem-sucedida.

Essa operação exige permissão para a ação `lex:DeleteIntent`.

Sintaxe da Solicitação

```
DELETE /intents/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome da intenção. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteIntentVersion

Serviço: Amazon Lex Model Building Service

Exclui uma versão específica de uma intenção. Para excluir todas as versões de uma intenção, use a operação DeleteIntent.

Essa operação exige permissões para a ação `lex:DeleteIntentVersion`.

Sintaxe da Solicitação

```
DELETE /intents/name/versions/version HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome da intenção.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

version

A versão da intenção a ser excluída. Você não pode excluir a versão `$LATEST` da intenção. Para excluir a versão `$LATEST`, use a operação DeleteIntent.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[0-9]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteSlotType

Serviço: Amazon Lex Model Building Service

Exclui todas as versões do tipo de slot, incluindo a versão `$LATEST`. Para excluir uma versão específica do tipo de slot, use a operação DeleteSlotTypeVersion.

Você só pode excluir uma versão de um tipo de slot se ela não for referenciada. Para excluir um tipo de slot mencionado em uma ou mais intenções, primeiro você deve remover essas referências.

> **Note**
>
> Se você receber a exceção `ResourceInUseException`, ela fornecerá um exemplo de referência que mostra a intenção onde o tipo de slot é referenciado. Para remover a referência ao tipo de slot, atualize a intenção ou exclua-o. Se você receber a mesma exceção ao tentar excluir o tipo de slot novamente, repita até que o tipo de slot não tenha referências e a chamada para `DeleteSlotType` seja bem-sucedida.

Essa operação exige permissão para a ação `lex:DeleteSlotType`.

Sintaxe da Solicitação

```
DELETE /slottypes/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do tipo de slot. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteSlotTypeVersion

Serviço: Amazon Lex Model Building Service

Exclui uma versão específica de um tipo de slot. Para excluir todas as versões de um tipo de slot, use a operação DeleteSlotType.

Essa operação exige permissões para a ação `lex:DeleteSlotTypeVersion`.

Sintaxe da Solicitação

```
DELETE /slottypes/name/version/version HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do tipo de slot.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

version

A versão do tipo de slot a ser excluída. Você não pode excluir a versão `$LATEST` do tipo de slot. Para excluir a versão `$LATEST`, use a operação DeleteSlotType.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[0-9]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

ResourceInUseException

O recurso que você está tentando excluir é referenciado por outro recurso. Use essas informações para remover referências ao recurso que você está tentando excluir.

O corpo da exceção contém um objeto JSON que descreve o recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de Status HTTP: 400

Consulte Também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteUtterances

Serviço: Amazon Lex Model Building Service

Exclui as declarações armazenadas.

O Amazon Lex armazena as declarações que os usuários enviam para o seu bot. As declarações são armazenadas por 15 dias para uso com a operação GetUtterancesView e, em seguida, armazenadas indefinidamente para uso na melhoria da capacidade do seu bot de responder às entradas do usuário.

Use a operação `DeleteUtterances` para excluir declarações armazenadas por um usuário específico. Quando você usa a operação `DeleteUtterances`, as declarações armazenadas para melhorar a capacidade do seu bot de responder às entradas do usuário são excluídas imediatamente. As declarações armazenadas para uso com a operação `GetUtterancesView` são excluídas após 15 dias.

Essa operação exige permissões para a ação `lex:DeleteUtterances`.

Sintaxe da Solicitação

```
DELETE /bots/botName/utterances/userId HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botName

O nome do bot que armazenou as declarações.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

userId

Um identificador exclusivo para o usuário que fez as declarações. Esse é o ID do usuário que foi enviado na solicitação de PostTextoperação PostContentou que continha o enunciado.

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma linguagem específica AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBot

Serviço: Amazon Lex Model Building Service

Retorna informações de metadados para um bot específico. Você deve fornecer o nome do bot e a versão ou o alias do bot.

Essa operação exige permissões para a ação `lex:GetBot`.

Sintaxe da Solicitação

```
GET /bots/name/versions/versionoralias HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do bot. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

versionoralias

A versão do alias do bot.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "abortStatement": {
      "messages": [
```

```
            {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
    },
    "checksum": "string",
    "childDirected": boolean,
    "clarificationPrompt": {
        "maxAttempts": number,
        "messages": [
            {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
    },
    "createdDate": number,
    "description": "string",
    "detectSentiment": boolean,
    "enableModelImprovements": boolean,
    "failureReason": "string",
    "idleSessionTTLInSeconds": number,
    "intents": [
        {
            "intentName": "string",
            "intentVersion": "string"
        }
    ],
    "lastUpdatedDate": number,
    "locale": "string",
    "name": "string",
    "nluIntentConfidenceThreshold": number,
    "status": "string",
    "version": "string",
    "voiceId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

abortStatement

A mensagem que o Amazon Lex retorna quando o usuário decide encerrar a conversa sem concluí-la. Para obter mais informações, consulte PutBot.

Tipo: objeto Statement

checksum

Soma de verificação do bot usada para identificar uma revisão específica da versão `$LATEST` do bot.

Tipo: String

childDirected

Para cada bot do Amazon Lex criado com o Serviço de criação de modelo do Amazon Lex, você deve especificar se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à Lei de Proteção à Privacidade Online Infantil (COPPA), especificando `true` ou `false` no `childDirected` campo. Ao especificar `true` no campo `childDirected`, você confirma que seu uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Ao especificar `false` no campo `childDirected`, você confirma que seu uso do Amazon Lex não está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Você não pode especificar um valor padrão para o campo `childDirected` que não reflita com precisão se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA.

Se o uso do Amazon Lex estiver relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos, você deverá obter qualquer consentimento parental verificável exigido pela COPPA. Para obter informações sobre o uso do Amazon Lex em conexão com sites, programas ou outros aplicativos direcionados, total ou parcialmente, a crianças menores de 13 anos, consulte as perguntas frequentes do Amazon Lex.

Tipo: booliano

clarificationPrompt

A mensagem que o Amazon Lex usa quando não entende a solicitação do usuário. Para obter mais informações, consulte PutBot.

Tipo: objeto Prompt

createdDate

A data e a hora em que o bot foi criado.

Tipo: Timestamp

description

Uma descrição do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

detectSentiment

Indica se as declarações inseridas pelo usuário devem ser enviadas ao Amazon Comprehend para análise de sentimento.

Tipo: booliano

enableModelImprovements

Indica se o bot usa melhorias de precisão. `true` indica que o bot está usando as melhorias, caso contrário, é `false`.

Tipo: booliano

failureReason

Se `status` for `FAILED`, o Amazon Lex explica por que não conseguiu criar o bot.

Tipo: String

idleSessionTTLInSeconds

O tempo máximo em segundos que o Amazon Lex retém os dados coletados em uma conversa. Para obter mais informações, consulte PutBot.

Tipo: inteiro

Faixa válida: valor mínimo de 60. Valor máximo de 86.400.

intents

Uma matriz de objetos `intent`. Para obter mais informações, consulte PutBot.

Tipo: matriz de objetos Intent

lastUpdatedDate

A data em que o bot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

locale

O local de destino deste bot.

Tipo: String

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

O nome do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

A pontuação que determina onde o Amazon Lex insere o`AMAZON.FallbackIntent`,`AMAZON.KendraSearchIntent`, ou ambos ao retornar intenções alternativas em uma resposta PostContent. PostText `AMAZON.FallbackIntent`será inserido se a pontuação de confiança para todas as intenções estiver abaixo desse valor. `AMAZON.KendraSearchIntent`só é inserido se estiver configurado para o bot.

Tipo: duplo

Intervalo válido: valor mínimo de 0. Valor máximo de 1.

status

O status do bot.

Quando o status é `BUILDING`, o Amazon Lex está criando o bot para teste e uso.

Se o status do bot for `READY_BASIC_TESTING`, você poderá testar o bot usando as declarações exatas especificadas nas intenções do bot. Quando o bot estiver pronto para o teste completo ou para ser executado, o status será `READY`.

Se houve um problema com a criação do bot, o status será `FAILED` e o campo `failureReason` explica por que o bot não foi criado.

Se o bot tiver sido salvo, mas não criado, o status será `NOT_BUILT`.

Tipo: String

Valores Válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

A versão do bot. Para um novo bot, a versão é sempre `$LATEST`.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

voiceId

O ID de voz do Amazon Polly que o Amazon Lex usa para a interação de voz com o usuário. Para obter mais informações, consulte PutBot.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBotAlias

Serviço: Amazon Lex Model Building Service

Retorna informações sobre um alias de bot do Amazon Lex. Para obter mais informações sobre aliases, consulte Versionamento e aliases.

Essa operação exige permissões para a ação `lex:GetBotAlias`.

Sintaxe da Solicitação

```
GET /bots/botName/aliases/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botName

O nome do bot.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

name

O nome do alias do bot. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "botName": "string",
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string",
            "resourcePrefix": "string"
         }
      ]
   },
   "createdDate": number,
   "description": "string",
   "lastUpdatedDate": number,
   "name": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

botName

O nome do bot para o qual o alias aponta.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

botVersion

A versão do bot para o qual o alias aponta.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

checksum

Soma de verificação do alias do bot.

Tipo: String

conversationLogs

As configurações que determinam como o Amazon Lex usa logs de conversa para o alias.

Tipo: objeto ConversationLogsResponse

createdDate

A data em que o alias do bot foi criado.

Tipo: Timestamp

description

Uma descrição do alias do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

lastUpdatedDate

A data em que o alias do bot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome do alias do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetBotAliases

Serviço: Amazon Lex Model Building Service

Retorna uma lista de alias para um bot específico do Amazon Lex.

Essa operação exige permissões para a ação `lex:GetBotAliases`.

Sintaxe da Solicitação

```
GET /bots/botName/aliases/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botName

O nome do bot.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

maxResults

O número máximo de aliases a ser retornado na resposta. O padrão é 50.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nameContains

Substring para corresponder aos nomes de alias do bot. Um alias será retornado se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

nextToken

Um token de paginação para buscar a próxima página de aliases. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de aliases, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "BotAliases": [
      {
         "botName": "string",
         "botVersion": "string",
         "checksum": "string",
         "conversationLogs": {
            "iamRoleArn": "string",
            "logSettings": [
               {
                  "destination": "string",
                  "kmsKeyArn": "string",
                  "logType": "string",
                  "resourceArn": "string",
                  "resourcePrefix": "string"
               }
            ]
         },
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

BotAliases

Uma matriz de objetos `BotAliasMetadata`, cada um descrevendo um alias de bot.

Tipo: matriz de objetos BotAliasMetadata

nextToken

Um token de paginação para buscar a próxima página de aliases. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de aliases, especifique o token de paginação na próxima solicitação.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++

- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBotChannelAssociation

Serviço: Amazon Lex Model Building Service

Retorna informações sobre a associação entre um bot do Amazon Lex e uma plataforma de mensagens.

Essa operação exige permissões para a ação `lex:GetBotChannelAssociation`.

Sintaxe da Solicitação

```
GET /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

aliasName

Um alias que aponta para a versão específica do bot do Amazon Lex à qual essa associação está sendo feita.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

botName

O nome do bot do Amazon Lex.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

name

O nome da associação entre o bot e o canal. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "botAlias": "string",
   "botConfiguration": {
      "string" : "string"
   },
   "botName": "string",
   "createdDate": number,
   "description": "string",
   "failureReason": "string",
   "name": "string",
   "status": "string",
   "type": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

botAlias

Um alias que aponta para a versão específica do bot do Amazon Lex à qual essa associação está sendo feita.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

botConfiguration

Fornece as informações que a plataforma de mensagens precisa para se comunicar com o bot do Amazon Lex.

Tipo: mapa de string para string

Entradas do mapa: número máximo de 10 itens.

botName

O nome do bot do Amazon Lex.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

createdDate

A data em que a associação entre o bot e o canal foi criada.

Tipo: Timestamp

description

A descrição da associação entre o bot e o canal.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

failureReason

Se `status` for `FAILED`, o Amazon Lex fornece o motivo da falha na criação da associação.

Tipo: String

name

O nome da associação entre o bot e o canal.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

status

O status do canal do bot.

- `CREATED` - O canal foi criado e está pronto para uso.
- `IN_PROGRESS` - A criação do canal está em andamento.
- `FAILED` - Houve um erro ao criar o canal. Para obter informações sobre o motivo da falha, consulte o campo `failureReason`.

Tipo: String

Valores Válidos: `IN_PROGRESS | CREATED | FAILED`

type

O tipo da plataforma de mensagens.

Tipo: String

Valores Válidos: `Facebook | Slack | Twilio-Sms | Kik`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBotChannelAssociations

Serviço: Amazon Lex Model Building Service

Retorna uma lista de todos os canais associados a um bot específico.

A operação `GetBotChannelAssociations` exige permissões para a ação `lex:GetBotChannelAssociations`.

Sintaxe da Solicitação

```
GET /bots/botName/aliases/aliasName/channels/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

aliasName

Um alias que aponta para a versão específica do bot do Amazon Lex à qual essa associação está sendo feita.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^(-|^([A-Za-z]_?)+$)$`

Exigido: Sim

botName

O nome do bot do Amazon Lex na associação.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

maxResults

O número máximo de associações a ser retornado na resposta. O padrão é 50.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nameContains

Substring a ser correspondente nos nomes das associações de canais. Uma associação será retornada se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”. Para retornar todas as associações de canais de bots, use um hífen (“-”) como `nameContains` parâmetro.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

nextToken

Um token de paginação para buscar a próxima página de associações. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de associações, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "botChannelAssociations": [
      {
         "botAlias": "string",
         "botConfiguration": {
            "string" : "string"
         },
         "botName": "string",
         "createdDate": number,
         "description": "string",
         "failureReason": "string",
         "name": "string",
         "status": "string",
         "type": "string"
      }
   ],
   "nextToken": "string"
```

```
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

botChannelAssociations

Uma matriz de objetos, uma para cada associação, que fornece informações sobre o bot do Amazon Lex e sua associação com o canal.

Tipo: matriz de objetos BotChannelAssociation

nextToken

Um token de paginação que busca a próxima página de associações. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de associações, especifique o token de paginação na próxima solicitação.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBots

Serviço: Amazon Lex Model Building Service

Retorna as informações do bot da seguinte forma:

- Se você fornecer o campo `nameContains`, a resposta incluirá informações sobre a versão `$LATEST` de todos os bots cujo nome contém a string especificada.
- Se você não especificar o campo `nameContains`, a operação retornará informações sobre a versão `$LATEST` de todos os seus bots.

Essa operação exige permissão para a ação `lex:GetBots`.

Sintaxe da Solicitação

```
GET /bots/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de bots a ser retornado na resposta retornada pela solicitação. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nameContains

Substring para corresponder aos nomes do bot. Um bot será retornado se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

nextToken

Um token de paginação que busca a próxima página de bots. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de bots, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

bots

Uma matriz de objetos `botMetadata`, com uma entrada para cada bot.

Tipo: matriz de objetos BotMetadata

nextToken

Se a resposta for truncada, ela incluirá um token de paginação que você pode especificar em sua próxima solicitação para buscar a próxima página de bots.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetBotVersions

Serviço: Amazon Lex Model Building Service

Obtém informações sobre todas as versões de um bot.

A operação `GetBotVersions` retorna um objeto `BotMetadata` para cada versão de um bot. Por exemplo, se um bot tiver três versões numeradas, a operação `GetBotVersions` retornará quatro objetos `BotMetadata` na resposta, um para cada versão numerada e um para a versão `$LATEST`.

A operação `GetBotVersions` sempre retorna pelo menos uma versão, a versão`$LATEST`.

Essa operação exige permissões para a ação `lex:GetBotVersions`.

Sintaxe da Solicitação

```
GET /bots/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de versões de bot a ser retornado em uma resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

name

O nome do bot para o qual as versões devem ser retornadas.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

nextToken

Um token de paginação para buscar a próxima página de versões de bot. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de versões, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

bots

Uma matriz de objetos `BotMetadata`, um para cada versão numerada do bot mais um para a versão `$LATEST`.

Tipo: matriz de objetos BotMetadata

nextToken

Um token de paginação para buscar a próxima página de versões de bot. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de versões, especifique o token de paginação na próxima solicitação.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetBuiltinIntent

Serviço: Amazon Lex Model Building Service

Retorna informações sobre uma intenção embutida.

Essa operação exige permissão para a ação `lex:GetBuiltinIntent`.

Sintaxe da Solicitação

```
GET /builtins/intents/signature HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

signature

O identificador exclusivo de uma intenção integrada. Para encontrar a assinatura de uma intenção, consulte Intenções integradas padrão no Alexa Skills Kit.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "signature": "string",
   "slots": [
      {
         "name": "string"
      }
   ],
   "supportedLocales": [ "string" ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

signature

O identificador exclusivo de uma intenção integrada.

Tipo: String

slots

Uma matriz de objetos `BuiltinIntentSlot`, uma entrada para cada tipo de slot na intenção.

Tipo: matriz de objetos BuiltinIntentSlot

supportedLocales

Uma lista de localidades compatíveis com a intenção.

Tipo: matriz de strings

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBuiltinIntents

Serviço: Amazon Lex Model Building Service

Obtém uma lista de intenções integradas que atendem aos critérios especificados.

Essa operação exige permissão para a ação `lex:GetBuiltinIntents`.

Sintaxe da Solicitação

```
GET /builtins/intents/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

locale

Uma lista de localidades compatíveis com a intenção.

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

maxResults

O número máximo de intenções a serem incluídas na resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nextToken

Um token de paginação que busca a próxima página de intenções. Se a chamada de API for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de intenções, especifique o token de paginação na próxima solicitação.

signatureContains

Substring para corresponder às assinaturas de intenção integradas. Uma intenção será retornada se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”. Para encontrar a assinatura de uma intenção, consulte Intenções integradas padrão no Alexa Skills Kit.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

intents

Uma matriz de objetos `builtinIntentMetadata`, uma para cada intenção na resposta.

Tipo: matriz de objetos BuiltinIntentMetadata

nextToken

Um token de paginação que busca a próxima página de intenções. Se a resposta a essa chamada de API for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de intenções, especifique o token de paginação na próxima solicitação.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBuiltinSlotTypes

Serviço: Amazon Lex Model Building Service

Obtém uma lista dos tipos de slots integrados que atendem aos critérios especificados.

Para obter uma lista de tipos de slot integrados, consulte Referência do tipo de slot no Alexa Skills Kit.

Essa operação exige permissão para a ação `lex:GetBuiltInSlotTypes`.

Sintaxe da Solicitação

```
GET /builtins/slottypes/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

locale

Uma lista de localidades compatíveis pelo tipo de slot.

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

maxResults

O número máximo de tipo de slot a serem incluídos na resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nextToken

Um token de paginação que busca a próxima página de tipos de slot. Se a resposta a essa chamada de API for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de tipos de slot, especifique o token de paginação na próxima solicitação.

signatureContains

Substring para corresponder às assinaturas de tipo de slot integradas. Um tipo de slot será retornado se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

nextToken

Se a resposta for truncada, ela incluirá um token de paginação que você pode usar em sua próxima solicitação para buscar a próxima página de tipos de slot.

Tipo: String

slotTypes

Uma matriz de objetos `BuiltInSlotTypeMetadata`, uma entrada para cada tipo de slot retornado.

Tipo: matriz de objetos BuiltinSlotTypeMetadata

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetExport

Serviço: Amazon Lex Model Building Service

Exporta o conteúdo de um recurso do Amazon Lex em um formato especificado.

Sintaxe da Solicitação

```
GET /exports/?exportType=exportType&name=name&resourceType=resourceType&version=version
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

exportType

O formato dos dados exportados.

Valores Válidos: `ALEXA_SKILLS_KIT | LEX`

Obrigatório: Sim

name

O nome do bot a ser exportado.

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 100.

Padrão: `[a-zA-Z_]+`

Exigido: Sim

resourceType

O tipo de recurso a ser exportado.

Valores Válidos: `BOT | INTENT | SLOT_TYPE`

Obrigatório: Sim

version

A versão do bot a ser exportado.

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 64.

Padrão: `[0-9]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "exportStatus": "string",
   "exportType": "string",
   "failureReason": "string",
   "name": "string",
   "resourceType": "string",
   "url": "string",
   "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

exportStatus

O status da exportação.

- `IN_PROGRESS` - A exportação está em andamento.
- `READY` - A exportação foi concluída.
- `FAILED` - A exportação não pôde ser concluída.

Tipo: String

Valores Válidos: `IN_PROGRESS | READY | FAILED`

exportType

O formato dos dados exportados.

Tipo: String

Valores Válidos: `ALEXA_SKILLS_KIT | LEX`

failureReason

Se `status` for `FAILED`, o Amazon Lex fornece o motivo da falha da exportação do recurso.

Tipo: String

name

O nome do bot que está sendo exportado.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `[a-zA-Z_]+`

resourceType

O tipo do recurso exportado.

Tipo: String

Valores Válidos: `BOT | INTENT | SLOT_TYPE`

url

Uma URL pré-assinada do S3 que fornece a localização do recurso exportado. O recurso exportado é um arquivo ZIP que contém o recurso exportado no formato JSON. A estrutura do arquivamento pode mudar. Seu código não deve depender da estrutura de arquivamento.

Tipo: String

version

A versão do bot que está sendo exportado.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin

- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetImport

Serviço: Amazon Lex Model Building Service

Obtém informações sobre um trabalho de importação iniciado com a operação `StartImport`.

Sintaxe da Solicitação

```
GET /imports/importId HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

importId

O identificador das informações do trabalho de importação a serem retornadas.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "createdDate": number,
   "failureReason": [ "string" ],
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

createdDate

Um carimbo de data e hora para a data e a hora em que a tarefa de importação foi criada.

Tipo: Timestamp

failureReason

Uma sequência de caracteres que descreve por que um trabalho de importação não foi concluído.

Tipo: matriz de strings

importId

O identificador para a tarefa de importação específica.

Tipo: String

importStatus

O status do trabalho de importação. Se o status for `FAILED`, você poderá obter o motivo da falha no campo `failureReason`.

Tipo: String

Valores Válidos: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

A ação tomada quando houve um conflito entre um recurso existente e um recurso no arquivo de importação.

Tipo: String

Valores Válidos: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

O nome dado ao trabalho de importação.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `[a-zA-Z_]+`

resourceType

O tipo do recurso a ser importado.

Tipo: String

Valores Válidos: `BOT | INTENT | SLOT_TYPE`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET

- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetIntent

Serviço: Amazon Lex Model Building Service

Retorna informações sobre uma intenção. Além do nome da intenção, você deve especificar a versão da intenção.

Essa operação exige permissões para executar a ação `lex:GetIntent`.

Sintaxe da Solicitação

```
GET /intents/name/versions/version HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome da intenção. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

version

A versão da intenção.

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "rejectionStatement": {
         "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      }
   },
   "fulfillmentActivity": {
      "codeHook": {
         "messageVersion": "string",
         "uri": "string"
      },
      "type": "string"
   },
   "inputContexts": [
      {
         "name": "string"
      }
   ],
   "kendraConfiguration": {
      "kendraIndex": "string",
      "queryFilterString": "string",
      "role": "string"
   },
   "lastUpdatedDate": number,
   "name": "string",
   "outputContexts": [
      {
         "name": "string",
         "timeToLiveInSeconds": number,
         "turnsToLive": number
      }
   ],
   "parentIntentSignature": "string",
   "rejectionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
```

```
        "responseCard": "string"
    },
    "sampleUtterances": [ "string" ],
    "slots": [
        {
            "defaultValueSpec": {
                "defaultValueList": [
                    {
                        "defaultValue": "string"
                    }
                ]
            },
            "description": "string",
            "name": "string",
            "obfuscationSetting": "string",
            "priority": number,
            "responseCard": "string",
            "sampleUtterances": [ "string" ],
            "slotConstraint": "string",
            "slotType": "string",
            "slotTypeVersion": "string",
            "valueElicitationPrompt": {
                "maxAttempts": number,
                "messages": [
                    {
                        "content": "string",
                        "contentType": "string",
                        "groupNumber": number
                    }
                ],
                "responseCard": "string"
            }
        }
    ],
    "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

checksum

A soma de verificação da intenção.

Tipo: String

conclusionStatement

Depois que a função do Lambda especificada no elemento `fulfillmentActivity` cumpre a intenção, o Amazon Lex transmite essa declaração ao usuário.

Tipo: objeto Statement

confirmationPrompt

Se definido no bot, o prompt que o Amazon Lex usa para confirmar a intenção do usuário antes de atendê-la. Para obter mais informações, consulte PutIntent.

Tipo: objeto Prompt

createdDate

A data em que a intenção foi criada.

Tipo: Timestamp

description

Uma descrição da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

dialogCodeHook

Se definido no bot, o Amazon Lex invoca essa função do Lambda para cada entrada do usuário. Para obter mais informações, consulte PutIntent.

Tipo: objeto CodeHook

followUpPrompt

Se definido no bot, o Amazon Lex usa esse prompt para solicitar atividades adicionais do usuário depois que a intenção for atendida. Para obter mais informações, consulte PutIntent.

Tipo: objeto FollowUpPrompt

fulfillmentActivity

Descreve como a intenção é atendida. Para obter mais informações, consulte PutIntent.

Tipo: objeto FulfillmentActivity

inputContexts

Uma matriz de objetos `InputContext` que lista os contextos que devem estar ativos para que o Amazon Lex escolha a intenção em uma conversa com o usuário.

Tipo: matriz de InputContext objetos

Membros da Matriz: número mínimo de 0 itens. Número máximo de 5 itens.

kendraConfiguration

Informações de configuração, se houver, para conectar um índice do Amazon Kendra com a intenção `AMAZON.KendraSearchIntent`.

Tipo: objeto KendraConfiguration

lastUpdatedDate

A data em que a intenção foi atualizada. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

outputContexts

Uma matriz de objetos `OutputContext` que lista os contextos que a intenção ativa quando a intenção é atendida.

Tipo: matriz de objetos OutputContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

parentIntentSignature

Um identificador exclusivo de uma intenção integrada.

Tipo: String

rejectionStatement

Quando o usuário responde “não” à pergunta definida em `confirmationPrompt`, o Amazon Lex responde com essa mensagem para confirmar que a intenção foi cancelada.

Tipo: objeto Statement

sampleUtterances

Uma matriz de amostra de declarações configurada para a intenção.

Tipo: matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 1.500 itens.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 200.

slots

Uma matriz de slots de intenção configurados para a intenção.

Tipo: matriz de objetos Slot

Membros da Matriz: número mínimo de 0 itens. Número máximo de 100 itens.

version

A versão da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetIntents

Serviço: Amazon Lex Model Building Service

Retorna as informações da intenção da seguinte forma:

- Se você especificar o campo `nameContains`, retornará a versão `$LATEST` de todas as intenções que contêm a string especificada.
- Se você não especificar o campo `nameContains`, a operação retornará informações sobre a versão `$LATEST` de todas as intenções.

A operação exige permissões para a ação `lex:GetIntents`.

Sintaxe da Solicitação

```
GET /intents/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de intenções a serem incluídas na resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nameContains

Substring para corresponder aos nomes da intenção. Uma intenção será retornada se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

nextToken

Um token de paginação que busca a próxima página de intenções. Se a resposta a essa chamada de API for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de intenções, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

intents

Uma matriz de objetos `Intent`. Para obter mais informações, consulte PutBot.

Tipo: matriz de objetos IntentMetadata

nextToken

Se a resposta for truncada, ela incluirá um token de paginação que você pode especificar em sua próxima solicitação para buscar a próxima página de intenções.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetIntentVersions

Serviço: Amazon Lex Model Building Service

Obtém informações sobre todas as versões de uma intenção.

A operação `GetIntentVersions` retorna um objeto `IntentMetadata` para cada versão de uma intenção. Por exemplo, se uma intenção tiver três versões numeradas, a operação `GetIntentVersions` retornará quatro objetos `IntentMetadata` na resposta, um para cada versão numerada e um para a versão `$LATEST`.

A operação `GetIntentVersions` sempre retorna pelo menos uma versão, a versão`$LATEST`.

Essa operação exige permissões para a ação `lex:GetIntentVersions`.

Sintaxe da Solicitação

```
GET /intents/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de versões de intenção a ser retornado em uma resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

name

O nome da intenção para a qual as versões devem ser retornadas.

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

nextToken

Um token de paginação para buscar a próxima página de versões de intenção. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de versões, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

intents

Uma matriz de objetos `IntentMetadata`, um para cada versão numerada da intenção mais um para a versão `$LATEST`.

Tipo: matriz de objetos IntentMetadata

nextToken

Um token de paginação para buscar a próxima página de versões de intenção. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de versões, especifique o token de paginação na próxima solicitação.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetMigration

Serviço: Amazon Lex Model Building Service

Fornece detalhes sobre uma migração contínua ou completa de um bot do Amazon Lex V1 para um bot do Amazon Lex V2. Use essa operação para visualizar os alertas e avisos de migração relacionados à migração.

Sintaxe da Solicitação

```
GET /migrations/migrationId HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

migrationId

O identificador exclusivo da migração a ser visualizada. O `migrationID` é retornado pela operação StartMigration.

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "alerts": [
      {
         "details": [ "string" ],
         "message": "string",
         "referenceURLs": [ "string" ],
         "type": "string"
      }
```

```
    ],
    "migrationId": "string",
    "migrationStatus": "string",
    "migrationStrategy": "string",
    "migrationTimestamp": number,
    "v1BotLocale": "string",
    "v1BotName": "string",
    "v1BotVersion": "string",
    "v2BotId": "string",
    "v2BotRole": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

alerts

Uma lista de alertas e avisos que indicam problemas com a migração do bot do Amazon Lex V1 para o Amazon Lex V2. Você recebe um aviso quando um atributo do Amazon Lex V1 tem uma implementação diferente no Amazon Lex V2.

Para obter mais informações, consulte Migração de um bot no Guia do desenvolvedor do Amazon Lex V2.

Tipo: matriz de objetos MigrationAlert

migrationId

O identificador exclusivo da migração. É o mesmo identificador usado ao chamar a operação `GetMigration`.

Tipo: String

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

migrationStatus

Indica o status da migração. Quando o status é `COMPLETE`, a migração é concluída e o bot está disponível no Amazon Lex V2. Pode haver alertas e avisos que precisem ser resolvidos para concluir a migração.

Tipo: String

Valores Válidos: `IN_PROGRESS | COMPLETED | FAILED`

migrationStrategy

A estratégia usada para conduzir a migração.

- `CREATE_NEW` - Cria um novo bot do Amazon Lex V2 e migra o bot do Amazon Lex V1 para o novo bot.
- `UPDATE_EXISTING` - Substitui os metadados existentes do bot do Amazon Lex V2 e a localidade que está sendo migrada. Isso não altera nenhuma outra localidade no bot do Amazon Lex V2. Se a localidade não existir, uma nova localidade será criada no bot do Amazon Lex V2.

Tipo: String

Valores Válidos: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

A data e hora em que a migração foi iniciada.

Tipo: Timestamp

v1BotLocale

A localidade do bot do Amazon Lex V1 foi migrada para o Amazon Lex V2.

Tipo: String

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

v1BotName

O nome do bot do Amazon Lex V1 foi migrado para o Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

v1BotVersion

A versão do bot do Amazon Lex V1 foi migrado para o Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

v2BotId

O identificador exclusivo do bot do Amazon Lex V2 para o qual o Amazon Lex V1 está sendo migrado.

Tipo: String

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

v2BotRole

O perfil do IAM que o Amazon Lex usa para executar a versão do bot do Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetMigrations

Serviço: Amazon Lex Model Building Service

Obtém uma lista das migrações entre o Amazon Lex V1 e o Amazon Lex V2.

Sintaxe da Solicitação

```
GET /migrations?
maxResults=maxResults&migrationStatusEquals=migrationStatusEquals&nextToken=nextToken&sortByAtt
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de migrações a serem retornadas na resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

migrationStatusEquals

Filtra a lista para conter somente migrações no estado especificado.

Valores Válidos: `IN_PROGRESS | COMPLETED | FAILED`

nextToken

Um token de paginação que busca a próxima página de migrações. Se a resposta a essa operação for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de migrações, especifique o token de paginação na solicitação.

sortByAttribute

O campo pelo qual classificar a lista de migrações. Você pode classificar pelo nome do bot do Amazon Lex V1 ou pela data e hora em que a migração foi iniciada.

Valores Válidos: `V1_BOT_NAME | MIGRATION_DATE_TIME`

sortByOrder

A ordem para classificar a lista.

Valores Válidos: `ASCENDING | DESCENDING`

v1BotNameContains

Filtra a lista para conter somente bots cujo nome contenha a string especificada. A string é correspondida em qualquer lugar no nome do bot.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "migrationSummaries": [
      {
         "migrationId": "string",
         "migrationStatus": "string",
         "migrationStrategy": "string",
         "migrationTimestamp": number,
         "v1BotLocale": "string",
         "v1BotName": "string",
         "v1BotVersion": "string",
         "v2BotId": "string",
         "v2BotRole": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

migrationSummaries

Uma matriz de resumos para migrações do Amazon Lex V1 para o Amazon Lex V2. Para ver detalhes da migração, use o `migrationId` do resumo em uma chamada para a operação GetMigration.

Tipo: matriz de objetos MigrationSummary

nextToken

Se a resposta for truncada, ela incluirá um token de paginação que você pode especificar em sua próxima solicitação para buscar a próxima página de migrações.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET

- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetSlotType

Serviço: Amazon Lex Model Building Service

Retorna informações sobre uma versão específica de um tipo de slot. Além de especificar o nome do tipo de slot, você deve especificar a versão do tipo de slot.

Essa operação exige permissões para a ação `lex:GetSlotType`.

Sintaxe da Solicitação

```
GET /slottypes/name/versions/version HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do tipo de slot. O nome diferencia maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

version

A versão do tipo de slot.

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "checksum": "string",
   "createdDate": number,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

checksum

Soma de verificação da versão `$LATEST` do tipo de slot.

Tipo: String

createdDate

A data em que o tipo de slot foi criado.

Tipo: Timestamp

description

Uma descrição do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

enumerationValues

Uma lista de objetos `EnumerationValue` que define os valores que o tipo de slot pode ter.

Tipo: matriz de objetos EnumerationValue

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10.000 itens.

lastUpdatedDate

A data em que o tipo de slot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

O tipo de slot integrado usado como pai do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informações de configuração que estendem o tipo de slot integrado principal.

Tipo: matriz de objetos SlotTypeConfiguration

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

valueSelectionStrategy

A estratégia que o Amazon Lex usa para determinar o valor de slot. Para obter mais informações, consulte PutSlotType.

Tipo: String

Valores Válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

A versão do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetSlotTypes

Serviço: Amazon Lex Model Building Service

Retorna as informações do tipo de slot da seguinte forma:

- Se você especificar o campo `nameContains`, retornará a versão `$LATEST` de todos os tipos de slot que contêm a string especificada.
- Se você não especificar o campo `nameContains`, a operação retornará informações sobre a versão `$LATEST` de todos os tipos de slot.

A operação exige permissões para a ação `lex:GetSlotTypes`.

Sintaxe da Solicitação

```
GET /slottypes/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de tipo de slot a serem incluídos na resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

nameContains

Substring para corresponder aos nomes do tipo de slot. Um tipo de slot será retornado se alguma parte de seu nome corresponder ao substring. Por exemplo, “xyz” corresponde a “xyzabc” e “abcxyz”.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

nextToken

Um token de paginação que busca a próxima página de tipos de slot. Se a resposta a essa chamada de API for truncada, o Amazon Lex retornará um token de paginação na resposta.

Para buscar a próxima página de tipos de slot, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

nextToken

Se a resposta for truncada, ela incluirá um token de paginação que você pode especificar em sua próxima solicitação para buscar a próxima página de tipos de slot.

Tipo: String

slotTypes

Uma matriz de objetos, um para cada tipo de slot, que fornece informações como o nome do tipo de slot, a versão e uma descrição.

Tipo: matriz de objetos SlotTypeMetadata

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin

- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetSlotTypeVersions

Serviço: Amazon Lex Model Building Service

Obtém informações sobre todas as versões de um tipo de slot.

A operação `GetSlotTypeVersions` retorna um objeto `SlotTypeMetadata` para cada versão de um tipo de slot. Por exemplo, se um tipo de slot tiver três versões numeradas, a operação `GetSlotTypeVersions` retornará quatro objetos `SlotTypeMetadata` na resposta, um para cada versão numerada e um para a versão `$LATEST`.

A operação `GetSlotTypeVersions` sempre retorna pelo menos uma versão, a versão`$LATEST`.

Essa operação exige permissões para a ação `lex:GetSlotTypeVersions`.

Sintaxe da Solicitação

```
GET /slottypes/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

maxResults

O número máximo de versões de tipo de slot a serem incluídos na resposta. O padrão é 10.

Faixa válida: valor mínimo de 1. Valor máximo de 50.

name

O nome do tipo de slot para o qual as versões devem ser retornadas.

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

nextToken

Um token de paginação para buscar a próxima página de versões de tipo de slot. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de versões, especifique o token de paginação na próxima solicitação.

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

nextToken

Um token de paginação para buscar a próxima página de versões de tipo de slot. Se a resposta a essa chamada for truncada, o Amazon Lex retornará um token de paginação na resposta. Para buscar a próxima página de versões, especifique o token de paginação na próxima solicitação.

Tipo: String

slotTypes

Uma matriz de objetos `SlotTypeMetadata`, um para cada versão numerada do tipo de versão mais um para a versão `$LATEST`.

Tipo: matriz de objetos SlotTypeMetadata

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetUtterancesView

Serviço: Amazon Lex Model Building Service

Use a operação `GetUtterancesView` para obter informações sobre as declarações que seus usuários fizeram ao seu bot. Você pode usar essa lista para ajustar as declarações aos quais seu bot responde.

Por exemplo, digamos que você crie um bot para pedir flores. Depois que seus usuários usarem seu bot por um tempo, use a operação `GetUtterancesView` para ver as solicitações que eles fizeram e se foram bem-sucedidas. Você pode descobrir que a declaração “Eu quero flores” não está sendo reconhecida. Você pode adicionar essa declaração à intenção `OrderFlowers` para que seu bot reconheça essa declaração.

Depois de publicar uma nova versão de um bot, você pode obter informações sobre a versão antiga e a nova para poder comparar o desempenho entre as duas versões.

As estatísticas de declaração são geradas uma vez por dia. Os dados ficam disponíveis nos últimos 15 dias. Você pode solicitar informações de até 5 versões do seu bot em cada solicitação. O Amazon Lex retorna as declarações mais frequentes recebidas pelo bot nos últimos 15 dias. A resposta contém informações sobre um máximo de 100 declarações para cada versão.

As estatísticas de declaração não são geradas nas seguintes condições:

- O campo `childDirected` foi definido como verdadeiro quando o bot foi criado.
- Você está usando a ofuscação de slots com um ou mais slots.
- Você optou por não participar da melhoria do Amazon Lex.

Essa operação exige permissões para a ação `lex:GetUtterancesView`.

Sintaxe da Solicitação

```
GET /bots/botname/utterances?
view=aggregation&bot_versions=botVersions&status_type=statusType HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botname

O nome do bot para o qual as informações da declaração devem ser retornadas.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

botVersions

Uma matriz de versões de bot para o qual as informações da declaração devem ser retornadas. O limite é de 5 versões por solicitação.

Membros da Matriz: número mínimo de 1 item. Número máximo de 5 itens.

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Exigido: Sim

statusType

Para retornar declarações que foram reconhecidas e tratadas, use `Detected`. Para retornar declarações que não foram reconhecidas, use `Missed`.

Valores Válidos: `Detected | Missed`

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "botName": "string",
   "utterances": [
      {
         "botVersion": "string",
```

```
        "utterances": [
           {
              "count": number,
              "distinctUsers": number,
              "firstUtteredDate": number,
              "lastUtteredDate": number,
              "utteranceString": "string"
           }
        ]
     }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

botName

O nome do bot para o qual as informações da declaração foram retornadas.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

utterances

Uma matriz de objetos UtteranceList, cada um contendo uma lista de UtteranceData objetos descrevendo as declarações que foram processadas pelo seu bot. A resposta contém no máximo 100 objetos `UtteranceData` para cada versão. O Amazon Lex retorna as declarações mais frequentes recebidas pelo bot nos últimos 15 dias.

Tipo: matriz de objetos UtteranceList

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListTagsForResource

Serviço: Amazon Lex Model Building Service

Obtém uma lista de tags associadas a um recurso especificado. Somente bots, aliases de bots e canais de bots podem ter tags associadas a eles.

Sintaxe da Solicitação

```
GET /tags/resourceArn HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

resourceArn

O nome do recurso da Amazon (ARN) do recurso para o qual você deseja a lista de tags.

Restrições de Tamanho: tamanho mínimo 1. Tamanho máximo de 1011.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

tags

As tags associadas a um recurso.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET

- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutBot

Serviço: Amazon Lex Model Building Service

Cria um bot de conversa do Amazon Lex ou substitui um bot existente. Ao criar ou atualizar um bot, você só precisa especificar um nome, uma localidade e se o bot é direcionado a crianças menores de 13 anos. Você pode usar isso para adicionar intenções posteriormente ou para remover intenções de um bot existente. Quando você cria um bot com o mínimo de informações, o bot é criado ou atualizado, mas o Amazon Lex retorna a resposta `FAILED`. Você pode criar o bot depois de adicionar uma ou mais intenções. Para obter mais informações sobre o Amazon Lex, consulte o Amazon Lex: como funciona.

Se você especificar o nome de um bot existente, os campos na solicitação substituirão os valores existentes na versão `$LATEST` do bot. O Amazon Lex remove todos os campos para os quais você não fornece valores na solicitação, exceto os campos `idleTTLInSeconds` e `privacySettings`, que são definidos com seus valores padrão. Se você não especificar valores para os campos obrigatórios, o Amazon Lex lançará uma exceção.

Essa operação exige permissões para a ação `lex:PutBot`. Para obter mais informações, consulte Gerenciamento de identidade e acesso para o Amazon Lex.

Sintaxe da Solicitação

```
PUT /bots/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
      "messages": [
```

```
            {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
    },
    "createVersion": boolean,
    "description": "string",
    "detectSentiment": boolean,
    "enableModelImprovements": boolean,
    "idleSessionTTLInSeconds": number,
    "intents": [
        {
            "intentName": "string",
            "intentVersion": "string"
        }
    ],
    "locale": "string",
    "nluIntentConfidenceThreshold": number,
    "processBehavior": "string",
    "tags": [
        {
            "key": "string",
            "value": "string"
        }
    ],
    "voiceId": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do bot. O nome não é sensível a maiúsculas e minúsculas.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

abortStatement

Quando o Amazon Lex não consegue entender a entrada do usuário no contexto, ele tenta extrair as informações algumas vezes. Depois disso, o Amazon Lex envia a mensagem definida no `abortStatement` para o usuário e, em seguida, cancela a conversa. Para definir o número de novas tentativas, use o campo `valueElicitationPrompt` para o tipo de slot.

Por exemplo, em um bot de pedidos de pizza, o Amazon Lex pode perguntar a um usuário "Que tipo de massa você quer?" Se a resposta do usuário não for uma das esperadas (por exemplo, "massa fina", "prato fundo" etc.), o Amazon Lex tentará obter uma resposta correta mais algumas vezes.

Por exemplo, em um aplicativo de pedidos de pizza, `OrderPizza` pode ser uma das intenções. Essa intenção pode exigir o slot `CrustType`. Você especifica o campo `valueElicitationPrompt` ao criar o slot `CrustType`.

Se você tiver definido uma intenção de fallback, a declaração de cancelamento não será enviada ao usuário; em vez disso, a intenção de fallback será usada. Para obter mais informações, consulte AMAZON. FallbackIntent.

Tipo: objeto Statement

Obrigatório: Não

checksum

Identifica uma revisão específica da versão `$LATEST`.

Ao criar um novo bot, deixe o campo `checksum` em branco. Se você especificar uma soma de verificação, obterá uma exceção `BadRequestException`.

Quando quiser atualizar um bot, defina o campo `checksum` como a soma de verificação da revisão mais recente da versão `$LATEST`. Se você não especificar o campo `checksum` ou se a soma de verificação não corresponder à versão `$LATEST`, você receberá uma exceção `PreconditionFailedException`.

Tipo: String

Obrigatório: Não

childDirected

Para cada bot do Amazon Lex criado com o Serviço de criação de modelo do Amazon Lex, você deve especificar se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à Lei de Proteção à Privacidade Online Infantil (COPPA), especificando `true` ou `false` no `childDirected` campo. Ao especificar `true` no campo `childDirected`, você confirma que seu uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Ao especificar `false` no campo `childDirected`, você confirma que seu uso do Amazon Lex não está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Você não pode especificar um valor padrão para o campo `childDirected` que não reflita com precisão se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA.

Se o uso do Amazon Lex estiver relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos, você deverá obter qualquer consentimento parental verificável exigido pela COPPA. Para obter informações sobre o uso do Amazon Lex em conexão com sites, programas ou outros aplicativos direcionados, total ou parcialmente, a crianças menores de 13 anos, consulte as perguntas frequentes do Amazon Lex.

Tipo: booliano

Obrigatório: Sim

clarificationPrompt

Quando o Amazon Lex não entende a intenção do usuário, ele usa essa mensagem para obter esclarecimentos. Para especificar quantas vezes o Amazon Lex deve repetir o prompt de esclarecimento, use o campo `maxAttempts`. Se o Amazon Lex ainda não entender, ele enviará a mensagem no campo `abortStatement`.

Ao criar um prompt de esclarecimento, certifique-se de que ela sugira a resposta correta do usuário. Por exemplo, para um bot que pede pizza e bebidas, você pode criar este prompt de esclarecimento: “O que você gostaria de fazer? Você pode dizer "Pedir uma bebida" ou "Pedir uma pizza".

Se você tiver definido uma intenção de fallback, ela será invocada se o prompt de esclarecimento for repetido o número de vezes definido no campo `maxAttempts`. Para obter mais informações, consulte AMAZON. FallbackIntent.

Se você não definir um prompt de esclarecimento, no runtime, o Amazon Lex retornará uma exceção de 400 solicitações inválidas em três casos:

- Prompt de acompanhamento - Quando o usuário responde a um prompt de acompanhamento, mas não fornece uma intenção. Por exemplo, em resposta a um prompt de acompanhamento que diz “Você gostaria de mais alguma coisa hoje?” O usuário diz: “Sim”. Como o Amazon Lex não tem um prompt de esclarecimento para obter uma intenção do usuário, ele retorna uma exceção 400 Solicitação inválida.
- Função do Lambda - Ao usar uma função do Lambda, você retorna um tipo de diálogo `ElicitIntent`. Como o Amazon Lex não tem um prompt de esclarecimento para obter uma intenção do usuário, ele retorna uma exceção 400 Solicitação inválida.
- PutSession operação - Ao usar a `PutSession` operação, você envia um tipo `ElicitIntent` de diálogo. Como o Amazon Lex não tem um prompt de esclarecimento para obter uma intenção do usuário, ele retorna uma exceção 400 Solicitação inválida.

Tipo: objeto Prompt

Obrigatório: Não

createVersion

Quando configurado para `true`, uma nova versão numerada do bot é criada. Isso é o mesmo que chamar a operação `CreateBotVersion`. Se `createVersion` não for especificado, o valor padrão será `false`.

Tipo: booliano

Obrigatório: não

description

Uma descrição do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: Não

detectSentiment

Quando configurado para o usuário `true`, as declarações do usuário são enviadas ao Amazon Comprehend para análise de sentimento. Se `detectSentiment` não for especificado, o valor padrão será `false`.

Tipo: booliano

Obrigatório: não

enableModelImprovements

Defina como `true` para permitir o acesso a melhorias na compreensão de linguagem natural.

Ao definir o parâmetro `enableModelImprovements` como `true`, você pode usar o parâmetro `nluIntentConfidenceThreshold` para configurar pontuações de confiança. Para obter mais informações, consulte Pontuações de confiança.

Você só pode definir o parâmetro `enableModelImprovements` em determinadas regiões. Se você definir o parâmetro como `true`, seu bot terá acesso a melhorias de precisão.

As regiões nas quais você pode definir o parâmetro `enableModelImprovements` como `false` a para a localidade en-US:

- Leste dos EUA (Norte da Virgínia) (us-east-1)
- Oeste dos EUA (Oregon) (us-west-2)
- Ásia-Pacífico (Sydney) (ap-southeast-2)
- Europa (Irlanda) (eu-west-1)

Em outras regiões e localidades, o parâmetro `enableModelImprovements` é definido como `true` por padrão. Nessas regiões e localidades, definir o parâmetro como `false` gera uma exceção `ValidationException`.

Tipo: booliano

Obrigatório: não

idleSessionTTLInSeconds

O tempo máximo em segundos que o Amazon Lex retém os dados coletados em uma conversa.

Uma sessão de interação do usuário permanecerá ativa pelo tempo especificado. Se nenhuma conversa ocorrer durante esse período, a sessão expirará, e o Amazon Lex excluirá todos os dados fornecidos antes do tempo limite.

Por exemplo, suponha que um usuário escolha a OrderPizza intenção, mas se distraia na metade do processo de fazer um pedido. Se o usuário não concluir o pedido dentro do prazo especificado, o Amazon Lex descartará as informações do slot coletadas e o usuário deverá começar de novo.

Se você não incluir o elemento `idleSessionTTLInSeconds` em uma solicitação da operação `PutBot`, o Amazon Lex usará o valor padrão. Isso também se aplica se a solicitação substituir um bot existente.

O padrão é 300 segundos (5 minutos).

Tipo: inteiro

Faixa válida: valor mínimo de 60. Valor máximo de 86.400.

Obrigatório: Não

intents

Uma matriz de objetos `Intent`. Cada intenção representa um comando que um usuário pode expressar. Por exemplo, um bot de pedido de pizza pode apoiar uma OrderPizza intenção. Para obter mais informações, consulte Amazon Lex: como funciona.

Tipo: matriz de objetos Intent

Obrigatório: Não

locale

Especifica a localidade de destino deste bot. Qualquer intenção usada no bot deve ser compatível com a localidade do bot.

O padrão é `en-US`.

Tipo: String

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Obrigatório: Sim

nluIntentConfidenceThreshold

Determina o limite em que o Amazon Lex inserirá o `AMAZON.FallbackIntent``AMAZON.KendraSearchIntent`,, ou ambos ao retornar intenções

alternativas em uma resposta PostContent. PostText `AMAZON.FallbackIntente` só `AMAZON.KendraSearchIntent` são inseridos se estiverem configurados para o bot.

Você deve definir o parâmetro `enableModelImprovements` como `true` para usar pontuações de confiança nas seguintes regiões.

- Leste dos EUA (Norte da Virgínia) (us-east-1)
- Oeste dos EUA (Oregon) (us-west-2)
- Ásia-Pacífico (Sydney) (ap-southeast-2)
- Europa (Irlanda) (eu-west-1)

Em outras regiões, o parâmetro `enableModelImprovements` é definido como `true` por padrão.

Por exemplo, suponha que um bot esteja configurado com o limite de confiança de 0,80 e o `AMAZON.FallbackIntent`. O Amazon Lex retorna três intenções alternativas com as seguintes pontuações de confiança: IntentA (0,70), IntentB (0,60), IntentC (0,50). A resposta da operação `PostText` seria:

- AMAZÔNIA. FallbackIntent
- IntentA
- IntentB
- IntentC

Tipo: duplo

Intervalo válido: valor mínimo de 0. Valor máximo de 1.

Obrigatório: Não

processBehavior

Se você definir o elemento `processBehavior` como `BUILD`, o Amazon Lex criará o bot para que ele possa ser executado. Se você definir o elemento como `SAVE`, o Amazon Lex salva o bot, mas não o cria.

Se você não especificar este valor, o valor padrão é `BUILD`.

Tipo: String

Valores Válidos: `SAVE | BUILD`

Obrigatório: Não

tags

Uma lista de tags a serem adicionadas ao bot. Você só pode adicionar tags ao criar um bot; não pode usar a operação `PutBot` para atualizar as tags em um bot. Para atualizar tags, use a operação `TagResource`.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Não

voiceId

O ID de voz do Amazon Polly que você quer que o Amazon Lex use para interações de voz com o usuário. A localidade configurada para a voz deve corresponder à localidade do bot. Para obter mais informações, consulte Vozes no Amazon Polly no Guia do desenvolvedor do Amazon Polly.

Tipo: String

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
```

```
    "clarificationPrompt": {
       "maxAttempts": number,
       "messages": [
          {
             "content": "string",
             "contentType": "string",
             "groupNumber": number
          }
       ],
       "responseCard": "string"
    },
    "createdDate": number,
    "createVersion": boolean,
    "description": "string",
    "detectSentiment": boolean,
    "enableModelImprovements": boolean,
    "failureReason": "string",
    "idleSessionTTLInSeconds": number,
    "intents": [
       {
          "intentName": "string",
          "intentVersion": "string"
       }
    ],
    "lastUpdatedDate": number,
    "locale": "string",
    "name": "string",
    "nluIntentConfidenceThreshold": number,
    "status": "string",
    "tags": [
       {
          "key": "string",
          "value": "string"
       }
    ],
    "version": "string",
    "voiceId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

abortStatement

A mensagem que o Amazon Lex usa para cancelar uma conversa. Para obter mais informações, consulte PutBot.

Tipo: objeto Statement

checksum

Soma de verificação do bot que você criou.

Tipo: String

childDirected

Para cada bot do Amazon Lex criado com o Serviço de criação de modelo do Amazon Lex, você deve especificar se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à Lei de Proteção à Privacidade Online Infantil (COPPA), especificando `true` ou `false` no `childDirected` campo. Ao especificar `true` no campo `childDirected`, você confirma que seu uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Ao especificar `false` no campo `childDirected`, você confirma que seu uso do Amazon Lex não está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA. Você não pode especificar um valor padrão para o campo `childDirected` que não reflita com precisão se o uso do Amazon Lex está relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos e sujeito à COPPA.

Se o uso do Amazon Lex estiver relacionado a um site, programa ou outro aplicativo direcionado, total ou parcialmente, a crianças menores de 13 anos, você deverá obter qualquer consentimento parental verificável exigido pela COPPA. Para obter informações sobre o uso do Amazon Lex em conexão com sites, programas ou outros aplicativos direcionados, total ou parcialmente, a crianças menores de 13 anos, consulte as perguntas frequentes do Amazon Lex.

Tipo: booliano

clarificationPrompt

Os prompts que o Amazon Lex usa quando não entende a intenção do usuário. Para obter mais informações, consulte PutBot.

Tipo: objeto Prompt

createdDate

A data e a hora em que o bot foi criado.

Tipo: Timestamp

createVersion

`True` se uma nova versão do bot tiver sido criada. Se o campo `createVersion` não tiver sido especificado na solicitação, o campo `createVersion` será definido como falso na resposta.

Tipo: booliano

description

Uma descrição do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

detectSentiment

`true` se o bot estiver configurado para enviar declarações do usuário ao Amazon Comprehend para análise de sentimento. Se o campo `detectSentiment` não tiver sido especificado na solicitação, o campo `detectSentiment` será `false` na resposta.

Tipo: booliano

enableModelImprovements

Indica se o bot usa melhorias de precisão. `true` indica que o bot está usando as melhorias, caso contrário, é `false`.

Tipo: booliano

failureReason

Se `status` for `FAILED`, o Amazon Lex fornece o motivo da falha na criação do bot.

Tipo: String

idleSessionTTLInSeconds

O tempo máximo que o Amazon Lex retém os dados coletados em uma conversa. Para obter mais informações, consulte PutBot.

Tipo: inteiro

Faixa válida: valor mínimo de 60. Valor máximo de 86.400.

intents

Uma matriz de objetos `Intent`. Para obter mais informações, consulte PutBot.

Tipo: matriz de objetos Intent

lastUpdatedDate

A data em que o bot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

locale

O local de destino deste bot.

Tipo: String

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

O nome do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

A pontuação que determina onde o Amazon Lex insere o`AMAZON.FallbackIntent`,`AMAZON.KendraSearchIntent`, ou ambos ao retornar intenções alternativas em uma resposta PostContent. PostText `AMAZON.FallbackIntent`será inserido se a pontuação de confiança para todas as intenções estiver abaixo desse valor. `AMAZON.KendraSearchIntent`só é inserido se estiver configurado para o bot.

Tipo: duplo

Intervalo válido: valor mínimo de 0. Valor máximo de 1.

status

Ao enviar uma solicitação para criar um bot com `processBehavior` definido como `BUILD`, o Amazon Lex define o elemento de resposta `status` como `BUILDING`.

No estado `READY_BASIC_TESTING`, você pode testar o bot com entradas do usuário que correspondem exatamente as declarações configuradas para as intenções e valores do bot nos tipos de slot.

Se o Amazon Lex não puder criar o bot, ele define `status` como `FAILED`. O Amazon Lex retorna o motivo da falha no elemento de resposta `failureReason`.

Quando você define `processBehavior` como `SAVE`, o Amazon Lex define o código de status como `NOT BUILT`.

Quando o bot está no estado `READY`, você pode testar e publicar o bot.

Tipo: String

Valores Válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

tags

Uma lista de tags associadas ao bot.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

version

A versão do bot. Para um novo bot, a versão é sempre `$LATEST`.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

voiceId

O ID de voz do Amazon Polly que o Amazon Lex usa para a interação de voz com o usuário. Para obter mais informações, consulte PutBot.

Tipo: String

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET

- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutBotAlias

Serviço: Amazon Lex Model Building Service

Cria um alias para a versão especificada do bot ou substitui um alias para o bot especificado. Para alterar a versão do bot para a qual o alias aponta, substitua o alias. Para obter mais informações sobre aliases, consulte Versionamento e aliases.

Essa operação exige permissões para a ação `lex:PutBotAlias`.

Sintaxe da Solicitação

```
PUT /bots/botName/aliases/name HTTP/1.1
Content-type: application/json

{
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string"
         }
      ]
   },
   "description": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botName

O nome do bot.

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

name

O nome do alias. O nome não é sensível a maiúsculas e minúsculas.

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

botVersion

A versão do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Exigido: Sim

checksum

Identifica uma revisão específica da versão `$LATEST`.

Ao criar um novo alias de bot, deixe o campo `checksum` em branco. Se você especificar uma soma de verificação, obterá uma exceção `BadRequestException`.

Quando quiser atualizar um alias de bot, defina o campo `checksum` como a soma de verificação da revisão mais recente da versão `$LATEST`. Se você não especificar o campo `checksum` ou se a soma de verificação não corresponder à versão `$LATEST`, você receberá uma exceção `PreconditionFailedException`.

Tipo: String

Obrigatório: Não

conversationLogs

Configurações para logs de conversas para o alias.

Tipo: objeto ConversationLogsRequest

Obrigatório: Não

description

Uma descrição do alias.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: Não

tags

Uma lista de tags a serem adicionadas ao alias de bot. Você só pode adicionar tags ao criar um alias de bot; não pode usar a operação `PutBotAlias` para atualizar as tags em um alias de bot. Para atualizar tags, use a operação `TagResource`.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "botName": "string",
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string",
            "resourcePrefix": "string"
         }
      ]
   },
   "createdDate": number,
   "description": "string",
   "lastUpdatedDate": number,
   "name": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

botName

O nome do bot para o qual o alias aponta.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

botVersion

A versão do bot para o qual o alias aponta.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

checksum

A soma de verificação para a versão atual do alias.

Tipo: String

conversationLogs

As configurações que determinam como o Amazon Lex usa logs de conversa para o alias.

Tipo: objeto ConversationLogsResponse

createdDate

A data em que o alias do bot foi criado.

Tipo: Timestamp

description

Uma descrição do alias.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

lastUpdatedDate

A data em que o alias do bot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome do alias.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

tags

Uma lista de tags associadas a um bot.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutIntent

Serviço: Amazon Lex Model Building Service

Cria uma intenção ou substitui uma intenção existente.

Para definir a interação entre o usuário e seu bot, você usa uma ou mais intenções. Para um bot de pedidos de pizza, por exemplo, você criaria uma intenção `OrderPizza`.

Para criar uma intenção ou substituir uma intenção existente, você deve fornecer o seguinte:

- Nome da intenção. Por exemplo, .`OrderPizza`
- Amostra de declarações. Por exemplo, “Posso pedir uma pizza, por favor.” e “Quero pedir uma pizza”.
- Informações a serem coletadas. Você especifica os tipos de slots para as informações que seu bot solicitará do usuário. Você pode especificar tipos de compartimentos padrão, como data ou hora, ou tipos de compartimentos personalizados, como o tamanho e a massa de uma pizza.
- Como a intenção será atendida. Você pode fornecer uma função do Lambda ou configurar a intenção para retornar as informações da intenção ao aplicativo cliente. Se você usa uma função do Lambda, quando todas as informações de intenção estão disponíveis, o Amazon Lex invoca sua função do Lambda. Se você configurar sua intenção para retornar as informações de intenção para o aplicativo cliente.

Você pode especificar outras informações opcionais na solicitação, como:

- Um prompt de confirmação para solicitar que o usuário confirme uma intenção. Por exemplo, “Devo pedir sua pizza?”
- Uma declaração de conclusão a ser enviada ao usuário após o atendimento da intenção. Por exemplo, “Devo pedir sua pizza?”.
- Um prompt de acompanhamento que solicita atividades adicionais ao usuário. Por exemplo, perguntar “Você quer pedir uma bebida com a sua pizza?”

Se você especificar um nome de intenção existente para atualizar a intenção, o Amazon Lex substituirá os valores na versão `$LATEST` da intenção pelos valores na solicitação. O Amazon Lex remove campos que você não fornece na solicitação. Se você não especificar os campos obrigatórios, o Amazon Lex lançará uma exceção. Quando você atualiza a versão `$LATEST` de uma intenção, o campo `status` de qualquer bot que usa a versão `$LATEST` da intenção é definido como `NOT_BUILT`.

Para obter mais informações, consulte Amazon Lex: como funciona.

Essa operação exige permissões para a ação `lex:PutIntent`.

Sintaxe da Solicitação

```
PUT /intents/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createVersion": boolean,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
```

```
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
    },
    "rejectionStatement": {
        "messages": [
            {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
    }
},
"fulfillmentActivity": {
    "codeHook": {
        "messageVersion": "string",
        "uri": "string"
    },
    "type": "string"
},
"inputContexts": [
    {
        "name": "string"
    }
],
"kendraConfiguration": {
    "kendraIndex": "string",
    "queryFilterString": "string",
    "role": "string"
},
"outputContexts": [
    {
        "name": "string",
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
],
"parentIntentSignature": "string",
"rejectionStatement": {
    "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "sampleUtterances": [ "string" ],
      "slots": [
         {
            "defaultValueSpec": {
               "defaultValueList": [
                  {
                     "defaultValue": "string"
                  }
               ]
            },
            "description": "string",
            "name": "string",
            "obfuscationSetting": "string",
            "priority": number,
            "responseCard": "string",
            "sampleUtterances": [ "string" ],
            "slotConstraint": "string",
            "slotType": "string",
            "slotTypeVersion": "string",
            "valueElicitationPrompt": {
               "maxAttempts": number,
               "messages": [
                  {
                     "content": "string",
                     "contentType": "string",
                     "groupNumber": number
                  }
               ],
               "responseCard": "string"
            }
         }
      ]
   }
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome da intenção. O nome não é sensível a maiúsculas e minúsculas.

O nome não pode corresponder a um nome de intenção incorporado ou a um nome de intenção incorporado com “AMAZON.” removido. Por exemplo, como há uma intenção integrada chamada `AMAZON.HelpIntent`, você não pode criar uma intenção personalizada chamada `HelpIntent`.

Para obter uma lista de intenções integradas, consulte Intenções integradas padrão no Alexa Skills Kit.

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

checksum

Identifica uma revisão específica da versão `$LATEST`.

Ao criar uma nova intenção, deixe o campo `checksum` em branco. Se você especificar uma soma de verificação, obterá uma exceção `BadRequestException`.

Quando quiser atualizar uma intenção, defina o campo `checksum` como a soma de verificação da revisão mais recente da versão `$LATEST`. Se você não especificar o campo `checksum` ou se a soma de verificação não corresponder à versão `$LATEST`, você receberá uma exceção `PreconditionFailedException`.

Tipo: String

Obrigatório: Não

conclusionStatement

A declaração que você deseja que o Amazon Lex transmita ao usuário após a intenção ser atendida com sucesso pela função do Lambda.

Esse elemento é relevante somente se você fornecer uma função do Lambda no `fulfillmentActivity`. Se você retornar a intenção ao aplicativo cliente, não poderá especificar esse elemento.

> **Note**
>
> Os `followUpPrompt` e `conclusionStatement` são mutuamente exclusivos. Você pode especificar apenas um.

Tipo: objeto Statement

Obrigatório: Não

confirmationPrompt

Solicita que o usuário confirme a intenção. Essa pergunta deve ter uma resposta afirmativa ou negativa.

O Amazon Lex usa esse prompt para garantir que o usuário reconheça que a intenção está pronta para ser atendida. Por exemplo, com a intenção `OrderPizza`, talvez você queira confirmar que o pedido está correto antes de fazer o pedido. Para outras finalidades, como intenções que simplesmente respondem às perguntas do usuário, talvez não seja necessário pedir confirmação ao usuário antes de fornecer as informações.

> **Note**
>
> Você deve fornecer tanto o `rejectionStatement` quanto o `confirmationPrompt`, ou nenhum.

Tipo: objeto Prompt

Obrigatório: Não

createVersion

Quando configurado para `true`, uma nova versão numerada da intenção é criada. Isso é o mesmo que chamar a operação `CreateIntentVersion`. Se você não especificar `createVersion`, o valor padrão será `false`.

Tipo: booliano

Obrigatório: não

description

Uma descrição da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: Não

dialogCodeHook

Especifica uma função do Lambda do alias para invocar cada entrada do usuário. Você pode invocar essa função do Lambda para personalizar a interação do usuário.

Por exemplo, suponha que seu bot determine que o usuário é John. Sua função do Lambda pode recuperar as informações de John de um banco de dados de back-end e preencher previamente alguns dos valores. Por exemplo, se achar que John é intolerante ao glúten, você pode definir o intervalo de intenção correspondente, `GlutenIntolerant`, como verdadeiro. Você pode encontrar o número de telefone de John e definir o atributo de sessão correspondente.

Tipo: objeto CodeHook

Obrigatório: Não

followUpPrompt

O Amazon Lex usa esse prompt para solicitar atividades adicionais do usuário depois de atender uma intenção. Por exemplo, depois que a intenção `OrderPizza` for atendida, você pode solicitar que o usuário peça uma bebida.

A ação que o Amazon Lex executa depende da resposta do usuário, da seguinte forma:

- Se o usuário disser “Sim”, ele responderá com o prompt de esclarecimento configurado para o bot.

- Se o usuário disser “Sim” e continuar com uma declaração que aciona uma intenção, ele iniciará uma conversa sobre a intenção.
- Se o usuário disser “Não”, ele responderá com a declaração de rejeição configurada para o prompt de acompanhamento.
- Se ele não reconhecer a declaração, ele repetirá o prompt de acompanhamento novamente.

O campo `followUpPrompt` e o campo `conclusionStatement` são mutuamente exclusivos. Você pode especificar apenas um.

Tipo: objeto FollowUpPrompt

Obrigatório: Não

fulfillmentActivity

Obrigatório. Descreve como a intenção é atendida. Por exemplo, depois que um usuário fornece todas as informações de um pedido de pizza, `fulfillmentActivity` define como o bot faz um pedido em uma pizzaria local.

Você pode configurar o Amazon Lex para retornar todas as informações de intenção ao aplicativo cliente ou direcioná-lo para invocar uma função do Lambda que possa processar a intenção (por exemplo, fazer um pedido em uma pizzaria).

Tipo: objeto FulfillmentActivity

Obrigatório: Não

inputContexts

Uma matriz de objetos `InputContext` que lista os contextos que devem estar ativos para que o Amazon Lex escolha a intenção em uma conversa com o usuário.

Tipo: matriz de InputContext objetos

Membros da Matriz: número mínimo de 0 itens. Número máximo de 5 itens.

Obrigatório: Não

kendraConfiguration

Informações de configuração necessárias para usar a intenção `AMAZON.KendraSearchIntent`para se conectar a um índice do Amazon Kendra. Para obter mais informações, consulte AMAZON. KendraSearchIntent.

Tipo: objeto KendraConfiguration

Obrigatório: Não

outputContexts

Uma matriz de objetos `OutputContext` que lista os contextos que a intenção ativa quando a intenção é atendida.

Tipo: matriz de objetos OutputContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

Obrigatório: Não

parentIntentSignature

O identificador exclusivo da intenção integrada a ser usada como base para essa intenção. Para encontrar a assinatura de uma intenção, consulte Intenções integradas padrão no Alexa Skills Kit.

Tipo: String

Obrigatório: Não

rejectionStatement

Quando o usuário responde “não” à pergunta definida em `confirmationPrompt`, o Amazon Lex responde com essa mensagem para confirmar que a intenção foi cancelada.

> **Note**
>
> Você deve fornecer tanto o `rejectionStatement` quanto o `confirmationPrompt`, ou nenhum.

Tipo: objeto Statement

Obrigatório: Não

sampleUtterances

Uma lista das declarações (strings) que um usuário pode fazer para sinalizar a intenção. Por exemplo, “Eu quero {PizzaSize} pizza”, “Pedir {Quantidade} {PizzaSize} pizzas”.

Em cada declaração, o nome de um slot é colocado entre chaves.

Tipo: matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 1.500 itens.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 200.

Obrigatório: Não

slots

Uma matriz de slots de intenção. Em runtime, o Amazon Lex extrai os valores de slot necessários do usuário usando prompts definidos nos slots. Para obter mais informações, consulte Amazon Lex: como funciona.

Tipo: matriz de objetos Slot

Membros da Matriz: número mínimo de 0 itens. Número máximo de 100 itens.

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
```

```
        "responseCard": "string"
    },
    "createdDate": number,
    "createVersion": boolean,
    "description": "string",
    "dialogCodeHook": {
        "messageVersion": "string",
        "uri": "string"
    },
    "followUpPrompt": {
        "prompt": {
            "maxAttempts": number,
            "messages": [
                {
                    "content": "string",
                    "contentType": "string",
                    "groupNumber": number
                }
            ],
            "responseCard": "string"
        },
        "rejectionStatement": {
            "messages": [
                {
                    "content": "string",
                    "contentType": "string",
                    "groupNumber": number
                }
            ],
            "responseCard": "string"
        }
    },
    "fulfillmentActivity": {
        "codeHook": {
            "messageVersion": "string",
            "uri": "string"
        },
        "type": "string"
    },
    "inputContexts": [
        {
            "name": "string"
        }
    ],
```

```
    "kendraConfiguration": {
       "kendraIndex": "string",
       "queryFilterString": "string",
       "role": "string"
    },
    "lastUpdatedDate": number,
    "name": "string",
    "outputContexts": [
       {
          "name": "string",
          "timeToLiveInSeconds": number,
          "turnsToLive": number
       }
    ],
    "parentIntentSignature": "string",
    "rejectionStatement": {
       "messages": [
          {
             "content": "string",
             "contentType": "string",
             "groupNumber": number
          }
       ],
       "responseCard": "string"
    },
    "sampleUtterances": [ "string" ],
    "slots": [
       {
          "defaultValueSpec": {
             "defaultValueList": [
                {
                   "defaultValue": "string"
                }
             ]
          },
          "description": "string",
          "name": "string",
          "obfuscationSetting": "string",
          "priority": number,
          "responseCard": "string",
          "sampleUtterances": [ "string" ],
          "slotConstraint": "string",
          "slotType": "string",
          "slotTypeVersion": "string",
```

```
        "valueElicitationPrompt": {
           "maxAttempts": number,
           "messages": [
              {
                 "content": "string",
                 "contentType": "string",
                 "groupNumber": number
              }
           ],
           "responseCard": "string"
        }
     }
  ],
  "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

checksum

Soma de verificação da versão `$LATEST` de intenção criada ou atualizada.

Tipo: String

conclusionStatement

Depois que a função do Lambda especificada no elemento `fulfillmentActivity` atende a intenção, o Amazon Lex transmite essa declaração ao usuário.

Tipo: objeto Statement

confirmationPrompt

Se definido na intenção, o Amazon Lex solicita que o usuário confirme a intenção antes de atendê-la.

Tipo: objeto Prompt

createdDate

A data em que a intenção foi criada.

Tipo: Timestamp

createVersion

`True` se uma nova versão da intenção tiver sido criada. Se o campo `createVersion` não tiver sido especificado na solicitação, o campo `createVersion` será definido como falso na resposta.

Tipo: booliano

description

Uma descrição da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

dialogCodeHook

Se definido na intenção, o Amazon Lex invoca essa função do Lambda para cada entrada do usuário.

Tipo: objeto CodeHook

followUpPrompt

Se definido na intenção, o Amazon Lex usa esse prompt para solicitar atividades adicionais do usuário depois que a intenção for atendida.

Tipo: objeto FollowUpPrompt

fulfillmentActivity

Se definido na intenção, o Amazon Lex invoca essa função do Lambda para cumprir a intenção depois que o usuário fornece todas as informações exigidas pela intenção.

Tipo: objeto FulfillmentActivity

inputContexts

Uma matriz de objetos `InputContext` que lista os contextos que devem estar ativos para que o Amazon Lex escolha a intenção em uma conversa com o usuário.

Tipo: matriz de InputContext objetos

Membros da Matriz: número mínimo de 0 itens. Número máximo de 5 itens.

kendraConfiguration

Informações de configuração, se houver, necessárias para se conectar a um índice do Amazon Kendra e usar na intenção `AMAZON.KendraSearchIntent`.

Tipo: objeto KendraConfiguration

lastUpdatedDate

A data em que a intenção foi atualizada. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

outputContexts

Uma matriz de objetos `OutputContext` que lista os contextos que a intenção ativa quando a intenção é atendida.

Tipo: matriz de objetos OutputContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

parentIntentSignature

Um identificador exclusivo da intenção integrada na qual essa intenção se baseia.

Tipo: String

rejectionStatement

Quando o usuário responde “não” à pergunta definida em `confirmationPrompt`, o Amazon Lex responde com essa mensagem para confirmar que a intenção foi cancelada.

Tipo: objeto Statement

sampleUtterances

Uma matriz de amostras de declarações que são configuradas para a intenção.

Tipo: matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 1.500 itens.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 200.

slots

Uma matriz de slots de intenção configurados para a intenção.

Tipo: matriz de objetos Slot

Membros da Matriz: número mínimo de 0 itens. Número máximo de 100 itens.

version

A versão da intenção. Para uma nova intenção, a versão é sempre `$LATEST`.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutSlotType

Serviço: Amazon Lex Model Building Service

Cria um tipo de slot personalizado ou substitui um tipo de slot personalizado existente.

Para criar um tipo de slot personalizado, especifique um nome para o tipo de slot e um conjunto de valores de enumeração, que são os valores que um slot desse tipo pode assumir. Para obter mais informações, consulte Amazon Lex: como funciona.

Se você especificar o nome de um tipo de slot existente, os campos na solicitação substituirão os valores existentes na versão `$LATEST` do tipo de slot. O Amazon Lex remove campos que você não fornece na solicitação. Se você não especificar os campos obrigatórios, o Amazon Lex lançará uma exceção. Quando você atualiza a versão `$LATEST` de um tipo de slot, o campo `$LATEST` de qualquer bot que usa a versão `status` do tipo de slot é definido como `NOT_BUILT`.

Essa operação exige permissões para a ação `lex:PutSlotType`.

Sintaxe da Solicitação

```
PUT /slottypes/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "checksum": "string",
   "createVersion": boolean,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do tipo de slot. O nome não é sensível a maiúsculas e minúsculas.

O nome não pode corresponder a um nome de tipo de slot incorporado ou a um nome de tipo de slot incorporado com “AMAZON.” removido. Por exemplo, como há um tipo de slot integrado chamado `AMAZON.DATE`, você não pode criar um tipo de slot personalizado chamado `DATE`.

Para obter uma lista de tipos de slot integrados, consulte Referência do tipo de slot no Alexa Skills Kit.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

checksum

Identifica uma revisão específica da versão `$LATEST`.

Ao criar um novo tipo de slot, deixe o campo `checksum` em branco. Se você especificar uma soma de verificação, obterá uma exceção `BadRequestException`.

Quando quiser atualizar um tipo de slot, defina o campo `checksum` como a soma de verificação da revisão mais recente da versão `$LATEST`. Se você não especificar o campo `checksum` ou se a soma de verificação não corresponder à versão `$LATEST`, você receberá uma exceção `PreconditionFailedException`.

Tipo: String

Obrigatório: Não

createVersion

Quando configurado para `true`, uma nova versão numerada do tipo de slot é criada. Isso é o mesmo que chamar a operação `CreateSlotTypeVersion`. Se você não especificar `createVersion`, o valor padrão será `false`.

Tipo: booliano

Obrigatório: não

description

Uma descrição do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: Não

enumerationValues

Uma lista de objetos `EnumerationValue` que define os valores que o tipo de slot pode ter. Cada valor pode ter uma lista de `synonyms`, que são valores adicionais que ajudam a treinar o modelo de machine learning sobre os valores que ele resolve para o slot.

Um tipo de slot de expressão regular não exige valores de enumeração. Todos os outros tipos de slots exigem uma lista de valores de enumeração.

Quando o Amazon Lex resolve um valor de slot, ele gera uma lista de resolução que contém até cinco valores possíveis para o slot. Se você estiver usando uma função do Lambda, essa lista de resolução será passada para a função. Se você não estiver usando uma função do Lambda, poderá optar por retornar o valor que o usuário inseriu ou o primeiro valor na lista como o valor de slot. O campo `valueSelectionStrategy` indica a opção a ser usada.

Tipo: matriz de objetos EnumerationValue

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10.000 itens.

Obrigatório: Não

parentSlotTypeSignature

O tipo de slot integrado usado como pai do tipo de slot. Quando você define um tipo de slot pai, o novo tipo de slot tem todas as mesmas configurações do pai.

Somente `AMAZON.AlphaNumeric` é suportado.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Obrigatório: Não

slotTypeConfigurations

Informações de configuração que estendem o tipo de slot integrado principal. A configuração é adicionada às configurações do tipo de slot principal.

Tipo: matriz de objetos SlotTypeConfiguration

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

Obrigatório: Não

valueSelectionStrategy

Determina a estratégia de resolução de slots que o Amazon Lex usa para retornar valores do tipo de slot. O campo pode ser definido com os dos seguintes valores:

- `ORIGINAL_VALUE` - Retorna o valor inserido pelo usuário, se o valor do usuário for semelhante ao valor de um slot.
- `TOP_RESOLUTION` - Se houver uma lista de resolução para o slot, retornará o primeiro valor na lista como o valor do tipo de slot. Se não houver uma lista de resolução, retornará o valor “null”.

Se `valueSelectionStrategy` não for especificado, o valor padrão será `ORIGINAL_VALUE`.

Tipo: String

Valores Válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "checksum": "string",
   "createdDate": number,
   "createVersion": boolean,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

checksum

Soma de verificação da versão `$LATEST` do tipo de slot.

Tipo: String

createdDate

A data em que o tipo de slot foi criado.

Tipo: Timestamp

createVersion

`True` se uma nova versão do tipo de slot tiver sido criada. Se o campo `createVersion` não tiver sido especificado na solicitação, o campo `createVersion` será definido como falso na resposta.

Tipo: booliano

description

Uma descrição do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

enumerationValues

Uma lista de objetos `EnumerationValue` que define os valores que o tipo de slot pode ter.

Tipo: matriz de objetos EnumerationValue

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10.000 itens.

lastUpdatedDate

A data em que o tipo de slot foi atualizado. Quando você cria um tipo de slot, a data de criação e a data da última atualização são as mesmas.

Tipo: Timestamp

name

O nome do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

O tipo de slot integrado usado como pai do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informações de configuração que estendem o tipo de slot integrado principal.

Tipo: matriz de objetos SlotTypeConfiguration

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

valueSelectionStrategy

A estratégia de resolução de slot que o Amazon Lex usa para determinar o valor de slot. Para obter mais informações, consulte PutSlotType.

Tipo: String

Valores Válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

A versão do tipo de slot. Para um novo tipo de slot, a versão é sempre `$LATEST`.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

PreconditionFailedException

A soma de verificação do recurso que você está tentando alterar não corresponde à soma de verificação na solicitação. Verifique a soma de verificação e tente novamente.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

StartImport

Serviço: Amazon Lex Model Building Service

Inicia um trabalho para importar um recurso para o Amazon Lex.

Sintaxe da Solicitação

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
   "mergeStrategy": "string",
   "payload": blob,
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parâmetros da solicitação de URI

A solicitação não usa nenhum parâmetro de URI.

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

mergeStrategy

Especifica a ação que a operação `StartImport` deve realizar quando há um recurso existente com o mesmo nome.

- FAIL_ON_CONFLICT - A operação de importação é interrompida no primeiro conflito entre um recurso no arquivo de importação e um recurso existente. O nome do recurso que está causando o conflito está no campo `failureReason` da resposta à operação `GetImport`.

  OVERWRITE_LATEST - A operação de importação prossegue mesmo se houver um conflito com um recurso existente. A versão $LASTEST do recurso existente é substituída pelos dados do arquivo de importação.

Tipo: String

Valores Válidos: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

Obrigatório: Sim

payload

Um arquivo zip em formato binário. O arquivo deve conter um arquivo, um arquivo JSON que contém o recurso a ser importado. O recurso deve corresponder ao tipo especificado no campo `resourceType`.

Tipo: Objeto de dados binários codificado em Base64

Obrigatório: Sim

resourceType

Especifica o tipo de recurso a ser exportado. Cada recurso também exporta todos os recursos dos quais ele depende.

- Um bot exporta intenções dependentes.
- Uma intenção exporta os tipos de slots dependentes.

Tipo: String

Valores Válidos: `BOT | INTENT | SLOT_TYPE`

Obrigatório: Sim

tags

Uma lista de tags a serem adicionadas ao bot importado. Apenas é possível adicionar tags ao importar um bot. Não é possível adicionar tags a uma intenção ou tipo de slot.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 201
Content-type: application/json
```

```
{
   "createdDate": number,
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 201.

Os dados a seguir são retornados no formato JSON pelo serviço.

createdDate

Um carimbo de data e hora para a data e a hora em que a tarefa de importação foi solicitada.

Tipo: Timestamp

importId

O identificador para a tarefa de importação específica.

Tipo: String

importStatus

O status do trabalho de importação. Se o status for `FAILED`, você poderá obter o motivo da falha usando a operação `GetImport`.

Tipo: String

Valores Válidos: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

A ação a ser executada quando houver um conflito de fusão.

Tipo: String

Valores Válidos: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

O nome dado ao trabalho de importação.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `[a-zA-Z_]+`

resourceType

O tipo do recurso a ser importado.

Tipo: String

Valores Válidos: `BOT | INTENT | SLOT_TYPE`

tags

Uma lista de tags a serem adicionadas ao bot importado.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

StartMigration

Serviço: Amazon Lex Model Building Service

Começa a migrar um bot do Amazon Lex V1 para o Amazon Lex V2. Migre seu bot quando quiser aproveitar os novos atributos do Amazon Lex V2.

Para obter mais informações, consulte Migração de um bot no Guia do desenvolvedor do Amazon Lex.

Sintaxe da Solicitação

```
POST /migrations HTTP/1.1
Content-type: application/json

{
   "migrationStrategy": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotName": "string",
   "v2BotRole": "string"
}
```

Parâmetros da solicitação de URI

A solicitação não usa nenhum parâmetro de URI.

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

migrationStrategy

A estratégia usada para conduzir a migração.

- `CREATE_NEW` - Cria um novo bot do Amazon Lex V2 e migra o bot do Amazon Lex V1 para o novo bot.
- `UPDATE_EXISTING` - Substitui os metadados existentes do bot do Amazon Lex V2 e a localidade que está sendo migrada. Isso não altera nenhuma outra localidade no bot do Amazon Lex V2. Se a localidade não existir, uma nova localidade será criada no bot do Amazon Lex V2.

Tipo: String

Valores Válidos: `CREATE_NEW | UPDATE_EXISTING`

Obrigatório: Sim

v1BotName

O nome do bot do Amazon Lex V1 que você está migrando para o Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

v1BotVersion

A versão do bot a ser migrado para o Amazon Lex V2. Você pode migrar a versão `$LATEST`, bem como qualquer versão numerada.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Exigido: Sim

v2BotName

O nome do bot do Amazon Lex V2 para o qual você está migrando o bot do Amazon Lex V1.

- Se o bot do Amazon Lex V2 não existir, você deverá usar a estratégia de migração `CREATE_NEW`.
- Se o bot do Amazon Lex V2 existir, você deverá usar a estratégia de migração `UPDATE_EXISTING` para alterar o conteúdo do bot do Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([0-9a-zA-Z][_-]?)+$`

Exigido: Sim

v2BotRole

O perfil do IAM que o Amazon Lex usa para executar a versão do bot do Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 202
Content-type: application/json

{
   "migrationId": "string",
   "migrationStrategy": "string",
   "migrationTimestamp": number,
   "v1BotLocale": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotId": "string",
   "v2BotRole": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 202.

Os dados a seguir são retornados no formato JSON pelo serviço.

migrationId

O identificador exclusivo que o Amazon Lex atribuiu à migração.

Tipo: String

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

migrationStrategy

A estratégia usada para conduzir a migração.

Tipo: String

Valores Válidos: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

A data e hora em que a migração foi iniciada.

Tipo: Timestamp

v1BotLocale

A localidade usada para o bot do Amazon Lex V1.

Tipo: String

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

v1BotName

O nome do bot do Amazon Lex V1 que você está migrando para o Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

v1BotVersion

A versão do bot a ser migrado para o Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

v2BotId

O identificador exclusivo do bot do Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

v2BotRole

O perfil do IAM que o Amazon Lex usa para executar a versão do bot do Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Erros

AccessDeniedException

Seu usuário ou função do IAM não tem permissão para chamar o Amazon Lex V2 APIs necessário para migrar seu bot.

Código de status HTTP: 403

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

TagResource

Serviço: Amazon Lex Model Building Service

Adiciona as tags especificadas ao recurso especificado. Se uma chave de tag já existir, o valor existente será substituído pelo novo valor.

Sintaxe da Solicitação

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

resourceArn

O nome do recurso da Amazon (ARN) do bot, do alias do bot ou do canal do bot a ser marcado.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1011.

Obrigatório: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

tags

A lista de chaves de tag a serem adicionadas a um recurso. Se uma chave de tag já existir, o valor existente será substituído pelo novo valor.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UntagResource

Serviço: Amazon Lex Model Building Service

Remove as tags de um bot, alias de bot ou canal de bot.

Sintaxe da Solicitação

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

resourceArn

O nome do recurso da Amazon (ARN) do recurso do qual remover as tags.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1011.

Obrigatório: Sim

tagKeys

Uma lista de chaves de tag a serem removidas do recurso. Se uma chave de tag não existir no recurso, ela será ignorada.

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 204
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 204 com um corpo HTTP vazio.

Erros

BadRequestException

A solicitação não está bem formada. Por exemplo, um valor é inválido ou um campo obrigatório está faltando. Verifique os valores dos campos e tente novamente.

Código de status HTTP: 400

ConflictException

Houve um conflito ao processar a solicitação. Tente sua solicitação novamente.

Código de Status HTTP: 409

InternalFailureException

Ocorreu um erro interno do Amazon Lex. Tente sua solicitação novamente.

Código de status HTTP: 500

LimitExceededException

A solicitação excedeu um limite. Tente sua solicitação novamente.

Código de status HTTP: 429

NotFoundException

O recurso especificado na solicitação não foi encontrado. Verifique o recurso e tente novamente.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2

- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

Serviço de runtime do Amazon Lex

As seguintes ações são compatíveis com o Serviço de runtime do Amazon Lex:

- DeleteSession
- GetSession
- PostContent
- PostText
- PutSession

DeleteSession

Serviço: Amazon Lex Runtime Service

Remove informações de sessão para um bot, alias e ID de usuário especificado.

Sintaxe da Solicitação

```
DELETE /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botAlias

O alias em uso para o bot que contém os dados da sessão.

Obrigatório: Sim

botName

O nome do bot que contém os dados da sessão.

Obrigatório: Sim

userId

O identificador do usuário associado aos dados da sessão.

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Padrão: `[0-9a-zA-Z._:-]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "botAlias": "string",
    "botName": "string",
    "sessionId": "string",
    "userId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

botAlias

O alias em uso para o bot associado aos dados da sessão.

Tipo: String

botName

O nome do bot associado aos dados de sessão.

Tipo: String

sessionId

O identificador exclusivo da sessão.

Tipo: String

userId

O ID do usuário do aplicativo cliente.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Padrão: `[0-9a-zA-Z._:-]+`

Erros

BadRequestException

A validação da solicitação falhou, não há mensagem utilizável no contexto ou a compilação do bot falhou, ainda está em andamento ou contém alterações não criadas.

Código de status HTTP: 400

ConflictException

Dois clientes estão usando a mesma conta da AWS, o bot do Amazon Lex e o mesmo ID de usuário.

Código de Status HTTP: 409

InternalFailureException

Erro de serviço interno. Tente a chamada novamente.

Código de status HTTP: 500

LimitExceededException

Excedeu um limite.

Código de status HTTP: 429

NotFoundException

O recurso (como o bot Amazon Lex ou um alias) mencionado não foi encontrado.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetSession

Serviço: Amazon Lex Runtime Service

Retorna informações de sessão para um bot, alias e ID de usuário especificado.

Sintaxe da Solicitação

```
GET /bot/botName/alias/botAlias/user/userId/session/?
checkpointLabelFilter=checkpointLabelFilter HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botAlias

O alias em uso para o bot que contém os dados da sessão.

Obrigatório: Sim

botName

O nome do bot que contém os dados da sessão.

Obrigatório: Sim

checkpointLabelFilter

Uma string usada para filtrar as intenções retornadas na estrutura `recentIntentSummaryView`.

Quando você especifica um filtro, somente as intenções com o campo `checkpointLabel` definido para essa string são retornadas.

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 255.

Padrão: `[a-zA-Z0-9-]+`

userId

O ID do usuário do aplicativo cliente. O Amazon Lex usa isso para identificar a conversa de um usuário com seu bot.

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Padrão: `[0-9a-zA-Z._:-]+`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "dialogAction": {
      "fulfillmentState": "string",
      "intentName": "string",
      "message": "string",
      "messageFormat": "string",
      "slots": {
         "string" : "string"
      },
      "slotToElicit": "string",
      "type": "string"
   },
   "recentIntentSummaryView": [
      {
         "checkpointLabel": "string",
         "confirmationStatus": "string",
         "dialogActionType": "string",
         "fulfillmentState": "string",
```

```
            "intentName": "string",
            "slots": {
               "string" : "string"
            },
            "slotToElicit": "string"
         }
      ],
      "sessionAttributes": {
         "string" : "string"
      },
      "sessionId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

activeContexts

Uma lista de contextos ativos para a sessão. Um contexto pode ser definido quando uma intenção é cumprida ou chamando a operação `PostContent`, `PostText` ou `PutSession`.

Você pode usar um contexto para controlar as intenções que podem acompanhar uma intenção ou para modificar a operação do seu aplicativo.

Tipo: matriz de objetos ActiveContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 20 itens.

dialogAction

Descreve o estado atual do bot.

Tipo: objeto DialogAction

recentIntentSummaryView

Uma matriz de informações sobre as intenções usadas na sessão. A matriz pode conter no máximo três resumos. Se mais de três intenções forem usadas na sessão, a operação `recentIntentSummaryView` conterá informações sobre as últimas três intenções usadas.

Se você definir o parâmetro `checkpointLabelFilter` na solicitação, a matriz conterá somente as intenções com o rótulo especificado.

Tipo: matriz de objetos IntentSummary

Membros da Matriz: número mínimo de 0 itens. Número máximo de 3 itens.

sessionAttributes

Mapa de pares de chaves/valores que representam as informações de contexto específicas da sessão. Ele contém informações do aplicativo passadas entre o Amazon Lex e um aplicativo cliente.

Tipo: mapa de string para string

sessionId

Um identificador exclusivo da sessão.

Tipo: String

Erros

BadRequestException

A validação da solicitação falhou, não há mensagem utilizável no contexto ou a compilação do bot falhou, ainda está em andamento ou contém alterações não criadas.

Código de status HTTP: 400

InternalFailureException

Erro de serviço interno. Tente a chamada novamente.

Código de status HTTP: 500

LimitExceededException

Excedeu um limite.

Código de status HTTP: 429

NotFoundException

O recurso (como o bot Amazon Lex ou um alias) mencionado não foi encontrado.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PostContent

Serviço: Amazon Lex Runtime Service

Envia a entrada do usuário (texto ou fala) ao Amazon Lex. Os clientes usam essa API para enviar solicitações de texto e áudio para o Amazon Lex em runtime. O Amazon Lex interpreta a entrada do usuário usando o modelo de machine learning criado para o bot.

A operação `PostContent` oferece suporte a entrada de áudio em 8 kHz e 16 kHz. Você pode usar áudio de 8 kHz para obter maior precisão de reconhecimento de fala em aplicativos de áudio de telefone.

Em resposta, o Amazon Lex retorna a próxima mensagem para transmitir ao usuário. Considere as seguintes mensagens de exemplo:

- Para uma entrada do usuário “Eu gostaria de uma pizza”, o Amazon Lex pode retornar uma resposta com uma mensagem gerando dados do slot (por exemplo, `PizzaSize`): “Qual tamanho de pizza você gostaria?”.
- Depois que o usuário fornece todas as informações do pedido de pizza, o Amazon Lex pode retornar uma resposta com uma mensagem para obter a confirmação do usuário: “Pedir a pizza?”.
- Depois que o usuário responder “Sim” ao prompt de confirmação, o Amazon Lex poderá retornar uma declaração de conclusão: “Obrigado, sua pizza de queijo foi pedida.”.

Nem todas as mensagens do Amazon Lex exigem uma resposta do usuário. Por exemplo, declarações de conclusão não exigem uma resposta. Algumas mensagens exigem apenas uma resposta sim ou não. Além do `message`, o Amazon Lex fornece contexto adicional sobre a mensagem na resposta que você pode usar para aprimorar o comportamento do cliente, como exibir a interface de usuário apropriada do cliente. Considere os seguintes exemplos:

- Se a mensagem for para obter dados de slots, o Amazon Lex retornará as seguintes informações de contexto:
  - cabeçalho `x-amz-lex-dialog-state` definido como `ElicitSlot`
  - cabeçalho `x-amz-lex-intent-name` definido com o nome da intenção no contexto atual
  - cabeçalho `x-amz-lex-slot-to-elicit` definido com o nome do slot para o qual `message` está obtendo informações
  - cabeçalho `x-amz-lex-slots` definido como um mapa de slots configurados para a intenção com seus valores atuais

- Se a mensagem for um prompt de confirmação, o cabeçalho `x-amz-lex-dialog-state` será definido como `Confirmation` e o cabeçalho `x-amz-lex-slot-to-elicit` será omitido.
- Se a mensagem for um prompt de esclarecimento configurado para a intenção, indicando que a intenção do usuário não foi compreendida, o cabeçalho `x-amz-dialog-state` será definido como `ElicitIntent` e o cabeçalho `x-amz-slot-to-elicit` será omitido.

Além disso, o Amazon Lex também retorna seu aplicativo específico`sessionAttributes`. Para obter mais informações, consulte Gerenciar o contexto de conversação.

Sintaxe da Solicitação

```
POST /bot/botName/alias/botAlias/user/userId/content HTTP/1.1
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-request-attributes: requestAttributes
Content-Type: contentType
Accept: accept
x-amz-lex-active-contexts: activeContexts

inputStream
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

accept

Você passa este valor como o cabeçalho HTTP `Accept`.

A mensagem que o Amazon Lex retorna na resposta pode ser texto ou fala com base no valor do cabeçalho HTTP `Accept` na solicitação.

- Se o valor for `text/plain; charset=utf-8`, o Amazon Lex retornará o texto na resposta.
- Se o valor começar com `audio/`, o Amazon Lex retornará a fala na resposta. O Amazon Lex usa o Amazon Polly para gerar a fala (usando a configuração especificada no cabeçalho `Accept`). Por exemplo, se você especificar `audio/mpeg` como valor, o Amazon Lex retornará a fala no formato MPEG.
- Se o valor for `audio/pcm`, a fala retornada está `audio/pcm` no formato little-endian de 16 bits.
- Os valores aceitos são os seguintes:

- audio/mpeg
- audio/ogg
- audio/pcm
- text/plain; charset=utf-8
- audio/* (o padrão é mpeg)

activeContexts

Uma lista de contextos ativos para a solicitação. Um contexto pode ser ativado quando uma intenção anterior é atendida ou incluindo o contexto na solicitação,

Se você não especificar uma lista de contextos, o Amazon Lex usará a lista atual de contextos para a sessão. Se você especificar uma lista vazia, todos os contextos da sessão serão apagados.

botAlias

Alias do bot Amazon Lex.

Obrigatório: Sim

botName

Nome do bot do Amazon Lex.

Obrigatório: Sim

contentType

Você passa este valor como o cabeçalho HTTP `Content-Type`.

Indica o formato de áudio ou texto. O valor do cabeçalho deve começar com um dos seguintes prefixos:

- No formato PCM, os dados de áudio devem estar na ordem de bytes little-endian.
  - audio/l16; rate=16000; channels=1
  - audio/x-l16; sample-rate=16000; channel-count=1
  - áudio/lpcm; taxa de amostragem = 8000; = 16; contagem de canais = 1; = falso sample-size-bits is-big-endian
- Formato Opus

  - áudio/ x-cbr-opus-with -preâmbulo; tamanho do preâmbulo = 0; taxa de bits = 256000; =4 frame-size-milliseconds
- Formato de texto
  - text/plain; charset=utf-8

Obrigatório: Sim

requestAttributes

Você passa este valor como o cabeçalho HTTP `x-amz-lex-request-attributes`.

Informações específicas da solicitação passadas entre o Amazon Lex e um aplicativo cliente. O valor deve ser um mapa serializado JSON e codificado em base64 com chaves e valores de string. O tamanho total dos cabeçalhos `requestAttributes` e `sessionAttributes` está limitado a 12 KB.

O namespace `x-amz-lex:` é reservado para atributos especiais. Não crie atributos de solicitação com o prefixo `x-amz-lex:`.

Para obter mais informações, consulte Definição de atributos de solicitação.

sessionAttributes

Você passa este valor como o cabeçalho HTTP `x-amz-lex-session-attributes`.

Informações específicas do aplicativo passadas entre o Amazon Lex e um aplicativo cliente. O valor deve ser um mapa serializado JSON e codificado em base64 com chaves e valores de string. O tamanho total dos cabeçalhos `sessionAttributes` e `requestAttributes` está limitado a 12 KB.

Para obter mais informações, consulte Definição de atributos de sessão.

userId

O ID do usuário do aplicativo cliente. O Amazon Lex usa isso para identificar a conversa de um usuário com seu bot. No runtime, cada solicitação deve conter o campo `userID`.

Para decidir o ID de usuário a ser usado em seu aplicativo, considere os seguintes fatores.

- O campo `userID` não deve conter nenhuma informação de identificação pessoal do usuário, por exemplo, nome, números de identificação pessoal ou outras informações pessoais do usuário final.

- Se você quiser que um usuário inicie uma conversa em um dispositivo e continue em outro, use um identificador específico do usuário.
- Se você quiser que o mesmo usuário possa ter duas conversas independentes em dois dispositivos diferentes, escolha um identificador específico do dispositivo.
- Um usuário não pode ter duas conversas independentes com duas versões diferentes do mesmo bot. Por exemplo, um usuário não pode conversar com as versões PROD e BETA do mesmo bot. Se você prevê que um usuário precisará conversar com duas versões diferentes, por exemplo, durante o teste, inclua o alias do bot no ID do usuário para separar as duas conversas.

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Padrão: `[0-9a-zA-Z._:-]+`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados binários a seguir.

inputStream

Entrada do usuário no formato de áudio PCM ou Opus ou formato de texto, conforme descrito no cabeçalho HTTP `Content-Type`.

Você pode transmitir dados de áudio para o Amazon Lex ou criar um buffer local que capture todos os dados de áudio antes do envio. Em geral, você obtém melhor desempenho se transmitir dados de áudio em vez de armazenar os dados em buffer localmente.

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 200
Content-Type: contentType
x-amz-lex-intent-name: intentName
x-amz-lex-nlu-intent-confidence: nluIntentConfidence
x-amz-lex-alternative-intents: alternativeIntents
x-amz-lex-slots: slots
```

```
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-sentiment: sentimentResponse
x-amz-lex-message: message
x-amz-lex-encoded-message: encodedMessage
x-amz-lex-message-format: messageFormat
x-amz-lex-dialog-state: dialogState
x-amz-lex-slot-to-elicit: slotToElicit
x-amz-lex-input-transcript: inputTranscript
x-amz-lex-encoded-input-transcript: encodedInputTranscript
x-amz-lex-bot-version: botVersion
x-amz-lex-session-id: sessionId
x-amz-lex-active-contexts: activeContexts

audioStream
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

A resposta retorna os cabeçalhos HTTP a seguir.

activeContexts

Uma lista de contextos ativos para a sessão. Um contexto pode ser definido quando uma intenção é cumprida ou chamando a operação `PostContent`, `PostText` ou `PutSession`.

Você pode usar um contexto para controlar as intenções que podem acompanhar uma intenção ou para modificar a operação do seu aplicativo.

alternativeIntents

Uma a quatro intenções alternativas que podem ser aplicáveis à intenção do usuário.

Cada alternativa inclui uma pontuação que indica o grau de confiança do Amazon Lex de que a intenção corresponde à intenção do usuário. As intenções são classificadas pela pontuação de confiança.

botVersion

A versão do bot que respondeu à conversa. Você pode usar essas informações para ajudar a determinar se uma versão de um bot tem um desempenho melhor do que outra versão.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[0-9]+|\$LATEST`

contentType

Tipo de conteúdo conforme especificado no cabeçalho HTTP `Accept` na solicitação.

dialogState

Identifica o estado atual da interação do usuário. O Amazon Lex retorna um dos seguintes valores como `dialogState`. Opcionalmente, o cliente pode usar essas informações para personalizar a interface do usuário.

- `ElicitIntent` - O Amazon Lex quer obter a intenção do usuário. Considere os seguintes exemplos:

  Por exemplo, um usuário pode expressar uma intenção (“Quero pedir uma pizza”). Se o Amazon Lex não puder deduzir a intenção do usuário a partir dessa declaração, ele retornará esse estado de diálogo.
- `ConfirmIntent` - O Amazon Lex espera uma resposta “sim” ou “não”.

  Por exemplo, o Amazon Lex quer a confirmação do usuário antes de atender uma intenção. Em vez de uma simples resposta de “sim” ou “não”, um usuário pode responder com informações adicionais. Por exemplo, “sim, mas peça uma pizza de massa grossa” ou “não, quero pedir uma bebida”. O Amazon Lex pode processar essas informações adicionais (nesses exemplos, atualizar o slot do tipo de crosta ou alterar a intenção de OrderPizza para OrderDrink).
- `ElicitSlot` - O Amazon Lex espera o valor de um slot para a intenção atual.

  Por exemplo, suponha que, na resposta, o Amazon Lex envie esta mensagem: “Qual tamanho de pizza você gostaria?”. Um usuário pode responder com o valor de slot (por exemplo, “média”). O usuário também pode fornecer informações adicionais na resposta (por exemplo, “pizza média de massa grossa”). O Amazon Lex pode processar essas informações adicionais de forma adequada.
- `Fulfilled` - Transmite que a função do Lambda atendeu com sucesso a intenção.
- `ReadyForFulfillment` - Transmite que o cliente deve atender à solicitação.
- `Failed` - Transmite que a conversa com o usuário falhou.

  Isso pode acontecer por vários motivos, incluindo o fato de o usuário não fornecer uma resposta adequada aos prompts do serviço (você pode configurar quantas vezes o Amazon Lex pode solicitar informações específicas a um usuário) ou se a função do Lambda não atender à intenção.

Valores Válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedInputTranscript

O texto usado para processar a solicitação.

Se a entrada tiver sido um fluxo de áudio, o campo `encodedInputTranscript` conterá o texto extraído do fluxo de áudio. Esse é o texto que é processado para reconhecer as intenções e os valores de slot. Você pode usar essas informações para determinar se o Amazon Lex está processando corretamente o áudio que você envia.

O campo `encodedInputTranscript` é codificado em base 64. Você deve decodificar o campo antes de poder usar o valor.

encodedMessage

A mensagem a ser transmitida ao usuário. A mensagem pode vir da configuração do bot ou de uma função do Lambda.

Se a intenção não estiver configurada com uma função do Lambda, ou se a função do Lambda tiver retornado `Delegate` como `dialogAction.type` em sua resposta, o Amazon Lex decide o próximo curso de ação e seleciona uma mensagem apropriada da configuração do bot com base no contexto de interação atual. Por exemplo, se o Amazon Lex não conseguir entender a entrada do usuário, ele usa uma mensagem de prompt de esclarecimento.

Ao criar uma intenção, você pode atribuir mensagens a grupos. Quando as mensagens são atribuídas a grupos, o Amazon Lex retorna uma mensagem de cada grupo na resposta. O campo de mensagem é uma string JSON de escape que contém as mensagens. Para obter mais informações sobre a estrutura da string JSON retornada, consulte Formatos de mensagem suportados.

Se a função do Lambda retornar uma mensagem, o Amazon Lex a enviará para o cliente em sua resposta.

O campo `encodedMessage` é codificado em base 64. Você deve decodificar o campo antes de poder usar o valor.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.366.

inputTranscript

Esse cabeçalho foi descontinuado.

O texto usado para processar a solicitação.

Você pode usar esse campo somente nas localidades de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. Em todas as outras localidades, o campo `inputTranscript` é nulo. Em vez disso, use o campo `encodedInputTranscript`.

Se a entrada tiver sido um fluxo de áudio, o campo `inputTranscript` conterá o texto extraído do fluxo de áudio. Esse é o texto que é processado para reconhecer as intenções e os valores de slot. Você pode usar essas informações para determinar se o Amazon Lex está processando corretamente o áudio que você envia.

intentName

Intenção atual do usuário que o Amazon Lex conhece.

message

Esse cabeçalho foi descontinuado.

Você só pode usar esse campo nas localidades de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. Em todas as outras localidades, o campo `message` é nulo. Em vez disso, use o campo `encodedMessage`.

A mensagem a ser transmitida ao usuário. A mensagem pode vir da configuração do bot ou de uma função do Lambda.

Se a intenção não estiver configurada com uma função do Lambda, ou se a função do Lambda tiver retornado `Delegate` como `dialogAction.type` em sua resposta, o Amazon Lex decide o próximo curso de ação e seleciona uma mensagem apropriada da configuração do bot com base no contexto de interação atual. Por exemplo, se o Amazon Lex não conseguir entender a entrada do usuário, ele usa uma mensagem de prompt de esclarecimento.

Ao criar uma intenção, você pode atribuir mensagens a grupos. Quando as mensagens são atribuídas a grupos, o Amazon Lex retorna uma mensagem de cada grupo na resposta. O campo de mensagem é uma string JSON de escape que contém as mensagens. Para obter mais informações sobre a estrutura da string JSON retornada, consulte Formatos de mensagem suportados.

Se a função do Lambda retornar uma mensagem, o Amazon Lex a enviará para o cliente em sua resposta.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.024.

messageFormat

O formato da mensagem de resposta. Um dos seguintes valores:

- `PlainText` - A mensagem contém texto sem formatação UTF-8.
- `CustomPayload` - A mensagem é um formato personalizado para o cliente.
- `SSML` - A mensagem contém texto formatado para saída de voz.
- `Composite` - A mensagem contém um objeto JSON de escape que contém uma ou mais mensagens dos grupos aos quais as mensagens foram atribuídas quando a intenção foi criada.

Valores Válidos: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Fornece uma pontuação que indica o quanto o Amazon Lex tem certeza de que a intenção retornada é aquela que corresponde à intenção do usuário. A pontuação está entre 0,0 e 1,0.

A pontuação é relativa, não absoluta. A pontuação pode mudar com base nas melhorias no Amazon Lex.

sentimentResponse

O sentimento expresso em uma declaração.

Quando o bot está configurado para enviar declarações ao Amazon Comprehend para análise de sentimentos, esse campo contém o resultado da análise.

sessionAttributes

Mapa de pares de chaves/valores que representam as informações de contexto específicas da sessão.

sessionId

O identificador exclusivo da sessão.

slots

Mapa de zero ou mais slots de intenção (pares de chaves/valores) do Amazon Lex detectados na entrada do usuário durante a conversação. O campo é codificado em base 64.

O Amazon Lex cria uma lista de resolução que contém valores prováveis para um slot. O valor que ele retorna é determinado pelo `valueSelectionStrategy` selecionado quando o tipo de slot foi criado ou atualizado. Se `valueSelectionStrategy` for definido como

`ORIGINAL_VALUE`, o valor fornecido pelo usuário será retornado, se o valor do usuário for semelhante ao valor de slot. Se `valueSelectionStrategy` estiver definido como `TOP_RESOLUTION`, o Amazon Lex retornará o primeiro valor na lista de resolução ou, se não houver lista de resolução, nulo. Se `valueSelectionStrategy` não for especificado, o padrão será `ORIGINAL_VALUE`.

slotToElicit

Se o valor `dialogState` for `ElicitSlot`, retornará o nome do slot para o qual o Amazon Lex está obtendo um valor.

A resposta retorna as informações a seguir como corpo HTTP.

audioStream

O prompt (ou declaração) a ser transmitido ao usuário. Isso se baseia na configuração e no contexto do bot. Por exemplo, se o Amazon Lex não entender a intenção do usuário, ele envia o `clarificationPrompt` configurado para o bot. Se a intenção exigir confirmação antes de realizar a ação de atendimento, ela envia o `confirmationPrompt`. Outro exemplo: suponha que a função do Lambda tenha atendido com sucesso a intenção e enviado uma mensagem para transmitir ao usuário. Em seguida, o Amazon Lex envia essa mensagem na resposta.

Erros

BadGatewayException

Ou o bot do Amazon Lex ainda está sendo construído ou um dos serviços dependentes (Amazon Polly, AWS Lambda) falhou com um erro interno de serviço.

Código de status HTTP: 502

BadRequestException

A validação da solicitação falhou, não há mensagem utilizável no contexto ou a compilação do bot falhou, ainda está em andamento ou contém alterações não criadas.

Código de status HTTP: 400

ConflictException

Dois clientes estão usando a mesma conta da AWS, o bot do Amazon Lex e o mesmo ID de usuário.

Código de Status HTTP: 409

DependencyFailedException

Uma das dependências, como AWS Lambda ou Amazon Polly, gerou uma exceção. Por exemplo,

- Se o Amazon Lex não tiver permissões suficientes para chamar uma função do Lambda.
- Se uma função do Lambda levar mais de 30 segundos para ser executada.
- Se uma função do Lambda de atendimento retornar uma ação `Delegate` de diálogo sem remover nenhum valor de slot.

Código de status HTTP: 424

InternalFailureException

Erro de serviço interno. Tente a chamada novamente.

Código de status HTTP: 500

LimitExceededException

Excedeu um limite.

Código de status HTTP: 429

LoopDetectedException

Essa exceção não é usada.

Código de status HTTP: 508

NotAcceptableException

O cabeçalho de aceitação na solicitação não tem um valor válido.

Código de status HTTP: 406

NotFoundException

O recurso (como o bot Amazon Lex ou um alias) mencionado não foi encontrado.

Código de status HTTP: 404

RequestTimeoutException

A fala de entrada é muito longa.

Código de status HTTP: 408

UnsupportedMediaTypeException

O cabeçalho Content-Type (API `PostContent`) tem um valor inválido.

Código de status HTTP: 415

Exemplos

Exemplo 1

Nessa solicitação, o URI identifica um bot (Tráfego), uma versão do bot ($LATEST) e um nome de usuário final (someuser). O cabeçalho `Content-Type` identifica o formato do áudio no corpo. O Amazon Lex também oferece suporte a outros formatos. Para converter áudio de um formato para outro, se necessário, você pode usar o software de código aberto SoX. Você especifica o formato no qual deseja obter a resposta adicionando o cabeçalho HTTP `Accept`.

Na resposta, o cabeçalho `x-amz-lex-message` mostra a resposta que o Amazon Lex retornou. O cliente pode então enviar essa resposta ao usuário. A mesma mensagem é enviada no formato áudio/MPEG por meio de codificação em partes (conforme solicitado).

Exemplo de solicitação

```
"POST /bot/Traffic/alias/$LATEST/user/someuser/content HTTP/1.1[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
"Content-Type: audio/x-l16; channel-count=1; sample-rate=16000f[\r][\n]"
"Accept: audio/mpeg[\r][\n]"
"Host: runtime.lex.us-east-1.amazonaws.com[\r][\n]"
"Authorization: AWS4-HMAC-SHA256 Credential=BLANKED_OUT/20161230/us-east-1/lex/
aws4_request,
SignedHeaders=accept;content-type;host;x-amz-content-sha256;x-amz-date;x-amz-lex-
session-attributes,
 Signature=78ca5b54ea3f64a17ff7522de02cd90a9acd2365b45a9ce9b96ea105bb1c7ec2[\r][\n]"
"X-Amz-Date: 20161230T181426Z[\r][\n]"
"X-Amz-Content-Sha256:
 e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855[\r][\n]"
"Transfer-Encoding: chunked[\r][\n]"
"Connection: Keep-Alive[\r][\n]"
"User-Agent: Apache-HttpClient/4.5.x (Java/1.8.0_112)[\r][\n]"
"Accept-Encoding: gzip,deflate[\r][\n]"
"[\r][\n]"
"1000[\r][\n]"
"[0x7][0x0][0x7][0x0][\n]"
```

```
"[0x0][0x7][0x0][0xfc][0xff][\n]"
"[0x0][\n]"
…
```

Resposta da amostra

```
"HTTP/1.1 200 OK[\r][\n]"
"x-amzn-RequestId: cc8b34af-cebb-11e6-a35c-55f3a992f28d[\r][\n]"
"x-amz-lex-message: Sorry, can you repeat that?[\r][\n]"
"x-amz-lex-dialog-state: ElicitIntent[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
"Content-Type: audio/mpeg[\r][\n]"
"Transfer-Encoding: chunked[\r][\n]"
"Date: Fri, 30 Dec 2016 18:14:28 GMT[\r][\n]"
"[\r][\n]"
"2000[\r][\n]"
"ID3[0x4][0x0][0x0][0x0][0x0][0x0]#TSSE[0x0][0x0][0x0][0xf][0x0][0x0]
[0x3]Lavf57.41.100[0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0xff]
[0xf3]`[0xc4][0x0][0x1b]{[0x8d][0xe8][0x1]C[0x18][0x1][0x0]J[0xe0]`b[0xdd][0xd1]
[0xb][0xfd][0x11][0xdf][0xfe]";[0xbb][0xbb][0x9f][0xee][0xee][0xee][0xee]|DDD/[0xff]
[0xff][0xff][0xff]www?D[0xf7]w^?[0xff][0xfa]h[0x88][0x85][0xfe][0x88][0x88][0x88]
[[0xa2]'[0xff][0xfa]"{[0x9f][0xe8][0x88]]D[0xeb][0xbb][0xbb][0xa2]!u[0xfd][0xdd][0xdf]
[0x88][0x94][0x0]F[0xef][0xa1]8[0x0][0x82]w[0x88]N[0x0][0x0][0x9b][0xbb][0xe8][0xe
…
```

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3

- AWS SDK para Python
- AWS SDK para Ruby V3

PostText

Serviço: Amazon Lex Runtime Service

Envia a entrada do usuário ao Amazon Lex. Os aplicativos clientes podem usar essa API para enviar solicitações ao Amazon Lex em runtime. Em seguida, o Amazon Lex interpreta a entrada do usuário usando o modelo de machine learning criado para o bot.

Em resposta, o Amazon Lex retorna a próxima `message` para transmitir um `responseCard` opcional ao usuário para exibir. Considere as seguintes mensagens de exemplo:

- Para uma entrada do usuário “Eu gostaria de uma pizza”, o Amazon Lex pode retornar uma resposta com uma mensagem gerando dados do slot (por exemplo, PizzaSize): “Qual tamanho de pizza você gostaria?”
- Depois que o usuário fornece todas as informações do pedido de pizza, o Amazon Lex pode retornar uma resposta com uma mensagem para obter a confirmação do usuário: “Continuar com o pedido de pizza?”.
- Depois que o usuário responder o prompt de confirmação com “Sim”, o Amazon Lex poderá retornar uma declaração de conclusão: “Obrigado, sua pizza de queijo foi pedida.”.

Nem todas as mensagens do Amazon Lex exigem uma resposta do usuário. Por exemplo, uma declaração de conclusão não exige uma resposta. Algumas mensagens exigem apenas uma resposta “sim” ou “não” do usuário. Além do `message`, o Amazon Lex fornece contexto adicional sobre a mensagem na resposta que você pode usar para aprimorar o comportamento do cliente, por exemplo, para exibir a interface de usuário apropriada do cliente. Esses são os campos `slotToElicit`, `dialogState`, `intentName` e `slots` na resposta. Considere os seguintes exemplos:

- Se a mensagem for para obter dados de slots, o Amazon Lex retornará as seguintes informações de contexto:
  - `dialogState`definido como ElicitSlot
  - `intentName` definido como o nome da intenção no contexto atual
  - `slotToElicit` definido como o nome do slot para o qual `message` está obtendo informações
  - `slots` definido como um mapa de slots configurados para a intenção com valores atuais conhecidos
- Se a mensagem for um prompt de confirmação, o `dialogState` será definido como ConfirmIntent e `SlotToElicit` definido como nulo.

- Se a mensagem for um prompt de esclarecimento (configurado para a intenção) que indica que a intenção do usuário não foi compreendida, a `dialogState` é definida como ElicitIntent e definida como `slotToElicit` nula.

Além disso, o Amazon Lex também retorna seu aplicativo específico`sessionAttributes`. Para obter mais informações, consulte Gerenciar o contexto de conversação.

Sintaxe da Solicitação

```
POST /bot/botName/alias/botAlias/user/userId/text HTTP/1.1
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "inputText": "string",
   "requestAttributes": {
      "string" : "string"
   },
   "sessionAttributes": {
      "string" : "string"
   }
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

botAlias

O alias do bot do Amazon Lex.

Obrigatório: Sim

botName

O nome do bot do Amazon Lex.

Obrigatório: Sim

userId

O ID do usuário do aplicativo cliente. O Amazon Lex usa isso para identificar a conversa de um usuário com seu bot. No runtime, cada solicitação deve conter o campo `userID`.

Para decidir o ID de usuário a ser usado em seu aplicativo, considere os seguintes fatores.

- O campo `userID` não deve conter nenhuma informação de identificação pessoal do usuário, por exemplo, nome, números de identificação pessoal ou outras informações pessoais do usuário final.
- Se você quiser que um usuário inicie uma conversa em um dispositivo e continue em outro, use um identificador específico do usuário.
- Se você quiser que o mesmo usuário possa ter duas conversas independentes em dois dispositivos diferentes, escolha um identificador específico do dispositivo.
- Um usuário não pode ter duas conversas independentes com duas versões diferentes do mesmo bot. Por exemplo, um usuário não pode conversar com as versões PROD e BETA do mesmo bot. Se você prevê que um usuário precisará conversar com duas versões diferentes, por exemplo, durante o teste, inclua o alias do bot no ID do usuário para separar as duas conversas.

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Padrão: `[0-9a-zA-Z._:-]+`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

activeContexts

Uma lista de contextos ativos para a solicitação. Um contexto pode ser ativado quando uma intenção anterior é atendida ou incluindo o contexto na solicitação,

Se você não especificar uma lista de contextos, o Amazon Lex usará a lista atual de contextos para a sessão. Se você especificar uma lista vazia, todos os contextos da sessão serão apagados.

Tipo: matriz de objetos ActiveContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 20 itens.

Obrigatório: Não

inputText

O texto que o usuário inseriu (o Amazon Lex interpreta esse texto).

Ao usar a AWS CLI, você não pode passar uma URL no parâmetro `--input-text`. Em vez disso, passe o URL usando o parâmetro `--cli-input-json`.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.024.

Obrigatório: Sim

requestAttributes

Informações específicas da solicitação passadas entre o Amazon Lex e um aplicativo cliente.

O namespace `x-amz-lex:` é reservado para atributos especiais. Não crie atributos de solicitação com o prefixo `x-amz-lex:`.

Para obter mais informações, consulte Definição de atributos de solicitação.

Tipo: mapa de string para string

Obrigatório: Não

sessionAttributes

Informações específicas do aplicativo passadas entre o Amazon Lex e um aplicativo cliente.

Para obter mais informações, consulte Definição de atributos de sessão.

Tipo: mapa de string para string

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "alternativeIntents": [
      {
         "intentName": "string",
         "nluIntentConfidence": {
            "score": number
         },
         "slots": {
            "string" : "string"
         }
      }
   ],
   "botVersion": "string",
   "dialogState": "string",
   "intentName": "string",
   "message": "string",
   "messageFormat": "string",
   "nluIntentConfidence": {
      "score": number
   },
   "responseCard": {
      "contentType": "string",
      "genericAttachments": [
         {
            "attachmentLinkUrl": "string",
            "buttons": [
               {
```

```
                    "text": "string",
                    "value": "string"
                }
            ],
            "imageUrl": "string",
            "subTitle": "string",
            "title": "string"
        }
    ],
    "version": "string"
  },
  "sentimentResponse": {
    "sentimentLabel": "string",
    "sentimentScore": "string"
  },
  "sessionAttributes": {
    "string" : "string"
  },
  "sessionId": "string",
  "slots": {
    "string" : "string"
  },
  "slotToElicit": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

activeContexts

Uma lista de contextos ativos para a sessão. Um contexto pode ser definido quando uma intenção é cumprida ou chamando a operação `PostContent`, `PostText` ou `PutSession`.

Você pode usar um contexto para controlar as intenções que podem acompanhar uma intenção ou para modificar a operação do seu aplicativo.

Tipo: matriz de objetos ActiveContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 20 itens.

alternativeIntents

Uma a quatro intenções alternativas que podem ser aplicáveis à intenção do usuário.

Cada alternativa inclui uma pontuação que indica o grau de confiança do Amazon Lex de que a intenção corresponde à intenção do usuário. As intenções são classificadas pela pontuação de confiança.

Tipo: matriz de objetos PredictedIntent

Membros da matriz: número máximo de 4 itens.

botVersion

A versão do bot que respondeu à conversa. Você pode usar essas informações para ajudar a determinar se uma versão de um bot tem um desempenho melhor do que outra versão.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[0-9]+|\$LATEST`

dialogState

Identifica o estado atual da interação do usuário. O Amazon Lex retorna um dos seguintes valores como `dialogState`. Opcionalmente, o cliente pode usar essas informações para personalizar a interface do usuário.

- `ElicitIntent` - O Amazon Lex quer obter a intenção do usuário.

  Por exemplo, um usuário pode expressar uma intenção (“Quero pedir uma pizza”). Se o Amazon Lex não puder deduzir a intenção do usuário a partir dessa declaração, ele retornará esse dialogState.
- `ConfirmIntent` - O Amazon Lex espera uma resposta “sim” ou “não”.

  Por exemplo, o Amazon Lex quer a confirmação do usuário antes de atender uma intenção.

  Em vez de um “sim” ou “não”, um usuário pode responder com informações adicionais. Por exemplo, “sim, mas peça uma pizza de massa grossa” ou “não, quero pedir uma bebida”. O Amazon Lex pode processar essas informações adicionais (nesses exemplos, atualizar o valor do slot do tipo de crosta ou alterar a intenção de OrderPizza para OrderDrink).
- `ElicitSlot` - O Amazon Lex espera um valor de slot para a intenção atual.

Por exemplo, suponha que, na resposta, o Amazon Lex envie esta mensagem: “Qual tamanho de pizza você gostaria?”. Um usuário pode responder com o valor de slot (por exemplo, “média”). O usuário também pode fornecer informações adicionais na resposta (por exemplo, “pizza média de massa grossa”). O Amazon Lex pode processar essas informações adicionais de forma adequada.

- `Fulfilled` - Transmite que a função do Lambda configurada para a intenção atendeu com sucesso a intenção.
- `ReadyForFulfillment` - Transmite que o cliente deve atender à solicitação.
- `Failed` - Transmite que a conversa com o usuário falhou.

  Isso pode acontecer por vários motivos, incluindo o fato de o usuário não ter fornecido uma resposta adequada aos prompts do serviço (você pode configurar quantas vezes o Amazon Lex pode solicitar informações específicas a um usuário) ou se a função do Lambda não atendeu à intenção.

Tipo: String

Valores Válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

intentName

A intenção atual do usuário que o Amazon Lex conhece.

Tipo: String

message

A mensagem a ser transmitida ao usuário. A mensagem pode vir da configuração do bot ou de uma função do Lambda.

Se a intenção não estiver configurada com uma função do Lambda, ou se a função do Lambda tiver retornado `Delegate` como `dialogAction.type` em sua resposta, o Amazon Lex decide o próximo curso de ação e seleciona uma mensagem apropriada da configuração do bot com base no contexto de interação atual. Por exemplo, se o Amazon Lex não conseguir entender a entrada do usuário, ele usa uma mensagem de prompt de esclarecimento.

Ao criar uma intenção, você pode atribuir mensagens a grupos. Quando as mensagens são atribuídas a grupos, o Amazon Lex retorna uma mensagem de cada grupo na resposta. O campo de mensagem é uma string JSON de escape que contém as mensagens. Para obter

mais informações sobre a estrutura da string JSON retornada, consulte Formatos de mensagem suportados.

Se a função do Lambda retornar uma mensagem, o Amazon Lex a enviará para o cliente em sua resposta.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.024.

messageFormat

O formato da mensagem de resposta. Um dos seguintes valores:

- `PlainText` - A mensagem contém texto sem formatação UTF-8.
- `CustomPayload` - A mensagem é um formato personalizado definido pela função do Lambda.
- `SSML` - A mensagem contém texto formatado para saída de voz.
- `Composite` - A mensagem contém um objeto JSON de escape que contém uma ou mais mensagens dos grupos aos quais as mensagens foram atribuídas quando a intenção foi criada.

Tipo: String

Valores Válidos: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Fornece uma pontuação que indica o quanto o Amazon Lex tem certeza de que a intenção retornada é aquela que corresponde à intenção do usuário. A pontuação está entre 0,0 e 1,0. Para obter mais informações, consulte Pontuações de confiança.

A pontuação é relativa, não absoluta. A pontuação pode mudar com base nas melhorias no Amazon Lex.

Tipo: objeto IntentConfidence

responseCard

Representa as opções que o usuário tem para responder ao prompt atual. O cartão de resposta pode vir da configuração do bot (no console Amazon Lex, escolha o botão de configurações ao lado de um slot) ou de um hook de código (função do Lambda).

Tipo: objeto ResponseCard

sentimentResponse

O sentimento expresso em uma declaração.

Quando o bot está configurado para enviar declarações ao Amazon Comprehend para análise de sentimentos, esse campo contém o resultado da análise.

Tipo: objeto SentimentResponse

sessionAttributes

Um mapa de pares de chaves/valores que representam as informações de contexto específicas da sessão.

Tipo: mapa de string para string

sessionId

Um identificador exclusivo da sessão.

Tipo: String

slots

Os slots de intenção que o Amazon Lex detectou a partir da entrada do usuário na conversa.

O Amazon Lex cria uma lista de resolução que contém valores prováveis para um slot. O valor que ele retorna é determinado pelo `valueSelectionStrategy` selecionado quando o tipo de slot foi criado ou atualizado. Se `valueSelectionStrategy` for definido como `ORIGINAL_VALUE`, o valor fornecido pelo usuário será retornado, se o valor do usuário for semelhante ao valor de slot. Se `valueSelectionStrategy` estiver definido como `TOP_RESOLUTION`, o Amazon Lex retornará o primeiro valor na lista de resolução ou, se não houver lista de resolução, nulo. Se `valueSelectionStrategy` não for especificado, o padrão será `ORIGINAL_VALUE`.

Tipo: mapa de string para string

slotToElicit

Se o valor `dialogState` for `ElicitSlot`, retornará o nome do slot para o qual o Amazon Lex está obtendo um valor.

Tipo: String

Erros

BadGatewayException

Ou o bot do Amazon Lex ainda está sendo construído ou um dos serviços dependentes (Amazon Polly, AWS Lambda) falhou com um erro interno de serviço.

Código de status HTTP: 502

BadRequestException

A validação da solicitação falhou, não há mensagem utilizável no contexto ou a compilação do bot falhou, ainda está em andamento ou contém alterações não criadas.

Código de status HTTP: 400

ConflictException

Dois clientes estão usando a mesma conta da AWS, o bot do Amazon Lex e o mesmo ID de usuário.

Código de Status HTTP: 409

DependencyFailedException

Uma das dependências, como AWS Lambda ou Amazon Polly, gerou uma exceção. Por exemplo,

- Se o Amazon Lex não tiver permissões suficientes para chamar uma função do Lambda.
- Se uma função do Lambda levar mais de 30 segundos para ser executada.
- Se uma função do Lambda de atendimento retornar uma ação `Delegate` de diálogo sem remover nenhum valor de slot.

Código de status HTTP: 424

InternalFailureException

Erro de serviço interno. Tente a chamada novamente.

Código de status HTTP: 500

LimitExceededException

Excedeu um limite.

Código de status HTTP: 429

LoopDetectedException

Essa exceção não é usada.

Código de status HTTP: 508

NotFoundException

O recurso (como o bot Amazon Lex ou um alias) mencionado não foi encontrado.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutSession

Serviço: Amazon Lex Runtime Service

Cria uma nova sessão ou modifica uma sessão existente com um bot do Amazon Lex. Use essa operação para permitir que seu aplicativo defina o estado do bot.

Para obter mais informações, consulte Gerenciamento de sessões.

Sintaxe da Solicitação

```
POST /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
Accept: accept
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "dialogAction": {
      "fulfillmentState": "string",
      "intentName": "string",
      "message": "string",
      "messageFormat": "string",
      "slots": {
         "string" : "string"
      },
      "slotToElicit": "string",
      "type": "string"
   },
   "recentIntentSummaryView": [
      {
         "checkpointLabel": "string",
         "confirmationStatus": "string",
         "dialogActionType": "string",
```

```
            "fulfillmentState": "string",
            "intentName": "string",
            "slots": {
               "string" : "string"
            },
            "slotToElicit": "string"
         }
      ],
      "sessionAttributes": {
         "string" : "string"
      }
   }
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

accept

A mensagem que o Amazon Lex retorna na resposta pode ser texto ou fala dependendo do valor deste campo.

- Se o valor for `text/plain; charset=utf-8`, o Amazon Lex retornará o texto na resposta.
- Se o valor começar com `audio/`, o Amazon Lex retornará a fala na resposta. O Amazon Lex usa o Amazon Polly para gerar a fala na configuração que você especificar. Por exemplo, se você especificar `audio/mpeg` como valor, o Amazon Lex retornará a fala no formato MPEG.
- Se o valor for `audio/pcm`, a fala será retornada como `audio/pcm` no formato little-endian de 16 bits.
- Os valores aceitos são os seguintes:
  - `audio/mpeg`
  - `audio/ogg`
  - `audio/pcm`
  - `audio/*` (o padrão é mpeg)
  - `text/plain; charset=utf-8`

botAlias

O alias em uso para o bot que contém os dados da sessão.

Obrigatório: Sim

botName

O nome do bot que contém os dados da sessão.

Obrigatório: Sim

userId

O ID do usuário do aplicativo cliente. O Amazon Lex usa isso para identificar a conversa de um usuário com seu bot.

Restrições de tamanho: tamanho mínimo 2. Comprimento máximo de 100.

Padrão: `[0-9a-zA-Z._:-]+`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

activeContexts

Uma lista de contextos ativos para a solicitação. Um contexto pode ser ativado quando uma intenção anterior é atendida ou incluindo o contexto na solicitação,

Se você não especificar uma lista de contextos, o Amazon Lex usará a lista atual de contextos para a sessão. Se você especificar uma lista vazia, todos os contextos da sessão serão apagados.

Tipo: matriz de objetos ActiveContext

Membros da Matriz: número mínimo de 0 itens. Número máximo de 20 itens.

Obrigatório: Não

dialogAction

Define a próxima ação que o bot deve realizar para atender a conversa.

Tipo: objeto DialogAction

Obrigatório: Não

recentIntentSummaryView

Um resumo das intenções recentes do bot. Você pode usar a visualização de resumo de intenções para definir um rótulo de ponto de verificação em uma intenção e modificar os atributos das intenções. Você também pode usá-la para remover ou adicionar objetos de resumo de intenção à lista.

A intenção que você modifica ou adiciona à lista deve fazer sentido para o bot. Por exemplo, o nome da intenção deve ser válida para o bot. Você deve fornecer valores válidos para:

- `intentName`
- nomes de slot
- `slotToElict`

Se você enviar o parâmetro `recentIntentSummaryView` em uma solicitação `PutSession`, o conteúdo da nova visualização de resumo substituirá a antiga visualização de resumo. Por exemplo, se uma solicitação `GetSession` retornar três intenções na exibição de resumo e você chamar `PutSession` com uma intenção na visualização de resumo, a próxima chamada `GetSession` retornará apenas uma intenção.

Tipo: matriz de objetos IntentSummary

Membros da Matriz: número mínimo de 0 itens. Número máximo de 3 itens.

Obrigatório: Não

sessionAttributes

Mapa de pares de chaves/valores que representam as informações de contexto específicas da sessão. Ele contém informações do aplicativo passadas entre o Amazon Lex e um aplicativo cliente.

Tipo: mapa de string para string

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-Type: contentType
x-amz-lex-intent-name: intentName
x-amz-lex-slots: slots
```

```
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-message: message
x-amz-lex-encoded-message: encodedMessage
x-amz-lex-message-format: messageFormat
x-amz-lex-dialog-state: dialogState
x-amz-lex-slot-to-elicit: slotToElicit
x-amz-lex-session-id: sessionId
x-amz-lex-active-contexts: activeContexts

audioStream
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

A resposta retorna os cabeçalhos HTTP a seguir.

activeContexts

Uma lista de contextos ativos para a sessão.

contentType

Tipo de conteúdo conforme especificado no cabeçalho HTTP `Accept` na solicitação.

dialogState

- `ConfirmIntent` - O Amazon Lex espera uma resposta “sim” ou “não” para confirmar a intenção antes de atendê-la.
- `ElicitIntent` - O Amazon Lex quer obter a intenção do usuário.
- `ElicitSlot` - O Amazon Lex espera o valor de um slot para a intenção atual.
- `Failed` - Transmite que a conversa com o usuário falhou. Isso pode acontecer por vários motivos, incluindo o usuário não fornecer uma resposta adequada aos prompts do serviço ou a função do Lambda não atender a intenção.
- `Fulfilled` - Transmite que a função do Lambda atendeu com sucesso a intenção.
- `ReadyForFulfillment` - Transmite que o cliente deve atender à solicitação.

Valores Válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedMessage

A próxima mensagem que deve ser apresentada ao usuário.

O campo `encodedMessage` é codificado em base 64. Você deve decodificar o campo antes de poder usar o valor.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.366.

intentName

O nome da intenção atual.

message

Esse cabeçalho foi descontinuado.

A próxima mensagem que deve ser apresentada ao usuário.

Você só pode usar esse campo nas localidades de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. Em todas as outras localidades, o campo `message` é nulo. Em vez disso, use o campo `encodedMessage`.

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 1.024.

messageFormat

O formato da mensagem de resposta. Um dos seguintes valores:

- `PlainText` - A mensagem contém texto sem formatação UTF-8.
- `CustomPayload` - A mensagem é um formato personalizado para o cliente.
- `SSML` - A mensagem contém texto formatado para saída de voz.
- `Composite` - A mensagem contém um objeto JSON de escape que contém uma ou mais mensagens dos grupos aos quais as mensagens foram atribuídas quando a intenção foi criada.

Valores Válidos: `PlainText | CustomPayload | SSML | Composite`

sessionAttributes

Mapa de pares de chaves/valores que representam as informações de contexto específicas da sessão.

sessionId

Um identificador exclusivo da sessão.

slots

Mapa de zero ou mais slots de intenção do Amazon Lex detectados na entrada do usuário durante a conversação.

O Amazon Lex cria uma lista de resolução que contém valores prováveis para um slot. O valor que ele retorna é determinado pelo `valueSelectionStrategy` selecionado quando o tipo de slot foi criado ou atualizado. Se `valueSelectionStrategy` for definido como `ORIGINAL_VALUE`, o valor fornecido pelo usuário será retornado, se o valor do usuário for semelhante ao valor de slot. Se `valueSelectionStrategy` estiver definido como `TOP_RESOLUTION`, o Amazon Lex retornará o primeiro valor na lista de resolução ou, se não houver lista de resolução, nulo. Se `valueSelectionStrategy` não for especificado, o padrão será `ORIGINAL_VALUE`.

slotToElicit

Se o `dialogState` for `ElicitSlot`, retornará o nome do slot para o qual o Amazon Lex está obtendo um valor.

A resposta retorna as informações a seguir como corpo HTTP.

audioStream

A versão em áudio da mensagem a ser transmitida ao usuário.

Erros

BadGatewayException

Ou o bot do Amazon Lex ainda está sendo construído ou um dos serviços dependentes (Amazon Polly, AWS Lambda) falhou com um erro interno de serviço.

Código de status HTTP: 502

BadRequestException

A validação da solicitação falhou, não há mensagem utilizável no contexto ou a compilação do bot falhou, ainda está em andamento ou contém alterações não criadas.

Código de status HTTP: 400

ConflictException

Dois clientes estão usando a mesma conta da AWS, o bot do Amazon Lex e o mesmo ID de usuário.

Código de Status HTTP: 409

DependencyFailedException

Uma das dependências, como AWS Lambda ou Amazon Polly, gerou uma exceção. Por exemplo,

- Se o Amazon Lex não tiver permissões suficientes para chamar uma função do Lambda.
- Se uma função do Lambda levar mais de 30 segundos para ser executada.
- Se uma função do Lambda de atendimento retornar uma ação `Delegate` de diálogo sem remover nenhum valor de slot.

Código de status HTTP: 424

InternalFailureException

Erro de serviço interno. Tente a chamada novamente.

Código de status HTTP: 500

LimitExceededException

Excedeu um limite.

Código de status HTTP: 429

NotAcceptableException

O cabeçalho de aceitação na solicitação não tem um valor válido.

Código de status HTTP: 406

NotFoundException

O recurso (como o bot Amazon Lex ou um alias) mencionado não foi encontrado.

Código de status HTTP: 404

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK para .NET
- AWS SDK para C++

- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para Kotlin
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

Tipos de dados

Os seguintes tipos de dados são compatíveis com o Serviço de criação de modelos do Amazon Lex:

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata
- BuiltinIntentMetadata
- BuiltinIntentSlot
- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent
- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message

- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt
- ResourceReference
- Slot
- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration
- Statement
- Tag
- UtteranceData
- UtteranceList

Os seguintes tipos de dados são compatíveis com o Serviço de runtime do Amazon Lex:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

Serviço de criação de modelos do Amazon Lex

Os seguintes tipos de dados são compatíveis com o Serviço de criação de modelos do Amazon Lex:

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata
- BuiltinIntentMetadata
- BuiltinIntentSlot
- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent
- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message
- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt
- ResourceReference
- Slot
- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration

- Statement
- Tag
- UtteranceData
- UtteranceList

BotAliasMetadata

Serviço: Amazon Lex Model Building Service

Fornece informações sobre um alias de bot.

Conteúdo

botName

O nome do bot para o qual o alias aponta.

Tipo: string

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

botVersion

A versão do bot do Amazon Lex para o qual o alias aponta.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: não

checksum

Soma de verificação do alias do bot.

Tipo: String

Obrigatório: Não

conversationLogs

Configurações que determinam como o Amazon Lex usa logs de conversa para o alias.

Tipo: objeto ConversationLogsResponse

Obrigatório: não

createdDate

A data em que o alias do bot foi criado.

Tipo: carimbo de data/hora

Obrigatório: não

description

Uma descrição do alias do bot.

Tipo: string

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: não

lastUpdatedDate

A data em que o alias do bot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: carimbo de data/hora

Obrigatório: não

name

O nome do alias do bot.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

BotChannelAssociation

Serviço: Amazon Lex Model Building Service

Representa uma associação entre um bot do Amazon Lex e uma plataforma externa de mensagens.

Conteúdo

botAlias

Um alias que aponta para a versão específica do bot do Amazon Lex à qual essa associação está sendo feita.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

botConfiguration

Fornece as informações necessárias para se comunicar com a plataforma de mensagens.

Tipo: mapa de string para string

Entradas do mapa: número máximo de 10 itens.

Obrigatório: não

botName

O nome do bot Amazon Lex ao qual essa associação está sendo feita.

> **Note**
>
> Atualmente, o Amazon Lex oferece suporte a associações com o Facebook, o Slack e o Twilio.

Tipo: string

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

createdDate

A data em que a associação entre o bot do Amazon Lex e o canal foi criada.

Tipo: carimbo de data/hora

Obrigatório: não

description

Uma descrição de texto da associação que você está criando.

Tipo: string

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: não

failureReason

Se `status` for `FAILED`, o Amazon Lex fornece o motivo da falha na criação da associação.

Tipo: String

Obrigatório: Não

name

O nome da associação entre o bot e o canal.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

status

O status do canal do bot.

- `CREATED` - O canal foi criado e está pronto para uso.

- `IN_PROGRESS` - A criação do canal está em andamento.
- `FAILED` - Houve um erro ao criar o canal. Para obter informações sobre o motivo da falha, consulte o campo `failureReason`.

Tipo: string

Valores Válidos: `IN_PROGRESS | CREATED | FAILED`

Obrigatório: não

type

Especifica o tipo de associação indicando o tipo de canal que está sendo estabelecido entre o bot do Amazon Lex e a plataforma externa de mensagens.

Tipo: string

Valores Válidos: `Facebook | Slack | Twilio-Sms | Kik`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BotMetadata

Serviço: Amazon Lex Model Building Service

Fornece informações sobre um bot.

Conteúdo

createdDate

A data e a hora em que o bot foi criado.

Tipo: carimbo de data/hora

Obrigatório: não

description

Uma descrição do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: não

lastUpdatedDate

A data em que o bot foi atualizado. Quando você cria um bot, a data de criação e a data da última atualização são as mesmas.

Tipo: carimbo de data/hora

Obrigatório: não

name

O nome do bot.

Tipo: String

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

status

O status do bot.

Tipo: string

Valores Válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

Obrigatório: não

version

A versão do bot. Para um novo bot, a versão é sempre `$LATEST`.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BuiltinIntentMetadata

Serviço: Amazon Lex Model Building Service

Fornece metadados para uma intenção integrada.

Conteúdo

signature

Um identificador exclusivo de uma intenção integrada. Para encontrar a assinatura de uma intenção, consulte Intenções integradas padrão no Alexa Skills Kit.

Tipo: String

Obrigatório: Não

supportedLocales

Uma lista de identificadores das localidades compatíveis pela intenção.

Tipo: matriz de strings

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BuiltinIntentSlot

Serviço: Amazon Lex Model Building Service

Fornece informações sobre um slot usado em uma intenção integrada.

Conteúdo

name

Uma lista de slots definidos para a intenção.

Tipo: String

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BuiltinSlotTypeMetadata

Serviço: Amazon Lex Model Building Service

Fornece informações sobre um tipo de slot integrado.

Conteúdo

signature

Um identificador exclusivo de um tipo de slot integrado. Para encontrar a assinatura de um tipo de slot, consulte Referência do tipo de slot no Alexa Skills Kit.

Tipo: String

Obrigatório: Não

supportedLocales

Uma lista de localidades de destino para o slot.

Tipo: matriz de strings

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

CodeHook

Serviço: Amazon Lex Model Building Service

Especifica uma função do Lambda que verifica as solicitações feitas a um bot ou atende à solicitação do usuário a um bot.

Conteúdo

messageVersion

A versão da solicitação-resposta que você deseja que o Amazon Lex use para invocar a função do Lambda. Para obter mais informações, consulte Uso de funções do Lambda.

Tipo: String

Restrições de tamanho: o tamanho mínimo é 1. Tamanho máximo de 5.

Obrigatório: sim

uri

O nome do recurso da Amazon (ARN) da função do Lambda.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `arn:aws[a-zA-Z-]*:lambda:[a-z]+-[a-z]+(-[a-z]+)*-[0-9]:[0-9]{12}:function:[a-zA-Z0-9-_]+(\/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})?(:[a-zA-Z0-9-_]+)?`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ConversationLogsRequest

Serviço: Amazon Lex Model Building Service

Fornece as configurações necessárias para os logs de conversas.

Conteúdo

iamRoleArn

O Amazon Resource Name (ARN) de uma função do IAM com permissão para gravar em seus CloudWatch registros para registros de texto e em seu bucket do S3 para registros de áudio. Se a criptografia de áudio estiver habilitada, essa função também fornecerá permissão de acesso à chave do AWS KMS usada para criptografar logs de áudio. Para obter mais informações, consulte Criação de um perfil e política do IAM para logs de conversas.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Exigido: Sim

logSettings

Configurações para seus logs de conversas. Você pode registrar em log o texto da conversa, o áudio da conversa ou ambos.

Tipo: matriz de objetos LogSettingsRequest

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ConversationLogsResponse

Serviço: Amazon Lex Model Building Service

Contém informações sobre as configurações do log de conversas.

Conteúdo

iamRoleArn

O Amazon Resource Name (ARN) da função do IAM usado para gravar seus registros no Logs ou em um CloudWatch bucket do S3.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Obrigatório: não

logSettings

Configurações para seus logs de conversas. Você pode registrar em log texto, áudio ou ambos.

Tipo: matriz de objetos LogSettingsResponse

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

EnumerationValue

Serviço: Amazon Lex Model Building Service

Cada tipo de slot pode ter um conjunto de valores. Cada valor de enumeração representa um valor que o tipo de slot pode ter.

Por exemplo, um bot de pedido de pizza pode ter um tipo de slot que especifica o tipo de massa que a pizza deve ter. O tipo de slot pode incluir os valores

- grossa
- fina
- recheada

Conteúdo

value

O valor do tipo de slot.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 140.

Obrigatório: sim

synonyms

Valores adicionais relacionados ao valor do tipo de slot.

Tipo: matriz de strings

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 140.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

FollowUpPrompt

Serviço: Amazon Lex Model Building Service

Um prompt de atividade adicional após a realização de uma intenção. Por exemplo, depois que a intenção `OrderPizza` for atendida, você pode perguntar ao usuário para saber se ele quer pedir uma bebida.

Conteúdo

prompt

Solicita informações do usuário.

Tipo: objeto Prompt

Obrigatório: sim

rejectionStatement

Quando o usuário responde “não” à pergunta definida no campo `prompt`, o Amazon Lex responde com essa mensagem para confirmar que a intenção foi cancelada.

Tipo: objeto Statement

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

FulfillmentActivity

Serviço: Amazon Lex Model Building Service

Descreve como a intenção é atendida depois que o usuário fornece todas as informações necessárias para a intenção. Você pode fornecer uma função do Lambda ou configurar a intenção para retornar as informações da intenção ao aplicativo cliente. Recomendamos que você use uma função do Lambda para que a lógica relevante resida na nuvem e limite o código do lado do cliente principalmente à apresentação. Se precisar atualizar a lógica, atualize apenas a função do Lambda; você não precisa atualizar seu aplicativo cliente.

Considere os seguintes exemplos:

- Em um aplicativo de pedidos de pizza, depois que o usuário fornece todas as informações para fazer um pedido, você usa a função do Lambda para fazer um pedido em uma pizzaria.
- Em um aplicativo de jogos, quando um usuário diz “pegue uma pedra”, essas informações devem voltar para o aplicativo cliente para que ele possa realizar a operação e atualizar os gráficos. Nesse caso, você deseja que o Amazon Lex retorne os dados da intenção ao cliente.

Conteúdo

type

Como a intenção deve ser cumprida, seja executando uma função do Lambda ou retornando os dados do slot para o aplicativo cliente.

Tipo: string

Valores Válidos: `ReturnIntent | CodeHook`

Obrigatório: sim

codeHook

Uma descrição da função do Lambda que é executada para atender a intenção.

Tipo: objeto CodeHook

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

InputContext

Serviço: Amazon Lex Model Building Service

O nome de um contexto que deve estar ativo para que uma intenção seja selecionada pelo Amazon Lex.

Conteúdo

name

O nome do contexto.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Intent

Serviço: Amazon Lex Model Building Service

Identifica uma versão específica de uma intenção.

Conteúdo

intentName

O nome da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

intentVersion

A versão da intenção.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IntentMetadata

Serviço: Amazon Lex Model Building Service

Fornece informações sobre uma intenção.

Conteúdo

createdDate

A data em que a intenção foi criada.

Tipo: carimbo de data/hora

Obrigatório: não

description

Uma descrição da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: não

lastUpdatedDate

A data em que a intenção foi atualizada. Quando você cria uma intenção, a data de criação e a data da última atualização são as mesmas.

Tipo: carimbo de data/hora

Obrigatório: não

name

O nome da intenção.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

version

A versão da intenção.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

KendraConfiguration

Serviço: Amazon Lex Model Building Service

Fornece informações de configuração para o AMAZON. KendraSearchIntentintenção. Quando você usa essa intenção, o Amazon Lex pesquisa o índice do Amazon Kendra especificado e retorna documentos do índice que correspondem à declaração do usuário. Para obter mais informações, consulte AMAZON. KendraSearchIntent.

Conteúdo

kendraIndex

O Amazon Resource Name (ARN) do índice Amazon Kendra que você deseja que seja o AMAZON. KendraSearchIntent intenção de pesquisar. O índice precisa estar na mesma conta e região que o bot do Amazon Lex. Se o índice do Amazon Kendra não existir, você receberá uma exceção ao chamar a operação `PutIntent`.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9][a-zA-Z0-9_-]*`

Exigido: Sim

role

O nome do recurso da Amazon (ARN) de um perfil do IAM que tem permissão para pesquisar o índice do Amazon Kendra. O perfil precisa estar na mesma conta e região que o bot do Amazon Lex. Se o perfil não existir, você receberá uma exceção ao chamar a operação `PutIntent`.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `arn:aws:iam::[0-9]{12}:role/.*`

Exigido: Sim

queryFilterString

Um filtro de consulta que o Amazon Lex envia ao Amazon Kendra para filtrar a resposta de uma consulta. O filtro está no formato definido pelo Amazon Kendra. Para obter mais informações, consulte Filtragem de consultas.

Você pode substituir essa string de filtro por uma nova string de filtro em runtime.

Tipo: string

Restrições de tamanho: tamanho mínimo 0.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LogSettingsRequest

Serviço: Amazon Lex Model Building Service

Configurações usadas para configurar o modo de entrega e o destino dos logs de conversas.

Conteúdo

destination

Onde os logs serão entregues. Os registros de texto são enviados para um grupo de CloudWatch registros de registros. Os logs de áudio são entregues em um bucket do S3.

Tipo: string

Valores Válidos: `CLOUDWATCH_LOGS | S3`

Obrigatório: sim

logType

O tipo de logs a ser habilitado. Os registros de texto são enviados para um grupo de CloudWatch registros de registros. Os logs de áudio são entregues em um bucket do S3.

Tipo: string

Valores Válidos: `AUDIO | TEXT`

Obrigatório: sim

resourceArn

O Amazon Resource Name (ARN) do grupo de registros de CloudWatch registros ou do bucket do S3 em que os registros devem ser entregues.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:(?:logs:[\w\-]+:[\d]{12}:log-group:[\.\-_/#A-Za-z0-9]{1,512}(?::\*)?|s3:::[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9])$`

Exigido: Sim

kmsKeyArn

O nome do recurso da Amazon (ARN) da chave gerenciada pelo cliente do AWS KMS do para criptografar logs de áudio entregues a um bucket do S3. A chave não se aplica aos CloudWatch registros e é opcional para buckets do S3.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LogSettingsResponse

Serviço: Amazon Lex Model Building Service

As configurações para seus logs de conversas.

Conteúdo

destination

O destino em que os logs são entregues.

Tipo: string

Valores Válidos: `CLOUDWATCH_LOGS | S3`

Obrigatório: não

kmsKeyArn

O nome do recurso da Amazon (ARN) da chave usada para criptografar logs de áudio em um bucket do S3.

Tipo: string

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Obrigatório: não

logType

O tipo de log que está ativado.

Tipo: string

Valores Válidos: `AUDIO | TEXT`

Obrigatório: não

resourceArn

O Amazon Resource Name (ARN) do grupo de registros de CloudWatch registros ou bucket do S3 em que os registros são entregues.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:(?:logs:[\w\-]+:[\d]{12}:log-group:[\.\-_/#A-Za-z0-9]{1,512}(?::\*)?|s3:::[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9])$`

Obrigatório: não

resourcePrefix

O prefixo do recurso é a primeira parte da chave de objeto do S3 dentro do bucket do S3 que você especificou para conter logs de áudio. Para CloudWatch registros, é o prefixo do nome do fluxo de registros dentro do grupo de registros que você especificou.

Tipo: string

Restrições de comprimento: comprimento máximo de 1024.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Message

Serviço: Amazon Lex Model Building Service

O objeto de mensagem que fornece o texto e o tipo da mensagem.

Conteúdo

content

O texto da mensagem.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.000.

Obrigatório: sim

contentType

O tipo do conteúdo do string de mensagem.

Tipo: string

Valores Válidos: `PlainText | SSML | CustomPayload`

Obrigatório: sim

groupNumber

Identifica o grupo de mensagens ao qual a mensagem pertence. Quando um grupo é atribuído a uma mensagem, o Amazon Lex retorna uma mensagem de cada grupo na resposta.

Tipo: inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 5.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

MigrationAlert

Serviço: Amazon Lex Model Building Service

Fornece informações sobre alertas e avisos que o Amazon Lex envia durante uma migração. Os alertas incluem informações sobre como resolver o problema.

Conteúdo

details

Detalhes adicionais sobre o alerta.

Tipo: matriz de strings

Obrigatório: não

message

Uma mensagem que descreve por que o alerta foi emitido.

Tipo: String

Obrigatório: Não

referenceURLs

Um link para a documentação do Amazon Lex que descreve como resolver o alerta.

Tipo: matriz de strings

Obrigatório: não

type

O tipo de alerta. Há dois tipos de alertas:

- `ERROR` - Houve um problema com a migração que não pode ser resolvido. A migração é interrompida.
- `WARN` - Houve um problema com a migração que exige alterações manuais no novo bot do Amazon Lex V2. A migração continua.

Tipo: string

Valores Válidos: `ERROR` | `WARN`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

MigrationSummary

Serviço: Amazon Lex Model Building Service

Fornece informações sobre como migrar um bot do Amazon Lex V1 para o Amazon Lex V2.

Conteúdo

migrationId

O identificador exclusivo que o Amazon Lex atribuiu à migração.

Tipo: String

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

Obrigatório: não

migrationStatus

O status da operação. Quando o status é `COMPLETE`, o bot fica disponível no Amazon Lex V2. Pode haver alertas e avisos que precisem ser resolvidos para concluir a migração.

Tipo: String

Valores Válidos: `IN_PROGRESS | COMPLETED | FAILED`

Obrigatório: não

migrationStrategy

A estratégia usada para conduzir a migração.

Tipo: String

Valores Válidos: `CREATE_NEW | UPDATE_EXISTING`

Obrigatório: não

migrationTimestamp

A data e hora em que a migração foi iniciada.

Tipo: carimbo de data/hora

Obrigatório: não

v1BotLocale

A localidade do bot do Amazon Lex V1 que é a origem da migração.

Tipo: string

Valores Válidos: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Obrigatório: não

v1BotName

O nome do bot do Amazon Lex V1 que é a origem da migração.

Tipo: string

Restrições de tamanho: tamanho mínimo 2. Tamanho máximo de 50.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

v1BotVersion

A versão do bot do Amazon Lex V1 que é a origem da migração.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: não

v2BotId

O identificador exclusivo do Amazon Lex V2 que é o destino da migração.

Tipo: string

Restrições de tamanho: tamanho fixo de 10.

Padrão: `^[0-9a-zA-Z]+$`

Obrigatório: não

v2BotRole

O perfil do IAM que o Amazon Lex usa para executar a versão do bot do Amazon Lex V2.

Tipo: String

Restrições de tamanho: tamanho mínimo 20. Tamanho máximo de 2.048.

Padrão: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

OutputContext

Serviço: Amazon Lex Model Building Service

A especificação de um contexto de saída que é definido quando uma intenção é atendida.

Conteúdo

name

O nome do contexto.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

timeToLiveInSeconds

O número de segundos em que o contexto deve estar ativo após ser enviado pela primeira vez em uma resposta `PostText` ou `PostContent`. Você pode definir o valor entre 5 e 86.400 segundos (24 horas).

Tipo: inteiro

Faixa válida: valor mínimo de 5. Valor máximo de 86.400.

Obrigatório: sim

turnsToLive

O número de turnos de conversação em que o contexto deve ficar ativo. Um turno de conversa é uma solicitação `PostContent` ou `PostText` e a resposta correspondente do Amazon Lex.

Tipo: inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 20.

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Prompt

Serviço: Amazon Lex Model Building Service

Obtém informações do usuário. Para definir um prompt, forneça uma ou mais mensagens e especifique o número de tentativas de obter informações do usuário. Se você fornecer mais de uma mensagem, o Amazon Lex escolherá uma das mensagens para usar para avisar o usuário. Para obter mais informações, consulte Amazon Lex: como funciona.

Conteúdo

maxAttempts

O número de vezes para solicitar informações ao usuário.

Tipo: inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 5.

Obrigatório: sim

messages

Uma matriz de objetos, em que cada uma fornece uma string de mensagem e seu tipo. Você pode especificar a string da mensagem em texto sem formatação ou em Linguagem de marcação de síntese de fala (SSML).

Tipo: Matriz de objetos Message

Membros da matriz: número mínimo de 1 item. Número máximo de 15 itens.

Obrigatório: sim

responseCard

Um cartão de resposta. O Amazon Lex usa esse prompt em runtime, na resposta `PostText` da API. Ele substitui os atributos da sessão e os valores de slot pelos espaços reservados no cartão de resposta. Para obter mais informações, consulte Exemplo: uso de um cartão de resposta.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 50.000.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ResourceReference

Serviço: Amazon Lex Model Building Service

Descreve o recurso que se refere ao recurso que você está tentando excluir. Esse objeto é retornado como parte da exceção `ResourceInUseException`.

Conteúdo

name

O nome do recurso que está usando o recurso que você está tentando excluir.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `[a-zA-Z_]+`

Obrigatório: não

version

A versão do recurso que está usando o recurso que você está tentando excluir.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Slot

Serviço: Amazon Lex Model Building Service

Identifica uma versão de um slot específico.

Conteúdo

name

O nome do slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z](-|_|.)?)+$`

Exigido: Sim

slotConstraint

Especifica se o slot é obrigatório ou opcional.

Tipo: string

Valores Válidos: `Required | Optional`

Obrigatório: sim

defaultValueSpec

Uma lista de valores padrão para o slot. Os valores padrão serão usados quando o Amazon Lex não determinar um valor para o slot. Você pode especificar valores padrão a partir de variáveis de contexto, atributos de sessão e valores definidos.

Tipo: objeto SlotDefaultValueSpec

Obrigatório: não

description

Uma descrição do slot.

Tipo: string

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: não

obfuscationSetting

Determina se um slot está ofuscado nos logs de conversas e nas declarações armazenadas. Quando você ofusca um slot, o valor é substituído pelo nome do slot entre colchetes ({}). Por exemplo, se o nome do slot for “full_name”, os valores ofuscados serão substituídos por “{full_name}”. Para obter mais informações, consulte Ofuscação de slots.

Tipo: string

Valores Válidos: `NONE | DEFAULT_OBFUSCATION`

Obrigatório: não

priority

Informa ao Amazon Lex a ordem na qual o valor de slot do usuário deve ser obtido. Por exemplo, se a intenção tiver dois slots com prioridades 1 e 2, o AWS Amazon Lex primeiro obtém um valor para o slot com prioridade 1.

Se vários slots compartilharem a mesma prioridade, a ordem na qual o Amazon Lex extrai valores é arbitrária.

Tipo: inteiro

Intervalo válido: valor mínimo de 0. Valor máximo de 100.

Obrigatório: não

responseCard

Um conjunto de respostas possíveis para o tipo de slot usado por clientes baseados em texto. Um usuário escolhe uma opção no cartão de resposta, em vez de usar texto para responder.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 50.000.

Obrigatório: não

sampleUtterances

Se você conhecer um padrão específico com que os usuários poderão responder a uma solicitação do Amazon Lex para um valor de slot, poderá fornecer essas declarações para

melhorar a precisão. Isso é opcional. Na maioria dos casos, o Amazon Lex conseguirá entender as declarações do usuário.

Tipo: matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 200.

Obrigatório: não

slotType

O tipo do slot, seja um tipo de slot personalizado que você definiu ou um dos tipos de slot incorporado.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Obrigatório: não

slotTypeVersion

A versão do tipo de slot.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: não

valueElicitationPrompt

O prompt que o Amazon Lex usa para instigar o valor de slot do usuário.

Tipo: objeto Prompt

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotDefaultValue

Serviço: Amazon Lex Model Building Service

Um valor padrão para um slot.

Conteúdo

defaultValue

O valor padrão para o slot. É possível especificar um dos seguintes:

- `#context-name.slot-name` - O valor de slot “slot-name” no contexto “context-name”.
- `{attribute}` - O valor de slot do atributo “attribute” da sessão.
- `'value'` - O valor discreto “value”.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 202.

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotDefaultValueSpec

Serviço: Amazon Lex Model Building Service

Contém os valores padrão para o slot. Os valores padrão serão usados quando o Amazon Lex não determinar um valor para o slot.

Conteúdo

defaultValueList

Os valores padrão para o slot. Você pode especificar mais de um padrão. Por exemplo, você pode especificar um valor padrão a ser usado a partir de uma variável de contexto correspondente, um atributo de sessão ou um valor fixo.

O valor padrão escolhido é selecionado com base na ordem em que você os especifica na lista. Por exemplo, se você especificar uma variável de contexto e um valor fixo nessa ordem, o Amazon Lex usará a variável de contexto se ela estiver disponível, caso contrário, usará o valor fixo.

Tipo: matriz de objetos SlotDefaultValue

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotTypeConfiguration

Serviço: Amazon Lex Model Building Service

Fornece as informações de configuração para um tipo de slot.

Conteúdo

regexConfiguration

Uma expressão regular usada para validar o valor de slot.

Tipo: objeto SlotTypeRegexConfiguration

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotTypeMetadata

Serviço: Amazon Lex Model Building Service

Fornece informações sobre um tipo de slot.

Conteúdo

createdDate

A data em que o tipo de slot foi criado.

Tipo: carimbo de data/hora

Obrigatório: não

description

Uma descrição do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 0. Tamanho máximo de 200.

Obrigatório: não

lastUpdatedDate

A data em que o tipo de slot foi atualizado. Quando você cria um recurso, a data de criação e a data da última atualização são as mesmas.

Tipo: carimbo de data/hora

Obrigatório: não

name

O nome do tipo de slot.

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Obrigatório: não

version

A versão do tipo de slot.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotTypeRegexConfiguration

Serviço: Amazon Lex Model Building Service

Fornece uma expressão regular usada para validar o valor do slot.

Conteúdo

pattern

Uma expressão regular usada para validar o valor de slot.

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

- A-Z, a-z
- 0-9
- Caracteres unicode ("\ u<Unicode>")

Representa caracteres unicode com quatro dígitos, como "\u0041" ou "\u005A".

Os seguintes operadores de expressão regular não são aceitos:

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

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Statement

Serviço: Amazon Lex Model Building Service

Uma coleção de mensagens que transmitem informações ao usuário. Em runtime, o Amazon Lex seleciona a mensagem a ser transmitida.

Conteúdo

messages

Uma coleção de objetos de mensagem.

Tipo: Matriz de objetos Message

Membros da matriz: número mínimo de 1 item. Número máximo de 15 itens.

Obrigatório: sim

responseCard

Em tempo de execução, se o cliente estiver usando a PostTextAPI, o Amazon Lex incluirá o cartão de resposta na resposta. Ele substitui todos os atributos da sessão e os valores de slots pelos espaços reservados no cartão de resposta.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 50.000.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Tag

Serviço: Amazon Lex Model Building Service

Uma lista de pares de chave/valor que identificam um bot, um alias de bot ou um canal de bot. Os caracteres permitidos para chaves e valores são letras Unicode, dígitos, espaço em branco e qualquer um dos seguintes símbolos: _ . : / = + - @.

Conteúdo

key

A chave para a tag. As chaves não diferenciam maiúsculas de minúsculas.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: sim

value

O valor a ser associado a uma chave. O valor pode ser uma string vazia, mas não pode ser nulo.

Tipo: string

Restrições de tamanho: o tamanho mínimo é 0. O tamanho máximo é 256.

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

UtteranceData

Serviço: Amazon Lex Model Building Service

Fornece informações sobre uma única declaração que foi feita para seu bot.

Conteúdo

count

O número de vezes em que a declaração foi processada.

Tipo: inteiro

Obrigatório: não

distinctUsers

O número total de indivíduos que usaram a declaração.

Tipo: número inteiro

Obrigatório: não

firstUtteredDate

A data em que a declaração foi registrada pela primeira vez.

Tipo: carimbo de data/hora

Obrigatório: não

lastUtteredDate

A data em que a declaração foi registrada pela última vez.

Tipo: carimbo de data/hora

Obrigatório: não

utteranceString

O texto que foi inserido pelo usuário ou a representação de texto de um clipe de áudio.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.000.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

UtteranceList

Serviço: Amazon Lex Model Building Service

Fornece uma lista de declarações que foram feitas para uma versão específica do seu bot. A lista contém no máximo 100 declarações.

Conteúdo

botVersion

A versão do bot que processou a lista.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `\$LATEST|[0-9]+`

Obrigatório: não

utterances

Um ou mais objetos UtteranceData que contêm informações sobre as declarações feitas a um bot. O número máximo de objetos é 100.

Tipo: matriz de objetos UtteranceData

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Serviço de runtime do Amazon Lex

Os seguintes tipos de dados são compatíveis com o Serviço de runtime do Amazon Lex:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

ActiveContext

Serviço: Amazon Lex Runtime Service

Um contexto é uma variável que contém informações sobre o estado atual da conversa entre um usuário e o Amazon Lex. O contexto pode ser definido automaticamente pelo Amazon Lex quando uma intenção é cumprida ou pode ser definido em runtime usando a operação `PutContent`, `PutText` ou `PutSession`.

Conteúdo

name

O nome do contexto.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100.

Padrão: `^([A-Za-z]_?)+$`

Exigido: Sim

parameters

Variáveis de estado para o contexto atual. Você pode usar esses valores como valores padrão para slots em eventos subsequentes.

Tipo: mapa de string para string

Entradas do mapa: número mínimo de 0 itens. Número máximo de 10 itens.

Restrições de tamanho de chave: tamanho mínimo 1. Comprimento máximo de 100.

Restrições de tamanho do valor: tamanho mínimo 1. Tamanho máximo de 1.024.

Obrigatório: sim

timeToLive

O intervalo de tempo ou o número de turnos em que um contexto permanece ativo.

Tipo: objeto ActiveContextTimeToLive

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ActiveContextTimeToLive

Serviço: Amazon Lex Runtime Service

O intervalo de tempo ou o número de turnos em que um contexto permanece ativo.

Conteúdo

timeToLiveInSeconds

O número de segundos em que o contexto deve estar ativo após ser enviado pela primeira vez em uma resposta `PostText` ou `PostContent`. Você pode definir o valor entre 5 e 86.400 segundos (24 horas).

Tipo: inteiro

Faixa válida: valor mínimo de 5. Valor máximo de 86.400.

Obrigatório: não

turnsToLive

O número de turnos de conversação em que o contexto deve ficar ativo. Um turno de conversa é uma solicitação `PostContent` ou `PostText` e a resposta correspondente do Amazon Lex.

Tipo: inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 20.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Button

Serviço: Amazon Lex Runtime Service

Representa uma opção a ser exibida na plataforma do cliente (Facebook, Slack etc.)

Conteúdo

text

Texto visível para o usuário no botão.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 15.

Obrigatório: sim

value

O valor enviado para o Amazon Lex quando um usuário escolhe o botão. Por exemplo, considere o texto do botão “NYC”. Quando o usuário escolhe o botão, o valor enviado pode ser “Cidade de Nova York”.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.000.

Obrigatório: sim

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

DialogAction

Serviço: Amazon Lex Runtime Service

Descreve a próxima ação que o bot deve realizar em sua interação com o usuário e fornece informações sobre o contexto em que a ação ocorre. Use o tipo de dados `DialogAction` para definir a interação em um estado específico ou para retornar a interação a um estado anterior.

Conteúdo

type

A próxima ação que o bot deve realizar em sua interação com o usuário. Os valores possíveis são:

- `ConfirmIntent`: a próxima ação é perguntar ao usuário se a intenção está completa e pronta para ser atendida. Essa é uma pergunta de sim/não, como “Fazer o pedido?”
- `Close`: indica que não haverá uma resposta do usuário. Por exemplo, a afirmação "Seu pedido de pizza foi feito" não requer uma resposta.
- `Delegate`: a próxima ação é determinada pelo Amazon Lex.
- `ElicitIntent`: a próxima ação é determinar a intenção que o usuário deseja atender.
- `ElicitSlot`: a próxima ação é ilicitar um valor de slot do usuário.

Tipo: String

Valores Válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Obrigatório: sim

fulfillmentState

O estado de atendimento da intenção. Os valores possíveis são:

- `Failed`: a função do Lambda associada à intenção falhou em atender a intenção.
- `Fulfilled` - A intenção foi atendida pela função do Lambda associada à intenção.
- `ReadyForFulfillment`: todas as informações necessárias para que a intenção esteja presente e pronta para ser atendida pelo aplicativo do cliente.

Tipo: String

Valores Válidos: `Fulfilled | Failed | ReadyForFulfillment`

Obrigatório: não

intentName

O nome da intenção.

Tipo: String

Obrigatório: Não

message

A mensagem que deve ser exibida ao usuário. Se você não especificar uma mensagem, o Amazon Lex usará a mensagem configurada para a intenção.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.024.

Obrigatório: não

messageFormat

- `PlainText` - A mensagem contém texto sem formatação UTF-8.
- `CustomPayload` - A mensagem é um formato personalizado para o cliente.
- `SSML` - A mensagem contém texto formatado para saída de voz.
- `Composite`: a mensagem contém um objeto JSON de escape com uma ou mais mensagens. Para obter mais informações, consulte Grupos de mensagens.

Tipo: string

Valores Válidos: `PlainText | CustomPayload | SSML | Composite`

Obrigatório: não

slots

Mapa dos slots que foram coletados e seus valores.

Tipo: mapa de string para string

Obrigatório: não

slotToElicit

O nome do slot que deve ser obtido do usuário.

Tipo: String

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

GenericAttachment

Serviço: Amazon Lex Runtime Service

Representa uma opção renderizada para o usuário quando um prompt é exibido. Pode ser uma imagem, um botão, um link ou um texto.

Conteúdo

attachmentLinkUrl

O URL de um anexo ao cartão de resposta.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: não

buttons

A lista de opções a serem mostradas ao usuário.

Tipo: matriz de objetos Button

Membros da Matriz: número mínimo de 0 itens. Número máximo de 5 itens.

Obrigatório: não

imageUrl

O URL de uma imagem que é exibida ao usuário.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: não

subTitle

O subtítulo mostrado abaixo do título.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 80.

Obrigatório: não

title

O título da opção.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 80.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IntentConfidence

Serviço: Amazon Lex Runtime Service

Fornece uma pontuação que indica o quanto o Amazon Lex tem certeza de que uma intenção satisfaz a intenção do usuário.

Conteúdo

score

Uma pontuação que indica o quanto o Amazon Lex tem certeza de que a intenção satisfaz a intenção do usuário. Varia entre 0,00 e 1,00. Pontuações mais altas indicam maior confiança.

Tipo: duplo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IntentSummary

Serviço: Amazon Lex Runtime Service

Fornece informações sobre o estado de uma intenção. Você pode usar essas informações para obter o estado atual de uma intenção para poder processá-la ou retorná-la ao estado anterior.

Conteúdo

dialogActionType

A próxima ação que o bot deve realizar em sua interação com o usuário. Os valores possíveis são:

- `ConfirmIntent`: a próxima ação é perguntar ao usuário se a intenção está completa e pronta para ser atendida. Essa é uma pergunta de sim/não, como “Fazer o pedido?”
- `Close`: indica que não haverá uma resposta do usuário. Por exemplo, a afirmação "Seu pedido de pizza foi feito" não requer uma resposta.
- `ElicitIntent`: a próxima ação é determinar a intenção que o usuário deseja atender.
- `ElicitSlot`: a próxima ação é ilicitar um valor de slot do usuário.

Tipo: String

Valores Válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Obrigatório: sim

checkpointLabel

Um rótulo definido pelo usuário que identifica uma intenção específica. Você pode usar esse rótulo para retornar a uma intenção anterior.

Use o parâmetro `checkpointLabelFilter` da operação `GetSessionRequest` para filtrar as intenções retornadas pela operação para aquelas com somente o rótulo especificado.

Tipo: String

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 255.

Padrão: `[a-zA-Z0-9-]+`

Obrigatório: não

confirmationStatus

O status da intenção após o usuário responder ao prompt de confirmação. Se o usuário confirmar a intenção, o Amazon Lex define esse campo como `Confirmed`. Se o usuário negar a intenção, o Amazon Lex define esse valor como `Denied`. Os valores possíveis são:

- `Confirmed` - O usuário respondeu “Sim” ao prompt de confirmação, confirmando que a intenção está completa e que está pronta para ser atendida.
- `Denied` - O usuário respondeu “Não” ao prompt de confirmação.
- `None` - O usuário nunca recebeu um prompt de confirmação; ou o usuário recebeu um prompt, mas não o confirmou nem negou.

Tipo: String

Valores Válidos: `None | Confirmed | Denied`

Obrigatório: não

fulfillmentState

O estado de atendimento da intenção. Os valores possíveis são:

- `Failed`: a função do Lambda associada à intenção falhou em atender a intenção.
- `Fulfilled` - A intenção foi atendida pela função do Lambda associada à intenção.
- `ReadyForFulfillment`: todas as informações necessárias para que a intenção esteja presente e pronta para ser atendida pelo aplicativo do cliente.

Tipo: String

Valores Válidos: `Fulfilled | Failed | ReadyForFulfillment`

Obrigatório: não

intentName

O nome da intenção.

Tipo: String

Obrigatório: Não

slots

Mapa dos slots que foram coletados e seus valores.

Tipo: mapa de string para string

Obrigatório: não

slotToElicit

O próximo slot a ser obtido de um usuário. Se não houver espaço para extrair, o campo ficará em branco.

Tipo: String

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

PredictedIntent

Serviço: Amazon Lex Runtime Service

Uma intenção sugerida pelo Amazon Lex satisfaz a intenção do usuário. Inclui o nome da intenção, a confiança que o Amazon Lex tem de que a intenção do usuário está satisfeita e os slots definidos para a intenção.

Conteúdo

intentName

O nome de uma intenção sugerida pelo Amazon Lex que satisfaz a intenção do usuário.

Tipo: String

Obrigatório: Não

nluIntentConfidence

Indica o quanto o Amazon Lex tem certeza de que a intenção satisfaz a intenção do usuário.

Tipo: objeto IntentConfidence

Obrigatório: não

slots

O slot e os valores do slot associados à intenção prevista.

Tipo: mapa de string para string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ResponseCard

Serviço: Amazon Lex Runtime Service

Se você configurar um cartão de resposta ao criar seus bots, o Amazon Lex substituirá os atributos da sessão e os valores de slots disponíveis e os retornará. O cartão de resposta também pode vir de uma função do Lambda (`dialogCodeHook` e `fulfillmentActivity` em uma intenção).

Conteúdo

contentType

O tipo de conteúdo da resposta.

Tipo: string

Valores Válidos: `application/vnd.amazonaws.card.generic`

Obrigatório: não

genericAttachments

Uma matriz de objetos anexos representando opções.

Tipo: matriz de objetos GenericAttachment

Membros da Matriz: número mínimo de 0 itens. Número máximo de 10 itens.

Obrigatório: não

version

A versão do formato do cartão de resposta.

Tipo: String

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

SentimentResponse

Serviço: Amazon Lex Runtime Service

O sentimento expresso em uma declaração.

Quando o bot está configurado para enviar declarações ao Amazon Comprehend para análise de sentimentos, a estrutura desse campo contém o resultado da análise.

Conteúdo

sentimentLabel

O sentimento inferido no qual o Amazon Comprehend tem a maior confiança.

Tipo: String

Obrigatório: Não

sentimentScore

A probabilidade de o sentimento ter sido inferido corretamente.

Tipo: String

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em uma das linguagens específicas AWS SDKs, consulte o seguinte:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Histórico da documentação do Amazon Lex

- Última atualização da documentação: 9 de setembro de 2021

A tabela a seguir descreve as alterações importantes em cada versão do Amazon Lex. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

Alteração	Descrição	Data
Novo atributo	O Amazon Lex agora oferece suporte à localidade coreana (ko-KR). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	9 de setembro de 2021
Novo atributo	O Amazon Lex agora oferece suporte à localidade em inglês (indiano). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	15 de julho de 2021
Novo atributo	O Amazon Lex agora fornece uma ferramenta para migrar um bot para a API Amazon Lex V2. Para obter mais informações, consulte Migrar um bot.	13 de julho de 2021
Novo atributo	O Amazon Lex agora oferece suporte à localidade japonesa (Japão). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	1.º de abril de 2021

Novo atributo	O Amazon Lex agora oferece suporte às localidades alemã (alemã) (de-DE) e espanhola (latino-americana) (es-419). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	23 de novembro de 2020
Novo atributo	O Amazon Lex agora oferece suporte ao uso de contextos para gerenciar as intenções de ativação. Para obter mais informações, consulte Configurar contexto de intenção.	19 de novembro de 2020
Novo atributo	O Amazon Lex agora oferece suporte às localidades francesa (fr-FR), franco-canadense (fr-CA), italiana (it-IT) e espanhola (es-ES). Para obter uma lista completa das localidades compatíveis, consulte Idiomas compatíveis com o Amazon Lex.	11 de novembro de 2020
Novo atributo	O Amazon Lex agora oferece suporte à localidade espanhola (EUA) (es-US). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	22 de setembro de 2020

Novo atributo	O Amazon Lex agora oferece suporte à localidade em inglês (britânico) (en-GB). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	15 de setembro de 2020
Novo atributo	O Amazon Lex agora oferece suporte à localidade em inglês (australiano) (en-AU). Para obter mais informações, consulte Idiomas compatíveis com o Amazon Lex.	8 de setembro de 2020
Novo atributo	O Amazon Lex agora tem 7 novas intenções integrada s e 9 novos tipos de slots integrados. Para obter mais informações, consulte Intenções integradas e tipos de slots.	8 de setembro de 2020
Novo exemplo	Aprenda a criar um bot do Amazon Lex que os atendente s do suporte ao cliente possam usar para responder às perguntas dos clientes pesquisando respostas com o Amazon Kendra. Para obter mais informações, consulte Exemplo: Assistente de atendente de call center.	10 de agosto de 2020

Novo atributo	O Amazon Lex agora pode retornar até quatro intenções alternativas com base em pontuações de confiança. Para obter mais informações, consulte Usar pontuações de confiança.	6 de agosto de 2020
Expansão de região	O Amazon Lex já está disponível na Ásia-Pacífico (Tóquio) (ap-northeast-1).	30 de junho de 2020
Novo atributo	O Amazon Lex agora oferece suporte à pesquisa de índices do Amazon Kendra para obter respostas a perguntas frequentes. Para obter mais informações, consulte AMAZON. KendraSearchIntent .	11 de junho de 2020
Novo atributo	O Amazon Lex agora retorna mais informações em logs de conversa. Para obter mais informações, consulte Visualização de registros de texto no Amazon CloudWatch Logs.	9 de junho de 2020
Expansão de região	O Amazon Lex já está disponível nas regiões Ásia-Pacífico (Singapura) (ap-southeast-1), Europa (Frankfurt) (eu-central-1) e Europa (Londres) (eu-west-2).	23 de abril de 2020

Novo atributo	O Amazon Lex agora oferece suporte à marcação. É possível usar a marcação para identificar recursos, alocar custos e controlar o acesso. Para obter mais informações, consulte Marcar recursos do Amazon Lex.	12 de março de 2020
Novo atributo	O Amazon Lex agora oferece suporte a expressões regulares para a AMAZON. AlphaNumeric tipo de slot embutido. Para obter mais informações, consulte AMAZON. AlphaNumeric.	6 de fevereiro de 2020
Novo atributo	Agora o Amazon Lex pode registrar informações de conversa e ofuscar valores de slot nesses logs. Para obter mais informações, consulte Criar logs de conversa e Ofuscação de slot.	19 de dezembro de 2019
Expansão de região	O Amazon Lex já está disponível na região Ásia-Pacífico (Sydney) (ap-southeast-2).	17 de dezembro de 2019
Novo atributo	O Amazon Lex agora está em conformidade com a HIPAA. Para obter mais informações, consulte Validação de conformidade do Amazon Lex.	10 de dezembro de 2019

Novo atributo	Agora o Amazon Lex pode enviar declarações do usuário ao Amazon Comprehend para analisar o sentimento da declaração. Para obter mais informações, consulte Análise de sentimento.	21 de novembro de 2019
Novo atributo	O Amazon Lex agora está em conformidade com SOC. Para obter mais informaçõ es, consulte Validação de conformidade do Amazon Lex.	19 de novembro de 2019
Novo atributo	O Amazon Lex agora está em conformidade com PCI. Para obter mais informaçõ es, consulte Validação de conformidade do Amazon Lex.	17 de outubro de 2019
Novo atributo	Adicionado suporte à adição de um ponto de verificação a uma intenção para que seja possível facilmente retornar à intenção durante uma conversa. Para obter mais informações, consulte Gerenciamento de sessões.	10 de outubro de 2019
Novo atributo	Adição de suporte ao `AMAZON.FallbackIntent` para que seu bot possa lidar com situações em que a entrada do usuário não é a esperada. Para obter mais informações, consulte AMAZON. FallbackIntent.	3 de outubro de 2019

Novo atributo	O Amazon Lex permite que você gerencie informações da sessão para seus bots. Para obter mais informaçõ es, consulte Gerenciamento de sessões com a API do Amazon Lex.	8 de agosto de 2019
Expansão de região	O Amazon Lex já está disponível na região Oeste dos EUA (Oregon) (us-west-2).	8 de maio de 2018
Novo atributo	Adicionado suporte para exportação e importação e no formato do Amazon Lex. Para obter mais informações, consulte Importar e exportar bots, intenções e tipos de slot do Amazon Lex.	13 de fevereiro de 2018
Novo atributo	O Amazon Lex agora oferece suporte a outras mensagens de resposta para bots. Para obter mais informações, consulte Respostas.	8 de fevereiro de 2018
Expansão de região	O Amazon Lex já está disponível na região Europa (Irlanda) (eu-west-1).	21 de novembro de 2017
Novo atributo	Suporte adicionado para a implantação de bots do Amazon Lex no Kik. Para obter mais informações, consulte Integrar um bot do Amazon Lex ao Kik.	20 de novembro de 2017

Novo atributo	Suporte adicionado para novos tipos de slot e atributos de solicitação integrados. Para obter mais informações, consulte Tipos de slot integrados e Definir atributos de solicitação .	3 de novembro de 2017
Novo atributo	Exportação adicionada ao atributo Alexa Skills Kit. Para obter mais informações, consulte Exportar para uma habilidade do Alexa.	7 de setembro de 2017
Novo atributo	Suporte a sinônimo adicionado aos valores do tipo de slot. Para obter mais informações, consulte Tipos de slot personalizados .	31 de agosto de 2017
Novo recurso	AWS CloudTrail Integração adicionada. Para obter mais informações, consulte Monitoramento de chamadas de API do Amazon Lex com AWS CloudTrail registros.	15 de agosto de 2017
Documentação expandida	Foram adicionados exemplos de primeiros passos para AWS CLI o. Para ter mais informações, consulte https://docs.aws.amazon.com/lex/latest/dg/gs-cli.html	22 de maio de 2017
Novo guia	Esta é a primeira versão do Guia do usuário do Amazon Lex.	19 de abril de 2017

AWS Glossário

Para obter a AWS terminologia mais recente, consulte o AWS glossário na Glossário da AWS Referência.