aws

Guía para desarrolladores

Amazon Lex V1

Amazon Lex V1: Guía para desarrolladores

Table of Contents

Si utiliza Amazon Lex V2, consulte la guía de Amazon Lex V2.

Si utiliza Amazon Lex V1, le recomendamos que actualice los bots a Amazon Lex V2. Hemos dejado de agregar nuevas características a V1, por lo que recomendamos encarecidamente utilizar V2 para todos los nuevos bots.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.

¿Qué es Amazon Lex?

Amazon Lex es un servicio de AWS que permite crear interfaces de conversación para las aplicaciones que usan voz y texto. Con Amazon Lex, el mismo motor de conversación que utiliza Amazon Alexa ahora está disponible para cualquier desarrollador, lo que le permite crear sofisticados bots de chat de lenguaje natural en aplicaciones nuevas y existentes. Amazon Lex proporciona la amplia funcionalidad y flexibilidad de la comprensión del lenguaje natural (NLU) y el reconocimiento automático de voz (ASR) que le permiten crear experiencias de usuario sumamente atractivas con interacciones de conversación realistas y crear nuevas categorías de productos.

Amazon Lex permite a cualquier desarrollador crear rápidamente bots de chat de conversación. Con Amazon Lex no es necesaria una amplia experiencia en sistemas de aprendizaje profundo para crear un bot. Tan solo debe especificar el flujo de conversación básico en la consola de Amazon Lex. Amazon Lex administra el diálogo y ajusta dinámicamente las respuestas en la conversación. Mediante la consola puede crear, probar y publicar su chatbot de texto o voz. A continuación, puede añadir las interfaces de conversación a los bots en dispositivos móviles, aplicaciones web y plataformas de chat (por ejemplo, Facebook Messenger).

Amazon Lex está integrado de serie con AWS Lambda y se puede integrar fácilmente con muchos otros servicios de la plataforma de AWS, incluidos Amazon Cognito, AWS Mobile Hub, Amazon CloudWatch y Amazon DynamoDB. La integración con Lambda proporciona a los bots acceso a conectores empresariales sin servidor integrados previamente para vincular datos de aplicaciones de SaaS como, por ejemplo, Salesforce, HubSpot o Marketo.

Alguno de los beneficios de usar Amazon Lex son los siguientes:

- Sencillez: Amazon Lex le guía a través del uso de la consola para que pueda crear su propio bot de chat en cuestión de minutos. Solo tiene que proporcionar algunas frases de ejemplo y Amazon Lex generará un modelo de lenguaje natural completo con el que interactuar por medio de voz y texto para formular preguntas, obtener respuestas y realizar tareas sofisticadas.

- Tecnologías de aprendizaje profundo democratizadas: Amazon Lex cuenta con la misma tecnología que Alexa y ofrece tecnologías de ASR y NLU para crear un sistema de comprensión del lenguaje hablado (SLU). Por medio de SLU, Amazon Lex recibe la entrada de texto y voz en lenguaje natural, entiende la intención de la entrada y cumple con la intención del usuario al invocar la función empresarial adecuada.

La comprensión del lenguaje natural y el reconocimiento de voz son algunos de los problemas principales que deben resolver las ciencias computacionales, lo que requiere entrenar sofisticados algoritmos de aprendizaje profundo con enormes cantidades de datos e infraestructura. Amazon Lex pone las tecnologías de aprendizaje profundo al alcance de todos los desarrolladores, con la misma tecnología que Alexa. Los bots de chat de Amazon Lex convierten la voz entrante en texto y comprenden la intención del usuario para generar una respuesta inteligente, para que pueda centrarse en el desarrollo de bots y generar un valor añadido diferente para sus clientes. Asimismo, logrará definir categorías de productos totalmente nuevas gracias a las interfaces de conversación.

- Implementación y escalado sin problemas: con Amazon Lex, puede crear, probar e implementar sus bots de chat directamente desde la consola de Amazon Lex. Amazon Lex le permite publicar fácilmente los bots de chat de voz o texto para su uso en dispositivos móviles, aplicaciones web y servicios de chat (por ejemplo, Facebook Messenger). Amazon Lex se escala automáticamente para que no tenga que preocuparse por el aprovisionamiento del hardware y la administración de la infraestructura para impulsar la experiencia del bot.

- Integración de serie con la plataforma de AWS: Amazon Lex dispone de interoperabilidad nativa con otros servicios de AWS como Amazon Cognito, AWS Lambda, Amazon CloudWatch y AWS Mobile Hub. Puede aprovechar la eficacia de la plataforma de AWS para seguridad, monitorización, autenticación de usuarios, lógica de negocio, almacenamiento y desarrollo de aplicaciones móviles.

- Rentabilidad: con Amazon Lex, no hay costos por anticipado ni cuotas mínimas. Solo se le cobrará por las solicitudes de texto o voz que se realicen. Los precios del pago por uso y el bajo costo de cada solicitud hacen que el servicio sea un método rentable para crear interfaces de conversación. El nivel gratuito de Amazon Lex le permite probar fácilmente Amazon Lex sin ninguna inversión inicial.

¿Es la primera vez que usa Amazon Lex?

Si es la primera vez que utiliza Amazon Lex, le recomendamos que lea las siguientes secciones en orden:

1. Introducción a Amazon Lex: en esta sección va a configurar su cuenta y probar Amazon Lex.
2. Referencia de la API : esta sección proporciona ejemplos adicionales que puede utilizar para explorar Amazon Lex.

Funcionamiento de Amazon Lex

Amazon Lex le permite crear aplicaciones con una interfaz de voz o de texto basada en la misma tecnología que utiliza Amazon Alexa. A continuación presentamos los pasos habituales que debe llevar a cabo a la hora de trabajar con Amazon Lex:

1. Cree un bot y configúrelo con una o varias intenciones que desee admitir. Configure el bot para que entienda el objetivo del usuario (intención), participe en la conversación con el usuario para obtener información y cumpla la intención del usuario.
2. Pruebe el bot. Puede utilizar la ventana de prueba del cliente incluida en la consola de Amazon Lex.
3. Publique una versión y cree un alias.
4. Implemente el bot. Puede implementar el bot en plataformas como, por ejemplo, aplicaciones móviles o plataformas de mensajería como Facebook Messenger.

Antes de empezar, familiarícese con los siguientes términos y conceptos clave de Amazon Lex:

- Bot: un bot realiza tareas automatizadas como, por ejemplo, pedir una pizza, reservar un hotel, pedir flores, etc. Un bot de Amazon Lex emplea capacidades de reconocimiento automático de voz (ASR) y comprensión del lenguaje natural (NLU). Cada bot debe tener un nombre único en su cuenta.

  Los bots de Amazon Lex pueden comprender entradas del usuario en forma de voz o texto y conversar en lenguaje natural. Puede crear funciones de Lambda y agregarlas como enlaces de código a la configuración de las intenciones para realizar tareas de validación de los datos de usuario y de cumplimiento.

- Intención: una intención representa una acción que el usuario desea realizar. Puede crear un bot que admita una o más intenciones relacionadas. Por ejemplo, puede crear un bot que pida pizza y bebidas. Para cada intención, debe proporcionar la siguiente información obligatoria:

- Nombre de la intención: un nombre descriptivo de la intención. Por ejemplo, **`OrderPizza`**. Los nombres de las intenciones deben ser únicos en su cuenta.
- Enunciados de muestra: cómo podría comunicar la intención un usuario. Por ejemplo, un usuario puede decir "¿Puedo pedir una pizza, por favor?" o "Deseo pedir una pizza".
- Cómo se cumple con la intención: la manera en que cumplir con la intención después de que el usuario proporcione la información necesaria (por ejemplo, realizar un pedido en una pizzería local). Es recomendable crear una función de Lambda para el cumplimiento de la intención.

  De forma opcional, puede configurar la intención para que Amazon Lex simplemente devuelva la información a la aplicación cliente y que esta se ocupe del cumplimiento.

  Además de las intenciones personalizadas, como pedir una pizza, Amazon Lex proporciona intenciones integradas para configurar rápidamente un bot. Para obtener más información, consulte Intenciones y tipos de slot integrados.

- Ranura: una intención puede requerir ninguna o varias ranuras o parámetros. Puede añadir parámetros como parte de la configuración. En tiempo de ejecución, Amazon Lex solicita al usuario valores de ranura específicos. El usuario deben proporcionar valores para todas las ranuras obligatorias para que Amazon Lex pueda cumplir con la intención.

  Por ejemplo, la intención `OrderPizza` requiere slots como el tamaño de la pizza, el tipo de masa y el número de pizzas. En la configuración de la intención debe añadir estos slots. Para cada ranura, hay que proporcionar un tipo de ranura y una pregunta para que Amazon Lex los envíe al cliente y obtenga datos del usuario. Un usuario puede responder con un valor de ranura que contenga palabras adicionales, como “una pizza grande, por favor” o “prefiero el tamaño pequeño”. Amazon Lex entiende igualmente el valor de la ranura previsto.

- Tipo de ranura: cada ranura tiene un tipo. Puede crear sus propios tipos de slot personalizados o utilizar tipos de slot integrados. Cada tipo de ranura debe tener un nombre único en su cuenta. Por ejemplo, puede crear y utilizar los siguientes tipos de slot para la intención `OrderPizza`:

- Tamaño: con los valores de enumeración `Small`, `Medium` y `Large`.
- Masa: con los valores de enumeración `Thick` y `Thin`.

Amazon Lex también ofrece tipos de ranura integrados. Por ejemplo, `AMAZON.NUMBER` es un tipo de slot integrado que puede utilizar con el número de pizzas encargadas. Para obtener más información, consulte Intenciones y tipos de slot integrados.

Para ver una lista de las regiones de AWS en las que Amazon Lex está disponible, consulte Regiones y puntos de conexión en la Referencia general de Amazon Web Services.

Los siguientes temas aportan información adicional. Le recomendamos que los revise en orden y que luego pase a los ejercicios Introducción a Amazon Lex.

Temas

Idiomas admitidos en Amazon Lex

Amazon Lex V1 admite una gran variedad de idiomas y configuraciones regionales. Los idiomas admitidos y las características que los admiten se muestran en las siguientes tablas.

Amazon Lex V2 admite otros idiomas. Consúltelos en Idiomas admitidos en Amazon Lex V2.

Idiomas y configuraciones regionales admitidos

Amazon Lex V1 admite los siguientes idiomas y configuraciones regionales.

Code	Lenguaje y configuración
de-DE	Alemán (Alemania)
en-AU	Inglés (Australia)
en-GB	Inglés (Reino Unido)
en-IN	Inglés (India)
en-US	Inglés (EE. UU.)
es-419	Español (Latinoamérica)
es-ES	Español (España)
es-US	Español (EE. UU.)
fr-CA	Francés (Canadá)
fr-FR	Francés (Francia)
it-IT	Italiano (Italia)
ja-JP	Japonés (Japón)
ko-KR	Coreano (Corea)

Idiomas y configuraciones regionales compatibles con las características de Amazon Lex

Todas las características de Amazon Lex son compatibles con todos los idiomas y configuraciones regionales, excepto los que se indican en esta tabla.

Característica	Idiomas y configuraciones regionales compatibles
Establecimiento del contexto de la intención	Inglés (EE. UU.) (en-US)

Modelo de programación

Un bot es el tipo de recurso principal de Amazon Lex. Los otros tipos de recursos de Amazon Lex son intención, tipo de ranura, alias y asociación de canal del bot.

Puede crear un bot mediante la consola de Amazon Lex o la API de desarrollo de modelos. La consola ofrece una interfaz gráfica de usuario que se usa para compilar un bot listo para producción para la aplicación. Si lo prefiere, puede utilizar la API del modelo de desarrollo con la AWS CLI o un programa propio para crear un bot.

Después de crear un bot, puede implementarlo en una de las plataformas admitidas o integrarlo en su propia aplicación. Cuando un usuario interactúa con el bot, la aplicación cliente le envía solicitudes con la API en tiempo de ejecución de Amazon Lex. Por ejemplo, cuando un usuario dice “Quiero pedir una pizza”, el cliente envía esta entrada a Amazon Lex con una de las operaciones de la API en tiempo de ejecución. Los usuarios pueden proporcionar entradas en formato de texto o de voz.

También puede crear funciones de Lambda y usarlas en una intención. Estos enlaces de código en forma de funciones de Lambda le permiten llevar a cabo ciertas actividades en tiempo de ejecución como la inicialización, la validación de entradas del usuario y el cumplimiento de intenciones. Las secciones siguientes facilitarán información adicional.

Temas

- Operaciones de la API de desarrollo de modelo
- Operaciones de la API en tiempo de ejecución
- Funciones de Lambda como enlaces de código

Operaciones de la API de desarrollo de modelo

Para crear mediante programación tipos de slots, intenciones y bots, utilice las operaciones de la API de desarrollo de modelo. También puede utilizar la API de desarrollo de modelo para administrar, actualizar y eliminar recursos del bot. Las operaciones de la API de desarrollo de modelo incluyen:

- PutBot, PutBotAlias, PutIntent y PutSlotType para crear y actualizar bots, alias de bots, intenciones y tipos de slots, respectivamente.
- CreateBotVersion, CreateIntentVersion y CreateSlotTypeVersion para crear y publicar versiones de los bots, alias de bots, intenciones y tipos de slots, respectivamente.
- GetBot y GetBots para obtener un bot específico o una lista de bots que haya creado, respectivamente.
- GetIntent y GetIntents para obtener una intención o una lista de intenciones específicas que haya creado, respectivamente.
- GetSlotType y GetSlotTypes para obtener un tipo de slot o una lista de tipo de slots específicos que haya creado, respectivamente.
- GetBuiltinIntent, GetBuiltinIntents y GetBuiltinSlotTypes para obtener una intención integrada de Amazon Lex, una lista de intenciones integradas de Amazon Lex o una lista de tipos de ranura integrados que puede utilizar en el bot, respectivamente.
- GetBotChannelAssociation y GetBotChannelAssociations para obtener una asociación entre el bot y una plataforma de mensajería, o una lista de las asociaciones entre el bot y las plataformas de mensajería, respectivamente.
- DeleteBot, DeleteBotAlias, DeleteBotChannelAssociation, DeleteIntent y DeleteSlotType para eliminar los recursos innecesarios de la cuenta.

Puede utilizar la API de desarrollo de modelos para crear herramientas personalizadas con las que administrar los recursos de Amazon Lex. Por ejemplo, existe un límite de 100 versiones para cada uno de los bots, las intenciones y los tipos de slots. Podría utilizar la API de desarrollo de modelos para crear una herramienta que elimine automáticamente versiones antiguas cuando el bot se acerca al límite.

Amazon Lex utiliza sumas de comprobación para asegurarse de que solo una operación actualiza un recurso a la vez. Cuando utiliza una operación de la API `Put` (PutBot, PutBotAlias PutIntent o PutSlotType) para actualizar un recurso, debe indicar la suma de comprobación actual del recurso en la solicitud. Si dos herramientas intentan actualizar un recurso al mismo tiempo, ambas proporcionan la misma suma de comprobación. La primera solicitud que alcanza

Amazon Lex tiene la suma de comprobación correcta del recurso. Para cuando llega la segunda solicitud, la suma de comprobación es diferente. La segunda herramienta recibe una excepción `PreconditionFailedException` y la actualización termina.

Las operaciones `Get` (GetBot, GetIntent y GetSlotType) cuentan con coherencia final. Si utiliza una operación `Get` inmediatamente después de crear o modificar un recurso con una de las operaciones `Put`, es posible que no se devuelvan cambios. Después de que una operación `Get` devuelva la versión más reciente, siempre devuelve el recurso actualizado hasta que se modifica de nuevo. Puede determinar si se ha devuelto un recurso actualizado consultando la suma de comprobación.

Operaciones de la API en tiempo de ejecución

Las aplicaciones cliente utilizan las siguientes operaciones de la API en tiempo de ejecución para comunicarse con Amazon Lex:

- PostContent: toma la entrada de voz o texto y devuelve información sobre la intención y un mensaje de texto o voz que se transmite al usuario. En la actualidad, Amazon Lex admite los siguientes formatos de audio:

  Formatos de audio de entrada: LPCM y Opus

  Formatos de audio de salida: MPEG, OGG y PCM

  La operación `PostContent` es compatible con la entrada de audio a 8 kHz y 16 kHz. Las aplicaciones en las que el usuario final habla con Amazon Lex por teléfono, como un centro de llamadas automático, pueden pasar audio a 8 kHz directamente.

- PostText: toma texto como entrada y devuelve información sobre la intención y un mensaje de texto que se transmite al usuario.

La aplicación cliente utiliza la API en tiempo de ejecución para llamar a un bot de Amazon Lex específico para que procese los enunciados, ya sean texto o voz, de la entrada del usuario. Por ejemplo, suponga que un usuario dice "Quiero pizza". El cliente envía esta entrada de usuario a un bot con una de las operaciones de la API en tiempo de ejecución de Amazon Lex. A partir de la

entrada del usuario, Amazon Lex reconoce que la solicitud corresponde a la intención `OrderPizza` definida en el bot. Amazon Lex inicia una conversación con el usuario para recopilar la información necesaria o los datos de la ranura como, por ejemplo, el tamaño, los ingredientes y el número de pizzas. Una vez que el usuario proporciona todos los datos de ranura necesarios, Amazon Lex invoca el enlace de código en forma de función de Lambda para llevar a cabo la intención o devuelve los datos de la intención al cliente, según cómo esté configurada.

Utilice la operación PostContent cuando el bot utiliza entrada de voz. Por ejemplo, una aplicación automatizada para un centro de llamadas puede enviar voz a un bot de Amazon Lex en vez de hacerlo a un agente para responder a las consultas de los clientes. Puede utilizar el formato de audio a 8 kHz para enviar audio directamente del teléfono a Amazon Lex.

La ventana de pruebas de la consola de Amazon Lex usa la API PostContent para enviar solicitudes de texto y voz a Amazon Lex. Puede utilizar esta ventana de prueba en los ejercicios de Introducción a Amazon Lex.

Funciones de Lambda como enlaces de código

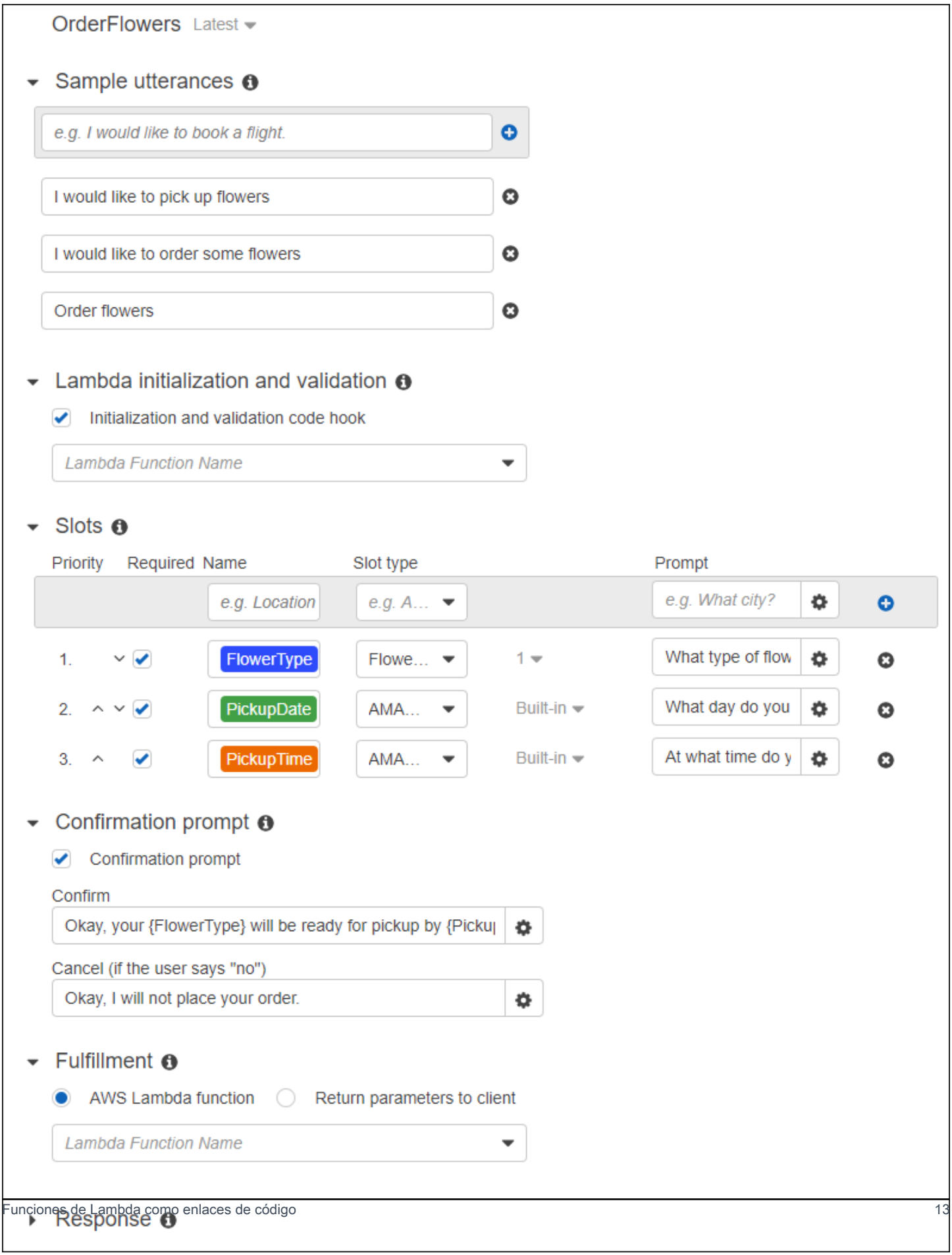
Puede configurar un bot de Amazon Lex para que invoque una función de Lambda como enlace de código. El enlace de código puede servir para varios propósitos:

- Personalizar la interacción del usuario: por ejemplo, cuando Joe pregunta por los ingredientes disponibles, puede utilizar el conocimiento previo de las preferencias de Joe para mostrar un subgrupo de ingredientes.
- Validar la entrada del usuario: imagine que Jen quiere recoger flores fuera del horario de apertura. Puede validar la hora que ha introducido Jen y enviar una respuesta adecuada.
- Cumplir la intención del usuario: una vez que Joe proporciona toda la información para el pedido de pizza, Amazon Lex puede llamar a una función de Lambda para que haga el pedido en una pizzería local.

Al configurar una intención, se especifican funciones de Lambda como enlaces de código en los siguientes lugares:

- Enlace de código de diálogo para inicialización y validación: se invoca esta función de Lambda en cada entrada del usuario, suponiendo que Amazon Lex haya entendido la intención del usuario.
- Enlace de código de cumplimiento: esta función de Lambda se invoca después de que el usuario proporcione todos los datos de ranura requeridos para llevar a cabo la intención.

Puede elegir la intención y establecer los enlaces de código en la consola de Amazon Lex, tal como se muestra en la siguiente captura de pantalla:


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

También puede definir los enlaces de código usando los campos `dialogCodeHook` y `fulfillmentActivity` de la operación PutIntent.

Una función de Lambda puede encargarse de la inicialización, la validación y el cumplimiento. Los datos de evento que recibe la función de Lambda incluyen un campo que identifica a quien la llama como un diálogo o como un enlace de código de cumplimiento. Puede utilizar esta información para ejecutar la parte apropiada del código.

Puede utilizar una función de Lambda para compilar un bot que se desenvuelva en diálogos complejos. Puede utilizar el campo `dialogAction` de la respuesta de la función de Lambda para hacer que Amazon Lex lleve a cabo acciones concretas. Por ejemplo, puede utilizar la acción de diálogo `ElicitSlot` para indicar a Amazon Lex que solicite al usuario un valor de ranura que no sea obligatorio. Si tiene una pregunta aclaratoria definida, puede utilizar la acción de diálogo `ElicitIntent` para obtener una nueva intención cuando el usuario haya terminado con la anterior.

Para obtener más información, consulte Uso de funciones de Lambda.

Administración de mensajes

Temas

- Tipos de mensajes
- Contextos para la configuración de mensajes
- Formatos de mensajes admitidos
- Grupos de mensajes
- Tarjetas de respuesta

Al crear un bot, puede configurar los mensajes de aclaración o informativos que desea que envíe al cliente. Considere los siguientes ejemplos:

- Podría configurar el bot con la siguiente pregunta aclaratoria:

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

  Amazon Lex envía este mensaje al cliente si no entiende la intención del usuario.

- Suponga que crea un bot para respaldar una intención llamada `OrderPizza`. Para pedir una pizza, desea que los usuarios proporcionen información como el tamaño, los ingredientes y el tipo de masa de la pizza. Podría configurar las siguientes preguntas:

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

  En cuanto Amazon Lex determina la intención del usuario de pedir una pizza, envía estos mensajes al cliente para obtener información del usuario.

En esta sección se explica el diseño de las interacciones del usuario en la configuración de su bot.

Tipos de mensajes

Un mensaje puede ser una pregunta o una afirmación.

- Una pregunta normalmente es una consulta y espera la respuesta del usuario.
- Una afirmación es informativa. No espera una respuesta.

Un mensaje puede incluir referencias a slots, a atributos de sesión o a atributos de solicitud. En el tiempo de ejecución, Amazon Lex reemplaza estas referencias por valores reales.

Para hacer referencia a los valores de slot que se han configurado, utilice la sintaxis siguiente:

```
{SlotName}
```

Para incluir los atributos de la sesión, utilice la sintaxis siguiente:

```
[SessionAttributeName]
```

Para hacer referencia a los atributos de sesión, utilice la sintaxis siguiente:

```
((RequestAttributeName))
```

Los mensajes pueden incluir valores de slot, atributos de sesión y atributos de solicitud.

Por ejemplo, suponga que configura el siguiente mensaje en la intención OrderPizza de su bot:

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

Este mensaje se refiere tanto a los atributos de slot (`PizzaTopping`) como a los de la sesión (`FirstName` y `DeliveryTime`). En el tiempo de ejecución, Amazon Lex sustituye estos marcadores de posición por valores y devuelve el siguiente mensaje al cliente:

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

Para incluir corchetes ([]) o llaves ({}) en un mensaje, utilice el carácter de barra inversa (\). Por ejemplo, el siguiente mensaje incluye las llaves y los corchetes:

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

El texto que se devuelve a la aplicación se parece al siguiente:

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

Para obtener información sobre los atributos de sesión, consulte las operaciones de la API en tiempo de ejecución PostText y PostContent. Para ver un ejemplo, consulte Reserva de viaje.

Las funciones de Lambda también pueden generar mensajes y devolverlos a Amazon Lex para que se los envíe al usuario. Si agrega funciones de Lambda al configurar la intención, puede crear mensajes de forma dinámica. Al proporcionar los mensajes mientras se configura el bot, puede eliminar la necesidad de construir una pregunta en la función de Lambda.

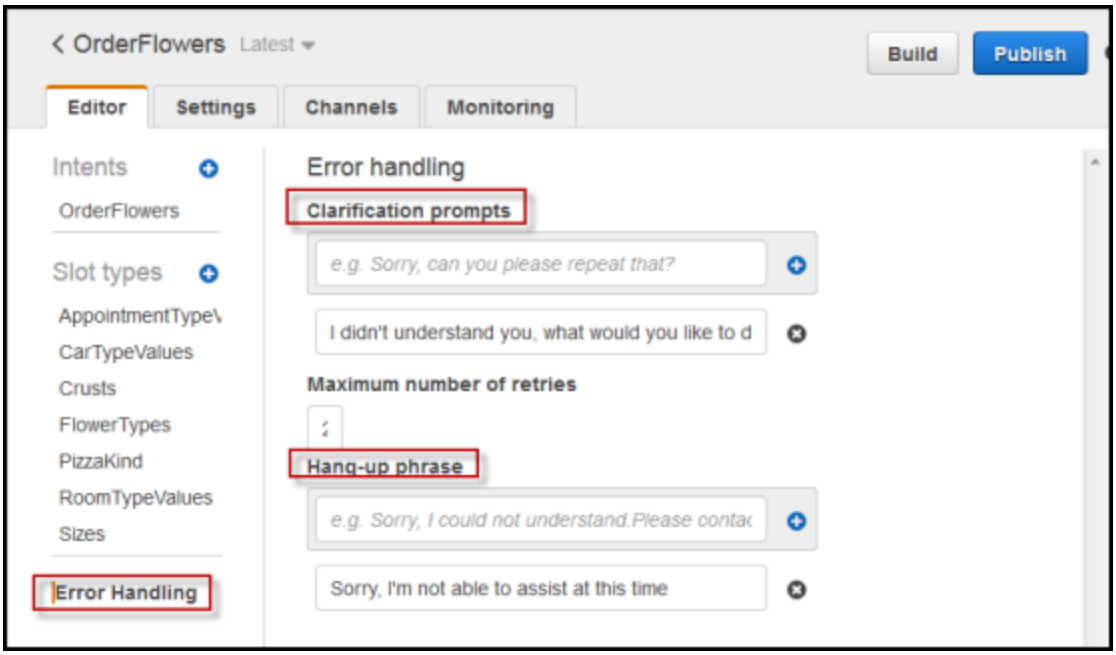
Contextos para la configuración de mensajes

Al crear el bot, puede crear los mensajes en diferentes contextos, como preguntas aclaratorias en el bot, preguntas para los valores de ranura y mensajes a partir de las intenciones. Amazon Lex elige un mensaje adecuado en cada contexto para devolverlo al usuario. Puede proporcionar un grupo de mensajes para cada contexto. Si lo hace así, Amazon Lex elige aleatoriamente un mensaje del grupo. También puede especificar el formato del mensaje o agrupar los mensajes. Para obtener más información, consulte Formatos de mensajes admitidos.

Si tiene una función de Lambda asociada con una intención, puede anular cualquiera de los mensajes que ha configurado en tiempo de compilación. Sin embargo, no se necesita una función de Lambda para utilizar cualquiera de estos mensajes.

Mensajes de bot

Puede configurar el bot con preguntas aclaratorias y mensajes de final de sesión. En tiempo de ejecución, Amazon Lex utiliza la pregunta aclaratoria si no comprende la intención del usuario. Puede configurar el número de veces que Amazon Lex solicita una aclaración antes de enviar el mensaje de fin de sesión. Puede configurar los mensajes en el nivel del bot en la sección Gestión de errores de la consola de Amazon Lex, tal como se indica en la siguiente imagen:



Con la API, puede configurar mensajes mediante el establecimiento de los campos `clarificationPrompt` y `abortStatement` en la operación PutBot.

Si utiliza una función de Lambda con una intención, la función de Lambda podría devolver una respuesta que indicara a Amazon Lex que preguntara la intención de un usuario. Si la función de Lambda no proporciona ese mensaje, Amazon Lex utiliza la pregunta aclaratoria.

Preguntas de slot

Debe especificar al menos un mensaje con pregunta para cada uno de los slots necesarios en una intención. En tiempo de ejecución, Amazon Lex utiliza uno de estos mensajes para solicitar al usuario que proporcione un valor para la ranura. Por ejemplo, para un slot `cityName`, una pregunta válida sería la siguiente:

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

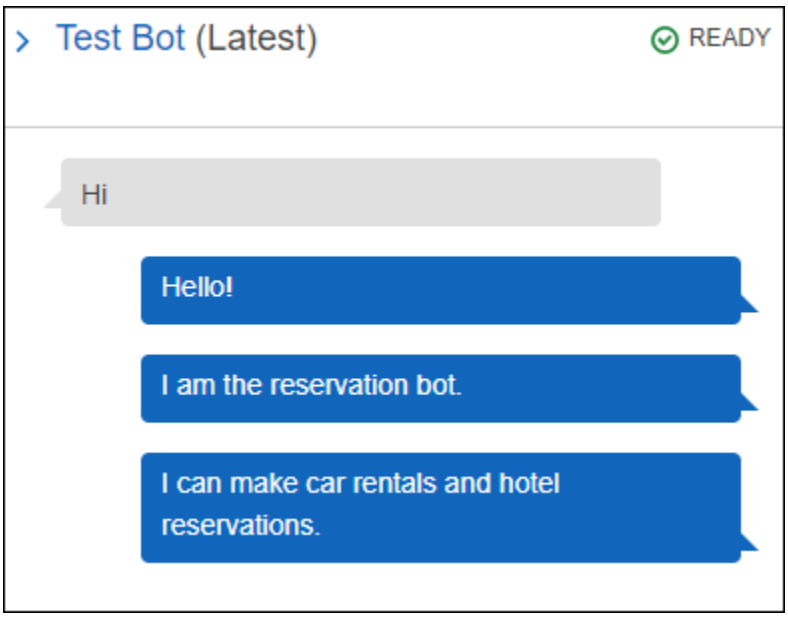
Puede configurar una o más preguntas para cada ranura utilizando la consola. También puede crear grupos de preguntas mediante la operación PutIntent. Para obtener más información, consulte Grupos de mensajes.

Respuestas

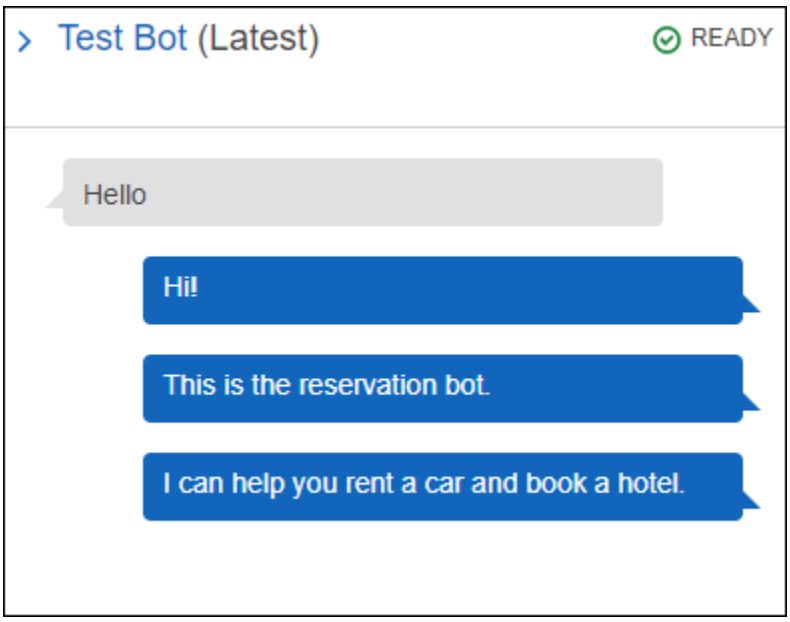
En la consola, utilice la sección Responses (Respuestas) para crear conversaciones dinámicas y participativas con el bot. Puede crear uno o varios grupos de mensajes para una respuesta. En el tiempo de ejecución, Amazon Lex crea una respuesta mediante la selección de un mensaje de cada grupo de mensajes. Para obtener más información acerca de los grupos de mensajes, consulte Grupos de mensajes.

Por ejemplo, el primer grupo de mensajes podría contener diferentes saludos: "Hola", "Buenas" y "Saludos". El segundo grupo de mensajes podría contener diferentes formas de presentación: "Soy el bot de reservas" y "Este es el bot de reservas". Un tercer grupo de mensajes podría comunicar las capacidades del bot: "Puedo ayudarle con el alquiler de automóviles y las reservas de hotel", "Puede alquilar automóviles y hacer reservas de hotel" y "Puedo ayudarle a alquilar un automóvil y a reservar un hotel".

Lex utiliza un mensaje de cada uno de los grupos de mensajes para crear de forma dinámica las respuestas en una conversación. Por ejemplo, una posible interacción puede ser:

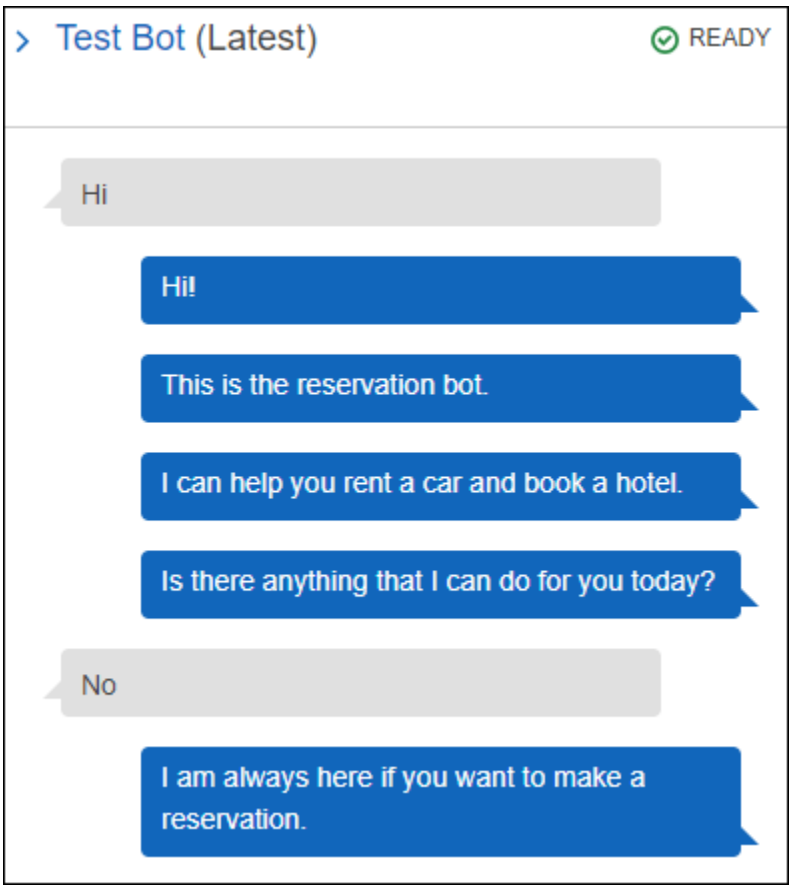


Otra posible interacción puede ser la siguiente:



En cualquier caso, el usuario podría responder con una nueva intención, como la intención `BookCar` o `BookHotel`.

Puede configurar el bot para que formule una pregunta de seguimiento en la respuesta. Por ejemplo, en el caso de la interacción anterior, podría crear un cuarto grupo de mensajes con las siguientes preguntas: "¿Puedo ayudarle con un automóvil o un hotel?", "¿Desea realizar una reserva ahora?" y "¿Hay algo en lo que pueda ayudarle?". En el caso de los mensajes que incluyen "No" como respuesta, puede crear una pregunta de seguimiento. En la siguiente imagen se muestra un ejemplo:



Para crear una pregunta de seguimiento, seleccione Wait for user reply (Esperar respuesta del usuario). A continuación, escriba el mensaje o los mensajes que desea enviar cuando el usuario diga "No". Al crear una respuesta para usar como pregunta de seguimiento, también debe especificar una afirmación adecuada cuando la respuesta a la declaración sea "No". Para ver un ejemplo, consulte la siguiente imagen:

Para añadir respuestas a una intención con la API, utilice la operación `PutIntent`. Para especificar una respuesta, defina el campo `conclusionStatement` en la solicitud `PutIntent`. Para establecer una pregunta de seguimiento, defina el campo `followUpPrompt` e incluya la declaración que se enviará cuando el usuario diga "No". No se puede establecer los campos `conclusionStatement` y `followUpPrompt` en el mismo intento.

Formatos de mensajes admitidos

Al utilizar la operación PostText, o al utilizar la operación PostContent con el encabezado `Accept` establecido en `text/plain;charset=utf8`, Amazon Lex admite mensajes en los siguientes formatos:

- `PlainText`: el mensaje contiene texto UTF-8 sin formato.
- `SSML`: el mensaje contiene texto con formato para salida de voz.
- `CustomPayload`: el mensaje contiene un formato personalizado que ha creado para el cliente. Puede definir la carga para satisfacer las necesidades de su aplicación.
- `Composite`: el mensaje es una recopilación de mensajes, uno de cada grupo de mensajes. Para obtener más información acerca de los grupos de mensajes, consulte Grupos de mensajes.

De forma predeterminada, Amazon Lex devuelve cualquiera de los mensajes definidos para una determinada pregunta. Por ejemplo, si define cinco mensajes para obtener un valor de ranura, Amazon Lex elige uno de los mensajes de forma aleatoria y lo devuelve al cliente.

Si desea que Amazon Lex devuelva un tipo de mensaje específico para el cliente en una solicitud en tiempo de ejecución, defina el parámetro de solicitud `x-amzn-lex:accept-content-types`. La respuesta se limita a los tipos solicitados. Si hay más de un mensaje del tipo especificado, Amazon Lex devuelve uno al azar. Para obtener más información acerca del encabezado `x-amz-lex:accept-content-types`, consulte Configuración del tipo de respuesta.

Grupos de mensajes

Un grupo de mensajes es un conjunto de respuestas adecuadas para una determinada pregunta. Utilice los grupos de mensajes cuando desee que el bot cree dinámicamente las respuestas en una conversación. Cuando Amazon Lex devuelve una respuesta a la aplicación cliente, elige de forma aleatoria un mensaje de cada grupo. Puede crear un máximo de cinco grupos de mensajes para cada respuesta. Cada grupo puede contener un máximo de cinco mensajes. Si desea obtener ejemplos de cómo crear grupos de mensajes en la consola, consulte Respuestas.

Para crear un grupo de mensajes, puede utilizar la consola o las operaciones PutBot, PutIntent o PutSlotType para asignar un número de grupo a un mensaje. Si no crea un grupo de mensajes, o si crea un único grupo de mensajes, Amazon Lex envía un solo mensaje en el campo `Message`. Las aplicaciones cliente obtienen varios mensajes en una respuesta solo cuando ha creado más de un grupo de mensajes en la consola o cuando crea más de un grupo de mensajes al crear o actualizar una intención con la operación PutIntent.

Cuando Amazon Lex envía un mensaje de un grupo, el campo `Message` de la respuesta contiene un objeto JSON con formato de escape que contiene los mensajes. El siguiente ejemplo muestra el contenido del campo `Message` cuando contiene varios mensajes.

> **Note**
>
> El ejemplo está formateado para facilitar su lectura. Una respuesta no contiene 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\"}
]}
```

Puede establecer el formato de los mensajes. El formato puede ser uno de los siguientes:

- PlainText: el mensaje está en texto UTF-8 sin formato.
- SSML: el mensaje es Speech Synthesis Markup Language (SSML, Lenguaje de marcado de síntesis de voz).
- CustomPayload: el mensaje está en un formato personalizado que ha especificado.

Para controlar el formato de los mensajes que devuelven las operaciones `PostContent` y `PostText` en el campo `Message`, defina el atributo de solicitud `x-amz-lex:accept-content-types`. Por ejemplo, si establece el encabezado en el valor siguiente, solo recibirá mensajes en texto sin formato y SSML en la respuesta:

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

Si solicita un formato de mensaje específico y un grupo de mensajes no contiene ese mensaje con ese formato, obtendrá una excepción `NoUsableMessageException`. Al utilizar un grupo de mensajes para agrupar mensajes por tipo, no uses el encabezado `x-amz-lex:accept-content-types`.

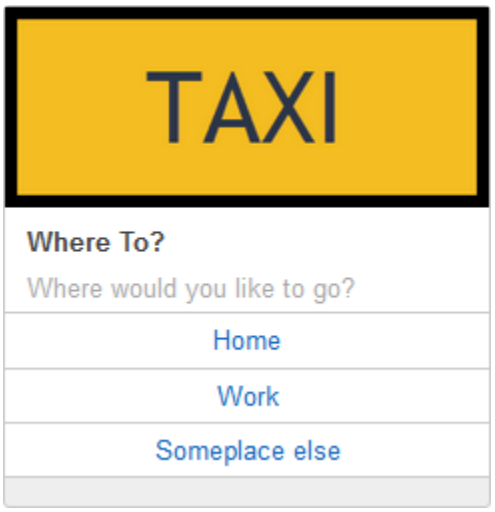
Para obtener más información acerca del encabezado `x-amz-lex:accept-content-types`, consulte Configuración del tipo de respuesta.

Tarjetas de respuesta

> **Note**
>
> Las tarjetas de respuesta no funcionan con el chat de Amazon Connect. Consulte Adición de mensajes interactivos al chat para descubrir una funcionalidad similar.

Una tarjeta de respuesta contiene un conjunto de respuestas adecuadas para una pregunta. Use las tarjetas de respuesta para simplificar las interacciones para los usuarios y aumentar la precisión del bot gracias a la reducción de errores tipográficos en las interacciones de texto. Puede enviar una tarjeta de respuesta para cada mensaje que Amazon Lex envía a la aplicación cliente. Puede utilizar tarjetas de respuesta con Facebook Messenger, Slack, Twilio y sus aplicaciones cliente.

Por ejemplo, en una aplicación de taxi, puede configurar una opción en la tarjeta de respuesta para "Domicilio" y establecer el valor en la dirección postal del usuario. Cuando el usuario seleccione esta opción, Amazon Lex recibirá toda la dirección como la entrada de texto. Vea la siguiente imagen:



Puede definir una tarjeta de respuesta para las siguientes preguntas:

- Instrucción de conclusión
- Pregunta de confirmación
- Pregunta de seguimiento
- Instrucción de rechazo
- Enunciados de tipo de slot

Puede definir una única tarjeta de respuesta por pregunta.

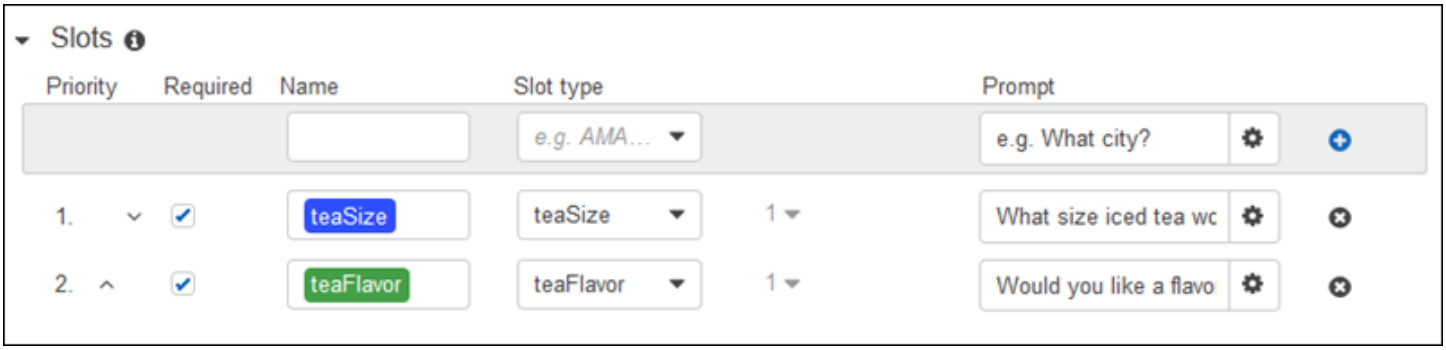
Puede configurar tarjetas de respuesta cuando crea una intención. Puede definir una tarjeta de respuesta estática en tiempo de compilación usando la consola o la operación PutIntent. También puede definir una tarjeta de respuesta dinámica en tiempo de ejecución en una función de Lambda. Si define tarjetas de respuesta estática y dinámica, la tarjeta de respuesta dinámica tiene preferencia.

Amazon Lex envía tarjetas de respuesta en el formato que el cliente comprende. Transforma las tarjetas de respuesta para Facebook Messenger, Slack y Twilio. Para otros clientes, Amazon Lex envía una estructura JSON en la respuesta PostText. Por ejemplo, si el cliente es Facebook Messenger, Amazon Lex transforma la tarjeta de respuesta en una plantilla genérica. Para obtener más información sobre las plantillas genéricas de Facebook Messenger, consulte Generic Template en el sitio web de Facebook. Para ver un ejemplo de la estructura del JSON, consulte Generación dinámica de tarjetas de respuesta.

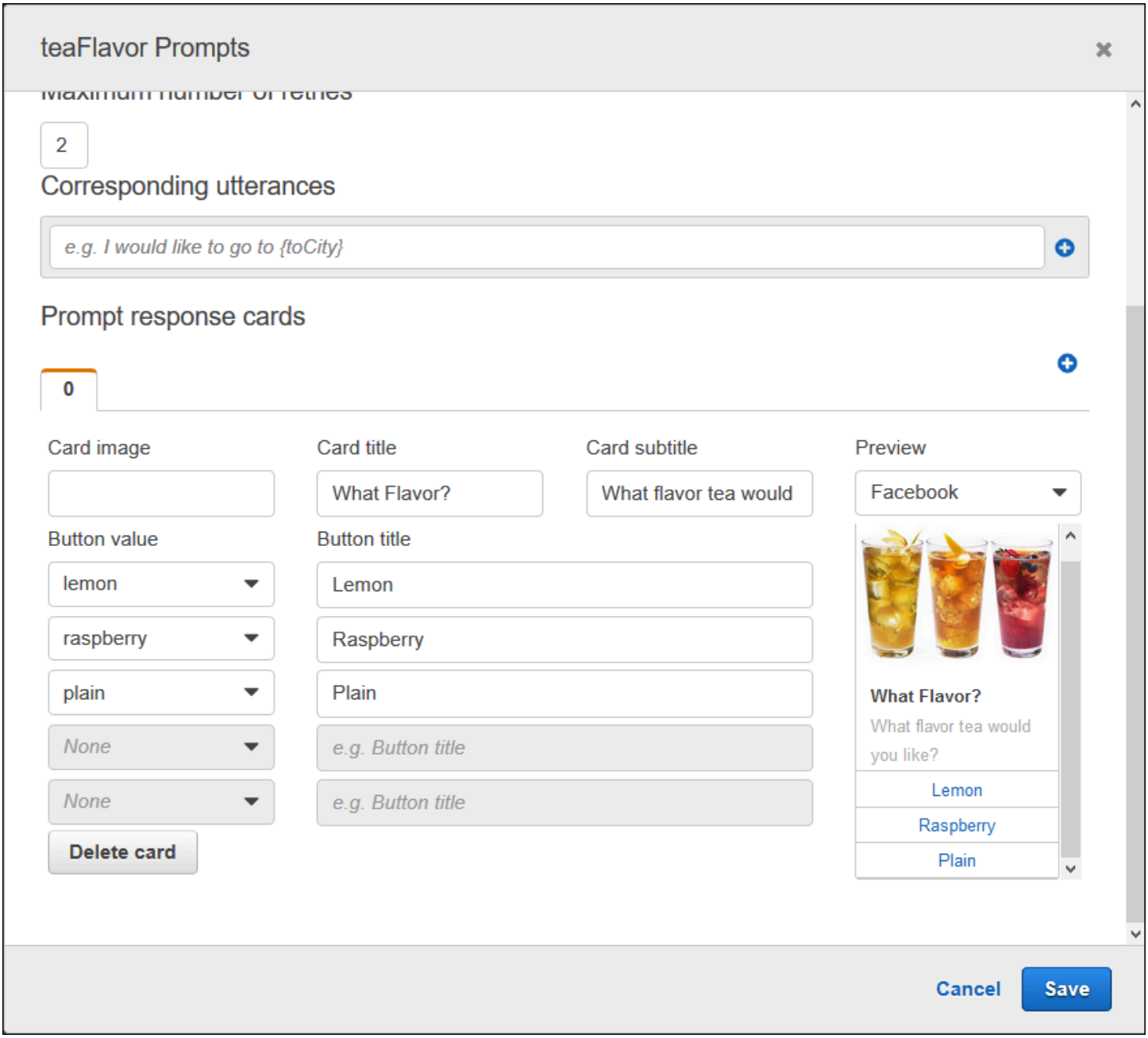
Puede utilizar tarjetas de respuesta solo con la operación PostText. No puede utilizar tarjetas de respuesta solo con la operación PostContent.

Definición de tarjetas de respuesta estática

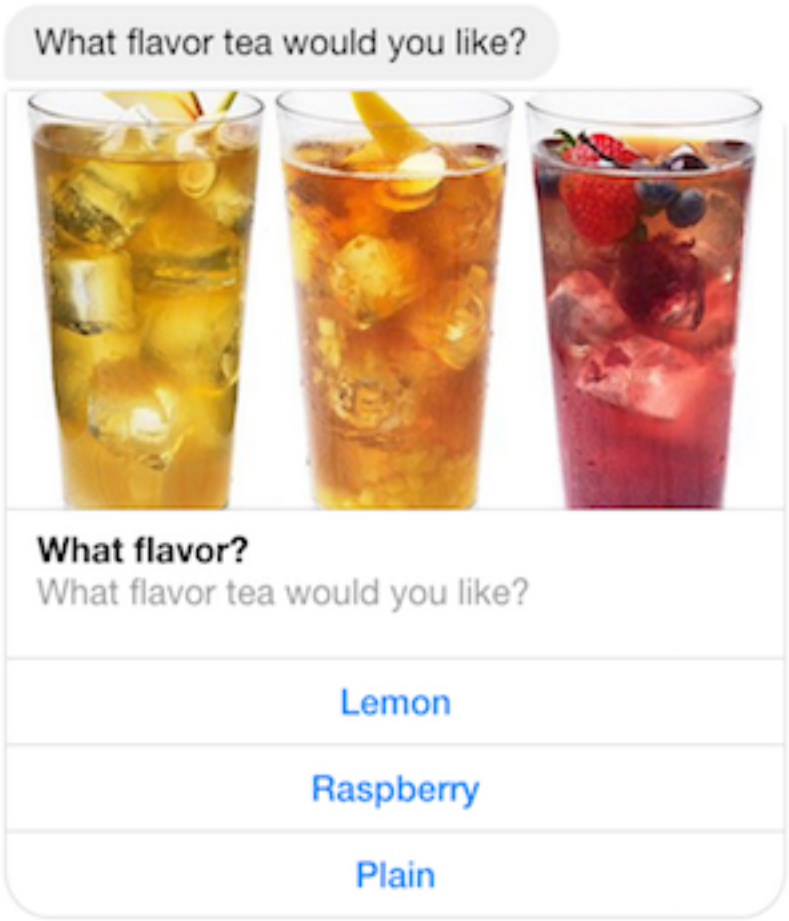
Para definir tarjetas de respuesta estática, use la operación PutBot o la consola de Amazon Lex cuando cree una intención. Una tarjeta de respuesta estática se define al mismo tiempo que las intenciones. Utilice una tarjeta de respuesta estática cuando las respuestas son fijas. Supongamos que está creando un bot con una intención que tiene un slot para sabor. Al definir el slot de sabor, debe especificar preguntas, como se muestra en la siguiente captura de pantalla de la consola:



Al definir las preguntas, puede asociar de manera opcional una tarjeta de respuesta y definir detalles con la operación PutBot o en la consola de Amazon Lex, tal como se muestra en el siguiente ejemplo:



Ahora imagine que ha integrado el bot con Facebook Messenger. El usuario puede hacer clic en los botones para elegir un sabor, tal como se muestra en la siguiente ilustración:



Para personalizar el contenido de una tarjeta de respuesta, puede hacer referencia a los atributos de sesión. En tiempo de ejecución, Amazon Lex sustituye estas referencias por los valores adecuados de los atributos de la sesión. Para obtener más información, consulte Definición de atributos de la sesión. Para ver un ejemplo, consulte Uso de una tarjeta de respuesta.

Generación dinámica de tarjetas de respuesta

Para generar tarjetas de respuesta dinámicamente en tiempo de ejecución, utilice la función de Lambda de inicialización y validación para la intención. Utilice una tarjeta de respuesta dinámica cuando las respuestas se determinan en tiempo de ejecución en la función de Lambda. En respuesta a las acciones de los usuarios, la función de Lambda genera una tarjeta de respuesta y la devuelve en la sección `dialogAction` de la respuesta. Para obtener más información, consulte Formato de respuesta.

A continuación se ofrece una respuesta parcial de una función de Lambda que muestra el elemento `responseCard`. Se genera una experiencia de usuario similar a la que se muestra en la sección 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 ver un ejemplo, consulte Programación de citas.

Gestión del contexto de la conversación

El contexto de la conversación es la información que un usuario, la aplicación o una función de Lambda proporciona a un bot de Amazon Lex para cumplir una intención. El contexto de conversación incluye datos de slot que el usuario proporciona, atributos de solicitud definidos por la aplicación cliente y atributos de sesión que crean la aplicación cliente y las funciones de Lambda.

Temas

- Establecimiento del contexto de la intención
- Uso de valores de ranuras predeterminados

- Definición de atributos de la sesión
- Definición de los atributos de solicitud
- Definición del tiempo de espera de la sesión
- Compartir información entre intenciones
- Definición de atributos complejos

Establecimiento del contexto de la intención

Puede hacer que Amazon Lex active las intenciones en función del contexto. Un contexto es una variable de estado que se puede asociar a una intención al definir un bot.

Los contextos de una intención se configuran cuando se crea la intención mediante la consola o mediante la operación PutIntent. Solo puede utilizar contextos en la configuración regional en inglés (EE. UU.) (en-US), y eso únicamente si ha establecido el parámetro `enableModelImprovements` en `true` al crear el bot con la operación PutBot.

Hay dos tipos de relaciones para los contextos: los contextos de salida y los contextos de entrada. Un contexto de salida se activa cuando se cumple una intención asociada. Se devuelve un contexto de salida a la aplicación en respuesta de la operación PostText o PostContent y se establece para la sesión actual. Una vez activado un contexto, permanece activo durante el número de turnos o el límite de tiempo configurados cuando se definió el contexto.

Un contexto de entrada especifica las condiciones en las que se puede reconocer una intención. Una intención solo se puede reconocer durante una conversación cuando todos sus contextos de entrada están activos. Una intención sin contextos de entrada siempre es apta para el reconocimiento.

Amazon Lex administra automáticamente el ciclo de vida de los contextos que se activan al cumplir las intenciones con los contextos de salida. También puede configurar los contextos activos en una llamada a la operación `PostContent` o `PostText`.

También puede establecer el contexto de una conversación con la función de Lambda para la intención. El contexto de salida de Amazon Lex se envía al evento de entrada de la función de Lambda. La función de Lambda puede enviar contextos en su respuesta. Para obtener más información, consulte Formato del evento de entrada y de la respuesta de la función de Lambda.

Por ejemplo, supongamos que tiene la intención de reservar un coche de alquiler que está configurado para devolver un contexto de salida denominado «book_car_fulfilled». Cuando se

cumple la intención, Amazon Lex establece la variable de contexto de salida «book_car_fulfilled». Dado que «book_car_fulfilled» es un contexto activo, ahora se tiene en cuenta el reconocimiento de una intención con el contexto «book_car_filled» establecido como contexto de entrada, siempre y cuando un enunciado del usuario se reconozca como un intento de provocar esa intención. Puede usarlo para fines que solo tengan sentido después de reservar un vehículo, como enviar un recibo por correo electrónico o modificar una reserva.

Contexto de salida

Amazon Lex activa los contextos de salida de una intención cuando se cumple la intención. Puede utilizar el contexto de salida para controlar las intenciones aptas para hacer un seguimiento de la intención actual.

Cada contexto tiene una lista de parámetros que se mantienen en la sesión. Los parámetros son los valores de los slots correspondientes a la intención cumplida. Puede utilizar estos parámetros para rellenar previamente los valores de los slots para otros propósitos. Para obtener más información, consulte Uso de valores de ranuras predeterminados.

El contexto de salida se configura cuando crea una intención con la consola o con la operación PutIntent. Puede configurar una intención con más de un contexto de salida. Cuando se cumple la intención, todos los contextos de salida se activan y se devuelven en la respuesta PostText o PostContent.

A continuación se muestra cómo asignar un contexto de salida a una intención mediante la consola.

Cuando define un contexto de salida, también define su tiempo activo, la duración o el número de turnos que el contexto incluye en las respuestas de Amazon Lex. Un turno es una solicitud de su aplicación a Amazon Lex. Una vez transcurrido el número de turnos o el tiempo, el contexto deja de estar activo.

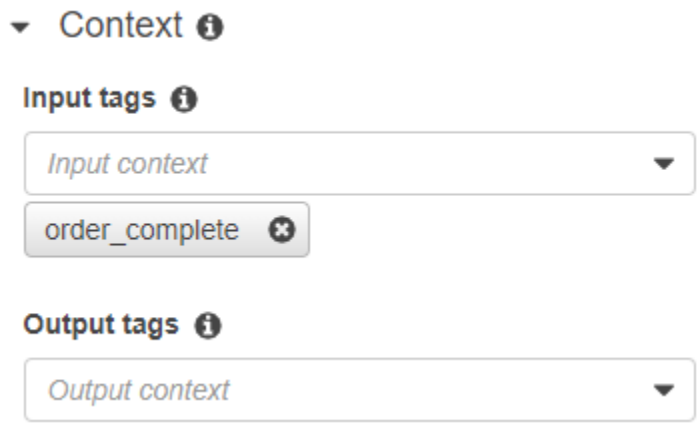
La aplicación puede usar el contexto de salida según sea necesario. Por ejemplo, su aplicación puede usar el contexto de salida para:

- Cambiar el comportamiento de la aplicación en función del contexto. Por ejemplo, una aplicación de viajes podría tener una acción diferente para el contexto «book_car_filled» que para «rental_hotel_fulfilled».
- Devolver el contexto de salida a Amazon Lex como contexto de entrada para el siguiente enunciado. Si Amazon Lex reconoce el enunciado como un intento de obtener una intención, utiliza el contexto para limitar las intenciones que se pueden devolver a las que tienen el contexto especificado.

Contexto de entrada

Establezca un contexto de entrada para limitar los puntos de la conversación en los que se reconoce la intención. Las intenciones sin un contexto de entrada siempre son aptas para ser reconocidas.

Los contextos de entrada a los que responde una intención se establecen mediante la consola o la operación `PutIntent`. Una intención puede tener más de un contexto de entrada. A continuación se muestra cómo asignar un contexto de entrada a una intención mediante la consola.



En el caso de una intención con más de un contexto de entrada, todos los contextos deben estar activos para activar la intención. Puede establecer un contexto de entrada al llamar a la operación PostText, PostContent o PutSession.

Puede configurar los slots con la intención de tomar los valores predeterminados del contexto activo actual. Los valores predeterminados se utilizan cuando Amazon Lex reconoce una nueva intención, pero no recibe un valor de slot. Al definir el slot, debe especificar el nombre del contexto y el nombre

del slot en el formulario `#context-name.parameter-name`. Para obtener más información, consulte Uso de valores de ranuras predeterminados.

Uso de valores de ranuras predeterminados

Cuando utiliza un valor por defecto, se especifica una fuente para que el valor de un slot se rellene con nuevas intenciones cuando la entrada del usuario no proporciona ningún slot. Esta fuente puede ser un cuadro de diálogo anterior, un atributo de una solicitud o una sesión, o un valor fijo que se establezca en el momento de la compilación.

Puede utilizar lo siguiente como origen de los valores por defecto.

- Diálogo anterior (contextos): #context -name.parameter-name
- Atributos de sesión: [nombre-atributo]
- Atributos de solicitud: <attribute-name>
- Valor fijo: cualquier valor que no coincida con el anterior

Cuando se utiliza la operación PutIntent para agregar ranuras a una intención, se puede agregar una lista de valores predeterminados. Los valores por defecto se utilizan en el orden en el que se muestran. Por ejemplo, supongamos que tiene una intención con un slot con la siguiente definición:

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

Cuando se reconoce la intención, el slot denominado «reservation-start-date» tiene su valor establecido en uno de los siguientes valores.

1. Si el contexto «book-car-filled» está activo, el valor del parámetro «StartDate» se utiliza como valor predeterminado.
2. Si el contexto «book-car-filled» no está activo o si el valor del parámetro «StartDate» no está establecido, el valor del atributo de la sesión «reservationStartDate» se utiliza como valor predeterminado.
3. Si no se utiliza ninguno de los dos primeros valores predeterminados, el slot no tiene un valor predeterminado y Amazon Lex obtendrá un valor como de costumbre.

Si se utiliza un valor predeterminado para el slot, el slot no se obtiene aunque sea necesario.

Definición de atributos de la sesión

Los atributos de sesión contienen información específica de la aplicación que se transfiere entre un bot y una aplicación cliente durante una sesión. Amazon Lex V2 pasa los atributos de sesión a todas las funciones de Lambda para un bot. Si una función de Lambda añade o actualiza los atributos de sesión, Amazon Lex devuelve la nueva información a la aplicación cliente. Por ejemplo:

- En Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola), el bot de ejemplo utiliza el atributo de la sesión `price` para mantener el precio de las flores. La función de Lambda define este atributo en función del tipo de flores que se han encargado. Para obtener más información, consulte Paso 5 (opcional): revisión de los detalles del flujo de información (consola).
- En Reserva de viaje, el bot de ejemplo utiliza la sesión del atributo `currentReservation` para mantener una copia de los datos del tipo de slot durante la conversación para reservar un hotel o un coche de alquiler. Para obtener más información, consulte Detalles del flujo de información.

Utilice los atributos de sesión en sus funciones de Lambda para inicializar un bot y personalizar las solicitudes y las tarjetas de respuesta. Por ejemplo:

- Inicialización: en un bot para pedir pizzas, la aplicación cliente transmite la ubicación del usuario como un atributo de la sesión en la primera llamada a la operación PostContent o PostText. Por ejemplo, `"Location": "111 Maple Street"`. La función de Lambda utiliza esta información para encontrar la pizzería más cercana para realizar el pedido.
- Personalización de solicitudes: configure solicitudes y tarjetas de respuesta que hagan referencia a los atributos de la sesión. Por ejemplo, «Hola, [Nombre], ¿qué ingredientes quieres?» Si pasa el nombre del usuario como un atributo de sesión (`{"FirstName": "Jo"}`), Amazon Lex lo

reemplazará donde aparezca el marcador. A continuación, envía una pregunta personalizada al usuario, "Hey Jo, which toppings would you like?"

Los atributos de sesión persisten durante toda la sesión. Amazon Lex los almacena en un almacén de datos cifrados hasta que finaliza la sesión. El cliente puede crear atributos de la sesión en una solicitud llamando a la operación PostContent o PostText con el campo `sessionAttributes` definido en un valor. Una función de Lambda puede crear un atributo de sesión en una respuesta. Una vez que el cliente o una función de Lambda ha creado un atributo de sesión, el valor de atributo almacenado se utiliza siempre que la aplicación cliente no incluya el campo `sessionAttribute` en una solicitud a Amazon Lex.

Por ejemplo, supongamos que tiene dos atributos de sesión, `{"x": "1", "y": "2"}`. Si el cliente llama a la operación `PostContent` o `PostText` sin especificar el campo `sessionAttributes`, Amazon Lex llama a la función de Lambda con los atributos de sesión almacenados (`{"x": 1, "y": 2}`). Si la función de Lambda no devuelve atributos de sesión, Amazon Lex devuelve los atributos de sesión almacenados a la aplicación cliente.

Si la aplicación cliente o una función de Lambda pasan atributos de sesión, Amazon Lex actualiza los atributos de sesión almacenados. Al pasar un valor existente, como por ejemplo `{"x": 2}`, se actualiza el valor almacenado. Si pasa un nuevo conjunto de atributos de sesión, por ejemplo `{"z": 3}`, los valores existentes se eliminan y solo se mantiene el nuevo valor. Cuando se pasa una asignación vacía, `{}`, los valores almacenados se borran.

Para enviar atributos de sesión a Amazon Lex, cree una asignación de cadena a cadena de los atributos. A continuación, se muestra cómo asignar atributos de la sesión:

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

Para la operación `PostText`, debe insertar la asignación en el cuerpo de la solicitud utilizando el campo `sessionAttributes`, de la siguiente manera:

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

Para la operación `PostContent`, hay que codificar en base64 la asignación y luego enviarla como el encabezado de `x-amz-lex-session-attributes`.

Si va a enviar datos binarios o estructurados en un atributo de la sesión, primero debe transformar los datos en una cadena sencilla. Para obtener más información, consulte Definición de atributos complejos.

Definición de los atributos de solicitud

Los atributos de solicitud contienen información específica de solicitud y se aplican únicamente a la solicitud actual. Una aplicación cliente envía esta información a Amazon Lex. Utilice los atributos de solicitud para pasar información que no tiene por qué persistir durante toda la sesión. Puede crear sus propios atributos de solicitud o utilizar atributos predefinidos. Para enviar atributos de solicitud, utilice el encabezado `x-amz-lex-request-attributes` en un campo the section called "PostContent" o `requestAttributes` en una solicitud the section called "PostText". Dado que los atributos de solicitud no persisten en las solicitudes como atributos de la sesión, no se devuelven en las respuestas `PostContent` o `PostText`.

> **Note**
>
> Para enviar información que persiste en las solicitudes, utilice atributos de la sesión.

El espacio de nombres `x-amz-lex:` está reservado a los atributos de solicitud predefinidos. No cree atributos de solicitud con el prefijo `x-amz-lex:`.

Definición de atributos de solicitud predefinidos

Amazon Lex proporciona atributos de solicitud predefinidos para administrar la forma en que procesa la información enviada al bot. Los atributos no persisten en toda la sesión, por lo que debe enviar los atributos predefinidos en cada solicitud. Todos los atributos predefinidos se encuentran en el espacio de nombres `x-amz-lex:`.

Además de los siguientes atributos predefinidos, Amazon Lex también proporciona atributos predefinidos para plataformas de mensajería. Para obtener una lista de dichos atributos, consulte Implementación de un bot de Amazon Lex en una plataforma de mensajería.

Configuración del tipo de respuesta

Si tiene dos aplicaciones cliente que tienen capacidades diferentes, es posible que necesite limitar el formato de los mensajes en una respuesta. Por ejemplo, es recomendable que restrinja los mensajes enviados a un cliente web a texto sin formato, pero permita que un cliente móvil use texto sin formato y Speech Synthesis SSML Markup Language (SSML). Para establecer el formato de los mensajes que devuelven las operaciones PostContent y PostText, use el atributo de solicitud `x-amz-lex:accept-content-types"`.

Puede configurar el atributo en cualquier combinación de los siguientes tipos de mensajes:

- `PlainText`: el mensaje contiene texto UTF-8 sin formato.
- `SSML`: el mensaje contiene texto con formato para salida de voz.
- `CustomPayload`: el mensaje contiene un formato personalizado que ha creado para el cliente. Puede definir la carga para satisfacer las necesidades de su aplicación.

Amazon Lex devuelve únicamente los mensajes con el tipo especificado en el campo `Message` de la respuesta. Puede configurar más de un valor mediante la separación de los valores con una coma. Si utiliza grupos de mensajes, cada uno debe contener al menos un mensaje del tipo especificado. De lo contrario, obtendrá un error `NoUsableMessageException`. Para obtener más información, consulte Grupos de mensajes.

> **Note**
>
> El atributo de solicitud `x-amz-lex:accept-content-types` no modifica el contenido del cuerpo HTML. El contenido de una respuesta de operación `PostText` es siempre de texto UTF-8 sin formato. El cuerpo de una respuesta de operación `PostContent` contiene datos en el formato establecido en el encabezado `Accept` de la solicitud.

Definición de la zona horaria preferida

Para definir la zona horaria utilizada para resolver las fechas de forma que sea relativa a la zona horaria del usuario, utilice el atributo de solicitud `x-amz-lex:time-zone`. Si no especifica una zona horaria en el atributo `x-amz-lex:time-zone`, la opción predeterminada depende de la región que esté utilizando para el bot.

Región	Zona horaria predeterminada
Este de EE. UU. (Norte de Virginia)	`America/New_York`
Oeste de EE. UU. (Oregón)	`America/Los_Angeles`
Asia Pacífico (Singapur)	`Asia/Singapore`
Asia Pacífico (Sídney)	`Australia/Sydney`
Asia-Pacífico (Tokio)	`Asia/Tokyo`
Europa (Fráncfort)	`Europe/Berlin`
Europa (Irlanda)	`Europe/Dublin`
Europa (Londres)	`Europe/London`

Por ejemplo, si el usuario responde `tomorrow` a la pregunta “¿Qué día desea recibir el paquete?”, la fecha en la que se entrega el paquete depende de la zona horaria del usuario. Por ejemplo, cuando es la 01:00 del 16 de septiembre en Nueva York, son las 22:00 del 15 de septiembre en Los Ángeles. Si el servicio se ejecuta en la región Este de EE. UU. (Norte de Virginia) y una persona en Los Ángeles solicita que se entregue un paquete “mañana” utilizando la zona horaria predeterminada, el paquete se entregará el día 17, no el día 16. Sin embargo, si se define el atributo de solicitud `x-amz-lex:time-zone` en `America/Los_Angeles`, el paquete se entregaría el día 16.

Puede definir el atributo en cualquiera de los nombres de zonas horarias de la organización IANA (Internet Assigned Number Authority). Para obtener la lista de nombres de zona horaria, consulte la lista de zonas horarias de la base de datos tz en Wikipedia.

Definición de atributos de solicitud definidos por el usuario

Un atributo de solicitud definido por el usuario es un dato que se envía al bot en cada solicitud. La información se envía en el encabezado de `amz-lex-request-attributes` de una solicitud `PostContent` o en el campo `requestAttributes` de una solicitud `PostText`.

Para enviar atributos de solicitud a Amazon Lex, cree una asignación de cadena a cadena de los atributos. A continuación, se muestra cómo asignar atributos de solicitud:

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

Para la operación `PostText`, debe insertar la asignación en el cuerpo de la solicitud utilizando el campo `requestAttributes`, de la siguiente manera:

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

Para la operación `PostContent`, hay que codificar en base64 la asignación y luego enviarla como el encabezado de `x-amz-lex-request-attributes`.

Si va a enviar datos binarios o estructurados en un atributo de solicitud, primero debe transformar los datos en una cadena sencilla. Para obtener más información, consulte Definición de atributos complejos.

Definición del tiempo de espera de la sesión

Amazon Lex conserva la información contextual (los datos de slot y los atributos de sesión), hasta que la sesión de la conversación finaliza. Para controlar el tiempo que dura una sesión para un bot, defina el tiempo de espera de la sesión. De forma predeterminada, la duración de la sesión es de 5 minutos, pero puede especificar cualquier duración entre 0 y 1440 minutos (24 horas).

Por ejemplo, suponga que crea un bot `ShoeOrdering` que admite intenciones como `OrderShoes` y `GetOrderStatus`. Cuando Amazon Lex detecta que la intención del usuario es comprar zapatos, solicita datos de slot. Por ejemplo, pregunta la talla, el color, la marca, etc. Si el usuario proporciona algunos de los datos de slot, pero no completa la compra de los zapatos, Amazon Lex recuerda todos los datos de slot y los atributos de sesión durante toda la sesión. Si el usuario vuelve a la sesión antes de que venza, puede proporcionar los datos de slot restantes y completar la compra.

En la consola de Amazon Lex, defina el tiempo de espera de la sesión al crear un bot. Con la interfaz de línea de comandos de AWS (CLI de AWS) o la API, se establece el tiempo de espera al crear o actualizar un bot con la operación PutBot definiendo el campo idleSessionTTLInSeconds.

Compartir información entre intenciones

Amazon Lex permite compartir información entre intenciones. Para compartir información entre intenciones, utilice atributos de la sesión.

Por ejemplo, un usuario del bot `ShoeOrdering` empieza el proceso de compra de unos zapatos. El bot comienza una conversación con el usuario para recopilar datos de slot, como la talla, el color y la marca de los zapatos. Cuando el usuario realiza un pedido, la función de Lambda que lleva a cabo el pedido define el atributo de sesión `orderNumber`, que contiene el número de pedido. Para obtener el estado del pedido, el usuario utiliza la intención `GetOrderStatus`. El bot puede pedir al usuario los datos del slot, como el número de pedido y la fecha del pedido. Cuando el bot tiene la información necesaria, devuelve el estado del pedido.

Si cree que los usuarios pueden cambiar de intención durante la misma sesión, puede diseñar su bot para que devuelva el estado del último pedido. En lugar de pedir al usuario de nuevo la información del pedido, se utiliza el atributo de la sesión `orderNumber` para compartir información entre las intenciones y satisfacer la intención `GetOrderStatus`. Para hacer esto, el bot devuelve el estado del último pedido que ha realizado el usuario.

Para ver un ejemplo del intercambio de información entre intenciones, consulte Reserva de viaje.

Definición de atributos complejos

Los atributos de sesión y solicitud son asignaciones de cadena a cadena de atributos y valores. En muchos casos, puede utilizar la asignación de cadenas para transferir valores de atributos entre la aplicación cliente y un bot. En algunos casos, sin embargo, es posible que necesite transferir datos binarios o una estructura compleja que no se puede convertir fácilmente a una asignación de cadenas. Por ejemplo, el siguiente objeto JSON representa una matriz de las tres ciudades más pobladas de los 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"
         }
      }
   ]
}
```

Esta matriz de datos no se convierte bien en una asignación de cadena a cadena. En este caso, puede transformar un objeto en una cadena sencilla para poder enviársela a su bot con las operaciones PostContent y PostText.

Por ejemplo, si utiliza JavaScript, puede utilizar la operación `JSON.stringify` para convertir un objeto a JSON y la operación `JSON.parse` para convertir un texto JSON a un objeto JavaScript:

```
// To convert an object to a string.
var jsonString = JSON.stringify(object, null, 2);
// To convert a string to an object.
var obj = JSON.parse(JSON string);
```

Para enviar los atributos de la sesión con la operación `PostContent`, debe codificar en base64 los atributos antes de incluirlos en el encabezado de la solicitud, tal y como se muestra en el siguiente código JavaScript:

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

Puede enviar datos binarios a las operaciones `PostContent` y `PostText` convirtiendo primero los datos a una cadena codificada en base64 y, a continuación, enviar la cadena como el valor en los atributos de la sesión:

```
"sessionAttributes" : {
```

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

Uso de puntuaciones de confianza

Cuando un usuario hace un enunciado, Amazon Lex utiliza la comprensión del lenguaje natural (NLU) para entender la solicitud del usuario y devolverle la intención correcta. De forma predeterminada, Amazon Lex devuelve la intención más probable definida por su bot.

En algunos casos, puede resultar difícil para Amazon Lex determinar la intención más probable. Por ejemplo, el usuario puede hacer un enunciado ambiguo o puede haber dos intenciones similares. Para ayudar a determinar la intención correcta, puede combinar sus conocimientos del dominio con las puntuaciones de confianza de una lista de intenciones alternativas. Una puntuación de confianza es una calificación que proporciona Amazon Lex y que muestra el grado de confianza en que una intención es la correcta.

Para determinar la diferencia entre dos intenciones alternativas, puede comparar sus puntuaciones de confianza. Por ejemplo, si una intención tiene una puntuación de confianza de 0,95 y otra tiene una puntuación de 0,65, la primera intención probablemente sea correcta. Sin embargo, si una intención tiene una puntuación de 0,75 y otra tiene una puntuación de 0,72, existe una ambigüedad entre las dos intenciones y es posible que pueda discriminar utilizando el conocimiento del dominio en su aplicación.

También puede utilizar las puntuaciones de confianza para crear aplicaciones de prueba que determinen si los cambios en los enunciados de una intención marcan una diferencia en el comportamiento del bot. Por ejemplo, puede obtener las puntuaciones de confianza de las intenciones de un bot utilizando un conjunto de enunciados y, a continuación, actualizar las intenciones con nuevos enunciados. A continuación, puede comprobar las puntuaciones de confianza para ver si se ha producido una mejora.

Las puntuaciones de confianza que devuelve Amazon Lex son valores comparativos. No debe confiar en ellos como puntuación absoluta. Los valores pueden cambiar en función de las mejoras de Amazon Lex.

Cuando utiliza las puntuaciones de confianza, Amazon Lex devuelve la intención más probable y hasta 4 intenciones alternativas con sus puntuaciones asociadas en cada respuesta. Si todas las puntuaciones de confianza son inferiores a un umbral definido, Amazon Lex incluye `AMAZON.FallbackIntent`, `AMAZON.KendraSearchIntent` o ambas, si están configuradas. Puede utilizar el umbral predeterminado o puede definir el suyo propio.

El siguiente código JSON muestra el campo `alternativeIntents` de la respuesta de la operación PostText.

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

Defina el umbral al crear o actualizar el bot. Puede utilizar la API de o la consola de Amazon Lex. En las regiones que se indican a continuación, debe suscribirse para habilitar las mejoras en la precisión y las puntuaciones de confianza. En la consola, elija las puntuaciones de confianza de la sección Opciones avanzadas. Con la API, establezca el parámetro `enableModelImprovements` cuando llame a la operación PutBot.

- Este de EE. UU. (Norte de Virginia) (us-east-1)
- Oeste de EE. UU. (Oregón) (us-west-2)
- Asia Pacífico (Sídney) (ap-southeast-2)
- Europa (Irlanda) (eu-west-1)

En todas las demás regiones, las mejoras en la precisión y la compatibilidad con puntuaciones de confianza están disponibles de forma predeterminada.

Para modificar el umbral de confianza, defínalo en la consola o con la operación PutBot. El umbral debe ser un número entre 1,00 y 0,00.

Para utilizar la consola, defina el umbral de confianza al crear o actualizar el bot.

Definición del umbral de confianza al crear un bot (consola)

- En Crear bot, introduzca un valor en el campo Umbral de puntuación de confianza.

Actualización del umbral de confianza (consola)

1. Seleccione el bot para actualizar de la lista de bots.
2. Elija la pestaña Settings.
3. En el panel de navegación izquierdo, elija General.
4. Actualice el valor del campo Umbral de puntuación de confianza.

Definición o actualización del umbral de confianza (SDK)

- Defina el parámetro `nluIntentConfidenceThreshold` de la operación PutBot. El siguiente código JSON muestra el parámetro que se debe configurar.

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

Administración de sesiones

Para cambiar la intención que Amazon Lex utiliza en una conversación con el usuario, puede utilizar la respuesta del enlace de código del cuadro de diálogo (función de Lambda) o puede utilizar las API de administración de sesiones en su aplicación personalizada.

Usar una URL de función de Lambda

Cuando utiliza una función de Lambda, Amazon Lex la llama con una estructura JSON que contiene la entrada a la función. La estructura JSON contiene un campo denominado `currentIntent` que incluye la intención que Amazon Lex ha identificado como la intención más probable del enunciado del usuario. La estructura JSON también incluye un campo `alternativeIntents` que contiene hasta cuatro intenciones adicionales que pueden satisfacer la intención del usuario. Cada intención incluye un campo denominado `nluIntentConfidenceScore` que contiene la puntuación de confianza que Amazon Lex asignó a la intención.

Para utilizar la intención alternativa, debe especificarla en la `ConfirmIntent` o la acción de diálogo `ElicitSlot` de la función de Lambda.

Para obtener más información, consulte Uso de funciones de Lambda.

Usar la API de administración de sesiones

Para utilizar una intención diferente de la intención actual, utilice la operación PutSession. Por ejemplo, si decide que la primera alternativa es preferible a la intención que eligió Amazon Lex, puede utilizar la operación `PutSession` para cambiar las intenciones para que la siguiente intención con la que interactúe el usuario sea la que usted seleccionó.

Para obtener más información, consulte Administración de sesiones con la API de Amazon Lex.

Registros de conversación

Habilite los registros de conversación para almacenar interacciones de bots. Puede utilizar estos registros para revisar el rendimiento de su bot y solucionar problemas con las conversaciones. Puede registrar texto para la operación PostText. Puede registrar texto y audio para la operación PostContent. Al habilitar los registros de conversación, obtiene una vista detallada de las conversaciones que los usuarios tienen con su bot.

Por ejemplo, una sesión con su bot tiene un ID de sesión. Puede usar este ID para obtener la transcripción de la conversación, incluidos los enunciados del usuario y las respuestas del bot correspondientes. También obtiene metadatos como el nombre de la intención y los valores de slot para un enunciado.

> **Note**
>
> No puede usar registros de conversación con un bot sujeto a la Ley de Protección de la Privacidad en Línea para Niños (COPPA).

Los registros de conversación se configuran para un alias. Cada alias puede tener distintas configuraciones para sus registros de texto y audio. Puede habilitar registros de texto, registros de audio o ambos para cada alias. Los registros de texto almacenan la entrada de texto, las transcripciones de la entrada de audio y los metadatos asociados en Registros de CloudWatch. Los registros de audio almacenan la entrada de audio en Amazon S3. Puede habilitar el cifrado de registros de texto y audio mediante CMK administradas por el cliente de AWS KMS.

Para configurar el registro, utilice la consola o la operación PutBotAlias. No puede registrar conversaciones para el alias `$LATEST` del bot o para el bot de prueba disponible en la consola de Amazon Lex. Después de habilitar los registros de conversaciones para un alias, la operación

PostContent o PostText para dicho alias registra los enunciados de texto o audio en el grupo de registro de Registros de CloudWatch configurado o en el bucket de S3.

Temas

Políticas de IAM para registros de conversación

En función del tipo de registro que seleccione, Amazon Lex requiere permiso para usar Registros de Amazon CloudWatch y los buckets de Amazon Simple Storage Service (S3) para almacenar los registros. Debe crear roles y permisos de AWS Identity and Access Management para permitir a Amazon Lex el acceso a estos recursos.

Creación de un rol de IAM y políticas para registros de conversación

Para habilitar los registros de conversaciones, debe conceder permiso de escritura para Registros de CloudWatch y Amazon S3. Si habilita el cifrado de objetos para los objetos de S3, tiene que conceder permiso de acceso a las claves de AWS KMS utilizadas para cifrar los objetos.

Puede usar la AWS Management Console de IAM, la API de IAM o la AWS Command Line Interface para crear el rol y las políticas. Estas instrucciones utilizan la AWS CLI para crear el rol y las políticas. Para obtener información acerca de la creación de políticas con la consola, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de AWS Identity and Access Management.

> **Note**
>
> El siguiente código tiene formato para Linux y MacOS. Para Windows, reemplace el carácter de continuación de línea de Linux (\) por un signo de intercalación (^).

Crear un rol de IAM para los registros de conversación

1. Cree un documento en el directorio actual llamado **`LexConversationLogsAssumeRolePolicyDocument.json`**, agregue el código siguiente y guárdelo. En este documento de política se agrega a Amazon Lex como entidad de confianza para el rol. Esto permite a Lex asumir el rol para entregar registros a los recursos configurados para registros de conversación.

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

2. En la AWS CLI, ejecute el siguiente comando para crear el rol de IAM para los registros de conversación.

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

A continuación, cree y asocie una política al rol que permite a Amazon Lex escribir en Registros de CloudWatch.

Creación de una política de IAM para registrar texto de conversación en Registros de CloudWatch

1. Cree un documento en el directorio actual llamado **`LexConversationLogsCloudWatchLogsPolicy.json`**, agregue la siguiente política de IAM y guárdelo.

```
{
  "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. En la AWS CLI, cree la política de IAM que concede permiso de escritura al grupo de registro de Registros de CloudWatch.

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

3. Asocie la política al rol de IAM que creó para los registros de conversación.

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

Si está registrando audio en un bucket de S3, cree una política que permita a Amazon Lex escribir en el bucket.

Crear una política de IAM para el registro de audio en un bucket de S3

1. Cree un documento en el directorio actual llamado **`LexConversationLogsS3Policy.json`**, agregue la siguiente política y guárdelo.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:PutObject"
          ],
```

```
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

2. En la AWS CLI, cree la política de IAM que concede permiso de escritura a su bucket de S3.

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

3. Asocie la política al rol que creó para los registros de conversación.

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

Conceder permisos para pasar un rol de IAM

Cuando se utiliza la consola, la AWS Command Line Interface o un SDK de AWS para especificar un rol de IAM que se va a utilizar para los registros de conversaciones, el usuario que especifique el rol de IAM para los registros de conversación debe tener permiso para pasar el rol a Amazon Lex. Para permitir que el usuario pase el rol a Amazon Lex, debe conceder el permiso `PassRole` al usuario, rol o grupo.

La política siguiente define el permiso para conceder al usuario, rol o grupo. Puede utilizar las claves de condición `iam:AssociatedResourceArn` y `iam:PassedToService` para limitar el alcance del permiso. Para obtener más información, consulte Concesión de permisos de usuario para pasar un rol a un servicio de AWS y Claves de contexto de condición de AWS STS en la Guía del usuario de AWS Identity and Access Management.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/role-name",
            "Condition": {
                "StringEquals": {
```

```
                    "iam:PassedToService": "lex.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": "arn:aws:lex:region:account-
id:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

Configuración de registros de conversación

Los registros de conversación se habilitan y deshabilitan mediante la consola o el campo `conversationLogs` de la operación `PutBotAlias`. Puede activar o desactivar los registros de audio, los registros de texto o ambos. El registro comienza en las nuevas sesiones de bot. Los cambios en la configuración del registro no se reflejan en las sesiones activas.

Para almacenar registros de texto, utilice un grupo de registro de Registros de Amazon CloudWatch en su cuenta de AWS. Puede utilizar cualquier grupo de registro válido. El grupo de registro debe estar en la misma región que el bot de Amazon Lex. Para obtener más información sobre la creación de un grupo de registro de Registros de CloudWatch, consulte Trabajar con grupos de registro y flujos de registro en la Guía del usuario de Registros de Amazon CloudWatch.

Para almacenar registros de audio, use un bucket de S3 en su cuenta de AWS. Puede utilizar cualquier bucket de S3 válido. El bucket debe estar en la misma región que el bot de Amazon Lex. Para obtener más información acerca de la creación de un bucket de S3, consulte Creación de un bucket en la Guía de introducción de Amazon Simple Storage Service.

Debe proporcionar un rol de IAM con políticas que permitan a Amazon Lex escribir en el grupo de registro o en el bucket configurado. Para obtener más información, consulte Creación de un rol de IAM y políticas para registros de conversación.

Si crea un rol vinculado a servicios mediante la AWS Command Line Interface, debe agregar un sufijo personalizado al rol mediante la opción `custom-suffix`, tal como se indica a continuación:

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

El rol de IAM que utilice para habilitar los registros de las conversiones debe tener el permiso `iam:PassRole`. La siguiente política debe estar asociada al rol.

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

Habilitación de registros de conversación

Para activar los registros mediante la consola

1. Abra la consola de Amazon Lex en https://console.aws.amazon.com/lex.
2. En la lista, elija un bot.
3. Elija la pestaña Settings (Configuración) y, a continuación, en el menú izquierdo elija Conversation logs (Registros de conversación).
4. En la lista de alias, elija el icono de configuración del alias para el que desea configurar los registros de conversación.
5. Seleccione si desea registrar texto, audio o ambos.
6. Para los registros de texto, introduzca el nombre del grupo de registro de Registros de Amazon CloudWatch.
7. Para el registro de audio, introduzca la información del bucket de S3.
8. Opcional. Para cifrar los registros de audio, elija la clave de AWS KMS que desea utilizar para el cifrado.
9. Elija un rol de IAM con los permisos necesarios.
10. Elija Save (Guardar) para iniciar el registro de conversaciones.

Para activar registros de texto mediante la API

1. Llame a la operación PutBotAlias con una entrada en el miembro `logSettings` del campo `conversationLogs`

- Establezca el miembro `destination` en `CLOUDWATCH_LOGS`
- Establezca el miembro `logType` en `TEXT`
- Establezca el miembro `resourceArn` en el nombre de recurso de Amazon (ARN) del grupo de registro de Registros de CloudWatch que es el destino de los registros

2. Establezca el miembro `iamRoleArn` del campo `conversationLogs` en el nombre de recurso de Amazon (ARN) de un rol de IAM que tenga los permisos necesarios para habilitar los registros de conversaciones en los recursos especificados.

Para activar los registros de audio mediante la API

1. Llame a la operación PutBotAlias con una entrada en el miembro `logSettings` del campo `conversationLogs`

   - Establezca el miembro `destination` en `S3`
   - Establezca el miembro `logType` en `AUDIO`
   - Establezca el miembro `resourceArn` en el ARN del bucket de Amazon S3 donde se almacenan los registros de audio
   - Opcional. Para cifrar registros de audio con una clave de AWS KMS específica, establezca el miembro `kmsKeyArn` del ARN de la clave que se utiliza para el cifrado.

2. Establezca el miembro `iamRoleArn` del campo `conversationLogs` en el nombre de recurso de Amazon (ARN) de un rol de IAM que tenga los permisos necesarios para habilitar los registros de conversaciones en los recursos especificados.

Deshabilitación de registros de conversación

Para desactivar los registros mediante la consola

1. Abra la consola de Amazon Lex en https://console.aws.amazon.com/lex.
2. En la lista, elija un bot.
3. Elija la pestaña Settings (Configuración) y, a continuación, en el menú izquierdo elija Conversation logs (Registros de conversación).
4. En la lista de alias, elija el icono de configuración del alias para el que desea configurar los registros de conversación.
5. Quite la marca de la casilla de verificación de texto, audio o ambas para desactivar el registro.

6. Elija Save (Guardar) para detener el registro de conversaciones.

Para desactivar los registros mediante la API

- Llame a la operación `PutBotAlias` sin el campo `conversationLogs`.

Para desactivar los registros de texto mediante la API

- • Si está registrando audio
    - Llame a la operación PutBotAlias con una entrada `logSettings` solo para `AUDIO`.
    - La llamada a la operación `PutBotAlias` no debe tener una entrada `logSettings` para `TEXT`.
  - Si no está registrando audio
    - Llame a la operación PutBotAlias sin el campo `conversationLogs`.

Para desactivar los registros de audio mediante la API

- • Si está registrando texto
    - Llame a la operación PutBotAlias con una entrada `logSettings` solo para `TEXT`.
    - La llamada a la operación `PutBotAlias` no debe tener una entrada `logSettings` para `AUDIO`.
  - Si no está registrando texto
    - Llame a la operación PutBotAlias sin el campo `conversationLogs`.

Cifrado de registros de conversación

Puede usar el cifrado para ayudar a proteger el contenido de los registros de conversación. Para los registros de texto y audio, puede utilizar CMK administradas por el cliente de AWS KMS para cifrar los datos del grupo de registro de Registros de CloudWatch y del bucket de S3.

> **Note**
>
> Amazon Lex solo admite CMK simétricas. No puede utilizar una CMK asimétrica para cifrar sus datos.

Habilite el cifrado mediante una clave de AWS KMS en el grupo de registro de Registros de CloudWatch que utiliza Amazon Lex para los registros de texto. No puede proporcionar una clave de AWS KMS en la configuración del registro para habilitar el cifrado de AWS KMS de su grupo de registros. Para obtener más información, consulte Cifrado de datos de registro en Registros de CloudWatch con AWS KMS en la Guía del usuario de Registros de Amazon CloudWatch.

Para los registros de audio se utiliza el cifrado predeterminado en el bucket de S3 o se especifica una clave de AWS KMS para cifrar los objetos de audio. Incluso si el bucket de S3 utiliza el cifrado predeterminado, puede especificar una clave de AWS KMS diferente para cifrar los objetos de audio. Para obtener más información, consulte Cifrado predeterminado de Amazon S3 para buckets de S3 en la Guía para desarrolladores de Amazon Simple Storage Service.

Amazon Lex requiere permisos de AWS KMS si elige cifrar los registros de audio. Debe asociar políticas adicionales al rol de IAM que se utiliza en los registros de conversaciones. Si utiliza el cifrado predeterminado en el bucket de S3, la política debe conceder acceso a la clave de AWS KMS configurada para ese bucket. Si especifica una clave de AWS KMS en la configuración del registro de audio, debe conceder acceso a esa clave.

Si no ha creado un rol para los registros de conversación, consulte Políticas de IAM para registros de conversación.

Creación de una política de IAM para utilizar una clave de AWS KMS en el cifrado de registros de audio

1. Cree un documento en el directorio actual llamado **`LexConversationLogsKMSPolicy.json`**, agregue la siguiente política y guárdelo.

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

2. En la AWS CLI, cree la política de IAM que concede permiso para utilizar la clave AWS KMS en el cifrado de registros de audio.

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

3. Asocie la política al rol que creó para los registros de conversación.

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

Visualización de registros de texto en Registros de Amazon CloudWatch

Amazon Lex almacena los registros de texto de sus conversaciones en Registros de Amazon CloudWatch. Para ver los registros, puede utilizar la consola o la API de Registros de CloudWatch. Para obtener más información, consulte Búsqueda de datos de registro mediante patrones de filtro y Sintaxis de consultas de CloudWatch Logs Insights en la Guía del usuario de Registros de Amazon CloudWatch.

Visualización de registros mediante la consola de Amazon Lex

1. Abra la consola de Amazon Lex en https://console.aws.amazon.com/lex.
2. En la lista, elija un bot.
3. Elija la pestaña Settings (Configuración), a continuación, en el menú izquierdo elija Conversation logs (Registros de conversación).
4. Elija el enlace situado debajo de Registros de texto para ver los registros del alias en la consola de CloudWatch.

También puede utilizar la consola de CloudWatch o la API para ver las entradas de registro. Para buscar las entradas de registro, desplácese hasta el grupo de registros que configuró para el alias. Encontrará el prefijo de flujo de registro para los registros en la consola de Amazon Lex o con la operación GetBotAlias.

Las entradas de registro para un enunciado de usuario se encuentran en varios flujos de registro. Un enunciado en la conversación tiene una entrada en uno de los flujos de registro con el prefijo especificado. Una entrada en el flujo de registro contiene la siguiente información.

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

El contenido de la entrada de registro depende del resultado de una transacción y de la configuración del bot y la solicitud.

- Los campos `intent`, `slots` y `slotToElicit` no aparecen en una entrada si el campo `missedUtterance` es `true`.
- El campo `s3PathForAudio` no aparece si los registros de audio están deshabilitados o si el campo `inputDialogMode` es `Text`.
- El campo `responseCard` solo aparece cuando se ha definido una tarjeta de respuesta para el bot.
- El mapa `requestAttributes` solo aparece si ha especificado atributos de solicitud en la solicitud.

- El campo `kendraResponse` solo está presente cuando `AMAZON.KendraSearchIntent` realiza una solicitud para buscar en un índice de Amazon Kendra.
- El campo `developerOverride` es verdadero cuando se especificó una intención alternativa en la función de Lambda del bot.
- El mapa `sessionAttributes` solo aparece si ha especificado atributos de sesión en la solicitud.
- El mapa `sentimentResponse` solo aparece si configura el bot para que devuelva valores de opinión.

> **Note**
>
> El formato de entrada puede cambiar sin un cambio correspondiente en `messageVersion`. El código no debería devolver un error si hay nuevos campos.

Debe tener un rol y una política establecidos para habilitar que Amazon Lex escriba en Registros de CloudWatch. Para obtener más información, consulte Políticas de IAM para registros de conversación.

Acceso a los registros de audio en Amazon S3

Amazon Lex almacena registros de audio de sus conversaciones en un bucket de S3.

Para acceder a los registros de audio mediante la consola

1. Abra la consola de Amazon Lex en https://console.aws.amazon.com/lex.
2. En la lista, elija un bot.
3. Elija la pestaña Settings (Configuración), a continuación, en el menú izquierdo elija Conversation logs (Registros de conversación).
4. Elija el enlace situado debajo de Registros de audio para acceder a los registros del alias en la consola de Amazon S3.

También puede usar la consola de Amazon S3 o la API para acceder a los registros de audio. Puede ver el prefijo de clave de objeto de S3 de los archivos de audio en la consola de Amazon Lex o en el campo `resourcePrefix` en la respuesta de la operación `GetBotAlias`.

Supervisión del estado de un registro de conversaciones con métricas de CloudWatch

Utilice Amazon CloudWatch para monitorear las métricas de entrega de sus registros de conversación. Puede establecer alarmas en las métricas de modo que esté al tanto de los problemas de registro si se producen.

Amazon Lex proporciona cuatro métricas en el espacio de nombres `AWS/Lex` para los registros de conversaciones:

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

Para obtener más información, consulte Métricas de CloudWatch para registros de conversaciones.

Las métricas de éxito muestran que Amazon Lex ha escrito correctamente los registros de audio o texto en sus respectivos destinos.

Las métricas de error muestran que Amazon Lex no pudo entregar registros de audio o texto en el destino especificado. Normalmente, se trata de un error de configuración. Cuando las métricas de error estén por encima de cero, compruebe lo siguiente:

- Asegúrese de que Amazon Lex es una entidad de confianza para el rol de IAM.
- Para el registro de texto, asegúrese de que existe el grupo de registro de Registros de CloudWatch. Para el registro de audio, asegúrese de que existe el bucket S3.
- Asegúrese de que el rol de IAM que Amazon Lex utiliza para tener acceso al grupo de registro de Registros de CloudWatch o al bucket de S3 tiene permiso de escritura para el grupo de registro o el bucket.
- Asegúrese de que el bucket de S3 existe en la misma región que el bot de Amazon Lex y de que pertenece a su cuenta.
- Si está utilizando una clave de AWS KMS para el cifrado de S3, asegúrese de que no haya políticas que impidan a Amazon Lex usar la clave y asegúrese de que el rol de IAM que proporcione tenga los permisos de AWS KMS necesarios. Para obtener más información, consulte Políticas de IAM para registros de conversación.

Administración de sesiones con la API de Amazon Lex

Cuando un usuario inicia una conversación con un bot, Amazon Lex crea una sesión. La información que se intercambia entre la aplicación y Amazon Lex conforma el estado de la sesión de la conversación. Cuando realiza una solicitud, la sesión se identifica utilizando el nombre del bot y el identificador de usuario que especifique. Para obtener más información sobre el identificador de usuario, consulte el campo `userId` de la operación PostText o PostContent.

La respuesta de una operación de sesión incluye un identificador de sesión único que identifica una sesión específica con un usuario. Puede utilizar este identificador durante las pruebas o para ayudar a resolver los problemas del bot.

Puede modificar el estado de la sesión enviado entre la aplicación y el bot. Por ejemplo, puede crear y modificar atributos que contengan información personalizada sobre la sesión y cambiar el flujo de la conversación estableciendo el contexto del diálogo para poder interpretar el siguiente enunciado.

Hay dos formas de actualizar el estado de la sesión. La primera consiste en utilizar una función de Lambda con la operación `PostContent` o `PostText` que se invoca después de cada turno de la conversación. Para obtener más información, consulte Uso de funciones de Lambda. La otra es utilizar la API en tiempo de ejecución de Amazon Lex en la aplicación para realizar cambios en el estado de la sesión.

La API en tiempo de ejecución de Amazon Lex dispone de operaciones que le permiten administrar la información de la sesión en una conversación con el bot. Las operaciones son PutSession, GetSession y DeleteSession. Puede utilizar estas operaciones para obtener información sobre el estado de la sesión del usuario con el bot y para tener un control pormenorizado sobre dicho estado.

Utilice la operación `GetSession` cuando desee obtener el estado actual de la sesión. La operación devuelve el estado actual de la sesión, incluido el estado del diálogo con el usuario, los atributos de la sesión que se hayan establecido y los valores de slot de las tres últimas intenciones con las que interactuó el usuario.

La operación `PutSession` le permite manipular directamente la sesión actual. Puede definir el tipo de acción de diálogo que el bot realizará a continuación. De este modo, tendrá control sobre el flujo de la conversación con el bot. Establezca el campo `type` de la acción de diálogo en `Delegate` para que Amazon Lex determine la siguiente acción del bot.

Puede utilizar la operación `PutSession` para crear una nueva sesión con un bot y establecer la intención con la que el bot debería comenzar. También puede usar la operación `PutSession`

para cambiar de una intención a otra. Al crear una sesión o cambiar la intención, también puede establecer el estado de sesión; por ejemplo, los valores de slot y los atributos de sesión. Cuando la nueva intención haya terminado, tendrá la posibilidad de reiniciar la intención anterior. Puede utilizar la operación `GetSession` para obtener el estado de la intención anterior del diálogo de Amazon Lex y utilizar esta información para establecer el estado de la intención en el diálogo.

La respuesta de la operación `PutSession` contiene la misma información que la operación `PostContent`. Puede utilizar esta información para preguntar al usuario por la siguiente información, igual que haría con la respuesta de la operación `PostContent`.

Utilice la operación `DeleteSession` para eliminar una sesión existente y comenzar de nuevo con una nueva sesión. Por ejemplo, si va a probar un bot, puede utilizar la operación `DeleteSession` para eliminar las sesiones de prueba desde el bot.

Las operaciones de sesión pueden utilizarse con las funciones de Lambda de realización. Por ejemplo, si la función de Lambda devuelve `Failed` como estado de realización, puede utilizar la operación `PutSession` para establecer el tipo de acción del diálogo en `close` y `fulfillmentState` en `ReadyForFulfillment` para intentar de nuevo el paso de cumplimiento.

Estas son algunas de las cosas que puede hacer con las operaciones de sesión:

- Hacer que el bot inicie una conversación en lugar de esperar al usuario.
- Cambiar las intenciones durante una conversación.
- Volver a una intención anterior.
- Iniciar o reiniciar una conversación en mitad de la interacción.
- Validar los valores de slot y hacer que el bot vuelva a solicitar los valores que no son válidos.

Cada una de acciones se describe a continuación.

Cambio de intenciones

Puede utilizar la operación `PutSession` para cambiar de una intención a otra. También puede utilizar esta operación para volver a una intención anterior. Puede utilizar la operación `PutSession` para establecer los atributos de sesión o los valores de slot de la nueva intención.

- Llame a la operación `PutSession`. Especifique el nombre de la intención y establezca la acción del diálogo en `Delegate`. También puede definir los atributos de sesión o los valores de slot necesarios para la nueva intención.

- Amazon Lex comenzará una conversación con el usuario utilizando la nueva intención.

Reanudación de una intención anterior

Si desea reanudar una intención anterior, utilice la operación `GetSession` para obtener un resumen de la intención y utilice después la operación `PutSession` para establecer la intención en el estado del diálogo anterior.

- Llame a la operación `GetSession`. La respuesta de la operación incluye un resumen del estado de las últimas tres intenciones del diálogo con las que interactuó el usuario.
- Utilizando la información del resumen de intenciones, invoque la operación `PutSession`. De este modo, devolverá al usuario a la intención anterior en el mismo lugar de la conversación.

En algunos casos puede ser necesario reanudar la conversación del usuario con el bot. Por ejemplo, supongamos que ha creado un bot de atención al cliente. La aplicación determina que el usuario debe hablar con un representante del servicio de atención al cliente. Después de hablar con el usuario, el representante puede dirigir la conversación de nuevo al bot con la información que se ha recopilado.

Para reanudar una sesión, siga un procedimiento similar a este:

- La aplicación determina que el usuario debe hablar con un representante del servicio de atención al cliente.
- Utilice la operación `GetSession` para obtener el estado actual de la intención del diálogo.
- El representante del servicio de atención al cliente habla con el usuario y resuelve el problema.
- Utilice la operación `PutSession` para establecer el estado de la intención del diálogo. Para ello, tal vez necesite configurar los valores de slot, configurar los atributos de sesión o cambiar la intención.
- El bot reanuda la conversación con el usuario.

Puede utilizar el parámetro `checkpointLabel` de la operación `PutSession` para etiquetar una intención de modo que pueda encontrarla más tarde. Por ejemplo, un bot que solicita información a un cliente podría pasar a la intención `Waiting` mientras el cliente recopila dicha información. El bot crea una etiqueta de punto de comprobación para la intención actual y, a continuación, inicia la intención `Waiting`. Cuando el cliente devuelve dicha información, el bot puede encontrar la intención anterior mediante la etiqueta de punto de comprobación y volver a utilizarla.

La intención debe estar presente en la estructura `recentIntentSummaryView` devuelta por la operación `GetSession`. Si especifica una etiqueta de punto de comprobación en la solicitud de la operación `GetSession`, devolverá un máximo de tres intenciones con dicha etiqueta de punto de comprobación.

- Utilice la operación `GetSession` para obtener el estado actual de la sesión.
- Utilice la operación `PutSession` para añadir una etiqueta de punto de comprobación a la última intención. Si es necesario, puede utilizar la llamada `PutSession` para cambiar de intención.
- Cuando llegue el momento de volver a utilizar la intención etiquetada, llame a la operación `GetSession` para devolver una lista de intenciones recientes. Puede utilizar el parámetro `checkpointLabelFilter` para que Amazon Lex devuelva solo las intenciones con la etiqueta de punto de comprobación especificada.

Inicio de una nueva sesión

Si desea que el bot comience la conversación con el usuario, puede utilizar la operación `PutSession`.

- Cree una intención de bienvenida sin slots y un mensaje de conclusión que le pida al usuario que indique una intención. Por ejemplo, "¿Qué desea pedir?" Puede decir "Una bebida" o "Una pizza".
- Llame a la operación `PutSession`. Especifique el nombre de la intención de bienvenida y establezca la acción del diálogo en `Delegate`.
- Amazon Lex responderá con el mensaje de la intención de bienvenida para iniciar la conversación con el usuario.

Validación de valores de slot

Puede validar las respuestas que recibe el bot utilizando la aplicación cliente. Si la respuesta no es válida, puede utilizar la operación `PutSession` para obtener una nueva respuesta del usuario. Por ejemplo, suponga que un bot para pedir flores solo puede vender tulipanes, rosas y lirios. Si el usuario pide claveles, la aplicación puede hacer lo siguiente:

- Examinar el valor de slot devuelto desde `PostText` o `PostContent`.
- Si el valor de slot no es válido, llamar a la operación `PutSession`. La aplicación debe borrar el valor de slot, definir el campo `slotToElicit` y establecer el valor de `dialogAction.type`

en `elicitSlot`. También puede configurar los campos `message` y `messageFormat` si desea cambiar el mensaje que Amazon Lex utiliza para obtener el valor de slot.

Opciones de implementación del bot

En la actualidad, Amazon Lex proporciona las siguientes opciones de implementación del bot:

- AWS Mobile SDK: puede crear aplicaciones móviles que se comuniquen con Amazon Lex mediante los AWS Mobile SDK.
- Facebook Messenger: puede integrar su página de Facebook Messenger con su bot de Amazon Lex para que los usuarios finales en Facebook puedan comunicarse con el bot. En la implementación actual, esta integración admite únicamente mensajes de entrada de texto.
- Slack: puede integrar el bot de Amazon Lex con una aplicación de mensajería Slack.
- Twilio: puede integrar el bot de Amazon Lex con Twilio Simple Messaging Service (SMS).

Para ver ejemplos, consulte Implementación de bots de Amazon Lex.

Intenciones y tipos de slot integrados

Para facilitar la creación de bots, Amazon Lex permite utilizar las intenciones y los tipos de ranura integrados estándar.

Temas

- Intenciones integradas
- Tipos de intenciones integradas

Intenciones integradas

Para las acciones comunes, puede utilizar la biblioteca de intenciones integrada estándar. Para crear una intención a partir de una intención integrada, elíjala en la consola y asígnele otro nombre. La nueva intención tiene la misma configuración que la intención base, como los enunciados de ejemplo.

En la implementación actual, no puede hacer lo siguiente:

- Añadir ni eliminar enunciados de ejemplo de la intención base

- Configurar slots para intenciones integradas

Añadir una intención integrada a un bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot al cual se debe agregar la intención integrada.
3. En el panel de navegación, elija el signo más (+) situado junto a Intenciones.
4. En Agregar intención, elija Buscar intenciones existentes.
5. El cuadro Buscar intenciones, escriba el nombre de la intención integrada para agregarla al bot.
6. En Copiar intención integrada, escriba un nombre para la intención y elija Agregar.
7. Configure la intención según sea necesario para su bot.

Temas

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

> **Note**
>
> Para la configuración regional en inglés (EE. UU.) (en-US), Amazon Lex admite las intenciones integradas estándar de Alexa. Para obtener una lista de las intenciones integradas, consulte Intenciones estándar integradas en Alexa Skills Kit.
> Amazon Lex no admite las siguientes intenciones:
>
> - `AMAZON.YesIntent`

- `AMAZON.NoIntent`
- Las intenciones de la Biblioteca de intenciones integradas de Alexa Skills Kit.

AMAZON.CancelIntent

Responda a las palabras y frases que indican que el usuario quiere cancelar la interacción actual. Su aplicación puede utilizar esta intención para eliminar los valores de los tipos de slot y otros atributos antes de finalizar la interacción con el usuario.

Enunciados comunes:

- cancelar
- no importa
- olvidar

AMAZON.FallbackIntent

Cuando la entrada de un usuario a una intención no es lo que espera un bot, puede configurar Amazon Lex para que invoque una intención alternativa. Por ejemplo, si la entrada del usuario «Quiero pedir caramelos» no coincide con una intención del bot `OrderFlowers`, Amazon Lex invoca la intención alternativa para gestionar la respuesta.

Puede añadir una intención alternativa mediante la incorporación del tipo de intención `AMAZON.FallbackIntent` integrada al bot. Puede especificar la intención mediante la operación PutBot o eligiendo la intención en la lista de intenciones integradas en la consola.

La invocación de una intención alternativa se realiza en dos pasos. En el primer paso, la intención alternativa coincide según la entrada del usuario. Cuando la intención alternativa coincide, la forma en que se comporta el bot depende del número de reintentos configurados para una pregunta. Por ejemplo, si el número máximo de intentos para determinar una intención es 2, el bot devuelve la pregunta aclaratoria del bot dos veces antes de invocar la intención alternativa.

Amazon Lex hace coincidir la intención alternativa en estas situaciones:

- La entrada del usuario a una intención no coincide con la entrada que espera el bot
- La entrada de audio es ruido o la entrada de texto no se reconoce como palabras.

- La entrada del usuario es ambigua y Amazon Lex no puede determinar qué intención debe invocar.

La intención alternativa se invoca cuando:

- El bot no reconoce la entrada del usuario como una intención después del número configurado de intentos de clarificación cuando comienza la conversación.
- Una intención no reconoce la entrada del usuario como un valor de slot después del número de intentos configurado.
- Una intención no reconoce la entrada del usuario como respuesta a una pregunta de confirmación después del número de intentos configurado.

Puede utilizar lo siguiente con una intención alternativa:

- Una función de Lambda de cumplimiento
- Una instrucción de conclusión
- Una pregunta de seguimiento

No puede añadir lo siguiente a una intención alternativa:

- Enunciados
- Slots
- Una función de Lambda de inicialización y validación
- Una pregunta de confirmación

Si ha configurado una instrucción de cancelación y una intención alternativa para un bot, Amazon Lex utiliza la intención alternativa. Si necesita que su bot tenga una instrucción de cancelación, puede utilizar la función de cumplimiento para la intención alternativa con el fin de proporcionar el mismo comportamiento que una instrucción de cancelación. Para obtener más información, consulte el parámetro `abortStatement` de la operación PutBot.

Uso de las preguntas de aclaración

Si proporciona al bot una pregunta aclaratoria, esta se utiliza para solicitar una intención válida al usuario. La pregunta aclaratoria se repetirá el número de veces que haya configurado. Después de eso, se invocará la intención alternativa.

Si no establece una pregunta aclaratoria al crear un bot y el usuario no comienza la conversación con una intención válida, Amazon Lex llama inmediatamente a su intención alternativa.

Cuando se utiliza una intención alternativa sin una pregunta aclaratoria, Amazon Lex no llama a la alternativa en estas circunstancias:

- Cuando el usuario responde a una pregunta de seguimiento, pero no proporciona una intención. Por ejemplo, si el usuario responde “sí” a la pregunta de seguimiento “¿Desea algo más hoy?”. Amazon Lex devuelve la excepción 400 de solicitud errónea porque no tiene una pregunta aclaratoria que pueda enviar al usuario para obtener una intención.
- Cuando se utiliza una función AWS Lambda, se devuelve un tipo de diálogo `ElicitIntent`. Dado que Amazon Lex no tiene una pregunta aclaratoria para obtener una intención del usuario, devuelve la excepción 400 de solicitud errónea.
- Cuando se utiliza la operación `PutSession`, se envía un tipo de diálogo `ElicitIntent`. Dado que Amazon Lex no tiene una pregunta aclaratoria para obtener una intención del usuario, devuelve la excepción 400 de solicitud errónea.

Uso de una función de Lambda con una intención alternativa

Cuando se invoca una intención alternativa, la respuesta depende de la configuración del parámetro `fulfillmentActivity` para la operación PutIntent. El bot realiza una de las siguientes operaciones:

- Devuelve la información de la intención a la aplicación cliente.
- Llama a la función de Lambda de cumplimiento. Llama a la función con las variables de sesión que se establecen para la sesión.

Para obtener más información acerca de cómo configurar la respuesta cuando se invoca una intención alternativa, consulte el parámetro `fulfillmentActivity` de la operación PutIntent.

Si utiliza la función de Lambda de cumplimiento en su intención alternativa, puede utilizar esta función para llamar a otra intención o para realizar algún tipo de comunicación con el usuario, como recopilar un número de devolución de llamada o abrir una sesión con un representante del servicio de atención al cliente.

Puede realizar cualquier acción en una función de Lambda de intención alternativa que pueda realizar en la función de cumplimiento para otra intención. Para obtener más información acerca de

la creación de una función de cumplimiento mediante AWS Lambda, consulte Uso de funciones de Lambda.

Una intención alternativa se puede invocar varias veces en la misma sesión. Por ejemplo, suponga que la función de Lambda utiliza la acción de diálogo `ElicitIntent` para solicitar al usuario una intención diferente. Si Amazon Lex no puede deducir la intención del usuario después del número de intentos configurado, invoca de nuevo la intención alternativa. También invoca la intención alternativa cuando el usuario no responde con un valor de slot válido después del número de intentos configurados.

Puede configurar una función de Lambda para realizar un seguimiento del número de veces que se llama a la intención alternativa mediante una variable de sesión. La función de Lambda puede realizar una acción diferente si se llama más veces que el umbral establecido en la función de Lambda. Para obtener más información acerca de las variables de sesión, consulte Definición de atributos de la sesión.

AMAZON.HelpIntent

Responde a palabras o frases que indican que el usuario necesita ayuda para interactuar con el bot. Cuando se invoca esta intención, puede configurar la función o aplicación de Lambda para que proporcione información sobre las capacidades del bot, formule preguntas de seguimiento sobre áreas de ayuda o entregue la interacción a un agente humano.

Enunciados comunes:

- ayuda
- ayúdame
- ¿me puedes ayudar?

AMAZON.KendraSearchIntent

Para buscar documentos indexados con Amazon Kendra, utilice la intención `AMAZON.KendraSearchIntent`. Cuando Amazon Lex no puede determinar la siguiente acción en una conversación con el usuario, desencadena la intención de búsqueda.

Solo `AMAZON.KendraSearchIntent` está disponible en la configuración regional en inglés (EE. UU.) (en-US) y en el Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón) y Europa (Irlanda).

Amazon Kendra es un servicio de búsqueda basado en aprendizaje automático que indexa documentos en lenguaje natural, como documentos PDF o archivos de Microsoft Word. Puede buscar documentos indexados y devolver los siguientes tipos de contestaciones a una pregunta:

- Una respuesta
- Una entrada de una pregunta frecuente que podría dar respuesta a la pregunta
- Un documento relacionado con la pregunta

Para ver un ejemplo del uso de `AMAZON.KendraSearchIntent`, consulte Ejemplo: Creación de un bot de preguntas frecuentes sobre un índice de Amazon Kendra.

Si configura una intención `AMAZON.KendraSearchIntent` para su bot, Amazon Lex la llamará siempre que no pueda determinar el enunciado del usuario en una ranura o una intención. Por ejemplo, si el bot está obteniendo una respuesta de un tipo de ranura denominado “ingrediente de pizza” y el usuario dice “¿Qué es una pizza?”, Amazon Lex llamará a `AMAZON.KendraSearchIntent` para administrar la pregunta. Si no hay respuesta de Amazon Kendra, la conversación continuará tal y como está configurada en el bot.

Cuando `AMAZON.KendraSearchIntent` y `AMAZON.FallbackIntent` se usan en el mismo bot, Amazon Lex utiliza las intenciones de la siguiente manera:

1. Amazon Lex llama a `AMAZON.KendraSearchIntent`. La intención llama a la operación `Query` de Amazon Kendra.
2. Si Amazon Kendra devuelve una respuesta, Amazon Lex muestra el resultado al usuario.
3. Si no hay respuesta por parte de Amazon Kendra, Amazon Lex vuelve a preguntar al usuario. La siguiente acción dependerá de la respuesta del usuario.
   - Si la respuesta del usuario contiene un enunciado que Amazon Lex reconoce, como llenar un valor de ranura o confirmar una intención, la conversación con el usuario continúa conforme a la configuración del bot.
   - Si la respuesta del usuario no contiene un enunciado que Amazon Lex reconozca, Amazon Lex hará otra llamada a la operación `Query`.
4. Si tras un número establecido de nuevos intentos no hay ninguna respuesta, Amazon Lex llamará a `AMAZON.FallbackIntent` y finalizará la conversación con el usuario.

Hay tres formar de usar la `AMAZON.KendraSearchIntent` para hacer una solicitud a Amazon Kendra:

- Deje que la intención de búsqueda haga la solicitud por usted. Amazon Lex llama a Amazon Kendra con el enunciado del usuario como cadena de búsqueda. Cuando cree la intención, puede definir una cadena de filtro de consulta que limite el número de respuestas devueltas por Amazon Kendra. Amazon Lex utiliza el filtro en la solicitud de consulta.
- Agregue parámetros de consulta adicionales a la solicitud para acotar los resultados de la búsqueda mediante la función de Lambda de diálogo. Puede agregar un campo `kendraQueryFilterString` que contenga parámetros de consulta de Amazon Kendra a la acción de diálogo `delegate`. Cuando se agregan parámetros de consulta a la solicitud con la función de Lambda, estos tienen prioridad sobre el filtro de consulta que se definió al crear la intención.
- Crear una nueva consulta mediante la función de Lambda de diálogo. Puede crear una solicitud de consulta de Amazon Kendra completa para que Amazon Lex la envíe. Especifique la consulta en el campo `kendraQueryRequestPayload` de la acción de diálogo `delegate`. El campo `kendraQueryRequestPayload` tiene prioridad sobre el campo `kendraQueryFilterString`.

Si desea especificar el parámetro `queryFilterString` al crear un bot o el campo `kendraQueryFilterString` al llamar a la acción `delegate` en una función de Lambda de diálogo, especifique una cadena que se utilice como filtro de atributos en la consulta de Amazon Kendra. Si la cadena no es un filtro de atributos válido, aparecerá una excepción `InvalidBotConfigException` en tiempo de ejecución. Para obtener más información sobre los filtros de atributos, consulte Usar atributos de documentos para filtrar las consultas en la Guía para desarrolladores de Amazon Kendra.

Para mantener el control sobre la consulta que Amazon Lex envía a Amazon Kendra, puede especificar una consulta en el campo `kendraQueryRequestPayload` de la función de Lambda de diálogo. Si la consulta no es válida, Amazon Lex devolverá una excepción `InvalidLambdaResponseException`. Para obtener más información, consulte la Operación Query en la Guía para desarrolladores de Amazon Kendra.

Si desea ver un ejemplo de cómo se usa `AMAZON.KendraSearchIntent`, consulte Ejemplo: Creación de un bot de preguntas frecuentes sobre un índice de Amazon Kendra.

Política de IAM para Amazon Kendra Search

Para poder utilizar la intención `AMAZON.KendraSearchIntent`, debe tener un rol que proporcione políticas de AWS Identity and Access Management (IAM) que permitan a Amazon Lex adoptar un rol de tiempo de ejecución con permiso para llamar a la intención `Query` de Amazon Kendra. La

configuración de IAM que utilice dependerá de si crea la `AMAZON.KendraSearchIntent` con la consola de Amazon Lex, con un SDK de AWS o con la AWS Command Line Interface (AWS CLI). Si utiliza la consola, puede decidir si desea agregar permisos al rol vinculado al servicio de Amazon Lex para que llame a Amazon Kendra o si prefiere utilizar un rol específico para llamar a la operación `Query` de Amazon Kendra. Si utiliza AWS CLI o un SDK para crear la intención, debe usar un rol específico para llamar a la operación `Query`.

Asociación de permisos

Puede utilizar la consola para asociar permisos que permitan al rol vinculado al servicio de Amazon Lex predeterminado acceder a la operación `Query` de Amazon Kendra. Si asocia permisos al rol vinculado al servicio, no es necesario crear y administrar específicamente un rol en tiempo de ejecución para conectarse al índice de Amazon Kendra.

El usuario, el rol o el grupo que utilice para obtener acceso a la consola de Amazon Lex debe tener permisos para administrar políticas de roles. Asocie la siguiente política de IAM al rol de acceso de la consola. Al conceder estos permisos, el rol podrá cambiar la política del rol vinculado al servicio 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": "*"
        }
    ]
}
```

Especificación de un rol

Puede utilizar la consola, la AWS CLI o la API para especificar un rol en tiempo de ejecución que se utilice para llamar a la operación `Query` de Amazon Kendra.

El usuario, rol o grupo que utilice para especificar el rol en tiempo de ejecución debe tener el permiso `iam:PassRole`. La siguiente política define el permiso. Puede utilizar las claves de contexto de condición `iam:AssociatedResourceArn` y `iam:PassedToService` para limitar aún más el alcance de los permisos. Para obtener más información, consulte IAM y claves de contexto de condición de AWS STS en la Guía del usuario de AWS Identity and Access Management.

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

El rol en tiempo de ejecución que Amazon Lex tiene que usar para llamar a Amazon Kendra debe tener permisos `kendra:Query`. Cuando se utiliza un rol de IAM existente para obtener permiso para llamar a la operación `Query` de Amazon Kendra, el rol debe tener asociada la siguiente política.

Puede utilizar la consola de IAM, la API de IAM o la AWS CLI para crear una política y asociarla a un rol. Estas instrucciones utilizan la CLI de AWS para crear el rol y las políticas.

> **Note**
>
> El siguiente código tiene formato para Linux y MacOS. Para Windows, reemplace el carácter de continuación de línea de Linux (\) por un signo de intercalación (^).

Para agregar permisos de la operación Query a un rol

1. Cree un documento llamado **`KendraQueryPolicy.json`** en el directorio actual, agregue el código siguiente y guárdelo.

   ```
   {
   ```

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

2. En AWS CLI, ejecute el siguiente comando para crear la política de IAM y ejecutar la operación `Query` de Amazon Kendra.

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

3. Asocie la política al rol de IAM que esté utilizando para llamar a la operación `Query`.

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

Puede optar por actualizar el rol vinculado al servicio de Amazon Lex o utilizar el rol que creó al generar la `AMAZON.KendraSearchIntent` de su bot. En el siguiente procedimiento, se muestra cómo puede elegir el rol de IAM que se va a utilizar.

Para especificar el rol en tiempo de ejecución de AMAZON.KendraSearchIntent

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot al que desee agregar la `AMAZON.KendraSearchIntent`.
3. Seleccione el signo más (+) situado junto a Intenciones.
4. En Agregar intención, seleccione Buscar intenciones existentes.
5. En Intenciones de búsqueda, escriba **`AMAZON.KendraSearchIntent`** y seleccione Agregar.

6. En Copiar intención integrada, escriba un nombre para la intención, como **`KendraSearchIntent`**, y seleccione Agregar.
7. Abra la sección de consultas de Amazon Kendra.
8. En Rol de IAM, elija una de las opciones siguientes:

   - Para actualizar el rol vinculado al servicio de Amazon Lex y permitir que el bot consulte los índices de Amazon Kendra, seleccione Agregar permisos de Amazon Kendra.
   - Para utilizar un rol que tenga permiso para llamar a la operación `Query` de Amazon Kendra, seleccione Usar un rol existente.

Uso de atributos de solicitud y sesión como filtros

Para filtrar la respuesta de Amazon Kendra y obtener los elementos relacionados con la conversación actual, use los atributos de sesión y solicitud como filtros agregando el parámetro `queryFilterString` cuando cree el bot. Especifique un marcador de posición para el atributo cuando cree la intención. De ese modo, Amazon Lex V2 sustituirá el valor antes de llamar a Amazon Kendra. Para obtener más información sobre los atributos de solicitud, consulte Definición de los atributos de solicitud. Para obtener más información acerca de los atributos de la sesión, consulte Definición de atributos de la sesión.

A continuación, se muestra un ejemplo de un parámetro `queryFilterString` que utiliza una cadena para filtrar la consulta de Amazon Kendra.

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

A continuación, se muestra un ejemplo de un parámetro `queryFilterString` que utiliza un atributo de sesión llamado `"SourceURI"` para filtrar la consulta de Amazon Kendra.

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

A continuación, se muestra un ejemplo de un parámetro `queryFilterString` que utiliza un atributo de solicitud llamado `"DepartmentName"` para filtrar la consulta de Amazon Kendra.

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

Los filtros `AMAZON.KendraSearchInteng` utilizan el mismo formato que los filtros de búsqueda de Amazon Kendra. Para obtener más información, consulte Usar atributos de documentos para filtrar los resultados de búsqueda en la Guía para desarrolladores de Amazon Kendra.

La cadena de filtro de consultas utilizada con la `AMAZON.KendraSearchIntent` debe incluir letras minúsculas para la primera letra de cada filtro. Por ejemplo, a continuación se muestra un filtro de consulta válido para la `AMAZON.KendraSearchIntent`.

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

Uso de la respuesta de búsqueda

Amazon Kendra devuelve la respuesta a una búsqueda en la declaración `conclusion` de la intención. La intención debe tener una declaración `conclusion`, a menos que una función de Lambda de cumplimiento genere un mensaje de conclusión.

Amazon Kendra tiene cuatro tipos de respuestas.

- `x-amz-lex:kendra-search-response-question_answer-question-<N>`: la pregunta de una sección de preguntas frecuentes que coincide con la búsqueda.
- `x-amz-lex:kendra-search-response-question_answer-answer-<N>`: la respuesta de una sección de preguntas frecuentes que coincide con la búsqueda.
- `x-amz-lex:kendra-search-response-document-<N>`: un extracto de un documento del índice relacionado con el texto del enunciado.
- `x-amz-lex:kendra-search-response-document-link-<N>`: la URL de un documento del índice relacionado con el texto del enunciado.

- `x-amz-lex:kendra-search-response-answer-<N>`: un extracto de un documento del índice que responde a la pregunta.

Las respuestas se devuelven en atributos `request`. Puede haber hasta cinco respuestas para cada atributo, numeradas del 1 al 5. Para obtener más información sobre las respuestas, consulte Tipos de respuesta en la Guía para desarrolladores de Amazon Kendra.

La declaración `conclusion` debe tener uno o varios grupos de mensajes. Cada grupo contiene uno o varios mensajes. Cada mensaje puede contener una o varias variables de marcador de posición que se reemplazarán con los atributos de solicitud de la respuesta proporcionada por Amazon Kendra. En el grupo de mensajes debe haber al menos un mensaje en el que todas las variables se hayan sustituido por los valores de atributo de solicitud obtenidos de la respuesta en tiempo de ejecución o debe haber un mensaje sin variables de marcador de posición. Los atributos de solicitud se separan con paréntesis dobles («((»))»). Los siguientes mensajes del grupo coinciden con cualquier respuesta de Amazon Kendra:

- «Encontré una pregunta frecuente para usted: ((x-amz-lex:kendra-search-response-question_answer-question-1)) y la respuesta es ((x-amz-lex:kendra-search-response-question_answer-answer-1))»
- «Encontré un extracto en un documento útil: ((x-amz-lex:kendra-search-response-document-1))»
- «Creo que la respuesta a sus preguntas es ((x-amz-lex:kendra-search-response-answer-1))»

Usar una función de Lambda para administrar la solicitud y la respuesta

La intención `AMAZON.KendraSearchIntent` puede utilizar el enlace de código de diálogo y el enlace de código de cumplimentación para administrar la solicitud enviada a Amazon Kendra y la respuesta. Utilice la función de Lambda del enlace de código de diálogo cuando desee modificar la consulta que envía a Amazon Kendra, y la función de Lambda de enlace de código de cumplimentación cuando desee modificar la respuesta.

Crear una consulta con el enlace de código de diálogo

Puede utilizar el enlace de código de diálogo para crear una consulta y enviarla a Amazon Kendra. El uso del enlace de código de diálogo es opcional. Si no especifica ningún enlace de código de diálogo, Amazon Lex creará una consulta a partir del enunciado del usuario y utilizará la `queryFilterString` que se proporcionó al configurar la intención, si se proporcionó alguna.

Puede utilizar dos campos en la respuesta del enlace de código de diálogo para modificar la solicitud que se envía a Amazon Kendra:

- `kendraQueryFilterString`: utilice esta cadena para especificar los filtros de atributos para la solicitud de Amazon Kendra. Puede filtrar la consulta utilizando cualquiera de los campos definidos en el índice. Para obtener información sobre la estructura de la cadena de filtro, consulte Usar atributos de documentos para filtrar consultas en la Guía para desarrolladores de Amazon Kendra. Si la cadena de filtro especificada no es válida, aparecerá una excepción `InvalidLambdaResponseException`. La cadena `kendraQueryFilterString` invalida cualquier otra cadena de consulta especificada en el campo `queryFilterString` configurado para la intención.
- `kendraQueryRequestPayload`: utilice esta cadena para especificar una consulta de Amazon Kendra. La consulta puede utilizar cualquiera de las características de Amazon Kendra. Si no especifica una consulta válida, aparecerá una excepción `InvalidLambdaResponseException`. Para obtener más información, consulte Consulta en la Guía para desarrolladores de Amazon Kendra.

Una vez que haya creado el filtro o la cadena de consulta, envíe la respuesta a Amazon Lex con el campo `dialogAction` de la respuesta establecido en `delegate`. Amazon Lex envía la consulta a Amazon Kendra y, a continuación, devuelve la respuesta a la consulta al enlace de código de cumplimentación.

Uso del enlace de código de cumplimentación en la respuesta

Una vez que Amazon Lex envía una consulta a Amazon Kendra, la respuesta se devuelve a la función de Lambda de cumplimiento `AMAZON.KendraSearchIntent`. El evento de entrada del enlace de código contiene la respuesta completa de Amazon Kendra. Los datos de consulta tienen la misma estructura que los datos devueltos por la operación `Query` de Amazon Kendra. Para obtener más información, consulte Sintaxis de la respuesta a la consulta en la Guía para desarrolladores de Amazon Kendra.

El enlace de código de cumplimentación es opcional. Si no existe o si el enlace de código no devuelve un mensaje en la respuesta, Amazon Lex utilizará la declaración `conclusion` con las respuestas.

Ejemplo: Creación de un bot de preguntas frecuentes sobre un índice de Amazon Kendra

En este ejemplo, se crea un bot de Amazon Lex que utiliza un índice de Amazon Kendra para proporcionar respuestas a las preguntas de los usuarios. El bot de preguntas frecuentes

(FAQ) se encarga de administrar el diálogo con el usuario. Este bot utiliza la intención `AMAZON.KendraSearchIntent` para consultar el índice y presentar la respuesta al usuario. Para crear un bot, tiene que hacer lo siguiente:

1. Crear un bot con el que sus clientes puedan interactuar para obtener respuestas.
2. Crear una intención personalizada. El bot necesita al menos una intención que tenga, como mínimo, un enunciado. Esta intención permitirá crear el bot, pero no se utilizará de ninguna otra manera.
3. Agregar la intención `KendraSearchIntent` al bot y configurarlo para que funcione con el índice de Amazon Kendra.
4. Probar el bot haciendo preguntas que deben responderse a partir de los documentos almacenados en el índice de Amazon Kendra.

Para poder utilizar este ejemplo, primero debe crear un índice de Amazon Kendra. Para obtener más información, consulte Introducción a los buckets de S3 (consola) en la Guía del desarrollador de Amazon Kendra.

Para crear un bot de preguntas frecuentes

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el panel de navegación, elija Bots.
3. Seleccione Create (Crear).
4. Elija Custom bot (Personalizar bot). Configure el bot de la siguiente manera:
   - Nombre del bot: introduzca un nombre que indique la finalidad del bot, como **`KendraTestBot`**.
   - Voz de salida: elija Ninguna.
   - Tiempo de espera de la sesión: introduzca **5**.
   - Análisis de opiniones: elija No.
   - COPPA: elija No.
   - Almacenamiento de enunciados del usuario: elija No almacenar.
5. Seleccione Create (Crear).

Para generar correctamente un bot, debe crear al menos una intención que tenga, como mínimo, un enunciado de ejemplo. Esta intención es necesaria para compilar el bot de Amazon Lex, pero no se usa para responder a preguntas frecuentes. El enunciado de la intención no debe aplicarse a ninguna de las preguntas que haga el cliente.

Para crear la intención necesaria

1. En la página Introducción al bot, elija Crear intención.
2. En Agregar intención, elija Crear intención.
3. En el cuadro de diálogo Crear intención, escriba un nombre; por ejemplo **`RequiredIntent`**.
4. En Enunciados de ejemplo, escriba un enunciado; por ejemplo, **`Required utterance`**.
5. Elija Guardar intención.

Ahora, cree la intención para buscar un índice de Amazon Kendra y los mensajes de respuesta que debe devolver.

Para crear una intención AMAZON.KendraSearchIntent y un mensaje de respuesta

1. En el panel de navegación, elija el signo más (+) situado junto a Intenciones.
2. En Agregar intención, elija Buscar intenciones existentes.
3. En el cuadro de búsqueda Buscar intenciones, introduzca **`AMAZON.KendraSearchIntent`** y seleccione esta intención en la lista.
4. En Copiar intención integrada, escriba un nombre para la intención, como **`KendraSearchIntent`**, y elija Agregar.
5. En el editor de intenciones, seleccione la Consulta de Amazon Kendra para abrir las opciones de consulta.
6. En el menú Índice de Amazon Kendra, elija el índice donde desee buscar la intención.
7. En la sección Respuesta, agregue los tres mensajes siguientes:

```
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. Elija Guardar intención y Crear para generar el bot.

Por último, utilice la ventana de prueba de la consola para probar las respuestas del bot. Las preguntas deben restringirse a un ámbito incluido en el índice.

Para probar el bot de preguntas frecuentes

1. En la ventana de prueba de la consola, escriba una pregunta sobre el índice.
2. Compruebe la respuesta en la sección de respuestas de la ventana de prueba.
3. Si desea restablecer la ventana de prueba para hacer otra pregunta, seleccione Borrar historial del chat.

AMAZON.PauseIntent

Responde a las palabras y frases que permiten al usuario pausar una interacción con un bot para poder volver a ella más adelante. La aplicación o función de Lambda necesita guardar los datos de intención en las variables de sesión, o bien, debe usar la operación GetSession para recuperar los datos de intención cuando reanude la intención actual.

Enunciados comunes:

- Pause
- Pausar eso

AMAZON.RepeatIntent

Responde a palabras y frases que permiten al usuario repetir el mensaje anterior. La aplicación debe usar una función de Lambda para guardar la información de intención anterior en las variables de sesión, o bien, debe usar la operación GetSession para obtener la información de intención anterior.

Enunciados comunes:

- repetir
- dilo otra vez
- repite

AMAZON.ResumeIntent

Responde a palabras y frases que permiten al usuario reanudar un intento previamente pausado. La aplicación o función de Lambda debe gestionar la información necesaria para reanudar la intención anterior.

Enunciados comunes:

- resumir
- continuar
- seguir adelante

AMAZON.StartOverIntent

Responde a palabras y frases que permiten al usuario dejar de procesar la intención actual y volver a empezar desde el principio. Puede utilizar la función de Lambda o la operación `PutSession` para volver a obtener el valor del primer slot.

Enunciados comunes:

- empezar de nuevo
- reiniciar
- empezar de nuevo

AMAZON.StopIntent

Responde a las palabras y frases que indican que el usuario quiere dejar de procesar la intención actual y finalizar la interacción con un bot. La aplicación o función de Lambda debe borrar todos los atributos y valores de tipo de slot existentes y, a continuación, finalizar la interacción.

Enunciados comunes:

- parar
- apagar
- calla

Tipos de intenciones integradas

Amazon Lex admite tipos de slot integrados que definen cómo se reconocen y gestionan los datos del slot. Puede crear slots de estos tipos en sus intenciones. Esto elimina la necesidad de crear valores de enumeración para datos de slot de uso común como la fecha, la hora y la ubicación. Los tipos de slot integrados no tienen versiones.

Tipo de slot	Descripción breve	Configuraciones regionales admitidas	
AMAZON.Airport	Reconoce las palabras que representan un aeropuerto.	Todas las configuraciones locales	
AMAZON.AlphaNumeric	Reconoce palabras compuestas de letras y números.	Todas las configuraciones regionales, excepto el coreano (ko-KR)	
AMAZON.City	Reconoce las palabras que representan una ciudad.	Todas las configuraciones locales	
AMAZON.Country	Reconoce las palabras que representan un país.	Todas las configuraciones locales	
AMAZON.DATE	Reconoce las palabras que representan una fecha y las convierte a un formato estándar.	Todas las configuraciones locales	
AMAZON.DURATION	Reconoce las palabras que representan una	Todas las configuraciones locales	

Tipo de slot	Descripción breve	Configuraciones regionales admitidas	
	duración y las convierte a un formato estándar.		
AMAZON.EmailAddress	Reconoce las palabras que representan una dirección de correo electrónico y las convierte en una dirección de correo electrónico estándar.	Todas las configuraciones locales	
AMAZON.FirstName	Reconoce las palabras que representan un nombre.	Todas las configuraciones locales	
AMAZON.LastName	Reconoce las palabras que representan un apellido.	Todas las configuraciones locales	
AMAZON.NUMBER	Reconoce palabras numéricas y las convierte en dígitos.	Todas las configuraciones locales	
AMAZON.Percentage	Reconoce las palabras que representan un porcentaje y las convierte en un número y un signo de porcentaje (%).	Todas las configuraciones locales	

Tipo de slot	Descripción breve	Configuraciones regionales admitidas	
AMAZON.PhoneNumber	Reconoce palabras que representan un número de teléfono y las convierte en una cadena numérica.	Todas las configuraciones locales	
AMAZON.SpeedUnit	Reconoce palabras que representan una unidad de velocidad y las convierte en una abreviatura estándar.	Inglés (EE. UU.) (en-US)	
AMAZON.State	Reconoce las palabras que representan un estado.	Todas las configuraciones locales	
AMAZON.StreetName	Reconoce las palabras que representan el nombre de una calle.	Todas las configuraciones regionales, excepto en inglés (EE. UU.) (en-US)	
AMAZON.TIME	Reconoce las palabras que indican horas y las convierte en un formato de hora.	Todas las configuraciones locales	
AMAZON.WeightUnit	Reconoce palabras que representan una unidad de peso y las convierte en una abreviatura estándar.	Inglés (EE. UU.) (en-US)	

> **Note**
>
> Para la configuración regional en inglés (EE. UU.) (en-US), Amazon Lex admite los tipos de ranura de Alexa Skills Kit. Para obtener una lista de los tipos de slot integrados que están disponibles, consulte lareferencia sobre los tipos de slots en la documentación de Alexa Skills Kit.
>
> - Amazon Lex no admite `AMAZON.LITERAL` o los tipos de ranura integrados `AMAZON.SearchQuery`.

AMAZON.Airport

Proporciona una lista de aeropuertos. Entre los ejemplos se incluyen:

- Aeropuerto Internacional John F. Kennedy
- Aeropuerto de Melbourne

AMAZON.AlphaNumeric

Reconoce cadenas compuestas de letras y números, como **`APQ123`**.

Este tipo de slot no está disponible en la configuración regional coreana (ko-KR).

Puede usar el tipo de slot `AMAZON.AlphaNumeric` para las cadenas que contienen:

- Caracteres alfabéticos, como **`ABC`**
- Caracteres numéricos, como **`123`**
- Una combinación de caracteres alfanuméricos, como **`ABC123`**

Puede añadir una expresión regular al tipo de slot `AMAZON.AlphaNumeric` para validar los valores introducidos para el slot. Por ejemplo, puede utilizar una expresión regular para validar:

- Códigos postales del Reino Unido o Canadá
- Números de permiso de conducción
- Números de identificación de vehículo

Use una expresión regular estándar. Amazon Lex admite los siguientes caracteres en la expresión regular:

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

Amazon Lex también admite caracteres Unicode en las expresiones regulares. El formato es `\u`*`Unicode`*. Utilice cuatro dígitos para representar caracteres Unicode. Por ejemplo, `[\u0041-\u005A]` equivale a [A-Z].

No se admiten los siguientes operadores de expresiones regulares:

- Repetidores infinitos: *, + o {x,} sin límite superior.
- Comodín (.)

La longitud máxima de la expresión regular es de 300 caracteres. La longitud máxima de una cadena almacenada en un tipo de slot alfanumérico de AMAZON.AlphaNumeric que utiliza una expresión regular es de 30 caracteres.

A continuación se muestran algunos ejemplos de expresiones regulares.

- Cadenas alfanuméricas, como **`APQ123`** o **`APQ1`**: `[A-Z]{3}[0-9]{1,3}` o una cadena más restringida `[A-DP-T]{3} [1-5]{1,3}`
- Formato internacional de correo urgente del servicio postal de Estados Unidos, como **`CP123456789US`**: `CP[0-9]{9}US`
- Números de ruta bancaria, como **`123456789`**: `[0-9]{9}`

Para establecer la expresión regular de un tipo de slot, utilice la consola o la operación PutSlotType. La expresión regular se valida al guardar el tipo de slot. Si la expresión no es válida, Amazon Lex devuelve un mensaje de error.

Cuando se utiliza una expresión regular en un tipo de ranura, Amazon Lex comprueba la entrada en ranuras de ese tipo con la expresión regular. Si la entrada coincide con la expresión, el valor se acepta para el slot. Si la entrada no coincide, Amazon Lex solicita al usuario que repita la entrada.

AMAZON.City

Proporciona una lista de ciudades locales y mundiales. El tipo de slot reconoce las variaciones más comunes de los nombres de las ciudades. Amazon Lex no convierte de una variante a un nombre oficial.

Ejemplos:

- Nueva York
- Reikiavik
- Tokio
- Versalles

AMAZON.Country

Los nombres de los países de todo el mundo. Ejemplos:

- Australia
- Alemania
- Japón
- Estados Unidos
- Uruguay

AMAZON.DATE

Convierte las palabras que representan fechas en un formato de fecha.

La fecha se proporciona según su intención en el formato de fecha ISO-8601. La fecha en que su intención aparece en el slot puede variar según la frase específica pronunciada por el usuario.

- Los enunciados que se refieren a una fecha específica, como «hoy», «ahora» o «veinticinco de noviembre», se convierten en una fecha completa: `2020-11-25`. De forma predeterminada, son fechas iguales o posteriores a la fecha actual.
- Los enunciados que se refieren a una semana concreta, como “esta semana” o “la semana que viene”, se convierten en la fecha del primer día de la semana. En el formato ISO-8601, la semana comienza el lunes y termina el domingo. Por ejemplo, si hoy es 25 de noviembre de 2020, «la semana que viene» se convierte en `2020-11-30`.

- Los enunciados que se asignan a un mes, pero no a un día específico, como “el mes que viene”, se convierten en el último día del mes. Por ejemplo, si hoy es 25 de noviembre de 2020, «el mes que viene» se convierte en `2020-12-31`.
- Los enunciados que se asignan a un año, pero no a un día o mes específico, como “el año que viene”, se convierten en el último día del año. Por ejemplo, si hoy es 25 de noviembre de 2020, «el año que viene» se convierte en `2021-12-31`.

AMAZON.DURATION

Convierte las palabras que indican duraciones en una duración numérica.

La duración se resuelve en un formato basado en el formato de duración ISO-8601, `PnYnMnWnDTnHnMnS`. `P` indica que se trata de una duración, `n` es un valor numérico y la letra mayúscula que sigue a `n` es el elemento de fecha u hora específico. Por ejemplo, `P3D` significa 3 días. `T` se utiliza para indicar que los valores restantes representan elementos de tiempo y no de fecha.

Ejemplos:

- «diez minutos»: `PT10M`
- «cinco horas»: `PT5H`
- «tres días»: `P3D`
- «cuarenta y cinco segundos»: `PT45S`
- «ocho semanas»: `P8W`
- «siete años»: `P7Y`
- «cinco horas y diez minutos»: `PT5H10M`
- «dos años, tres horas y diez minutos»: `P2YT3H10M`

AMAZON.EmailAddress

Reconoce palabras que representan una dirección de correo electrónico especificada como nombredeusuario@dominio. Las direcciones pueden incluir los siguientes caracteres especiales en un nombre de usuario: guion bajo (_), guion (-), punto (.) y signo más (+).

AMAZON.FirstName

Nombres de uso común. Este tipo de ranura reconoce nombres formales y apodos informales. El nombre que se envía a su intención es el valor enviado por el usuario. Amazon Lex no convierte de apodos a nombres formales.

Para los nombres que suenan igual, pero que se escriben de forma diferente, Amazon Lex envía la intención en una única forma común.

En la configuración regional en inglés (EE. UU.) (en-US), utilice el nombre de ranura AMAZON.US_First_Name.

Ejemplos:

- Emily
- John
- Sofia

AMAZON.LastName

Apellidos de uso común. Para los apellidos que suenan igual, pero que se escriben de forma diferente, Amazon Lex envía la intención en una única forma común.

En la configuración regional en inglés (EE. UU.) (en-US), utilice el nombre de ranura AMAZON.US_Last_Name.

Ejemplos:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON.NUMBER

Convierte palabras o números que expresan un número en dígitos, incluidos los decimales. En la siguiente tabla se muestra cómo el tipo de slot `AMAZON.NUMBER` captura palabras numéricas.

Entrada	Respuesta
ciento veinte tres punto cuatro cinco	123.45
ciento veinte tres punto cuatro cinco	123.45
punto cuatro dos	0.42
punto cuarenta y dos	0.42
232.998	232.998
50	50

AMAZON.Percentage

Convierte palabras y símbolos que representan un porcentaje en un valor numérico con un signo de porcentaje (%).

Si el usuario introduce un número sin un signo de porcentaje ni la palabra «porcentaje», el valor del slot se establece en el número. En la siguiente tabla se muestra cómo el tipo de slot `AMAZON.Percentage` captura porcentajes.

Input	Respuesta
50 por ciento	50%
0.4 por ciento	0.4%
23.5%	23.5%
veinticinco por ciento	25%

AMAZON.PhoneNumber

Convierte los números o palabras que representan un número de teléfono en un formato de cadena sin puntuación del modo siguiente.

Tipo	Descripción	Input	Resultado
Número internacional con el signo más (+) inicial	Número de 11 dígitos con el signo más inicial.	+61 7 4445 1061 +1 (509) 555-1212	`+61744431061` `+15095551212`
Número internacional sin el signo más (+) inicial	Número de 11 dígitos sin el signo más inicial	1 (509) 555-1212 61 7 4445 1061	`15095551212` `61744451061`
Número nacional	Número de 10 dígitos sin código internaci onal	(03) 5115 4444 (509) 555-1212	`0351154444` `5095551212`
Número local	Número de teléfono de 7 dígitos sin código internacional ni código de área	555-1212	`5551212`

AMAZON.SpeedUnit

Convierte palabras que representan una unidad de velocidad en la abreviatura correspondiente. Por ejemplo, "millas por hora" se convierte en `mph`.

Este tipo de ranura solo está disponible en la configuración regional en inglés (EE. UU.) (en-US).

Los siguientes ejemplos muestran cómo el tipo de slot `AMAZON.SpeedUnit` captura las unidades de velocidad.

Unidad de velocidad	Abreviatura
millas por hora, mph, MPH, m/h	mph
kilómetros por hora, km por hora, km/h, KMPH, km/h	kmph
metros por segundo, mps, MPS, m/s	mps

Unidad de velocidad	Abreviatura
millas náuticas por hora, nudos, nudo	nudo

AMAZON.State

Los nombres de las regiones geográficas y políticas de los países.

Ejemplos:

- Baviera
- Prefectura de Fukushima
- Noroeste del Pacífico
- Queensland
- Gales

AMAZON.StreetName

Los nombres de las calles en una dirección postal típica. Esto incluye solo el nombre de la calle, no el número de la casa.

Este tipo de ranura no está disponible en la configuración regional en inglés (EE. UU.) (en-US).

Ejemplos:

- Avenida Canberra
- Front Street
- Market Road

AMAZON.TIME

Convierte palabras que representan horas en valores de horas. Incluye resoluciones de horas ambiguas. Cuando un usuario introduce una hora ambigua, Amazon Lex utiliza el atributo `slotDetails` de un evento de Lambda para pasar resoluciones de horas ambiguas a la función de Lambda. Por ejemplo, si el bot solicita al usuario una hora de entrega, el usuario puede responder diciendo "10 en punto". Esta hora es ambigua. Pueden ser las 10:00 de la mañana o las 10:00

de la noche. En este caso, el valor de la asignación `slots` es `null` y la entidad `slotDetails` contiene las dos posibles resoluciones de la hora. Amazon Lex introduce lo siguiente en la función de Lambda:

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

Cuando el usuario responde con una hora inequívoca, Amazon Lex envía la hora a la función de Lambda en el atributo `slots` del evento de Lambda y el atributo `slotDetails` está vacío. Por ejemplo, si el usuario responde a la petición de una hora de entrega con “10:00”, Amazon Lex introduce lo siguiente en la función de Lambda:

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

Para obtener más información acerca de los datos enviados de Amazon Lex a una función de Lambda, consulte Formato del evento de entrada.

AMAZON.WeightUnit

Convierte palabras que representan una unidad de peso en la abreviatura correspondiente. Por ejemplo, "kilogramo" se convierte en `kg`.

Este tipo de ranura solo está disponible en la configuración regional en inglés (EE. UU.) (en-US).

En los siguientes ejemplos se muestra cómo el tipo de slot `AMAZON.WeightUnit` captura unidades de peso:

Unidad de peso	Abreviatura
kilogramos, kilos, kgs, KGS	kg
gramos, grms, gm, GMS, g	g
miligramos, mg, mgs.	mg
libras, lbs, LBS	lbs
onzas, oz, OZ	oz
tonelada, ton, t	t
kilotón, kt	kt

Tipos de slots personalizados

Para cada intención, puede especificar los parámetros que indican la información que necesita para satisfacer la solicitud del usuario. Estos parámetros, o slots, tienen un tipo. Un tipo de ranura es una lista de valores que Amazon Lex utiliza para enseñar al modelo de machine learning a reconocer los valores de una ranura. Por ejemplo, puede definir un tipo de slot llamado "`Genres.`" Cada valor del tipo de slot es el nombre de un género, "comedia", "aventura", "documentales", etc. Puede definir un sinónimo para un valor de tipo de slot. Por ejemplo, puede definir los sinónimos "divertida" y "humorística" para el valor "comedia".

Puede configurar el tipo de slot para restringir la resolución a los valores del slot. Los valores de slot se utilizarán como una enumeración y el valor especificado por el usuario se resolverá en el valor del slot si es el mismo que uno de los valores de slot o un sinónimo. Un sinónimo se resuelve en el valor del slot correspondiente. Por ejemplo, si el usuario introduce "divertido", se resolverá en el slot "comedia".

Alternativamente, puede configurar el tipo de slot para ampliar los valores. Los valores del slot se utilizarán como datos de capacitación y el slot se resuelve en el valor que proporciona el usuario si es similar a los valores del slot y los sinónimos. Este es el comportamiento predeterminado.

Amazon Lex mantiene una lista de posibles resoluciones para una ranura. Cada entrada de la lista ofrece un valor de resolución que Amazon Lex reconoce como posibilidad adicional para la ranura.

Un valor de resolución es la mejor forma de que coincida con el valor de slot. La lista contiene hasta cinco valores.

Cuando el valor especificado por el usuario es un sinónimo, la primera entrada de la lista de valores de resolución es el valor del tipo de slot. Por ejemplo, si el usuario introduce "divertido", el campo `slots` contiene "divertido" y la primera entrada del campo `slotDetails` es "comedia". Puede configurar `valueSelectionStrategy` al crear o actualizar un tipo de slot con la operación PutSlotType, de manera que el valor de slot se rellene con el primer valor de la lista de resolución.

Si utiliza una función de Lambda, el evento de entrada a la función incluye una lista de resolución llamada `slotDetails`. El siguiente ejemplo muestra la ranura y la sección de detalles de la ranura de la entrada a una función de Lambda:

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

Para cada tipo de slot puede definir un máximo de 10 000 valores y sinónimos. Cada bot puede contender un total de 50 000 valores de tipo de slot y sinónimos. Por ejemplo, puede tener cinco tipos de slot, cada uno con 5000 valores y 5000 sinónimos, o puede tener diez tipos de slot, cada uno con 2500 valores y 2500 sinónimos. Si supera estos límites, obtendrá una `LimitExceededException` cuando llame a la operación PutBot.

Ofuscación de ranura

Amazon Lex le permite ofuscar u ocultar el contenido de las ranuras para que el contenido no sea visible. Si desea proteger la información confidencial recopilada como valores de ranura, puede habilitar la ofuscación de ranuras para enmascarar estos valores en registros de conversaciones.

Cuando elige ofuscar los valores de ranura, Amazon Lex reemplaza el valor de la ranura por el nombre de la ranura en los registros de conversaciones. Para una ranura llamada `full_name`, el valor de la ranura se ofuscaría de la siguiente manera:

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

Si un enunciado contiene caracteres entre llaves ({}), Amazon Lex oculta los caracteres entre llaves con dos barras invertidas (\\). Por ejemplo, el texto `{John Stiles}` se ofusca de la siguiente manera:

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

Los valores de slot se ofuscan en los registros de conversación. Los valores de slot siguen estando disponibles en la respuesta de las operaciones `PostContent` y `PostText`, y los valores de slot están disponibles para las funciones de Lambda de validación y cumplimiento. Si utiliza valores de ranura en sus mensajes o respuestas, esos valores de ranura no se ofuscan en los registros de conversación.

En el primer turno de una conversación, Amazon Lex ofusca los valores de ranura si reconoce una ranura y un valor de ranura en el enunciado. Si no se reconoce ningún valor de ranura, Amazon Lex no ofusca el enunciado.

En el segundo turno y los posteriores, Amazon Lex sabe cuál es la ranura que debe obtener y si el valor de ranura debe ofuscarse. Si Amazon Lex reconoce el valor de la ranura, el valor se ofusca. Si Amazon Lex no reconoce un valor, se ofusca todo el enunciado. Los valores de ranura en enunciados perdidos no se ofuscarán.

Amazon Lex tampoco ofusca los valores de ranura que almacena en atributos de solicitud o sesión. Si está almacenando valores de ranura que deben ofuscarse como un atributo, debe cifrar u ofuscar el valor de otro modo.

Amazon Lex no ofusca el valor de la ranura en el audio. Se ofusca el valor de ranura en la transcripción de audio.

No necesitas ofuscar todas las ranuras de un bot. Puede elegir qué ranuras se ofuscan con la consola o la API de Amazon Lex. En la consola, elija Ofuscación de ranura en la configuración de una ranura. Si está utilizando la API, establezca el `obfuscationSetting` campo de la ranura en `DEFAULT_OBFUSCATION` cuando llame a la PutIntent operación.

Análisis de opiniones

Puede utilizar el análisis de emociones para determinar los sentimientos expresados por un usuario. Con la información de las emociones, puede administrar el flujo de la conversación o realizar análisis tras la llamada. Por ejemplo, si la opinión del usuario es negativa, puede crear un flujo para transferir la conversación a un agente humano.

Amazon Lex se integra con Amazon Comprehend para detectar los sentimientos de los usuarios. La respuesta de Amazon Comprehend indica si el sentimiento general del texto es positivo, neutro, negativo o mixto. La respuesta contiene la opinión más probable del enunciado del usuario y las puntuaciones para cada una de las categorías de sentimiento. La puntuación representa la probabilidad de que la opinión se haya detectado correctamente.

Para habilitar el análisis de sentimientos de un bot, utilice la consola o la API de Amazon Lex. En la consola de Amazon Lex, elija la pestaña Configuración del bot y establezca la opción Análisis de opiniones en Sí. Si está utilizando la API, llame a la operación PutBot con el campo `detectSentiment` establecido en `true`.

Cuando se habilita el análisis de opiniones, la respuesta de las operaciones PostText y PostContent devuelve un campo llamado `sentimentResponse` en la respuesta del bot junto con otros metadatos. El campo `sentimentResponse` tiene a su vez dos campos: `SentimentLabel` y `SentimentScore`, que contienen el resultado del análisis de opinión. Si utiliza una función de Lambda, el campo `sentimentResponse` se incluye en los datos de evento enviados a la función.

A continuación, se muestra un ejemplo del campo `sentimentResponse` que se devuelve en la respuesta de `PostContent` o `PostText`. El campo `SentimentScore` es una cadena que contiene las puntuaciones de la respuesta.

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

Amazon Lex llama a Amazon Comprehend en su nombre para determinar el sentimiento en cada enunciado procesado por el bot. Al habilitar el análisis de opiniones, acepta los términos y acuerdos de servicio de Amazon Comprehend. Para obtener más información general acerca de Amazon Comprehend, consulte Precios de Amazon Comprehend.

Para obtener más información sobre cómo funciona el análisis de opiniones de Amazon Comprehend, consulte Determinación de opiniones en la Guía para desarrolladores de Amazon Comprehend.

Etiquetado de los recursos de Amazon Lex

Puede asignar metadatos a cada recurso como etiquetas, lo que le ayudará a administrar los bots, alias de bots y canales de bots de Amazon Lex. Una etiqueta es una marca que se asigna a un recurso de AWS. Cada etiqueta consta de una key (clave) y un value (valor).

Las etiquetas le permiten clasificar los recursos de AWS de diversas maneras, por ejemplo, según su finalidad, propietario o aplicación. Las etiquetas le ayudan a:

- Identificar y organizar sus recursos de AWS. Muchos recursos de AWS admiten el etiquetado, por lo que puede asignar la misma etiqueta a los recursos de distintos servicios para indicar que los recursos están relacionados. Por ejemplo, puede etiquetar un bot y las funciones de Lambda que utiliza con la misma etiqueta.
- Asigne costos. Las etiquetas se activan en el panel de AWS Billing and Cost Management. AWS usa las etiquetas para clasificar los costos y enviar un informe mensual de asignación de costos. En el caso de Amazon Lex, puede asignar costos para cada alias mediante etiquetas específicas para el alias, excepto para el alias `$LATEST`. Asigne costos para el alias `$LATEST` con etiquetas para bots de Amazon Lex. Para obtener más información, consulte Uso de etiquetas de asignación de costos en la Guía del usuario de AWS Billing and Cost Management.
- Controle el acceso a los recursos. Las etiquetas de Amazon Lex le permiten crear políticas para controlar el acceso a los recursos de Amazon Lex. Estas políticas se pueden asociar a un rol o usuario de IAM para habilitar el control de acceso basado en etiquetas. Para obtener más información, consulte ABAC con Amazon Lex. Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Usar una etiqueta para acceder a un recurso.

Puede trabajar con etiquetas desde la AWS Management Console, la AWS Command Line Interface o la API de Amazon Lex.

Etiquetado de los recursos de

Si está utilizando la consola de Amazon Lex, puede etiquetar recursos al crearlos o agregar las etiquetas más tarde. También puede utilizar la consola para actualizar o quitar etiquetas existentes.

Si está utilizando la AWS CLI o la API de Amazon Lex, utilice las siguientes operaciones para administrar las etiquetas de los recursos:

- ListTagsForResource: muestra las etiquetas asociadas a un recurso.
- PutBot y PutBotAlias: aplica etiquetas al crear un bot o un alias de bot.
- TagResource: agrega etiquetas a un recurso existente y las modifica.
- UntagResource: elimina las etiquetas de un recurso.

Los siguientes recursos de Amazon Lex admiten el etiquetado:

- Bots: utilice un nombre de recurso de Amazon (ARN) como el siguiente:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}`
- Alias de bot: use un ARN como el siguiente:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}:${bot-alias}`
- Canales de bot: use un ARN como el siguiente:
  - `arn:${partition}:lex:${region}:${account}:bot-channel:${bot-name}:${bot-alias}:${channel-name}`

Restricciones de las etiquetas

Las siguientes restricciones básicas se aplican a las etiquetas en los recursos de Amazon Lex:

- Número máximo de etiquetas: 50.
- Longitud máxima de la clave: 128 caracteres
- Longitud máxima del valor: 256 caracteres
- Caracteres válidos para claves y valores: a-z, A-Z, 0-9, espacio y los siguientes caracteres: _ . : / = + - y @
- Las claves y los valores distinguen entre mayúsculas y minúsculas.

- No utilice `aws:` como prefijo para claves, ya que su uso está reservado a AWS.

Etiquetado de recursos (Consola)

Puede usar la consola para administrar etiquetas en un bot, un alias de bot o un recurso de canal bot. Puede agregar etiquetas al crear un recurso, o puede agregar, modificar o quitar etiquetas de los recursos existentes.

Para agregar una etiqueta al crear un bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija Create (Crear) para crear un nuevo bot.
3. En la parte inferior de la página Create your bot (Crear tu bot), elija Tags (Etiquetas).
4. Elija Add tag (Agregar etiqueta) y agregue una o más etiquetas al bot. Puede añadir hasta 50 etiquetas.

Para agregar una etiqueta al crear un alias de bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija el bot al que desee agregar al alias del bot.
3. Elija Settings.
4. Agregue el nombre del alias, elija la versión del bot y, a continuación, elija Add tags (Agregar etiquetas).
5. Elija Add tag (Agregar etiqueta) y agregue una o más etiquetas al alias del bot. Puede añadir hasta 50 etiquetas.

Para agregar una etiqueta al crear un canal de bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija el bot que desea agregar al canal de bot.
3. Elija Channels (Canales) y, a continuación, elija el canal que desea agregar.
4. Agregue los detalles del canal de bot y, a continuación, elija Tags (Etiquetas).

5. Elija Add tag (Agregar etiqueta) y agregue una o más etiquetas al canal de bot. Puede añadir hasta 50 etiquetas.

Para agregar una etiqueta al importar un bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija Actions (Acciones) y, a continuación, Import (Importar).
3. Elija el archivo ZIP para importar el bot.
4. Elija Tags (Etiquetas), y, a continuación, elija Add tag (Agregar etiqueta) para agregar una o más etiquetas al bot. Puede añadir hasta 50 etiquetas.

Para agregar, eliminar o modificar una etiqueta en un bot existente

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el menú de la izquierda, elija Bots y, a continuación, elija el bot que desea modificar.
3. Elija Settings (Configuración) y, a continuación, en el menú de la izquierda, elija General.
4. Elija Tags (Etiquetas) y, a continuación, agregue, modifique o elimine etiquetas para el bot.

Para agregar, quitar o modificar una etiqueta en un alias de bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el menú de la izquierda, elija Bots y, a continuación, elija el bot que desea modificar.
3. Elija Settings (Configuración) y, a continuación, en el menú de la izquierda, elija Alias.
4. Elija Manage tags (Administrar etiquetas) para el alias que desee modificar y, a continuación, agregue, modifique o quite etiquetas para el alias de bot.

Para agregar, quitar o modificar una etiqueta en un canal de bot existente

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el menú de la izquierda, elija Bots y, a continuación, elija el bot que desea modificar.

3. Elija Channels.
4. Elija Tags (Etiquetas) y, a continuación, agregue, modifique o quite etiquetas del canal de bot.

Etiquetado de recursos (AWS CLI)

Puede usar la AWS CLI para administrar etiquetas en un bot, un alias de bot o un recurso de canal de bot. Puede agregar etiquetas cuando cree un bot o un alias de bot, o puede agregar, modificar o quitar etiquetas de un bot, un alias de bot o un canal de bot.

Todos los ejemplos están formateados para Linux y macOS. Para utilizar el comando en Windows, reemplace el carácter de continuación de Linux (\) por un símbolo de intercalación (^).

Para agregar una etiqueta al crear un bot

- El siguiente comando abreviado `put-bot` de la AWS CLI muestra los parámetros que debe utilizar para agregar una etiqueta al crear un bot. Para crear realmente un bot, debe proporcionar otros parámetros. Para obtener más información, consulte Paso 4: Introducción (AWS CLI).

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

Para agregar una etiqueta al crear un alias de bot

- El siguiente comando abreviado `put-bot-alias` de la AWS CLI muestra los parámetros que debe utilizar para agregar una etiqueta al crear un alias de bot. Para crear realmente un alias de bot, debe proporcionar otros parámetros. Para obtener más información, consulte Ejercicio 5: Crear un alias (AWS CLI).

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

Para mostrar etiquetas en un recurso

- Utilice el comando `list-tags-for-resource` de la AWS CLI para mostrar los recursos asociados con un bot, alias de bot, canal de bot.

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

Para agregar o modificar etiquetas en un recurso

- Utilice el comando `tag-resource` de la AWS CLI para agregar o modificar un bot, alias de bot o 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"}]'
```

Para quitar etiquetas de un recurso

- Utilice el comando `untag-resource` de la AWS CLI para quitar etiquetas de un bot, alias de bot o canal de bot.

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

Introducción a Amazon Lex

Amazon Lex ofrece operaciones de la API que puede integrar con las aplicaciones existentes. Para ver una lista de las operaciones admitidas, consulte Referencia de la API. Puede utilizar cualquiera de las siguientes opciones:

- SDK de AWS: cuando se usan los SDK, las solicitudes enviadas a Amazon Lex se firmarán y autenticarán automáticamente con las credenciales proporcionadas. Esta es la opción recomendada para compilar las aplicaciones.
- AWS CLI — Puede utilizarla AWS CLI para acceder a cualquier función de Amazon Lex sin tener que escribir ningún código.
- Consola de AWS: la consola es la forma más sencilla de comenzar a probar y utilizar Amazon Lex.

Si es la primera vez que utiliza Amazon Lex, le recomendamos que lea Funcionamiento de Amazon Lex antes de continuar.

Temas

Paso 1: configurar una AWS cuenta y crear un usuario administrador

Antes de usar Amazon Lex por primera vez, complete las siguientes tareas:

1. Inscríbase en AWS
2. Creación de un usuario

Inscríbase en AWS

Si ya tienes una AWS cuenta, omite esta tarea.

Cuando te registras en Amazon Web Services (AWS), tu AWS cuenta se registra automáticamente en todos los servicios de Amazon Lex AWS, incluido Amazon Lex. Solo se le cobrará por los servicios que utilice.

Con Amazon Lex, paga solo por los recursos que usa. Si es cliente nuevo de AWS , puede comenzar con Amazon Lex de forma gratuita. Para obtener más información, consulte Nivel de uso gratuito de AWS.

Si ya tienes una AWS cuenta, pasa a la siguiente tarea. Si no dispone de una cuenta de AWS , utilice el siguiente procedimiento para crear una.

Para crear una AWS cuenta

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en una Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea una. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

Anota el ID de tu AWS cuenta porque lo necesitarás para la siguiente tarea.

Creación de un usuario

Los servicios de AWS, como Amazon Lex, requieren que proporcione credenciales al acceder a ellos para que el servicio pueda determinar si tiene permisos para acceder a los recursos que son propiedad de ese servicio. La consola requiere que especifique la contraseña. Sin embargo, no le recomendamos que acceda AWS con las credenciales de su AWS cuenta. Le recomendamos que utilice en su lugar:

- Usa AWS Identity and Access Management (IAM) para crear un usuario
- Añada el usuario a un grupo de IAM con permisos administrativos.
- Conceda permisos administrativos al usuario que ha creado.

A continuación, puede acceder AWS mediante una URL especial y las credenciales del usuario.

En los ejercicios de introducción de esta guía se presupone que tiene un usuario (`adminuser`) con privilegios de administrador. Siga el procedimiento para crear `adminuser` en su cuenta.

Para crear un usuario administrador e iniciar sesión en la consola

1. Cree un usuario administrador llamado `adminuser` en su cuenta de AWS . Para obtener instrucciones, consulte Creación de su primer grupo de administradores y usuarios de IAM en la Guía del usuario de IAM.
2. Como usuario, puede iniciar sesión en el AWS Management Console mediante una URL especial. Para obtener más información, consulte Cómo inician sesión los usuarios en su cuenta en la Guía del usuario de IAM.

Para obtener más información sobre IAM, consulte lo siguiente:

- AWS Identity and Access Management (IAM)
- Introducción
- Guía del usuario de IAM

Paso siguiente

Paso 2: configurar el AWS Command Line Interface

Paso 2: configurar el AWS Command Line Interface

Si prefiere utilizar Amazon Lex con AWS Command Line Interface (AWS CLI), descárguelo y configúrelo.

> **Important**
>
> No los necesita AWS CLI para realizar los pasos de los ejercicios de introducción. Sin embargo, en algunos de los ejercicios posteriores de esta guía se utiliza la AWS CLI. Si prefiere empezar con la consola, omita este paso y vaya a Paso 3: Introducción (consola). Más adelante, cuando lo necesites AWS CLI, vuelve aquí para configurarlo.

Para configurar el AWS CLI

1. Descargue y configure la AWS CLI. Para obtener instrucciones, consulte los siguientes temas en la Guía del usuario de AWS Command Line Interface :

   - Cómo configurarse con el AWS Command Line Interface
   - Configuración de la AWS Command Line Interface
2. Agregue un perfil con nombre para el usuario administrador al final del archivo de AWS CLI configuración. Este perfil se utiliza al ejecutar AWS CLI comandos. Para obtener más información sobre los perfiles con nombre, consulte Perfiles con nombre en la Guía del usuario de la 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 obtener una lista de AWS las regiones disponibles, consulte Regiones y puntos finales en la Referencia general de Amazon Web Services.
3. Compruebe la configuración; para ello, escriba el comando de ayuda en el símbolo del sistema:

   ```
   aws help
   ```

Paso 3: Introducción (consola)

Paso 3: Introducción (consola)

La forma más sencilla de aprender a utilizar Amazon Lex es mediante la consola. Para comenzar, hemos creado los siguientes ejercicios, en todos los cuales se utiliza la consola:

- Ejercicio 1: crear un bot de Amazon Lex mediante un esquema, un bot predefinido que proporciona toda la configuración necesaria para el bot. Solo tiene que hacer un mínimo de trabajo para probar la end-to-end configuración.

  Además, se utiliza el esquema de funciones Lambda, proporcionado por AWS Lambda, para crear una función Lambda. La función es un enlace de código que utiliza código predefinido que es compatible con su bot.

- Ejercicio 2: crear y configurar manualmente un bot personalizado. También creará una función de Lambda como enlace de código. Se proporciona un código de muestra.
- Ejercicio 3: publicar un bot y crear una nueva versión. En este ejercicio también va a crear un alias que apunte a la versión del bot.

Temas

Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola)

En este ejercicio, hará lo siguiente:

- Crear su primer bot de Amazon Lex y probarlo en la consola de Amazon Lex.

  En este ejercicio se va a utilizar el proyecto OrderFlowers. Para obtener más información sobre los proyectos, consulte Esquemas de Amazon Lex y AWS Lambda.

- Crear una función de AWS Lambda y probarla en la consola de Lambda. Cuando procesa una solicitud, el bot llama a esta función de Lambda. En este ejercicio, utilizará un esquema de Lambda (lex-order-flowers-python) proporcionado en la consola de AWS Lambda para crear la función de Lambda. El código del esquema indica cómo se puede utilizar la misma función de Lambda para la inicialización y validación, así como para llevar a cabo la intención `OrderFlowers`.

- Actualizar el bot para agregar la función de Lambda como el enlace de código que permite cumplir con la intención. Probar la experiencia integral.

En las siguientes secciones se explica qué hacen los proyectos.

Bot de Amazon Lex: descripción general del esquema

Puede utilizar el esquema OrderFlowers para crear un bot de Amazon Lex. Para obtener más información acerca de la estructura de los bots, consulte Funcionamiento de Amazon Lex. El bot está preconfigurado del modo siguiente:

- Intención: OrderFlowers
- Tipos de slot: un tipo de slot personalizado denominado `FlowerTypes` con los valores de enumeración: `roses`, `lilies` y `tulips`.
- Slots: la intención requiere la siguiente información (es decir, slots) para que el bot pueda llevar a cabo la intención.
  - `PickupTime` (AMAZON.TIME built-in type)
  - `FlowerType` (tipo personalizado FlowerTypes)
  - `PickupDate` (tipo integrado AMAZON.DATE)
- Enunciados: los siguientes enunciados de muestra identifican la intención del usuario:
  - «Me gustaría recoger unas flores».
  - «Me gustaría pedir unas flores».
- Preguntas: una vez que el bot identifica la intención, utiliza las siguientes preguntas para rellenar los slots:
  - Pregunta para el slot `FlowerType`: «¿Qué tipo de flores deseas pedir?»
  - Pregunta para el slot `PickupDate`: «¿Qué día deseas recoger las {TipoDeFlores}»?
  - Pregunta para el slot `PickupTime`: «¿A qué hora deseas recoger las {TipoDeFlores}?»
  - Instrucción de confirmación: «Bien, tus {FlowerType} estarán listas para su recogida a las {PickupTime} del {PickupDate}. ¿Le parece bien?"

Función AWS Lambda: resumen del proyecto

La función de Lambda de este ejercicio realiza tanto las tareas de inicialización y validación como las de cumplimiento. Por lo tanto, después de crear la función de Lambda, deberá actualizar la configuración de la intención y especificar la misma función de Lambda como enlace de código para administrar las tareas de inicialización o validación y de cumplimiento.

- Como enlace de código de inicialización y validación, la función de Lambda realiza la validación básica. Por ejemplo, si el usuario indica una hora de recogida que está fuera del horario de

apertura habitual, la función de Lambda indica a Amazon Lex que solicite una hora nueva al usuario.

- Como parte del enlace de código de cumplimiento, la función de Lambda devuelve un mensaje de resumen que indica que se ha realizado el pedido de flores (es decir, que se ha cumplido con la intención).

Paso siguiente

Paso 1: creación de un bot de Amazon Lex (consola)

Paso 1: creación de un bot de Amazon Lex (consola)

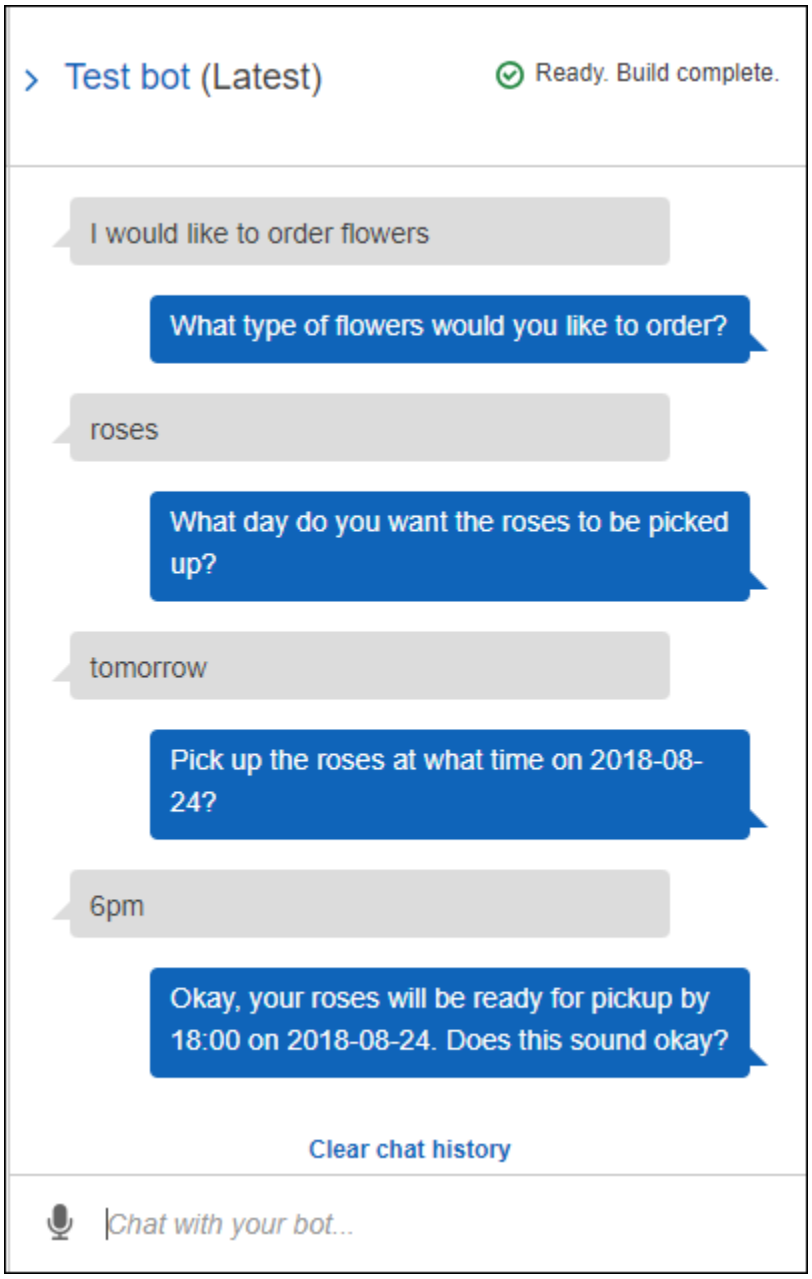
En este ejercicio, se crea un bot para hacer un pedido de flores, llamado OrderFlowersBot.

Creación de un bot de Amazon Lex (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Si es su primer bot, elija Get Started (Empezar); de lo contrario, en la página Bots, elija Create (Crear).
3. En la página Create your Lex bot, proporcione la información siguiente y, a continuación, seleccione Create.
   - Elija el proyecto OrderFlowers.
   - Deje el nombre de bot predeterminado (OrderFlowers).
   - En COPPA, elija **No**.
   - En Almacenamiento de enunciados del usuario, elija la respuesta adecuada.
4. Seleccione Create (Crear). La consola realiza las solicitudes necesarias para que Amazon Lex guarde la configuración. A continuación, la consola muestra la ventana del editor de bots.
5. Espere a que se le confirme que se creó su bot.
6. Pruebe el bot.

> **Note**
>
> Puede probar el bot escribiendo el texto en la ventana de prueba o, con navegadores compatibles, seleccionando el botón de micrófono en la misma ventana y hablando.

Utilice el siguiente texto de ejemplo para participar en la conversación con el bot para pedir flores:



A partir de esta entrada, el bot deduce la intención `OrderFlowers` y solicita los datos de slot. Cuando proporcione todos los datos necesarios del slot, el bot llevará a cabo la intención (`OrderFlowers`) devolviendo toda la información a la aplicación cliente (en este caso, la consola). La consola simplemente muestra la información en la ventana de prueba.

En concreto:

- En la instrucción "¿Qué día desea recoger las rosas?" aparece el término "rosas" porque la pregunta del slot `pickupDate` está configurada con sustituciones, `{FlowerType}`. Compruébelo en la consola.
- La instrucción "Bien, sus rosas estarán listas..." es lo que ha configurado para la confirmación.
- La última instrucción ("`FlowerType:roses...`") incluye solo los datos de slot que se devuelven al cliente, en este caso, en la ventana de prueba. En el siguiente ejercicio, utilizará una función de Lambda para llevar a cabo la intención, en cuyo caso obtendrá un mensaje que indica que se ha efectuado el pedido.

Paso siguiente

Paso 2 (opcional): revisión de los detalles del flujo de información (consola)

Paso 2 (opcional): revisión de los detalles del flujo de información (consola)

En esta sección se explica el flujo de información entre un cliente y Amazon Lex para cada entrada de usuario en nuestra conversación de ejemplo.

En el ejemplo se utiliza la ventana de pruebas de la consola para mostrar la conversación con el bot.

Visualización de la ventana de pruebas de Amazon Lex

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija el bot que quiera probar.
3. En la parte derecha de la consola, elija Probar bot de chat.

Para ver el flujo de información de contenido hablado o escrito, elija el tema correspondiente.

Temas

Paso 2a (opcional): revisión de los detalles del flujo de información escrita (consola)

En esta sección se explica el flujo de información entre el cliente y Amazon Lex cuando el cliente utiliza la voz para enviar solicitudes. Para obtener más información, consulte PostContent.

1. El usuario dice: Me gustaría pedir unas flores.

   a. El cliente (la consola) envía la siguiente solicitud PostContent a 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
```

   Tanto la URI de la solicitud como el cuerpo proporcionan información a Amazon Lex:

   - URI de la solicitud: proporciona el nombre del bot (`OrderFlowers`), su alias (`$LATEST`) y el nombre del usuario (una cadena aleatoria que identifica al usuario). `content` indica que se trata de una solicitud de la API `PostContent` (no una solicitud `PostText`).
   - Encabezados de solicitudes
     - `x-amz-lex-session-attributes`: el valor con codificación base64 representa “{}”. Cuando el cliente realiza la primera solicitud, no existen atributos de la sesión.
     - `Content-Type`: refleja el formato de audio.
   - Cuerpo de la solicitud: la transmisión de audio de la entrada del usuario ("Me gustaría pedir unas flores.").

   > **Note**
   >
   > En caso de que el usuario elija enviar texto ("Me gustaría pedir unas flores") a la API `PostContent` en lugar de hablar, el cuerpo de la solicitud será la entrada del usuario. El encabezado `Content-Type` se fija en consecuencia:
   >
   > ```
   > 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. Amazon Lex deduce la intención (`OrderFlowers`) a partir de la transmisión de entrada. A continuación, elige uno de los slots de la intención (en este caso, `FlowerType`) y una de sus preguntas para obtener valores y, luego, envía una respuesta con los siguientes encabezados:

```
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==
```

Los valores de encabezado proporcionan la siguiente información:

- `x-amz-lex-input-transcript`: proporciona la transcripción del audio (entrada del usuario) de la solicitud
- `x-amz-lex-message`: proporciona la transcripción del audio que devuelve Amazon Lex en la respuesta
- `x-amz-lex-slots`: la versión de la codificación de base64 de los slots y los valores:

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

- `x-amz-lex-session-attributes`: la versión de la codificación de base64 de los atributos de la sesión ({})

El cliente reproduce el audio en el cuerpo de la respuesta.

2. El usuario dice: rosas

a. El cliente (la consola) envía la siguiente solicitud PostContent a 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")
```

El cuerpo de la solicitud es la transmisión de audio de la entrada del usuario (rosas). No hay `sessionAttributes`.

b. Amazon Lex interpreta la transmisión de entrada en el contexto de la intención actual (recuerda que ha solicitado al usuario información sobre la ranura `FlowerType`). En primer lugar, Amazon Lex actualiza el valor de ranura de la intención actual. Después, elige otro slot (`PickupDate`), junto con uno de los mensajes de pregunta (¿Cuándo desea pasar a por las rosas?) y devuelve una respuesta con los siguientes encabezados:

```
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=
```

Los valores de encabezado proporcionan la siguiente información:

- `x-amz-lex-slots`: la versión de la codificación de base64 de los slots y los valores:

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

- `x-amz-lex-session-attributes`: la versión de la codificación de base64 de los atributos de la sesión ({})

El cliente reproduce el audio en el cuerpo de la respuesta.

3. El usuario dice: mañana

   a. El cliente (la consola) envía la siguiente solicitud PostContent a 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")
```

El cuerpo de la solicitud es la transmisión de audio de la entrada del usuario ("mañana"). No hay `sessionAttributes`.

b. Amazon Lex interpreta la transmisión de entrada en el contexto de la intención actual (recuerda que ha solicitado al usuario información sobre la ranura `PickupDate`). Amazon Lex actualiza el valor de ranura (`PickupDate`) de la intención actual. A continuación, elige otro slot para obtener el valor (`PickupTime`) y una de las preguntas para obtener valores (¿Cuándo desea recoger las rosas el 18/03/2017?), y devuelve una respuesta con los siguientes encabezados:

```
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
```

Los valores de encabezado proporcionan la siguiente información:

- `x-amz-lex-slots`: la versión de la codificación de base64 de los slots y los valores:

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

- `x-amz-lex-session-attributes`: la versión de la codificación de base64 de los atributos de la sesión ({})

El cliente reproduce el audio en el cuerpo de la respuesta.

4. El usuario dice: 18:00 h

a. El cliente (la consola) envía la siguiente solicitud PostContent a 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")
```

El cuerpo de la solicitud es la transmisión de audio de la entrada del usuario ("18:00 h"). No hay `sessionAttributes`.

b. Amazon Lex interpreta la transmisión de entrada en el contexto de la intención actual (recuerda que ha solicitado al usuario información sobre la ranura `PickupTime`). Primero actualiza el valor de slot de la intención actual.

Ahora Amazon Lex detecta que tiene información para todas las ranuras. Sin embargo, la intención `OrderFlowers` se ha configurado con un mensaje de confirmación. Por lo tanto, Amazon Lex necesita una confirmación explícita del usuario antes de cumplir con la intención. Envía una respuesta con los siguientes encabezados para solicitar la confirmación antes de encargar las 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
```

Los valores de encabezado proporcionan la siguiente información:

- `x-amz-lex-slots`: la versión de la codificación de base64 de los slots y los valores:

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

- `x-amz-lex-session-attributes`: la versión de la codificación de base64 de los atributos de la sesión ({})

El cliente reproduce el audio en el cuerpo de la respuesta.

5. El usuario dice: Sí

   a. El cliente (la consola) envía la siguiente solicitud PostContent a 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")
```

   El cuerpo de la solicitud es la transmisión de audio de la entrada del usuario ("Sí"). No hay `sessionAttributes`.

   b. Amazon Lex interpreta la transmisión de entrada y entiende que el usuario desea continuar con el pedido. La intención `OrderFlowers` se ha configurado con `ReturnIntent` como la actividad de cumplimiento. Esto provoca que Amazon Lex devuelva todos los datos de la intención al cliente. Amazon Lex devuelve una respuesta con lo siguiente:

```
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
```

   El encabezado de la respuesta `x-amz-lex-dialog-state` está configurado en `ReadyForFulfillment`. Ahora el cliente podrá llevar a cabo la intención.

6. Ahora, pruebe de nuevo el bot. Para establecer un nuevo contexto (usuario), elija el enlace Clear en la consola. Proporcione datos para la intención `OrderFlowers` e incluya algunos datos no válidos. Por ejemplo:

- Jazmín como tipo de flor (este tipo de flor no se admite)
- Ayer como el día en que desea recoger las flores

Observe que el bot acepta estos valores, ya que no tiene ningún código para inicializar y validar los datos de usuario. En la siguiente sección, añadirá una función de Lambda para que lo haga. Tenga en cuenta lo siguiente en relación con la función de Lambda:

- Valida los datos de slot después de cada entrada de usuario. Gestiona la intención al final. Es decir, el bot procesa el pedido de flores y devuelve un mensaje al usuario en lugar de simplemente devolver los datos de slot al cliente. Para obtener más información, consulte Uso de funciones de Lambda.
- También establece los atributos de la sesión. Para obtener más información acerca de los atributos de la sesión, consulte PostText.

  Una vez completada la sección Introducción, puede realizar los ejercicios adicionales (Ejemplos adicionales: creación de bots de Amazon Lex). Reserva de viaje usa los atributos de sesión para compartir información entre intenciones y entablar una conversación dinámica con el usuario.

Paso siguiente

Paso 3: creación de una función de Lambda (consola)

Paso 2b (opcional): revisión de los detalles del flujo de información escrita (consola)

En esta sección se explica el flujo de información entre el cliente y Amazon Lex en el que el cliente utiliza la API `PostText` para enviar solicitudes. Para obtener más información, consulte PostText.

1. El usuario escribe: Me gustaría pedir unas flores

   a. El cliente (la consola) envía la siguiente solicitud PostText a 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": {}
```

```
}
```

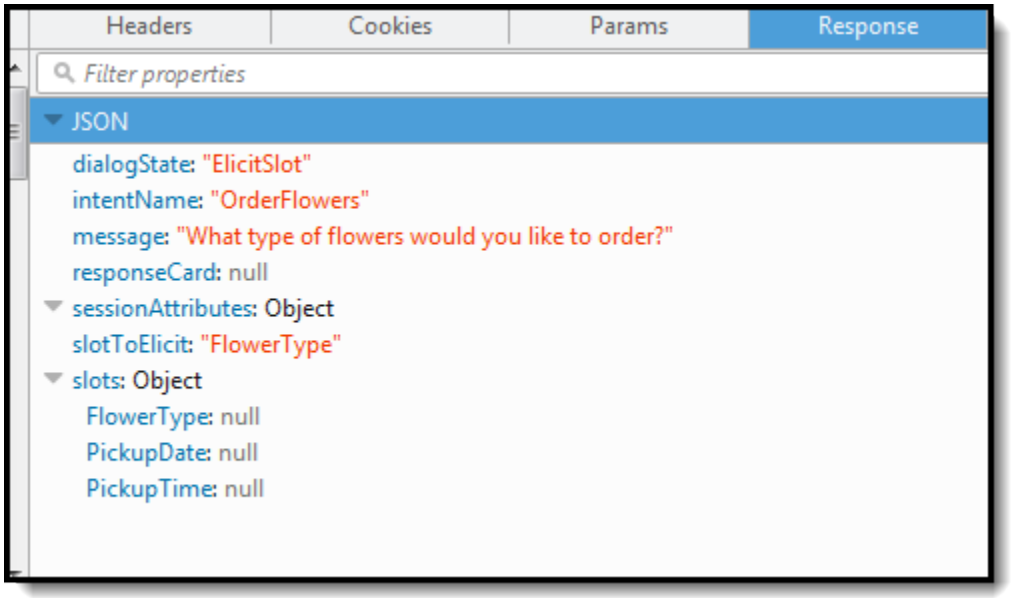
Tanto la URI de la solicitud como el cuerpo proporcionan información a Amazon Lex:

- URI de la solicitud: proporciona el nombre del bot (`OrderFlowers`), el alias del bot ($LATEST) y el nombre de usuario (una cadena aleatoria que identifica al usuario). El `text` de cola indica que se trata de una solicitud de API `PostText` (no `PostContent`).

- Cuerpo de la solicitud: incluye la entrada del usuario (`inputText`). No hay `sessionAttributes`. Cuando el cliente realiza la primera solicitud, no existen atributos de la sesión. La función Lambda los inicia más adelante.

b. Amazon Lex deduce la intención (`OrderFlowers`) a partir de `inputText`. Esta intención no tiene ningún enlace de código (es decir, funciones de Lambda) para la inicialización y validación de la entrada del usuario o el cumplimiento.

   Amazon Lex elige una de las ranuras de la intención (`FlowerType`) para obtener el valor. También selecciona una de las preguntas para obtener valores para el slot (que forman parte de la configuración de la intención) y, a continuación, envía la siguiente respuesta al cliente. La consola muestra el mensaje en la respuesta al usuario.



   El cliente muestra el mensaje en la respuesta.

2. El usuario escribe: rosas

a. El cliente (la consola) envía la siguiente solicitud PostText a Amazon Lex:

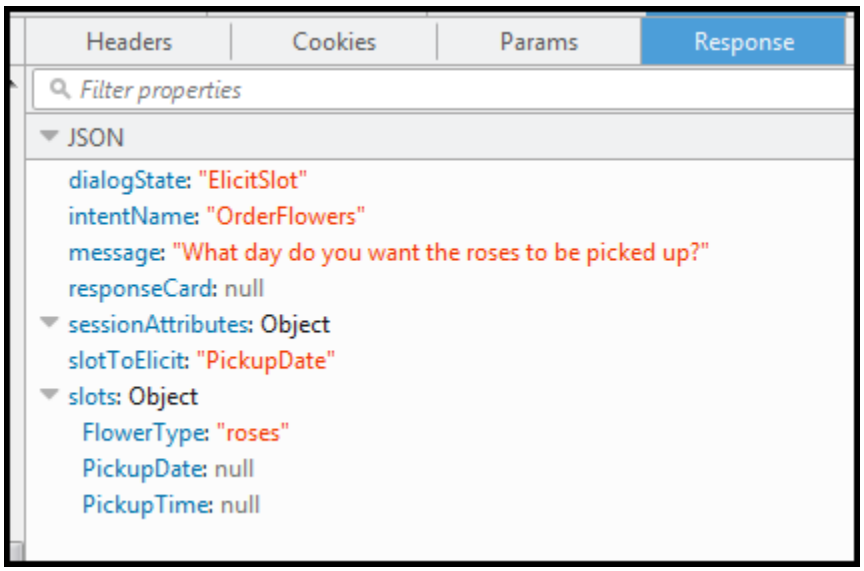
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

El `inputText` del cuerpo de la solicitud proporciona la entrada del usuario. No hay `sessionAttributes`.

b. En primer lugar, Amazon Lex interpreta `inputText` en el contexto de la intención actual (el servicio recuerda que ha solicitado al usuario información sobre la ranura `FlowerType`). Amazon Lex actualiza primero el valor del slot de la intención actual y elige otro slot (`PickupDate`) junto con uno de los mensajes de pregunta (¿Qué día desea recoger las rosas?)para el slot.

A continuación, Amazon Lex devuelve la siguiente respuesta:



El cliente muestra el mensaje en la respuesta.

3. El usuario escribe: mañana

a. El cliente (la consola) envía la siguiente solicitud PostText a Amazon Lex:

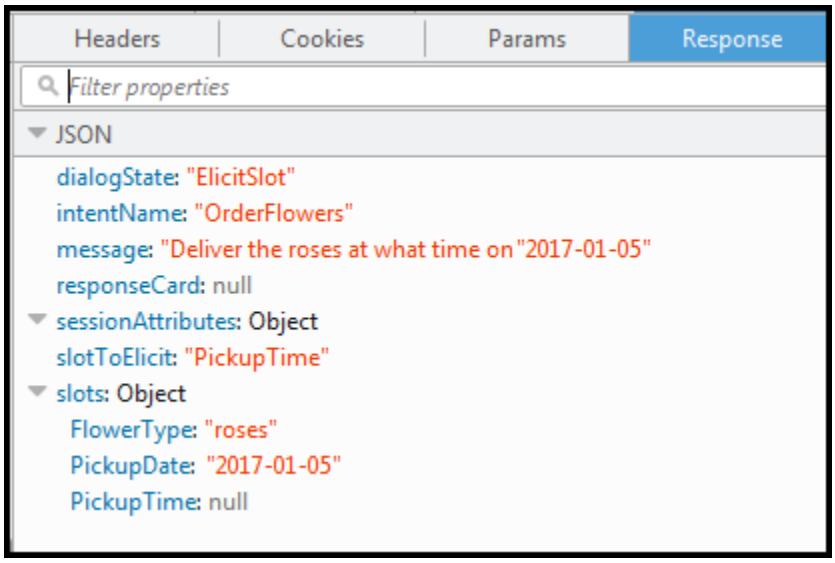
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

El `inputText` del cuerpo de la solicitud proporciona la entrada del usuario. No hay `sessionAttributes`.

b. En primer lugar, Amazon Lex interpreta `inputText` en el contexto de la intención actual (el servicio recuerda que ha solicitado al usuario información sobre la ranura `PickupDate`). Amazon Lex actualiza el valor de ranura (`PickupDate`) de la intención actual. Elige otro slot para obtener un valor para (`PickupTime`). Devuelve una de las preguntas para obtener valores (¿A qué hora quiere que se entreguen las rosas el 05/01/2017?)al cliente.

A continuación, Amazon Lex devuelve la siguiente respuesta:



El cliente muestra el mensaje en la respuesta.

4. El usuario escribe: 18:00 h

a. El cliente (la consola) envía la siguiente solicitud PostText a Amazon Lex:

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

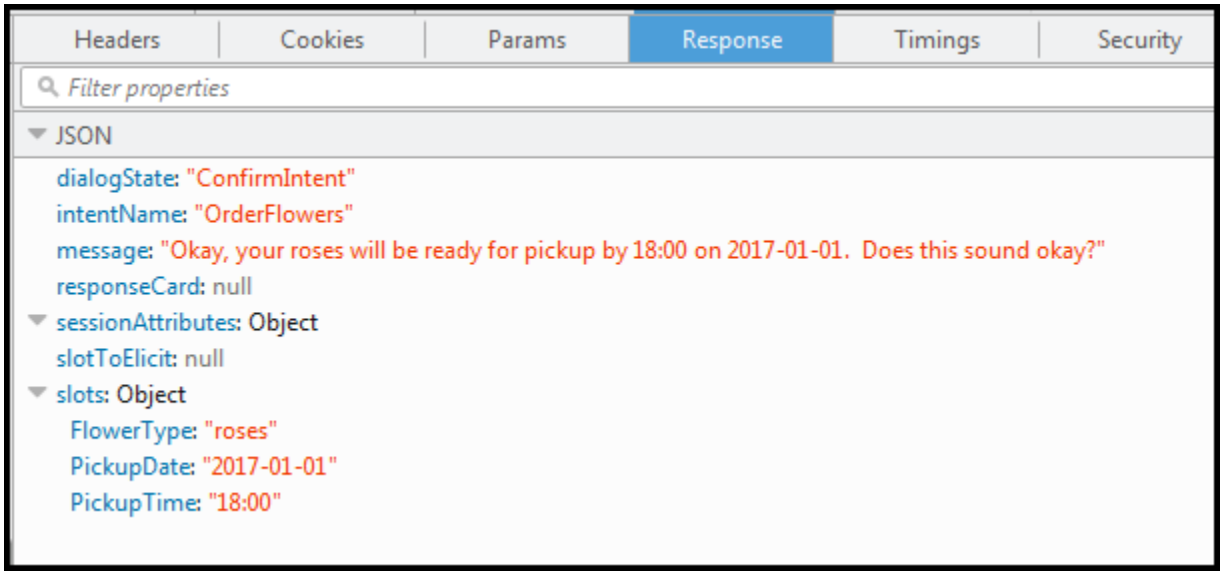
```
"Content-Encoding":"amz-1.0"

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

El `inputText` del cuerpo de la solicitud proporciona la entrada del usuario. No hay `sessionAttributes`.

b. En primer lugar, Amazon Lex interpreta `inputText` en el contexto de la intención actual (el servicio recuerda que ha solicitado al usuario información sobre la ranura `PickupTime`). En primer lugar, Amazon Lex actualiza el valor de ranura de la intención actual. Ahora Amazon Lex detecta que tiene información para todas las ranuras.

La intención `OrderFlowers` se ha configurado con un mensaje de confirmación. Por lo tanto, Amazon Lex necesita una confirmación explícita del usuario antes de cumplir con la intención. Amazon Lex envía el siguiente mensaje al cliente para solicitar su confirmación antes de pedir las flores:



El cliente muestra el mensaje en la respuesta.

5. El usuario escribe: Sí

a. El cliente (la consola) envía la siguiente solicitud PostText a Amazon Lex:

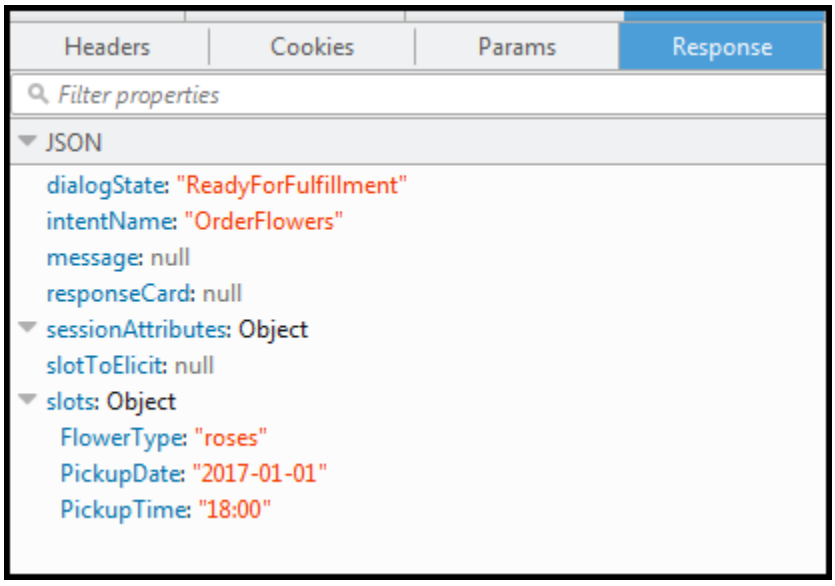
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
```

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

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

El `inputText` del cuerpo de la solicitud proporciona la entrada del usuario. No hay `sessionAttributes`.

b. Amazon Lex interpreta `inputText` en el contexto de la confirmación de la intención actual. Entiende que el usuario desea continuar con el pedido. La intención `OrderFlowers` se ha configurado con `ReturnIntent` como la actividad de cumplimiento (no existe ninguna función de Lambda para cumplir con la intención). Por ello, Amazon Lex devuelve los siguientes datos de ranura al cliente.



Amazon Lex establece `dialogState` en `ReadyForFulfillment`. Ahora el cliente podrá llevar a cabo la intención.

6. Ahora vuelva a probar el bot. Para ello, debe elegir el enlace Clear en la consola y establecer un nuevo contexto (usuario). Al proporcionar datos para la intención Pedir flores, intente que sean datos no válidos. Por ejemplo:

   - Jazmín como tipo de flor (este tipo de flor no se admite).
   - Ayer como el día en que desea recoger las flores.

Observe que el bot acepta estos valores, ya que no tiene ningún código para inicializar y validar los datos de usuario. En la siguiente sección, añadirá una función de Lambda para que lo haga. Tenga en cuenta lo siguiente en relación con la función de Lambda:

- La función de Lambda valida los datos de ranura después de cada entrada del usuario. Gestiona la intención al final. Es decir, el bot procesa el pedido de flores y devuelve un mensaje al usuario en lugar de simplemente devolver los datos de slot al cliente. Para obtener más información, consulte Uso de funciones de Lambda.
- La función de Lambda también establece los atributos de la sesión. Para obtener más información acerca de los atributos de la sesión, consulte PostText.

  Una vez completada la sección Introducción, puede realizar los ejercicios adicionales (Ejemplos adicionales: creación de bots de Amazon Lex). Reserva de viaje usa los atributos de sesión para compartir información entre intenciones y entablar una conversación dinámica con el usuario.

Paso siguiente

Paso 3: creación de una función de Lambda (consola)

Paso 3: creación de una función de Lambda (consola)

Cree una función (con el esquema lex-order-flowers-python) y pruebe a invocarla con los datos de evento de muestra en la consola de AWS Lambda.

Regrese a la consola de Amazon Lex y agregue la función de Lambda como el enlace de código para cumplir la intención `OrderFlowers` en `OrderFlowersBot` que ha creado en la sección anterior.

Para crear la función Lambda (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Crear función.
3. En la página Create function (Crear función), seleccione Use a blueprint (Utilizar un proyecto). Escriba **`lex-`** en el cuadro de texto de filtro y pulse `Enter` para buscar el proyecto. Seleccione el proyecto `lex-order-flowers-python`.

Los esquemas de funciones de Lambda están en formato Node.js y Python. En este ejercicio, utilizaremos el proyecto basado en Python.

4. En la página Basic information (Información básica), haga lo siguiente:
   - Escriba el nombre de la función de Lambda (`OrderFlowersCodeHook`).
   - Para el rol de ejecución, elija Crear un nuevo rol con permisos básicos de Lambda.
   - Deje los demás valores predeterminados.
5. Elija Crear función.
6. Si utiliza una configuración regional que no sea Inglés (EE. UU.) (en-US), actualice los nombres de las intenciones tal como se indica en Actualización de un esquema para una configuración regional específica.
7. Pruebe la función de Lambda.
   a. Haga clic en Select a test event (Seleccionar un evento de prueba) y en Configure test events (Configurar eventos de prueba).
   b. Seleccione Amazon Lex Order Flowers (Pedido de flores de Amazon Lex) en la lista Event template (Plantilla de eventos). Este ejemplo de evento sigue el modelo solicitud/respuesta de Amazon Lex (consulte Uso de funciones de Lambda). Asigne un nombre al evento de prueba (`LexOrderFlowersTest`).
   c. Seleccione Create (Crear).
   d. Seleccione Test (Probar) para probar el enlace de código.
   e. Compruebe que la función de Lambda se ha ejecutado correctamente. En este caso, la respuesta sigue el modelo de respuesta de Amazon Lex.

Paso siguiente

Paso 4: adición de la función de Lambda como enlace de código (consola)

Paso 4: adición de la función de Lambda como enlace de código (consola)

En esta sección, actualizará la configuración de la intención OrderFlowers para utilizar la función de Lambda de la siguiente manera:

- Primero debe utilizar la función de Lambda como enlace de código para cumplir con la intención `OrderFlowers`. Puede probar el bot y verificar que ha recibido un mensaje de cumplimiento

de la función de Lambda. Amazon Lex invoca la función de Lambda únicamente después de proporcionar datos para todas las ranuras necesarias para pedir flores.

- Configure la misma función de Lambda como enlace de código para llevar a cabo la inicialización y la validación. Puede probar y comprobar que la función de Lambda realiza la validación (a medida que proporciona datos de ranura).

Adición de una función de Lambda como enlace de código (consola)

1. En la consola de Amazon Lex, seleccione el bot OrderFlowers. La consola muestra la intención OrderFlowers. Asegúrese de que la versión de la intención está establecida en `$LATEST` porque esta es la única versión que podemos modificar.
2. Agregue la función de Lambda como enlace de código de cumplimiento y pruébela.

   a. En el editor, elija Función de AWS Lambda como Cumplimiento y seleccione la función de Lambda que ha creado en el paso anterior (`OrderFlowersCodeHook`). Elija Aceptar para dar a Amazon Lex permiso para invocar la función de Lambda.

      Está configurando esta función de Lambda como un enlace de código para cumplir con la intención. Amazon Lex invoca esta función únicamente después de disponer de todos los datos de ranura necesarios para llevar a cabo la intención.

   b. Especifique un mensaje de despedida.

   c. Elija Compilar.

   d. Pruebe el bot utilizando la conversación anterior.

   La última frase, “Gracias, su pedido de rosas…”, es una respuesta de la función de Lambda configurada como enlace de código. En la sección anterior, no había ninguna función de Lambda. Ahora utiliza una función de Lambda para cumplir realmente la intención `OrderFlowers`.
3. Añada la función de Lambda como un enlace de código de inicialización y validación y pruébela.

   El código del ejemplo de función de Lambda que utiliza se encarga tanto de la validación de la entrada del usuario como del cumplimiento. El evento de entrada que recibe la función de Lambda tiene un campo (`invocationSource`) que el código utiliza para determinar qué parte del código debe ejecutar. Para obtener más información, consulte Formato del evento de entrada y de la respuesta de la función de Lambda.

a. Seleccione la versión $LATEST de la intención `OrderFlowers`. Esa es la única versión que puede actualizar.

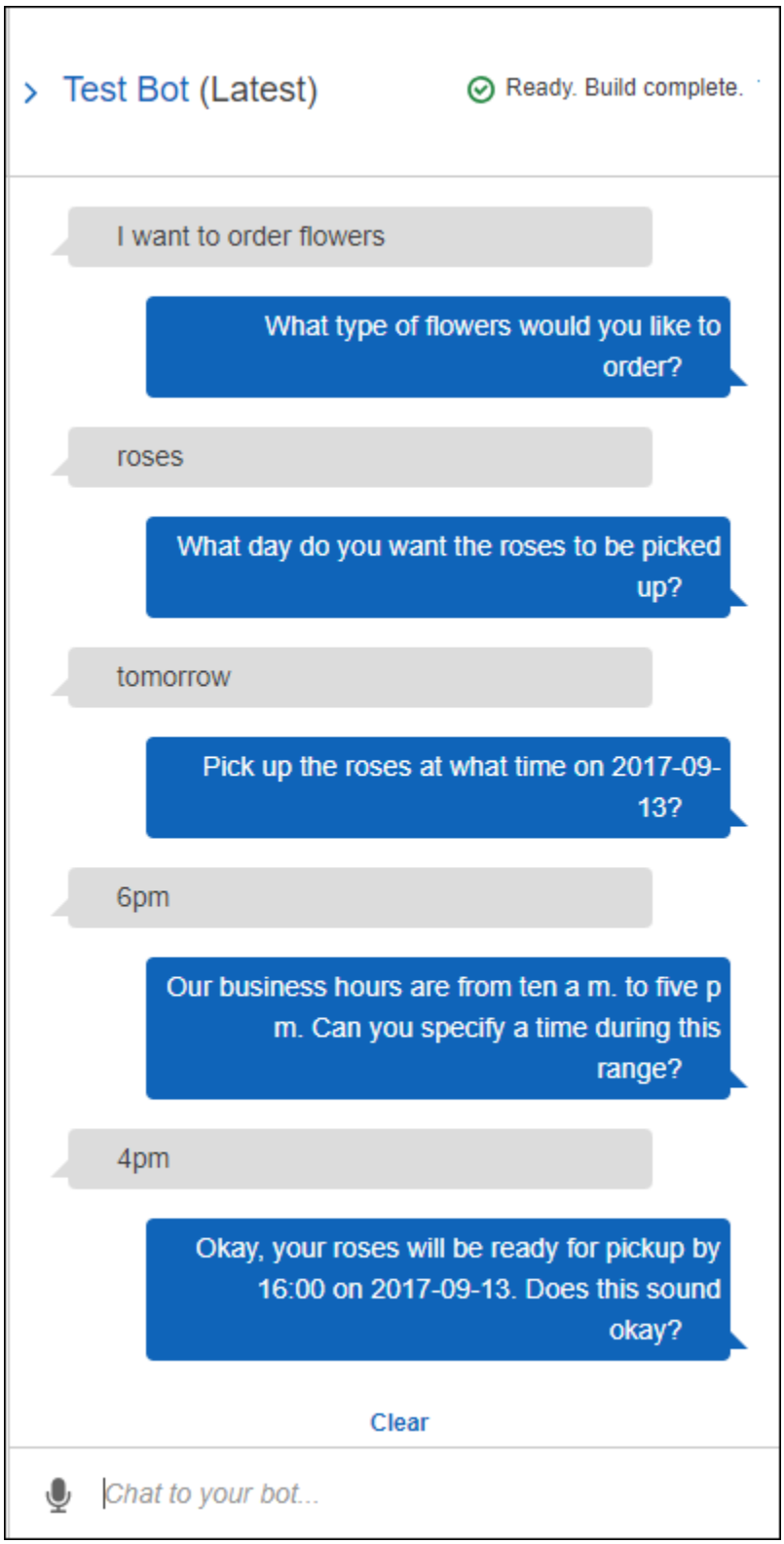
b. En el Editor, seleccione Initialization and validation en Options.

c. De nuevo, seleccione la misma función de Lambda.

d. Elija Compilar.

e. Pruebe el bot.

   Ahora está listo para conversar con Amazon Lex como se indica en la siguiente imagen. Para probar la parte de validación, elija la hora 18:00. La función de Lambda devolverá una respuesta (“Nuestro horario es de 10:00 a 17:00 horas”) y le preguntará de nuevo. Después de proporcionar todos los datos de ranura válidos, la función de Lambda realizará el pedido.



Paso siguiente

Paso 5 (opcional): revisión de los detalles del flujo de información (consola)

Paso 5 (opcional): revisión de los detalles del flujo de información (consola)

En esta sección se explica el flujo de información entre el cliente y Amazon Lex para cada entrada del usuario, incluida la integración de la función de Lambda.

> **Note**
>
> En esta sección se supone que el cliente envía solicitudes a Amazon Lex mediante la API en tiempo de ejecución `PostText` y se muestran los detalles correspondientes de la solicitud y la respuesta. Para ver un ejemplo de flujo de información entre el cliente y Amazon Lex en el que el usuario usa la API `PostContent`, consulte Paso 2a (opcional): revisión de los detalles del flujo de información escrita (consola) .

Para obtener más información sobre la API en tiempo de ejecución `PostText` y detalles adicionales sobre las solicitudes y las respuestas que se muestran en los siguientes pasos, consulte PostText.

1. Usuario: Me gustaría pedir unas flores.

   a. El cliente (la consola) envía la siguiente solicitud PostText a 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": {}
   }
   ```

   Tanto la URI de la solicitud como el cuerpo proporcionan información a Amazon Lex:

   - URI de la solicitud: proporciona el nombre del bot (`OrderFlowers`), el alias del bot ($LATEST) y el nombre de usuario (una cadena aleatoria que identifica al usuario). El `text` de cola indica que se trata de una solicitud de API `PostText` (no `PostContent`).
   - Cuerpo de la solicitud: incluye la entrada del usuario (`inputText`). No hay `sessionAttributes`. Cuando el cliente realiza la primera solicitud, no existen atributos de la sesión. La función Lambda los inicia más adelante.

b. Amazon Lex deduce la intención (`OrderFlowers`) a partir de `inputText`. Esta intención está configurada con una función de Lambda como enlace de código para las tareas de inicialización y validación de los datos del usuario. Por ello, Amazon Lex invoca la función de Lambda mediante la transferencia de la siguiente información como datos de eventos:

```
{
    "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 obtener más información, consulte Formato del evento de entrada.

Además de la información que envía el cliente, Amazon Lex también incluye los siguientes datos adicionales:

- `messageVersion`: actualmente Amazon Lex solo es compatible con la versión 1.0.
- `invocationSource`: indica la finalidad de invocar la función de Lambda. En este caso, es para inicializar y validar los datos del usuario. En este punto, Amazon Lex sabe que el usuario no ha proporcionado todos los datos de ranura necesarios para cumplir la intención.
- Información de `currentIntent` con todos los valores de slot establecidos en nulos.

c. En este momento, todos los valores de slot son nulos. La función de Lambda no tiene que validar nada y devuelve la siguiente respuesta a Amazon Lex:

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

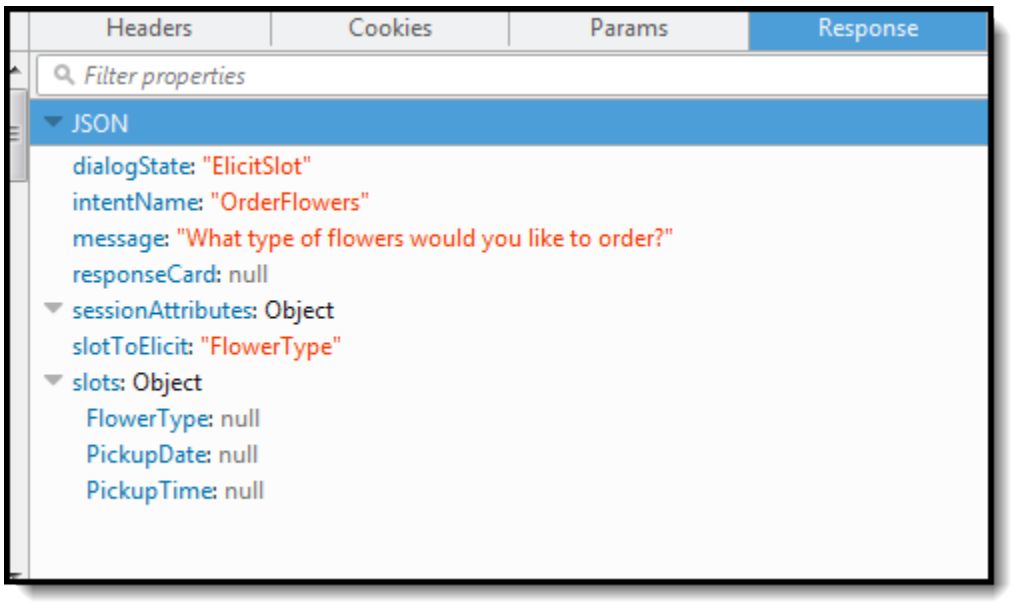
Para obtener más información sobre el formato de la respuesta, consulte Formato de respuesta.

Tenga en cuenta lo siguiente:

- `dialogAction.type`: al establecer este valor en `Delegate`, la función de Lambda delega la responsabilidad de decidir el siguiente procedimiento a Amazon Lex.

> **Note**
>
> Si la función de Lambda detecta algo en la validación de los datos de usuario, indica a Amazon Lex el siguiente paso, tal como se muestra a continuación.

d. De acuerdo con `dialogAction.type`, Amazon Lex decide cómo continuar. Puesto que ninguno de los slots tiene información, decide obtener el valor del slot `FlowerType`. Selecciona una de las preguntas para obtener valores para el slot ("¿Qué tipo de flores le gustaría pedir?") y envía la siguiente respuesta al cliente:



El cliente muestra el mensaje en la respuesta.

2. Usuario: rosas

   a. El cliente envía la siguiente solicitud PostText a Amazon Lex:

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

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

   En el cuerpo de la solicitud, `inputText` proporciona la entrada del usuario. No hay `sessionAttributes`.

   b. Amazon Lex interpreta primero `inputText` en el contexto de la confirmación de la intención actual. El servicio recuerda que ha solicitado al usuario específico información acerca del slot `FlowerType`. Actualiza el valor de la ranura para la intención actual e invoca la función de Lambda con los siguientes datos 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": "roses",
            "PickupDate": null
        },
        "confirmationStatus": "None"
    }
}
```

Tenga en cuenta lo siguiente:

- `invocationSource`: sigue siendo `DialogCodeHook` (simplemente validamos los datos de usuario).
- `currentIntent.slots`: Amazon Lex ha actualizado la ranura `FlowerType` con “roses”.

c. Al establecer el valor `invocationSource` como `DialogCodeHook`, la función de Lambda valida los datos del usuario. Reconoce `roses` como un valor de ranura válido (y establece `Price` como atributo de la sesión) y devuelve la siguiente respuesta a Amazon Lex.

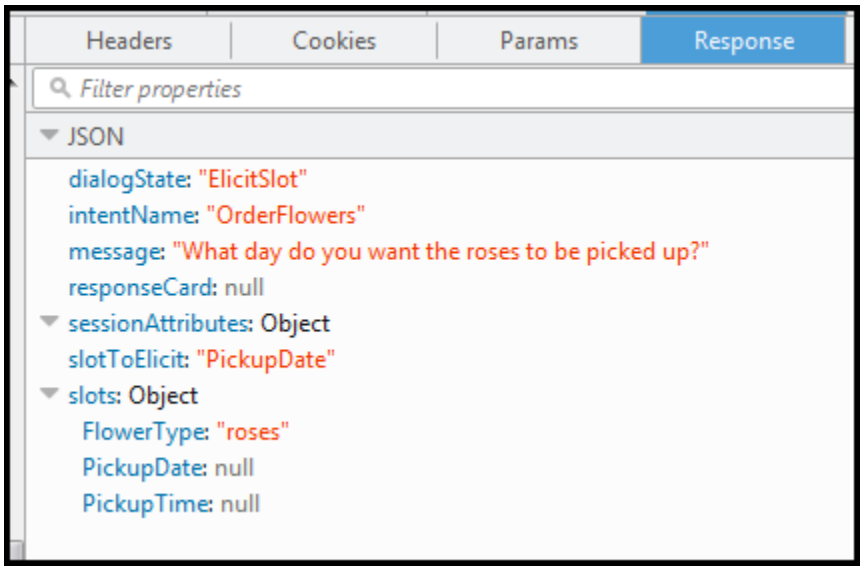
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": null
        }
    }
```

```
}
```

Tenga en cuenta lo siguiente:

- `sessionAttributes`: la función de Lambda ha agregado `Price` (de las rosas) como un atributo de la sesión.
- `dialogAction.type`: se establece en `Delegate`. Los datos del usuario son válidos, por lo que la función de Lambda indica a Amazon Lex que decida el siguiente paso.

d. Amazon Lex elige el siguiente paso en función de `dialogAction.type`. Amazon Lex sabe que necesita más datos de ranura, de modo que elige la siguiente ranura vacía (`PickupDate`) con la mayor prioridad según la configuración de la intención. Amazon Lex selecciona uno de los mensajes de pregunta ("¿Qué día desea recoger las rosas?") para esta ranura, según la configuración de la intención y, a continuación, envía esta respuesta al cliente:



El cliente simplemente muestra el mensaje en la respuesta: "¿Qué día desea recoger las rosas?."

3. Usuario: mañana

a. El cliente envía la siguiente solicitud PostText a Amazon Lex:

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

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

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

En el cuerpo de la solicitud, `inputText` proporciona la entrada del usuario y el cliente devuelve los atributos de la sesión al servicio.

b. Amazon Lex recuerda el contexto; es decir, que está obteniendo datos para la ranura `PickupDate`. En este contexto, sabe que el valor de `inputText` es para el slot `PickupDate`. A continuación, Amazon Lex llama la función de Lambda mediante el envío de los siguientes eventos:

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

Amazon Lex ha actualizado `currentIntent.slots` al establecer el valor `PickupDate`. Además, tenga en cuenta que el servicio pasa `sessionAttributes` tal cual a la función de Lambda.

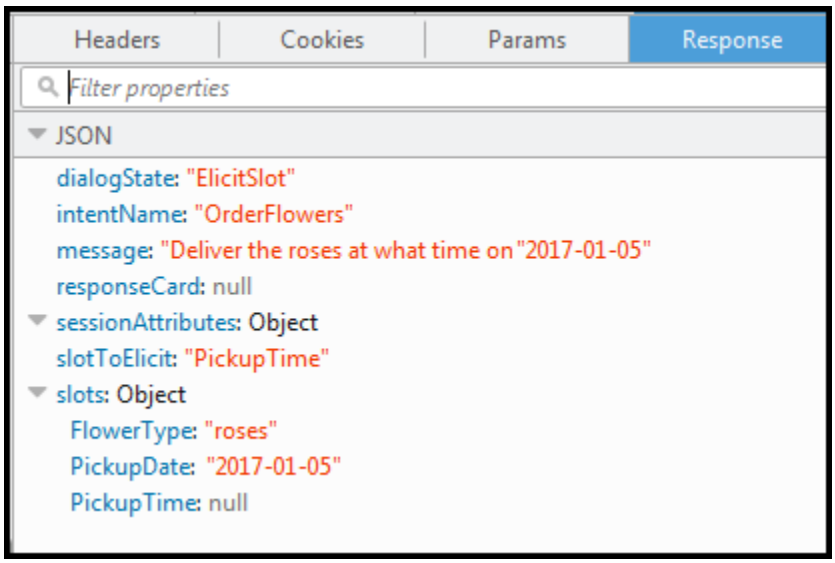
c. Al establecer el valor `invocationSource` como `DialogCodeHook`, la función de Lambda valida los datos de usuario. Reconoce que el valor de la ranura `PickupDate` es válido y devuelve la siguiente respuesta a Amazon Lex:

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

Tenga en cuenta lo siguiente:

- `sessionAttributes`: sin cambios.
- `dialogAction.type`: se establece en `Delegate`. Los datos del usuario son válidos, por lo que la función de Lambda indica a Amazon Lex que decida el siguiente paso.

d. Amazon Lex elige el siguiente paso en función de `dialogAction.type`. Amazon Lex sabe que necesita más datos de ranura, de modo que elige la siguiente ranura vacía (`PickupTime`) con la mayor prioridad según la configuración de la intención. Amazon Lex selecciona uno de los mensajes de pregunta (“¿A qué hora quiere que se entreguen las rosas el 05/01/2017?”) para esta ranura, según la configuración de la intención y, a continuación, envía esta respuesta al cliente:



El cliente muestra el mensaje en la respuesta: “¿A qué hora quiere que entreguen las rosas el 05/01/2017?”.

4. Usuario: 16:00 h

   a. El cliente envía la siguiente solicitud PostText a 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"
    }
}
```

   En el cuerpo de la solicitud, `inputText` proporciona la entrada del usuario. El cliente pasa los `sessionAttributes` en la solicitud.

   b. Amazon Lex entiende el contexto. Entiende que estaba obteniendo datos para el slot `PickupTime`. En este contexto, sabe que el valor `inputText` es para la ranura `PickupTime`. A continuación, Amazon Lex llama la función de Lambda mediante el envío de los siguientes eventos:

```
{
    "messageVersion": "1.0",
```

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

Amazon Lex ha actualizado `currentIntent.slots` al establecer el valor `PickupTime`.

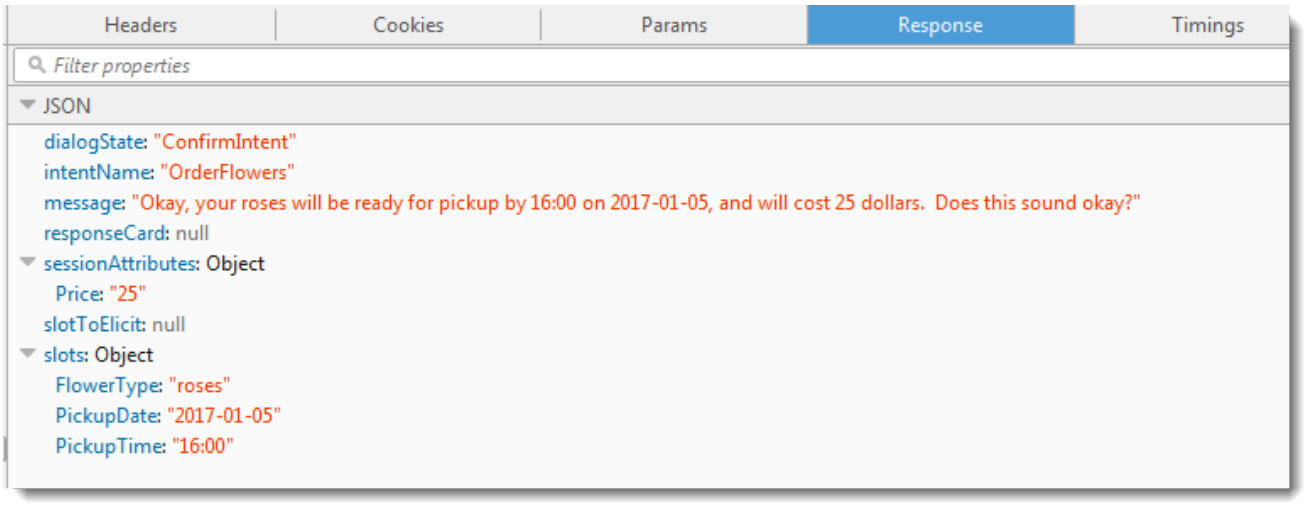
c. Al establecer el valor `invocationSource` como `DialogCodeHook`, la función de Lambda valida los datos del usuario. Reconoce que el valor de la ranura `PickupDate` es válido y devuelve la siguiente respuesta a Amazon Lex.

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

Tenga en cuenta lo siguiente:

- `sessionAttributes`: sin cambios en el atributo de la sesión.
- `dialogAction.type`: se establece en `Delegate`. Los datos del usuario son válidos, por lo que la función de Lambda indica a Amazon Lex que decida el siguiente paso.

d. En este punto, Amazon Lex sabe que dispone de todos los datos de ranura. Esta intención se ha configurado con un mensaje de confirmación. Por lo tanto, Amazon Lex envía la siguiente respuesta al usuario para solicitar su confirmación antes de cumplir con la intención:



El cliente simplemente muestra el mensaje en la respuesta y espera la respuesta del usuario.

5. Usuario: Sí

a. El cliente envía la siguiente solicitud PostText a Amazon Lex:

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

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

b. Amazon Lex interpreta `inputText` en el contexto de la confirmación de la intención actual. Amazon Lex entiende que el usuario desea continuar con el pedido. Esta vez, Amazon

Lex invoca la función de Lambda para cumplir la intención mediante el envío del siguiente evento, que establece `invocationSource` en `FulfillmentCodeHook` en el caso de que se envíe a la función. Amazon Lex también establece `confirmationStatus` en `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"
    }
}
```

Tenga en cuenta lo siguiente:

- `invocationSource`: esta vez, Amazon Lex configura este valor como `FulfillmentCodeHook`, por lo que indica a la función de Lambda que cumpla con la intención.
- `confirmationStatus`: se establece en `Confirmed`.

c. Esta vez, la función de Lambda cumple con la intención `OrderFlowers` y devuelve la siguiente respuesta:

```
{
    "sessionAttributes": {
        "Price": "25"
```

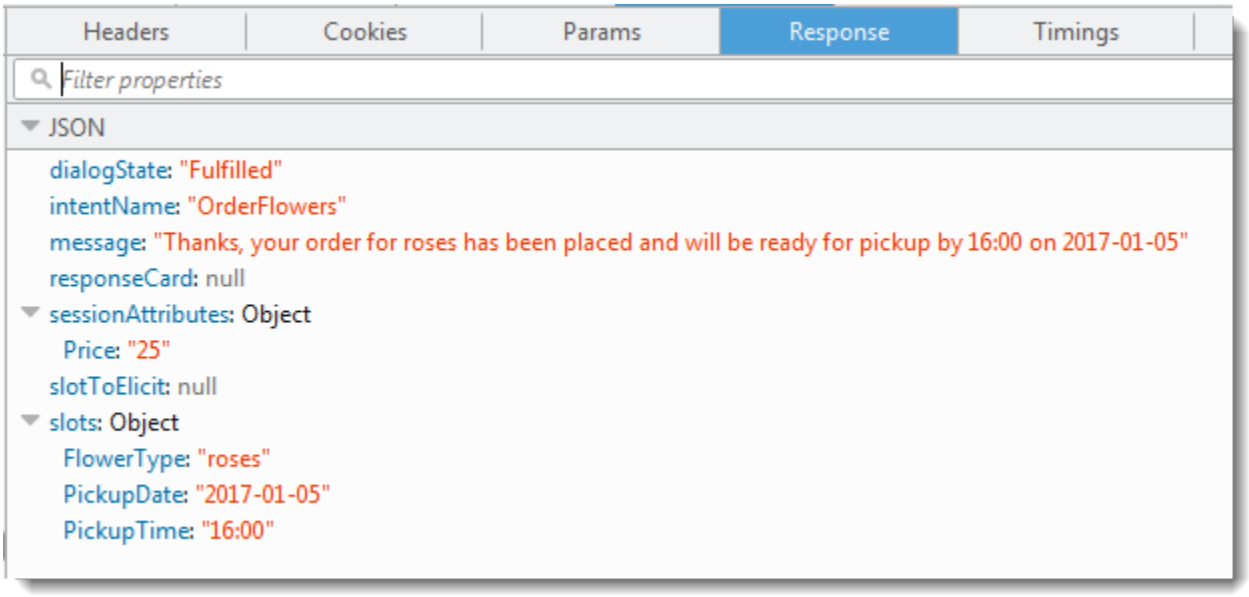
```
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}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

Tenga en cuenta lo siguiente:

- Establece `dialogAction.type`: la función de Lambda establece este valor en `Close` para que Amazon Lex no espere una respuesta del usuario.
- `dialogAction.fulfillmentState`: se establece en Fulfilled e incluye un `message` adecuado para transmitir al usuario.

d. Amazon Lex analiza `fulfillmentState` y envía la siguiente respuesta de vuelta al cliente:

A continuación, Amazon Lex devuelve lo siguiente al cliente:



Tenga en cuenta que:

- `dialogState`: Amazon Lex establece este valor en `fulfilled`.

- `message`: es el mismo mensaje que ha proporcionado la función de Lambda.

El cliente muestra el mensaje.

6. Ahora vuelva a probar el bot. Para establecer un nuevo contexto (usuario), elija el enlace Clear en la ventana de prueba. Ahora proporcione datos de slot no válidos para la intención `OrderFlowers`. Esta vez, la función de Lambda valida los datos, restablece el valor de los datos de ranura no válidos en nulo y pide a Amazon Lex que solicite al usuario datos válidos. Por ejemplo, pruebe lo siguiente:

   - Jazmín como tipo de flor (este tipo de flor no se admite).
   - Ayer como el día en que desea recoger las flores.
   - Después de realizar el pedido, escriba otro tipo de flores en lugar de responder "sí" para confirmar el pedido. Como respuesta, la función de Lambda actualiza `Price` en el atributo de la sesión y mantiene un recuento actual de los pedidos de flores.

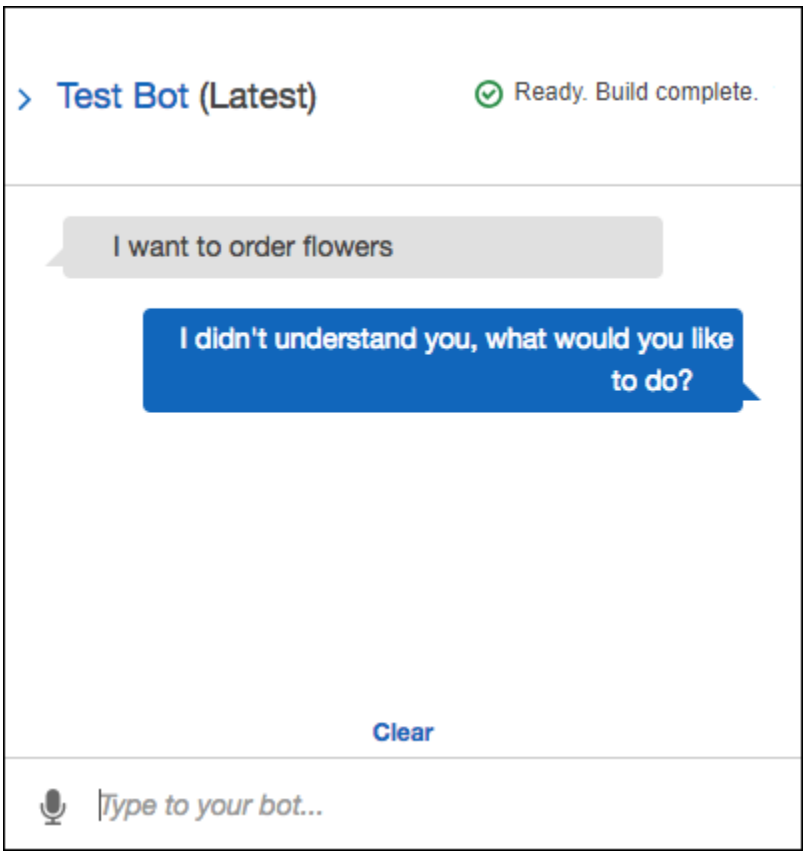
   La función de Lambda también lleva a cabo la actividad de cumplimiento.

Paso siguiente

Paso 6: actualización de la configuración de la intención para añadir un enunciado (consola)

Paso 6: actualización de la configuración de la intención para añadir un enunciado (consola)

El bot `OrderFlowers` está configurado con tan solo dos enunciados. Esto proporciona información limitada para que Amazon Lex genere un modelo de machine learning que reconozca la intención del usuario y responda a esta. Pruebe a escribir “Deseo encargar flores” en la siguiente ventana de pruebas. Amazon Lex no reconoce el texto y responde “No he entendido, ¿qué desea hacer?”. Puede mejorar el modelo de machine learning añadiendo más enunciados.



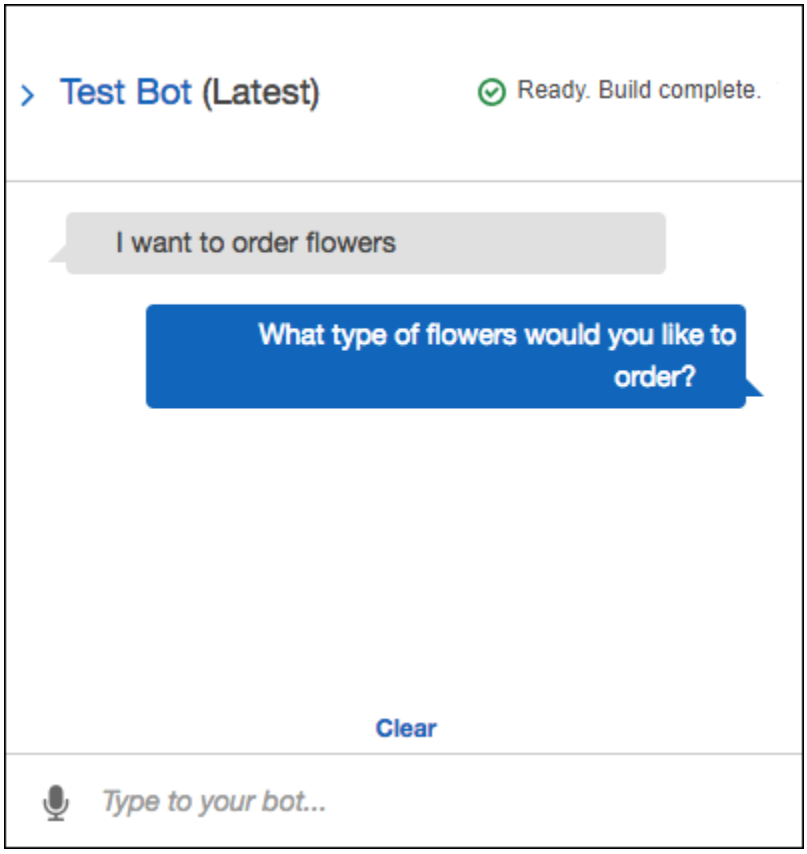
Cada enunciado que agrega proporciona a Amazon Lex más información sobre cómo responder a los usuarios. No es necesario añadir un enunciado, exacto, Amazon Lex generaliza a partir de las muestras que proporcione para reconocer las coincidencias exactas y entradas similares.

Para añadir un enunciado (consola)

1. Escriba el enunciado “Quiero flores” en la sección Ejemplos de enunciados del editor de intenciones para agregarlo, tal como se indica en la siguiente imagen, y, a continuación, haga clic en el icono del signo más situado junto a la nueva publicación.

2. Compile el bot para incluir el cambio. Elija Build y vuelva a elegir Build.

3. Pruebe el bot para confirmar que reconoce el nuevo enunciado. En la ventana de pruebas, tal como se indica en la siguiente imagen, escriba “Quiero pedir flores”. Amazon Lex reconoce la frase y responde con “¿Qué tipo de flores desea encargar?”.



Paso siguiente

Paso 7 (opcional): eliminar recursos (consola)

Paso 7 (opcional): eliminar recursos (consola)

Ahora va a eliminar los recursos que ha creado y a limpiar la cuenta.

Puede eliminar únicamente los recursos que no se usan. En general, debería eliminar los recursos en el orden que se indica a continuación:

- Eliminar bots para liberar los recursos de las intenciones.
- Eliminar intenciones para liberar los recursos de los tipos de slot.
- Por último, eliminar el tipo de slot.

Para limpiar la cuenta (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Desde la lista de bots, seleccione la casilla de verificación junto a OrderFlowers.
3. Para borrar el bot, elija Delete y luego elija Continue en el cuadro de diálogo de confirmación.
4. En el panel izquierdo, elija Intents.
5. En la lista de intenciones, elija OrderFlowersIntent.
6. Para eliminar la intención, elija Delete y luego elija Continue en el cuadro de diálogo de confirmación.
7. En el panel izquierdo, elija Slot types.
8. En la lista de tipos de slot, elija Flowers.
9. Para borrar el tipo de slot, elija Delete y luego elija Continue en el cuadro de diálogo de confirmación.

Ha eliminado todos los recursos de Amazon Lex que ha creado y ha limpiado su cuenta. Si lo desea, puede utilizar la consola de Lambda para eliminar la función de Lambda que se utiliza en este ejercicio.

Ejercicio 2: creación de un bot de Amazon Lex personalizado

En este ejercicio, utilizará la consola de Amazon Lex para crear un bot personalizado que encargue una pizza (`OrderPizzaBot`). Para configurar el bot, añada una intención personalizada (`OrderPizza`), defina los tipos de slot personalizados y defina los slots necesarios para llevar a cabo el pedido de pizza (la masa de la pizza, el tamaño de la pizza, etc.). Para obtener más información sobre los slots y los tipos de slot, consulte Funcionamiento de Amazon Lex.

Temas

- Paso 1: Crear una función Lambda
- Paso 2: creación de un bot
- Paso 3: compilación y prueba del bot
- Paso 4 (opcional): Eliminación

Paso 1: Crear una función Lambda

En primer lugar, cree una función de Lambda que pida una pizza. Deberá especificar esta función en el bot de Amazon Lex que creará en la siguiente sección.

Para crear una función Lambda

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Crear función.
3. En la página Create function, elija Author from scratch.

   Dado que está utilizando el código personalizado que se le ha proporcionado en este ejercicio para crear una función Lambda, debe seleccionar la opción para crear la función desde cero.

   Haga lo siguiente:

   a. Escriba el nombre (`PizzaOrderProcessor`).
   b. En Runtime, elija la última versión de Node.js.
   c. En Role, elija Create new role from template(s).
   d. Escriba un nombre nuevo para el rol (`PizzaOrderProcessorRole`).
   e. Elija Crear función.
4. En la página function, haga lo siguiente:

   En la sección Function code, elija Edit code inline y, a continuación, copie el siguiente código de función de Node.js y péguelo en la ventana.

```
'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. Seleccione Save.

Prueba de la función de Lambda con datos de eventos de ejemplo

En la consola, pruebe la función de Lambda con datos de eventos de ejemplo para invocarla de manera manual.

Para probar la función de Lambda:

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. En la página Función de Lambda, elija la función de Lambda (`PizzaOrderProcessor`).
3. En la página de la función, en la lista de eventos de prueba, elija Configure test events.
4. En la página Configure test event, haga lo siguiente:

   a. Elija Create new test event.

   b. En el campo Event name, escriba un nombre para el evento (`PizzaOrderProcessorTest`).

   c. Copie el siguiente evento de Amazon Lex en la ventana.

```
{
  "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. Seleccione Create (Crear).

AWS Lambda crea la prueba y vuelve a la página de la función. Elija Probar y Lambda ejecutará la función de Lambda.

En el cuadro de resultados, elija Details. La consola muestra la salida siguiente en el panel Execution result.

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

Paso siguiente

Paso 2: creación de un bot

Paso 2: creación de un bot

En este paso, debe crear un bot para gestionar los pedidos de pizza.

Temas

- Cree el bot
- Cree una intención
- Cree tipos de slot
- Configure la intención
- Configuración del bot de

Cree el bot

Cree el bot `PizzaOrderingBot` con el mínimo de información necesaria. Más adelante, puede añadir una intención, que es una acción que el usuario desea realizar, para el bot.

Para crear el bot

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Cree un bot.

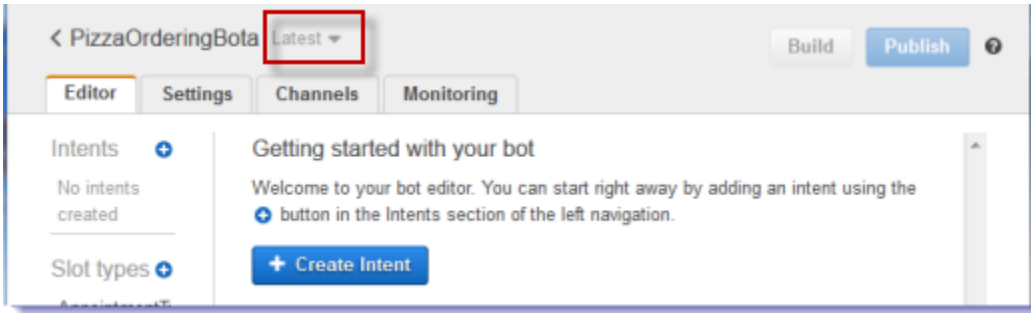
a. Si está creando su primer bot, elija Empezar. De lo contrario, elija Bots y, a continuación, seleccione Create.

b. En la página Crear un bot de Lex, elija Bot personalizado y proporcione la siguiente información:

- Nombre del bot: PizzaOrderingBot
- Idioma: elija el idioma y la configuración regional del bot.
- Output voice: Salli
- Session timeout : 5 minutos.
- COPPA: elija la respuesta adecuada.
- Almacenamiento de enunciados del usuario: elija la respuesta adecuada.

c. Seleccione Create (Crear).

La consola envía a Amazon Lex una solicitud para crear un nuevo bot. Amazon Lex establece la versión del bot en `$LATEST`. Después de crear el bot, Amazon Lex muestra la pestaña Editor del bot, tal como se indica en la siguiente imagen:



- La versión del bot, Latest, aparece junto al nombre del bot en la consola. Los nuevos recursos de Amazon Lex tienen la versión `$LATEST`. Para obtener más información, consulte Control de versiones y alias.
- Dado que todavía no ha creado ninguna intención ni tipo de slot, no se indica ninguno.
- Build y Publish son actividades del nivel de bot. Después de configurar todo el bot, podrá obtener más información sobre estas actividades.

Paso siguiente

Cree una intención

Cree una intención

Ahora, cree la intención `OrderPizza`, una acción que el usuario desea realizar, con el mínimo de información necesaria. Puede añadir tipos de slot para la intención y luego configurar la intención en otro momento.

Para crear una intención

1. En la consola de Amazon Lex, elija el signo más (+) junto a Intenciones y, a continuación, seleccione Crear nueva intención.
2. En el cuadro de diálogo Create intent, escriba el nombre de la intención (`OrderPizza`) y, a continuación, elija Add.

La consola envía una solicitud a Amazon Lex para crear la intención `OrderPizza`. En este ejemplo se crean ranuras para la intención después de crear los tipos de ranura.

Paso siguiente

Cree tipos de slot

Cree tipos de slot

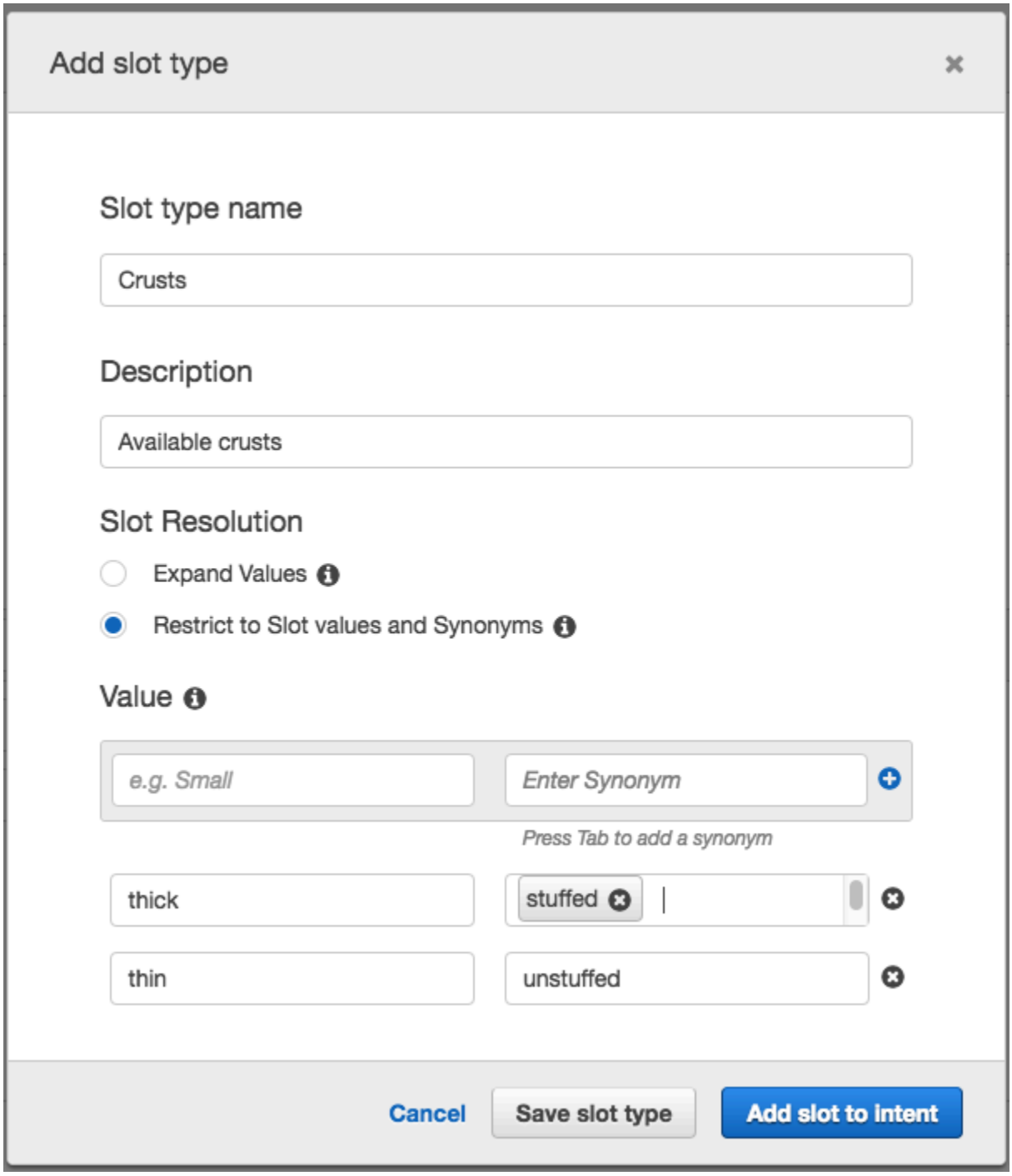
Cree los tipos de slot, o valores de parámetros, que utiliza la intención `OrderPizza`.

Para crear tipos de slot

1. En el menú de la izquierda, seleccione el signo más (+) junto a Slot types.
2. En el cuadro de diálogo Add slot type, añada lo siguiente:

   - Slot type name: Crusts
   - Description: Available crusts
   - Elija Restrict to Slot values and Synonyms
   - Valor: escriba **`thick`**. Pulse en la pestaña y, en el campo Synonym, escriba **`stuffed`**. Elija el signo más (+). Escriba **`thin`** y, a continuación, elija el signo más (+) de nuevo.

   El cuadro de diálogo debería ser similar al de la siguiente imagen:



3. Elija Add slot to intent.
4. En la página Intent, elija Required. Cambie el nombre del slot de **`slotOne`** a **`crust`**. Cambie la pregunta a **`What kind of crust would you like?`**
5. Repita el Step 1 al Step 4 utilizando los valores de la siguiente tabla:

Nombre	Descripción	Valores	Nombre de slot	Prompt
Sizes	Available sizes	small, medium, large	size	What size pizza?
PizzaKind	Available pizzas	veg, cheese	pizzaKind	Do you want a veg or cheese pizza?

Paso siguiente

Configure la intención

Configure la intención

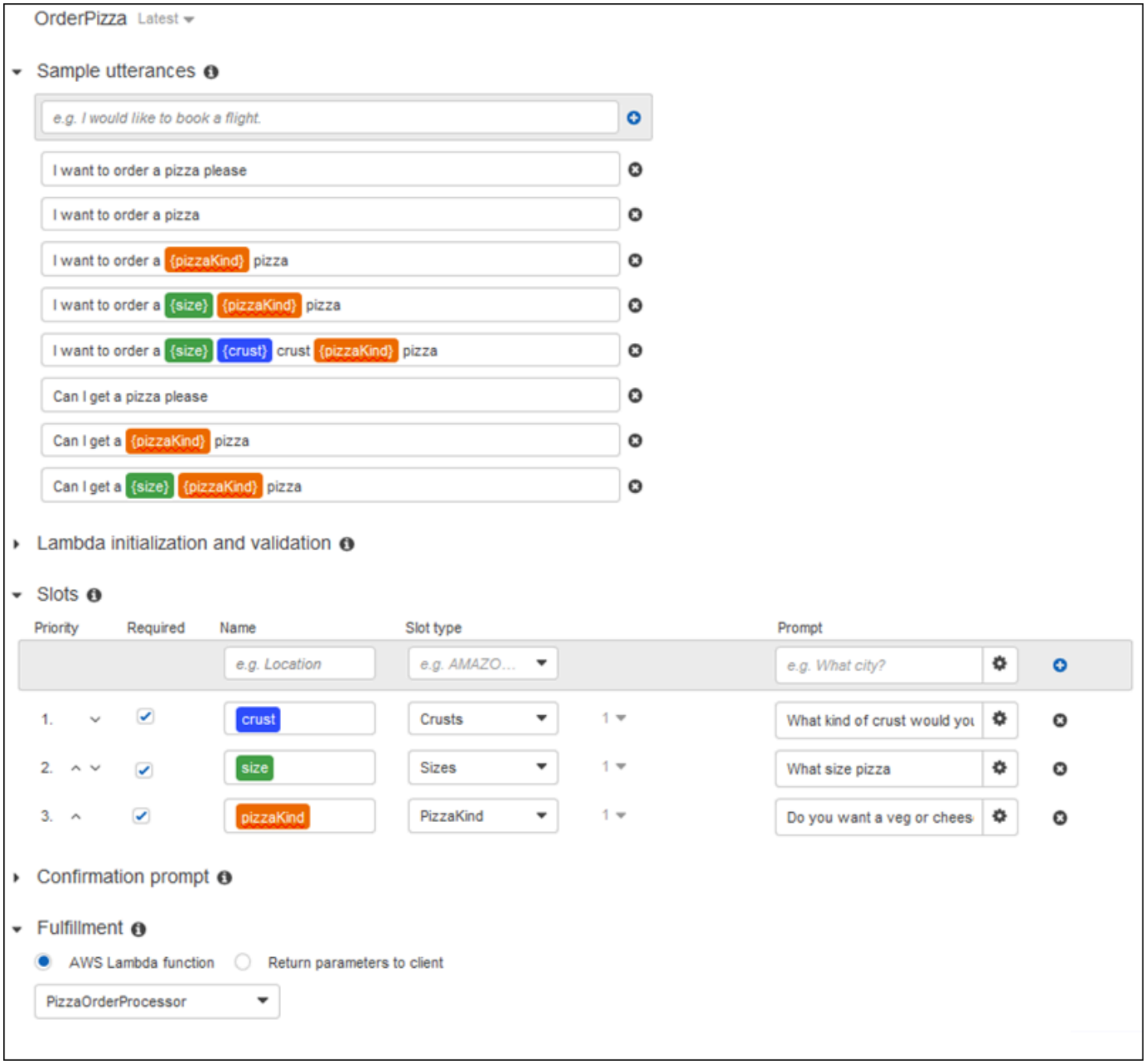
Configure la intención `OrderPizza` para llevar a cabo la solicitud de pizza de un usuario.

Para configurar una intención

- En la página OrderPizza, configure la intención de la siguiente manera:
  - Ejemplos de enunciados: escriba las siguientes cadenas. Las llaves {} incluyen los nombres de slot.
    - Deseo pedir una pizza, por favor
    - Deseo pedir una pizza
    - Deseo pedir una pizza {pizzaKind}
    - Deseo pedir una pizza {size} {pizzaKind}
    - Deseo una pizza {size} con masa {crust} {pizzaKind}
    - ¿Puede traerme una pizza, por favor?
    - ¿Puede traerme una pizza {pizzaKind}?
    - ¿Puede traerme una pizza {size} {pizzaKind}?
  - Lambda initialization and validation: deje el valor predeterminado.
  - Confirmation prompt: deje el valor predeterminado.
  - Cumplimiento: realice las siguientes tareas:
    - Elija función AWS Lambda.

- Elija **`PizzaOrderProcessor`**.
- Si se abre el cuadro de diálogo Agregar permiso a la función de Lambda, elija Aceptar para dar a la intención `OrderPizza` el permiso para llamar a la función de Lambda `PizzaOrderProcessor`.
- Deje seleccionado None.

La intención debe ser similar a la siguiente:



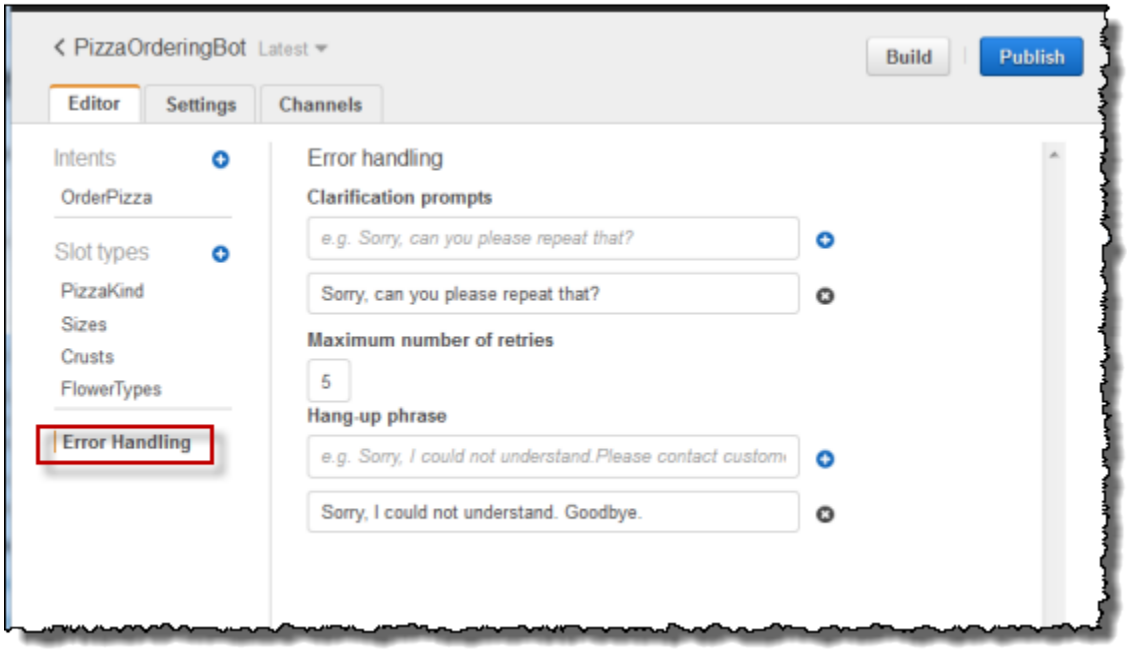
Paso siguiente

Configuración del bot de

Configuración del bot de

Configure la gestión de errores del bot `PizzaOrderingBot`.

1. Vaya al bot `PizzaOrderingBot`. Elija Editor y, a continuación, elija Gestión de errores, tal como se muestra en la siguiente imagen:



2. Utilice la pestaña Editor para configurar la gestión de errores del bot.

   - La información que facilita en Clarification Prompts se asigna a la configuración clarificationPrompt del bot.

     Si Amazon Lex no puede determinar la intención del usuario, el servicio devuelve una respuesta con este mensaje.
   - La información que facilita en la frase Hang-up se asigna a la configuración abortStatement del bot.

     Si el servicio no puede determinar la intención del usuario después de un número determinado de solicitudes consecutivas, Amazon Lex devuelve una respuesta con este mensaje.

   Deje los valores predeterminados.

Paso siguiente

Paso 3: compilación y prueba del bot

Paso 3: compilación y prueba del bot

Para asegurarse de que el bot funciona, compílelo y pruébelo.

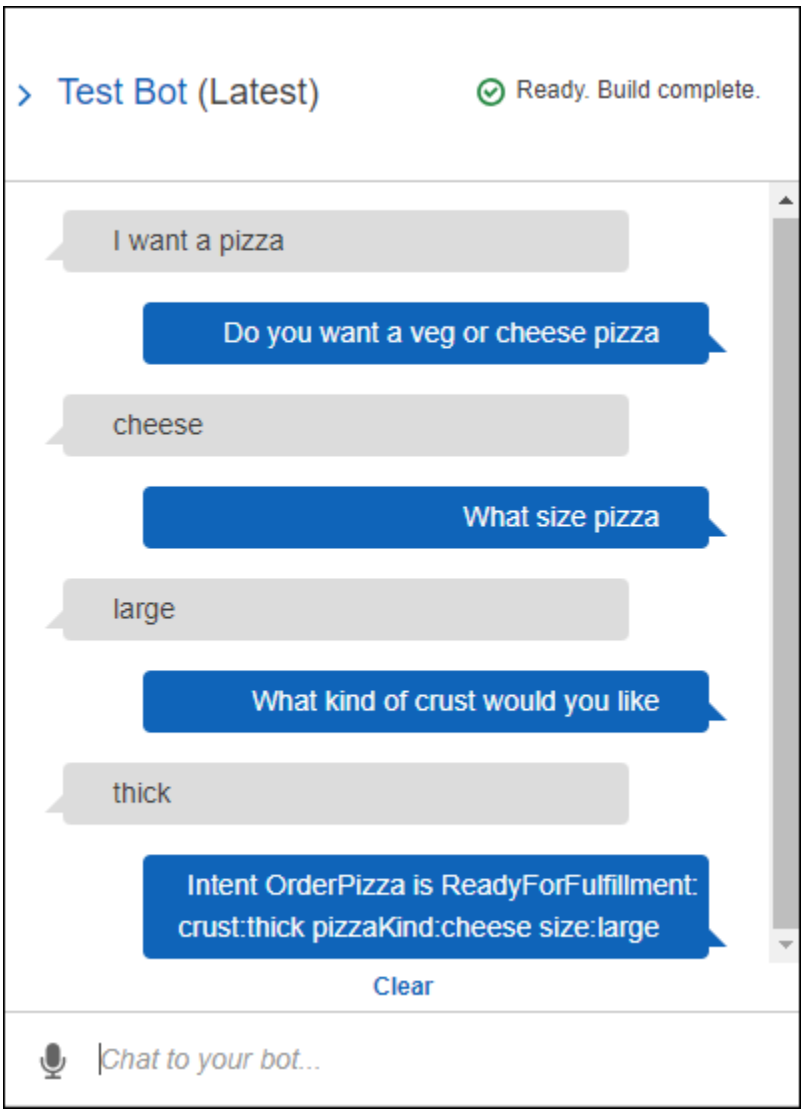
Para compilar y probar el bot

1. Para compilar el bot `PizzaOrderingBot`, elija Build.

   Amazon Lex crea un modelo de machine learning para el bot. Al probar el bot, la consola utiliza la API en tiempo de ejecución para enviar las entradas del usuario de nuevo a Amazon Lex. A continuación, Amazon Lex utiliza el modelo de machine learning para interpretar las entradas del usuario.

   La compilación puede llevar tiempo.
2. Para probar el bot, en la ventana Probar bot, empiece a comunicarse con el bot de Amazon Lex.

   - Por ejemplo, puede decir o escribir lo siguiente:



- Utilice los enunciados de ejemplo que haya configurado en la intención `OrderPizza` para probar el bot. Por ejemplo, a continuación tenemos uno de los enunciados de ejemplo que ha configurado para la intención `PizzaOrder`:

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

  Para probarlo, escriba lo siguiente:

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

Al escribir “Quiero pedir una pizza”, Amazon Lex detecta la intención (`OrderPizza`). A continuación, Amazon Lex solicita información de la ranura.

Cuando haya proporcionado toda la información de la ranura, Amazon Lex invoca la función de Lambda que ha configurado para la intención.

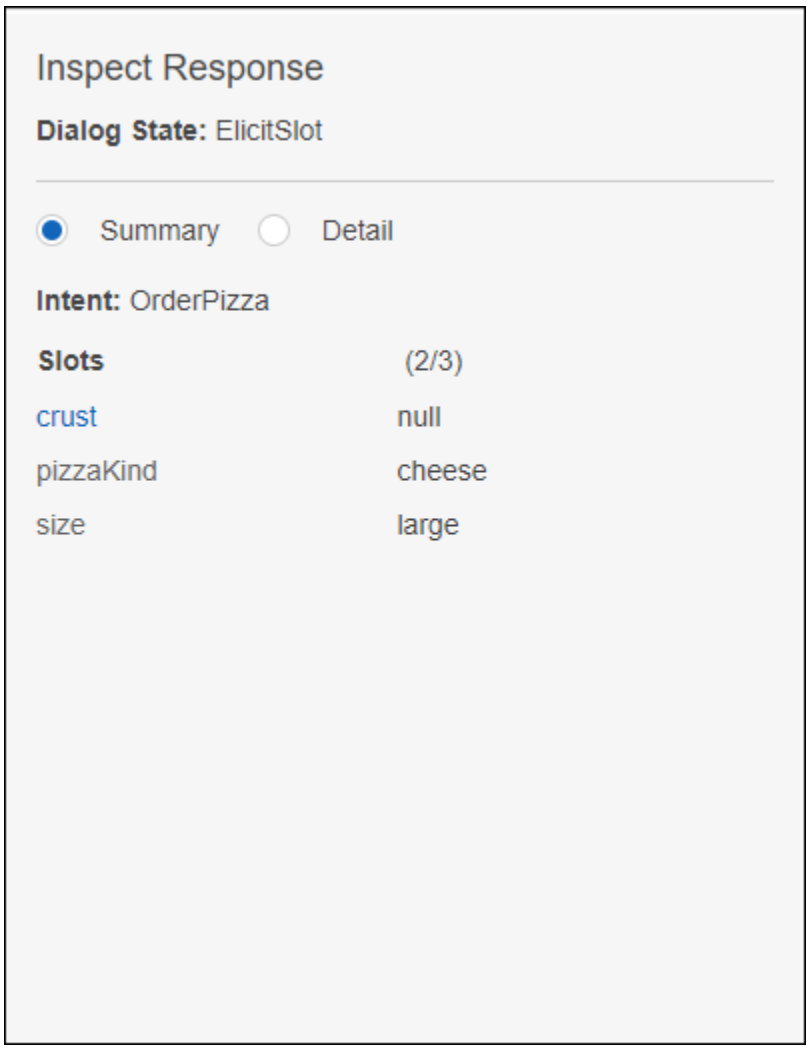
La función de Lambda devuelve un mensaje (“Bien, he pedido…”) a Amazon Lex, que este le devuelve a usted.

Inspección de la respuesta

Debajo de la ventana de chat hay un panel que le permite inspeccionar la respuesta de Amazon Lex. El panel ofrece información completa acerca del estado del bot, que cambia a medida que interactúa con el bot. El contenido del panel muestra el estado actual de la operación.

- Estado de diálogo: el estado actual de la conversación con el usuario. Puede ser `ElicitIntent`, `ElicitSlot`, `ConfirmIntent` o `Fulfilled`.

- Resumen: muestra una vista simplificada del cuadro de diálogo que indica los valores de ranura de la intención que se está realizando para que pueda realizar un seguimiento del flujo de información. Muestra el nombre de la intención, el número de slots y el número de slots rellenos, además de una lista de todos los slots y sus valores asociados. Vea la siguiente imagen:



- Detalle: muestra la respuesta JSON sin procesar del bot de chat para que pueda comprender mejor la interacción con el bot y el estado actual del cuadro de diálogo mientras prueba y depura el bot de chat. Si escribe en la ventana de chat, el panel de inspección muestra la respuesta JSON de la operación PostText. Si habla a la ventana de chat, el panel de inspección muestra los encabezados de la respuesta de la operación PostContent. Vea la siguiente imagen:

Paso siguiente

Paso 4 (opcional): Eliminación

Paso 4 (opcional): Eliminación

Elimine los recursos que ha creado y limpie su cuenta para evitar incurrir en más cargos por los recursos que ha creado.

Puede eliminar únicamente los recursos que no se usan. Por ejemplo, no puede eliminar un tipo de slot al que haga referencia una intención. No puede eliminar una intención a la que haga referencia un bot.

Elimine los recursos en el orden que se indica a continuación:

- Eliminar bots para liberar los recursos de las intenciones.

- Eliminar intenciones para liberar los recursos de los tipos de slot.
- Por último, eliminar el tipo de slot.

Para limpiar la cuenta

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, elija PizzaOrderingBot.
3. Para eliminar el bot, elija Delete y, a continuación, Continue.
4. En el panel izquierdo, elija Intents.
5. En la lista de intenciones, elija OrderPizza.
6. Para eliminar la intención, elija Delete y, a continuación, Continue.
7. En el menú de la izquierda, elija Slot types.
8. En la lista de tipos de slot, elija Crusts.
9. Para eliminar el tipo de slot, elija Delete y, a continuación, Continue.
10. Repita Step 8 y Step 9 para los tipos de slot `Sizes` y `PizzaKind`.

Ha eliminado todos los recursos que ha creado y ha limpiado su cuenta.

Pasos siguientes

- Publique una versión y cree un alias
- Creación de un bot de Amazon Lex con la AWS Command Line Interface

Ejercicio 3: publicación de una versión y creación de un alias

En los ejercicios de introducción 1 y 2, creó un bot y lo probó. En este ejercicio, hará lo siguiente:

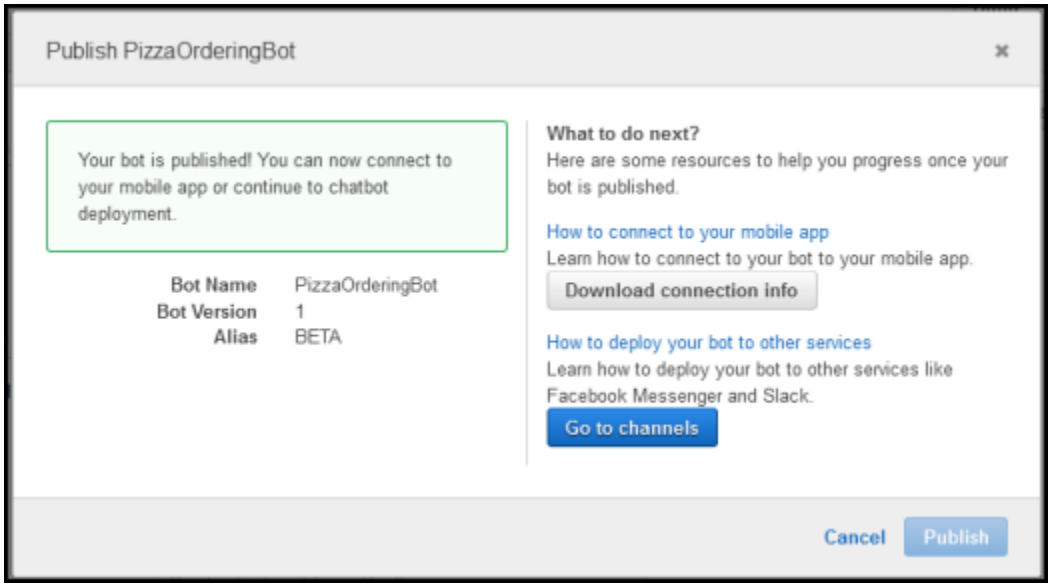
- Publicar una nueva versión del bot. Amazon Lex realiza una copia instantánea de la versión `$LATEST` para publicar una nueva versión.
- Cree un alias que apunte a la nueva versión.

Para obtener más información sobre el control de versiones y los alias, consulte Control de versiones y alias.

Haga lo siguiente para publicar una versión de un bot que ha creado para este ejercicio:

1. En la consola de Amazon Lex, elija uno de los bots que ha creado.

   Compruebe que la consola muestra `$LATEST` como la versión del bot junto al nombre del bot.
2. Elija Publish.
3. En el asistente Publicar ***`nombre del bot`***, especifique el alias **`BETA`** y, a continuación, elija Publicar.
4. Compruebe que la consola de Amazon Lex muestra la nueva versión del bot junto al nombre del bot, tal como se muestra en la siguiente imagen.



Ahora que tiene un bot de trabajo con la versión publicada y un alias, puede implementar el bot (en su aplicación móvil o integrar el bot en Facebook Messenger). Para ver un ejemplo, consulte Integración de un bot de Amazon Lex con Facebook Messenger.

Paso 4: Introducción (AWS CLI)

En este paso, se utiliza la AWS CLI para crear, probar y modificar un bot de Amazon Lex. Para completar estos ejercicios, es necesario conocer el uso de la CLI y disponer de un editor de texto. Para obtener más información, consultar Paso 2: configurar el AWS Command Line Interface

- Ejercicio 1: crear y probar un bot de Amazon Lex El ejercicio proporciona todos los objetos JSON necesarios para crear un slot personalizado, una intención y un bot. Para obtener más información, consultar Funcionamiento de Amazon Lex

- Ejercicio 2: actualizar el bot creado en el ejercicio 1 para agregar un ejemplo de enunciado adicional. Amazon Lex utiliza ejemplos de enunciados para crear el modelo de machine learning del bot.
- Ejercicio 3: actualizar el bot que ha creado en el ejercicio 1 para agregar una función de Lambda que valide las entradas del usuario y cumpla con la intención.
- Ejercicio 4: publicar una versión del tipo de ranura, la intención y los recursos de bot creados en el ejercicio 1. Una versión es una snapshot de un recurso que no se puede cambiar.
- Ejercicio 5: crear un alias para el bot creado en el ejercicio 1.
- Ejercicio 6: eliminar el tipo de ranura, la intención y el bot creados en el ejercicio 1 y el alias creado en el ejercicio 5 para limpiar la cuenta.

Temas

Ejercicio 1: creación de un bot de Amazon Lex (AWS CLI)

En general, cuando se crean bots, hay que:

1. Crear tipos de slots para definir la información con la que trabajará el bot.
2. Crear las intenciones que definen las acciones del usuario que admite el bot. Utilice los tipos de slots personalizadas que ha creado anteriormente para definir los slots, o parámetros, que requiere la intención.
3. Crear un bot que utilice las intenciones que ha definido.

En este ejercicio, creará y probará un nuevo bot de Amazon Lex con la CLI. Utilice las estructuras JSON que proporcionamos para ello. Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Temas

Paso 1: Crear un rol vinculado a servicio (AWS CLI)

Amazon Lex asume roles vinculados a servicios de AWS Identity and Access Management para llamar a los servicios de AWS en nombre de los bots. Los roles, que están en su cuenta, están vinculados a casos de uso de Amazon Lex y tienen permisos predefinidos. Para obtener más información, consulte Uso de roles vinculados a servicios para Amazon Lex.

Si ya ha creado un bot de Amazon Lex con la consola, el rol vinculado al servicio se habrá creado automáticamente. Vaya a Paso 2: Crear un tipo de slot personalizado (AWS CLI).

Para crear un rol vinculado a un servicio (AWS CLI)

1. En la AWS CLI, escriba el siguiente comando:

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

2. Compruebe la política usando el comando siguiente:

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

   La respuesta es:

```
{
    "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"
}
```

Paso siguiente

Paso 2: Crear un tipo de slot personalizado (AWS CLI)

Paso 2: Crear un tipo de slot personalizado (AWS CLI)

Cree un tipo de slot personalizado con valores de enumeración para las flores que puedan encargarse. Puede utilizar este tipo en el siguiente paso cuando cree la intención `OrderFlowers`. Un tipo de slot define los valores posibles para un slot, o parámetro, de la intención.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Para crear un tipo de slot personalizado (AWS CLI)

1. Cree un archivo de texto denominado **`FlowerTypes.json`**. Copie el código JSON de FlowerTypes.json en el archivo de texto.
2. Llame a la operación PutSlotType utilizando la AWS CLI para crear el tipo de slot. El ejemplo está preparado para Unix, Linux y macOS. Para Windows, sustituya la barra diagonal invertida (\) utilizada como carácter de continuación de Unix al final de cada línea por el signo de intercalación (^).

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

   La respuesta del servidor es:

   ```
   {
   ```

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

Paso siguiente

Paso 3: Crear una intención (AWS CLI)

FlowerTypes.json

El siguiente código contiene los datos JSON necesarios para crear el tipo de slot personalizado `FlowerTypes`:

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

Paso 3: Crear una intención (AWS CLI)

Cree una intención para el bot `OrderFlowersBot` y proporcione tres slots o parámetros. Los slots permiten que el bot satisfaga la intención:

- `FlowerType` es un tipo de slot personalizado que especifica los tipos de flores que pueden encargarse.
- `AMAZON.DATE` y `AMAZON.TIME` son tipos de slots integrados que se emplean para obtener del usuario la fecha y la hora de entrega de las flores.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Para crear la intención **`OrderFlowers`** (AWS CLI)

1. Cree un archivo de texto denominado **`OrderFlowers.json`**. Copie el código JSON de OrderFlowers.json en el archivo de texto.
2. En la AWS CLI, llame a la operación PutIntent para crear la intención. El ejemplo está preparado para Unix, Linux y macOS. Para Windows, sustituya la barra diagonal invertida (\) utilizada como carácter de continuación de Unix al final de cada línea por el signo de intercalación (^).

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

   El servidor responde con lo siguiente:

   ```
   {
       "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"
}
```

Paso siguiente

[Paso 4: Crear un bot (AWS CLI)](#)

OrderFlowers.json

El siguiente código contiene los datos JSON necesarios para crear la intención `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"
```

```
}
```

Paso 4: Crear un bot (AWS CLI)

El bot `OrderFlowersBot` tiene una intención, la intención `OrderFlowers` que ha creado en el paso anterior. Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie `"\$LATEST"` por `$LATEST`.

Para crear el bot **`OrderFlowersBot`** (AWS CLI)

1. Cree un archivo de texto denominado **`OrderFlowersBot.json`**. Copie el código JSON de OrderFlowersBot.json en el archivo de texto.
2. En la AWS CLI, llame a la operación PutBot para crear el bot. El ejemplo está preparado para Unix, Linux y macOS. Para Windows, sustituya la barra diagonal invertida (\) utilizada como carácter de continuación de Unix al final de cada línea por el signo de intercalación (^).

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

   La respuesta del servidor sigue a continuación. Cuando crea o actualiza el bot, el campo `status` se establece en `BUILDING`. Esto indica que el bot no está listo para su uso. Para determinar si el bot está listo, utilice la operación GetBot en el siguiente paso.

   ```
   {
       "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 si el nuevo bot está listo para su uso, ejecute este comando. Repita este comando hasta que el campo `status` devuelva `READY`. El ejemplo está preparado para Unix, Linux y macOS. Para Windows, sustituya la barra diagonal invertida (\) utilizada como carácter de continuación de Unix al final de cada línea por el signo de intercalación (^).

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

   Busque el campo `status` en la respuesta:

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

    ...

}
```

Paso siguiente

[Paso 5: Probar un bot (AWS CLI)]

OrderFlowersBot.json

El siguiente código JSON proporciona los datos necesarios para crear el bot de 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"
}
```

Paso 5: Probar un bot (AWS CLI)

Para probar el bot, utilice una prueba basada en texto o en voz.

Temas

- Probar el bot con entrada de texto (AWS CLI)
- Probar el bot con entrada de voz (AWS CLI)

Probar el bot con entrada de texto (AWS CLI)

Para verificar que el bot funciona correctamente con entradas de texto, utilice la operación PostText. Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de servicio en tiempo de ejecución.

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie `"\$LATEST"` por `$LATEST` y sustituya la barra diagonal invertida (\) utilizada como carácter de continuación al final de cada línea por el signo de intercalación (^).

Para probar el bot con texto (AWS CLI)

1. En la AWS CLI, comience una conversación con el bot `OrderFlowersBot`. El ejemplo está preparado para Unix, Linux y macOS. Para Windows, sustituya la barra diagonal invertida (\) utilizada como carácter de continuación de Unix al final de cada línea por el signo de intercalación (^).

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

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

Amazon Lex reconoce la intención del usuario y comienza una conversación al devolver la siguiente respuesta:

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

2. Ejecute los comandos siguientes para finalizar la conversación con el 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"
```

Después de confirmar el pedido, Amazon Lex envía una respuesta de cumplimiento para completar la conversación:

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

Paso siguiente

Probar el bot con entrada de voz (AWS CLI)

Probar el bot con entrada de voz (AWS CLI)

Para probar el bot utilizando archivos de audio, use la operación PostContent. Puede generar los archivos de audio mediante las operaciones de texto a voz de Amazon Polly.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos de Amazon Lex y Amazon Polly. Para ver la lista de regiones de Amazon Lex, consulte Cuotas de servicio en tiempo de ejecución. Para ver la lista de regiones de Amazon Polly, consulte Regiones y puntos de conexión de AWS en la referencia general de Amazon Web Services.

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie `"\$LATEST"` por `$LATEST` y sustituya la barra diagonal invertida (\) utilizada como carácter de continuación al final de cada línea por el signo de intercalación (^).

Para utilizar una entrada de voz para probar el bot (AWS CLI)

1. En la AWS CLI, cree un archivo de audio con Amazon Polly. El ejemplo está preparado para Unix, Linux y macOS. Para Windows, sustituya la barra diagonal invertida (\) utilizada como carácter de continuación de Unix al final de cada línea por el signo de intercalación (^).

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

2. Para enviar el archivo de audio a Amazon Lex, ejecute el siguiente comando. Amazon Lex guarda el audio de la respuesta en el archivo de salida 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
```

   Amazon Lex responde con una solicitud para la primera ranura. Guarda la respuesta de audio en el archivo de salida 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 realizar un pedido de rosas, cree el siguiente archivo de audio y envíelo a 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 establecer la fecha de la entrega, cree el siguiente archivo de audio y envíelo a 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 establecer la hora de la entrega, cree el siguiente archivo de audio y envíelo a 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 la entrega, cree el siguiente archivo de audio y envíelo a 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
```

Después de confirmar la entrega, Amazon Lex envía una respuesta que confirma el cumplimiento de la intención:

```
{
    "contentType": "text/plain;charset=utf-8",
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers",
    "inputTranscript": "yes",
```

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

Paso siguiente

Ejercicio 2: Añadir un nuevo enunciado (AWS CLI)

Ejercicio 2: Añadir un nuevo enunciado (AWS CLI)

Con el fin de mejorar el modelo de machine learning que Amazon Lex utiliza para reconocer las solicitudes de los usuarios, agregue otro enunciado de muestra al bot.

El proceso de añadir un nuevo enunciado consta de cuatro pasos.

1. Utilice la operación GetIntent para obtener una intención de Amazon Lex.
2. Actualice la intención.
3. Use la operación PutIntent para enviar la intención actualizada de vuelta a Amazon Lex.
4. Use las operaciones GetBot y PutBot para volver a compilar los bots que usen la intención.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

La respuesta de la operación `GetIntent` contiene un campo llamado `checksum` que identifica una revisión específica de la intención. Debe proporcionar el valor de la suma de comprobación cuando utilice la operación PutIntent para actualizar una intención. Si no, recibirá el siguiente mensaje de error:

```
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**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie "\$LATEST" por $LATEST y sustituya la barra diagonal invertida (\) utilizada como carácter de continuación al final de cada línea por el signo de intercalación (^).

Para actualizar la intención **`OrderFlowers`** (AWS CLI)

1. En la AWS CLI, obtenga la intención de Amazon Lex. Amazon Lex envía el resultado a un archivo llamado **`OrderFlowers-V2.json.`**

   ```
   aws lex-models get-intent \
       --region region \
       --name OrderFlowers \
       --intent-version "\$LATEST" > OrderFlowers-V2.json
   ```

2. Abra **`OrderFlowers-V2.json`** en un editor de texto.

   1. Busque y elimine los campos `createdDate`, `lastUpdatedDate` y `version`.
   2. Añada la línea siguiente al campo `sampleUtterances`:

      ```
      I want to order flowers
      ```

   3. Guarde el archivo.

3. Envíe la intención actualizada a Amazon Lex con el siguiente comando:

   ```
   aws lex-models put-intent  \
       --region region \
       --name OrderFlowers \
       --cli-input-json file://OrderFlowers-V2.json
   ```

   Amazon Lex envía la siguiente respuesta:

   ```
   {
       "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"
}
```

Ahora que ha actualizado la intención, recompile los bots que la usen.

Para recompilar el bot **`OrderFlowersBot`** (AWS CLI)

1. En la AWS CLI, obtenga la definición del bot `OrderFlowersBot` y guárdela en un archivo con el comando siguiente:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V2.json
```

2. En un editor de texto, abra **`OrderFlowersBot-V2.json`**. Elimine los campos `createdDate`, `lastUpdatedDate`, `status` y `version`.
3. En un editor de texto, añada la siguiente línea a la definición de bot:

```
"processBehavior": "BUILD",
```

4. En la AWS CLI, compile una nueva revisión del bot ejecutando el siguiente comando:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V2.json
```

   La respuesta del servidor es:

```
{
    "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"
}
```

Paso siguiente

Ejercicio 3: adición de una función de Lambda (AWS CLI)

Ejercicio 3: adición de una función de Lambda (AWS CLI)

Agregue al bot una función de Lambda que valide las entradas del usuario y cumpla la intención del usuario.

El proceso de agregar una expresión de Lambda consta de cinco pasos.

1. Utilice la función de Lambda AddPermission para permitir que la intención `OrderFlowers` llame a la operación Invoke de Lambda.
2. Utilice la operación GetIntent para obtener la intención de Amazon Lex.
3. Actualice la intención para agregar la función de Lambda.
4. Use la operación PutIntent para enviar la intención actualizada de vuelta a Amazon Lex.
5. Use las operaciones GetBot y PutBot para volver a compilar los bots que usen la intención.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Si agrega una función de Lambda a una intención antes de agregar el permiso `InvokeFunction`, obtendrá el siguiente mensaje de error:

```
An error occurred (BadRequestException) when calling the
PutIntent operation: Lex is unable to access the Lambda
function Lambda function ARN in the context of intent
intent ARN.  Please check the resource-based policy on
the function.
```

La respuesta de la operación `GetIntent` contiene un campo llamado `checksum` que identifica una revisión específica de la intención. Cuando se usa la operación PutIntent para actualizar una intención, debe proporcionar el valor de la suma de comprobación. Si no, recibirá el siguiente mensaje de error:

```
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.
```

En este ejercicio se utiliza la función de Lambda desde Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola). Para obtener instrucciones para crear la función de Lambda, consulte Paso 3: creación de una función de Lambda (consola).

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie `"\$LATEST"` por `$LATEST`.

Adición de una función de Lambda a una intención

1. En la AWS CLI, añada el permiso `InvokeFunction` a la intención `OrderFlowers`:

```
aws lambda add-permission \
    --region region \
    --function-name OrderFlowersCodeHook \
    --statement-id LexGettingStarted-OrderFlowersBot \
    --action lambda:InvokeFunction \
    --principal lex.amazonaws.com \
    --source-arn "arn:aws:lex:region:account ID:intent:OrderFlowers:*"
    --source-account account ID
```

Lambda envía la siguiente respuesta:

```
{
    "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. Obtenga la intención de Amazon Lex. Amazon Lex envía el resultado a un archivo llamado **`OrderFlowers-V3.json`**.

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V3.json
```

3. Abra **`OrderFlowers-V3.json`** en un editor de texto.

   1. Busque y elimine los campos `createdDate`, `lastUpdatedDate` y `version`.
   2. Actualice el campo `fulfillmentActivity`:

```
    "fulfillmentActivity": {
        "type": "CodeHook",
        "codeHook": {
```

```
            "uri": "arn:aws:lambda:region:account
ID:function:OrderFlowersCodeHook",
            "messageVersion": "1.0"
        }
    }
```

3. Guarde el archivo.

4. En la AWS CLI, envíe la intención actualizada a Amazon Lex.

```
aws lex-models put-intent \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V3.json
```

Ahora que ha actualizado la intención, recompile el bot.

Para recompilar el bot **`OrderFlowersBot`**

1. En la AWS CLI, obtenga la definición del bot `OrderFlowersBot` y guárdela en un archivo:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V3.json
```

2. En un editor de texto, abra **`OrderFlowersBot-V3.json`**. Elimine los campos `createdDate`, `lastUpdatedDate`, `status` y `version`.

3. En el editor de texto, añada la siguiente línea a la definición del bot:

```
"processBehavior": "BUILD",
```

4. En la AWS CLI, cree una nueva revisión del bot:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V3.json
```

   La respuesta del servidor es:

```
{
    "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"
}
```

Paso siguiente

Ejercicio 4: Publicar una versión (AWS CLI)

Ejercicio 4: Publicar una versión (AWS CLI)

Ahora, cree una versión del bot que ha creado en el ejercicio 1. Una versión es una snapshot del bot. No se puede cambiar una versión después de crearla. La única versión de un bot que puede actualizar es la versión `$LATEST`. Para obtener más información acerca de las versiones, consulte Control de versiones y alias.

Para poder publicar una versión de un bot, debe publicar las intenciones que este utilice. Del mismo modo, debe publicar los tipos de slot a los que hacen referencias las intenciones. En general, para publicar una versión de un bot, haga lo siguiente:

1. Publique una versión de un tipo de slot con la operación CreateSlotTypeVersion.
2. Publique una versión de una intención con la operación CreateIntentVersion.
3. Publique una versión de un bot con la operación CreateBotVersion.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Temas

- Paso 1: Publicar el tipo de slot (AWS CLI)
- Paso 2: Publicar la intención (AWS CLI)
- Paso 3: Publicar el bot (AWS CLI)

Paso 1: Publicar el tipo de slot (AWS CLI)

Antes de publicar una versión de alguna intención que use un tipo de slot, debe publicar una versión de ese tipo de slot. En este caso, publique el tipo de slot `FlowerTypes`.

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie `"\$LATEST"` por `$LATEST` y sustituya la barra diagonal invertida (\) utilizada como carácter de continuación al final de cada línea por el signo de intercalación (^).

Para publicar un tipo de slot (AWS CLI)

1. En la AWS CLI, obtenga la última versión del tipo de slot:

```
aws lex-models get-slot-type \
    --region region \
    --name FlowerTypes \
    --slot-type-version "\$LATEST"
```

La respuesta de Amazon Lex es la siguiente. Registre la suma de comprobación para la revisión actual de la versión $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 una nueva versión del tipo de slot. Use la suma de comprobación que ha registrado en el paso anterior.

```
aws lex-models create-slot-type-version \
    --region region \
    --name FlowerTypes \
    --checksum "checksum"
```

La respuesta de Amazon Lex es la siguiente. Registre el número de versión para el paso siguiente.

```
{
    "version": "1",
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Paso siguiente

Paso 2: Publicar la intención (AWS CLI)

Paso 2: Publicar la intención (AWS CLI)

Antes de publicar una intención, tiene que publicar todos los tipos de slot a los que esta haga referencia. Los tipos de slot deben ser versiones numeradas, no la versión `$LATEST`.

En primer lugar, actualice la intención `OrderFlowers` para que utilice la versión del tipo de slot `FlowerTypes` que ha publicado en el paso anterior. Después publique una nueva versión de la intención `OrderFlowers`.

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie "\$LATEST" por $LATEST y sustituya la barra diagonal invertida (\) utilizada como carácter de continuación al final de cada línea por el signo de intercalación (^).

Para publicar una versión de una intención (AWS CLI)

1. En la AWS CLI, obtenga la versión $LATEST de la intención `OrderFlowers` y guárdela en un archivo:

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4.json
```

2. Abra el archivo **`OrderFlowers_V4.json`** en un editor de texto. Elimine los campos `createdDate`, `lastUpdatedDate` y `version`. Busque el tipo de slot `FlowerTypes` y cambie la versión por el número de versión que ha registrado en el paso anterior. El siguiente fragmento del archivo **`OrderFlowers_V4.json`** muestra la ubicación del cambio:

```
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "version",
            "sampleUtterances": []
        },
```

3. En la AWS CLI, guarde la revisión de la intención:

```
aws lex-models put-intent \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers_V4.json
```

4. Obtenga la suma de comprobación de la última revisión de la intención:

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4a.json
```

El siguiente fragmento de la respuesta muestra la suma de comprobación de la intención. Guárdelo para el siguiente paso.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
```

5. Publicar una nueva versión de la intención:

```
aws lex-models create-intent-version \
    --region region \
    --name OrderFlowers \
    --checksum "checksum"
```

El siguiente fragmento de la respuesta muestra la nueva versión de comprobación de la intención. Registre el número de versión para el paso siguiente.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "version",
```

Paso siguiente

Paso 3: Publicar el bot (AWS CLI)

Paso 3: Publicar el bot (AWS CLI)

Una vez que publicados todos los tipos de slot y las intenciones que se utilizan en el bot, puede publicar el bot.

Actualice el bot `OrderFlowersBot` para usar la intención `OrderFlowers` que ha actualizado en el paso anterior. Después, publique una nueva versión del bot `OrderFlowersBot`.

> **Note**
>
> El ejemplo de AWS CLI siguiente tiene formato para Unix, Linux y macOS. Para Windows, cambie `"\$LATEST"` por `$LATEST` y sustituya la barra diagonal invertida (\) utilizada como carácter de continuación al final de cada línea por el signo de intercalación (^).

Para publicar una versión de un bot, (AWS CLI)

1. En la AWS CLI, obtenga la versión `$LATEST` del bot `OrderFlowersBot` y guárdela en un archivo:

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias "\$LATEST" > OrderFlowersBot_V4.json
   ```

2. Abra el archivo **`OrderFlowersBot_V4.json`** en un editor de texto. Elimine los campos `createdDate`, `lastUpdatedDate`, `status` y `version`. Busque la intención `OrderFlowers` y cambie la versión por el número de versión que ha registrado en el paso anterior. El siguiente fragmento de **`OrderFlowersBot_V4.json`** muestra la ubicación del cambio.

   ```
       "intents": [
           {
               "intentVersion": "version",
               "intentName": "OrderFlowers"
           }
   ```

3. En la AWS CLI, guarde la nueva revisión del bot. Anote el número de versión que devuelve la llamada a `put-bot`.

   ```
   aws lex-models put-bot \
   ```

```
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot_V4.json
```

4. Consiga la suma de comprobación de la última revisión del bot. Utilice el número de versión devuelto en el paso 3.

```
aws lex-models get-bot \
    --region region \
    --version-or-alias version \
    --name OrderFlowersBot > OrderFlowersBot_V4a.json
```

El siguiente fragmento de la respuesta muestra la suma de comprobación del bot. Guárdelo para el siguiente paso.

```
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
```

5. Publicar una nueva versión del bot:

```
aws lex-models create-bot-version \
    --region region \
    --name OrderFlowersBot \
    --checksum "checksum"
```

El siguiente fragmento de la respuesta muestra la nueva versión de comprobación del bot.

```
    "checksum": "checksum",
    "abortStatement": {
        ...
    },
    "version": "1",
    "lastUpdatedDate": timestamp,
```

Paso siguiente

Ejercicio 5: Crear un alias (AWS CLI)

Ejercicio 5: Crear un alias (AWS CLI)

Un alias es un puntero hacia una versión específica de un bot. Con un alias, puede actualizar fácilmente la versión que usan las aplicaciones cliente. Para obtener más información, consulte Control de versiones y alias. Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Para crear un alias (AWS CLI)

1. En la AWS CLI, obtenga la versión del bot `OrderFlowersBot` que ha creado en Ejercicio 4: Publicar una versión (AWS CLI).

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias version > OrderFlowersBot_V5.json
   ```

2. En un editor de texto, abra **`OrderFlowersBot_v5.json`**. Busque y registre el número de versión.

3. En la AWS CLI, cree el alias del bot:

   ```
   aws lex-models put-bot-alias  \
       --region region \
       --name PROD \
       --bot-name OrderFlowersBot \
       --bot-version version
   ```

   A continuación se muestra la respuesta del servidor:

   ```
   {
       "name": "PROD",
       "createdDate": timestamp,
       "checksum": "checksum",
       "lastUpdatedDate": timestamp,
       "botName": "OrderFlowersBot",
       "botVersion": "1"
   }}
   ```

Paso siguiente

Ejercicio 6: Limpieza (AWS CLI)

Ejercicio 6: Limpieza (AWS CLI)

Elimine los recursos que ha creado y limpie la cuenta.

Puede eliminar únicamente los recursos que no se usan. En general, debería eliminar los recursos en el orden que se indica a continuación.

1. Elimine los alias para liberar los recursos de los bots.
2. Eliminar bots para liberar los recursos de las intenciones.
3. Eliminar intenciones para liberar los recursos de los tipos de slot.
4. Elimine el tipo de slot.

Para ejecutar los comandos de este ejercicio, debe conocer la región donde se ejecutarán los comandos. Para obtener una lista de regiones, consulte Cuotas de creación de modelos .

Para limpiar la cuenta (AWS CLI)

1. En la línea de comandos de la AWS CLI, elimine el alias:

```
aws lex-models delete-bot-alias \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot
```

2. En la línea de comandos de la AWS CLI, elimine el bot:

```
aws lex-models delete-bot \
    --region region \
    --name OrderFlowersBot
```

3. En la línea de comandos de la AWS CLI, elimine la intención:

```
aws lex-models delete-intent \
    --region region \
    --name OrderFlowers
```

4. En la línea de comandos de la AWS CLI, elimine el tipo de slot:

```
aws lex-models delete-slot-type \
    --region region \
    --name FlowerTypes
```

Ha eliminado todos los recursos que ha creado y ha limpiado su cuenta.

Control de versiones y alias

Amazon Lex admite la publicación de versiones de bots, intenciones y tipos de ranura para que pueda determinar la implementación que usan las aplicaciones cliente. Una versión es una instantánea numerada de su trabajo que puede publicar para su uso en diferentes partes del flujo de trabajo, como, por ejemplo, el desarrollo, la implementación beta y la producción.

Los bots de Amazon Lex también admiten alias. Un alias es un puntero hacia una versión específica de un bot. Con un alias, puede actualizar fácilmente la versión que usan las aplicaciones cliente. Por ejemplo, puede apuntar un alias hacia la versión 1 de su bot. Cuando esté listo para actualizar el bot, puede publicar la versión 2 y cambiar el alias para que apunte a la nueva versión. Dado que sus aplicaciones utilizan el alias en lugar de una versión específica, todos los clientes obtienen las nuevas funcionalidades sin necesidad de actualizarse.

Temas

Control de versiones

Cuando define una versión de un recurso de Amazon Lex, crea una instantánea del recurso para utilizarlo tal y como estaba en el momento de crear la versión. Después de crear una versión, esta no cambiará mientras continúa trabajando en su aplicación.

La versión $LATEST

Al crear un bot, una intención o un tipo de ranura de Amazon Lex, solo hay una versión: `$LATEST`.

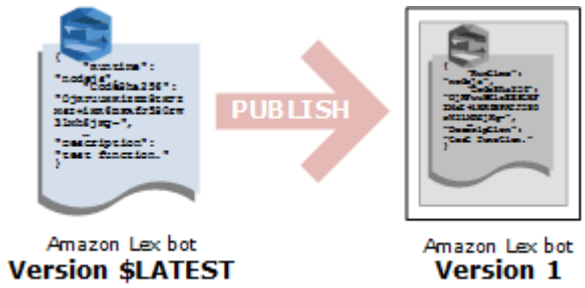
`$LATEST` es la copia de trabajo de su recurso. Solo puede actualizar la versión `$LATEST` y, hasta que publique la primera versión, `$LATEST` es la única versión del recurso con la que cuenta.

La versión $LATEST de un recurso es la única que puede utilizar la versión $LATEST de otro recurso. Por ejemplo, la versión $LATEST de un bot puede utilizar la versión $LATEST de una intención y la versión $LATEST de una intención puede utilizar la versión $LATEST de un tipo de slot.

La versión $LATEST del bot solo debería utilizarse para llevar a cabo pruebas manuales. Amazon Lex limita el número de solicitudes en tiempo de ejecución que puede realizar a la versión $LATEST del bot.

Publicación de una versión de un recurso de Amazon Lex

Al publicar un recurso, Amazon Lex realiza una copia de la versión $LATEST y la guarda como versión numerada. La versión publicada no se puede cambiar.



Puede crear y publicar versiones con la consola de Amazon Lex o la operación CreateBotVersion. Para ver un ejemplo, consulte Ejercicio 3: publicación de una versión y creación de un alias.

Si modifica la versión $LATEST de un recurso, puede publicar la nueva versión para que los cambios estén disponibles para sus aplicaciones cliente. Cada vez que publique una versión, Amazon Lex copiará la versión $LATEST para crear la nueva e incrementará el número de versión en 1. Los números de versión nunca se reutilizan. Por ejemplo, si suprime la versión numerada del recurso 10 y, a continuación, la vuelve a crear, el siguiente número de versión que asigne Amazon Lex será el 11.

Antes de publicar un bot, debe hacer que apunte a una versión numerada de cualquier intención que utilice. Si intenta publicar una nueva versión de un bot que utiliza la versión $LATEST de una intención, Amazon Lex devuelve la excepción HTTP 400 Bad Request. Antes de publicar una versión numerada de la intención, debe hacer que esta apunte a una versión numerada de cualquier tipo de slot que utilice. De lo contrario, obtendrá una excepción HTTP 400 Bad Request.

> **Note**
>
> Amazon Lex publica una nueva versión solo si la última versión publicada es diferente de la versión $LATEST. Si intenta publicar la versión $LATEST sin modificarla, Amazon Lex no creará ni publicará una nueva versión.

Actualización de un recurso de Amazon Lex

Solo puede actualizar la versión $LATEST de un bot, intención o tipo de slot de Amazon Lex. Las versiones publicadas no se pueden cambiar. Puede publicar una nueva versión en cualquier momento después de actualizar un recurso en la consola o con las operaciones CreateBotVersion, CreateIntentVersion o CreateSlotTypeVersion.

Eliminación de un recurso o versión de Amazon Lex

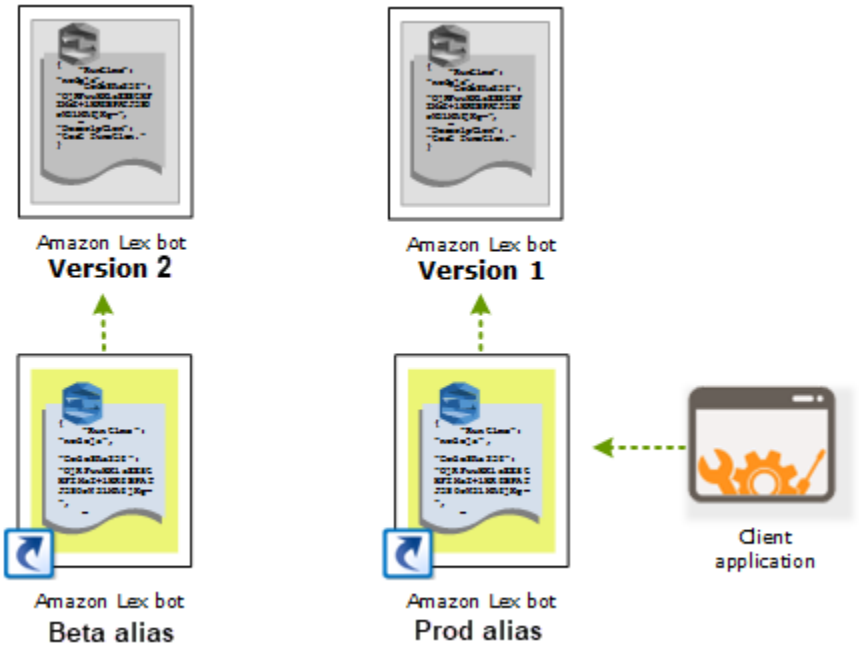
Amazon Lex permite eliminar un recurso o versión mediante la consola o una de las operaciones de la API:

- DeleteBot
- DeleteBotVersion
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion

Alias

Un alias es un puntero a una versión específica de un bot de Amazon Lex. Utilice un alias para permitir que las aplicaciones cliente utilicen una versión específica del bot sin necesidad de que la aplicación realice un seguimiento de la versión de que se trata.

En el siguiente ejemplo se muestran dos versiones de un bot de Amazon Lex, la versión 1 y la versión 2. Cada una de estas versiones de bot tiene un alias asociado, BETA y PROD, respectivamente. Las aplicaciones cliente usan el alias PROD para acceder al bot.



Al crear una segunda versión del bot, puede actualizar el alias para que apunte a la nueva versión del bot utilizando la consola o la operación PutBot. Al cambiar el alias, todas sus aplicaciones cliente utilizan la nueva versión. Si hay un problema con la nueva versión, puede volver a la versión anterior simplemente haciendo que el alias apunte hacia dicha versión.

> **Note**
>
> Aunque puede probar la versión $LATEST de un bot en la consola, le recomendamos que, al integrar un bot con la aplicación cliente, primero publique una versión y cree un alias que apunte a dicha versión. Utilice el alias de la aplicación cliente por las razones explicadas en esta sección. Al actualizar un alias, Amazon Lex esperará a que transcurra el tiempo de espera en todas las sesiones actuales antes de comenzar a utilizar la nueva versión. Para obtener más información acerca del tiempo de espera de la sesión, consulte the section called “Definición del tiempo de espera de la sesión”.

Uso de funciones de Lambda

Puede crear funciones de AWS Lambda para utilizarlas como enlaces de código en un bot de Amazon Lex. En la configuración de la intención, puede identificar funciones de Lambda para realizar las tareas de inicialización y validación, cumplimiento o ambas.

Le recomendamos que utilice una función de Lambda como un enlace de código en su bot. Sin una función de Lambda, el bot devuelve la información de la intención a la aplicación cliente para que la cumpla.

Temas

Formato del evento de entrada y de la respuesta de la función de Lambda

En esta sección se describe la estructura de los datos de eventos que Amazon Lex proporciona a una función de Lambda. Utilice esta información para analizar la entrada en el código de Lambda. También explica el formato de la respuesta que Amazon Lex espera que devuelva la función de Lambda.

Temas

Formato del evento de entrada

A continuación se muestra el formato general de un evento de Amazon Lex que se pasa a una función de Lambda. Utilice esta información cuando escriba la función de Lambda.

> **Note**
>
> El formato de entrada puede cambiar sin un cambio correspondiente en `messageVersion`. El código no debería devolver un error si hay nuevos campos.

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

Tenga en cuenta la siguiente información adicional sobre los campos del evento:

- currentIntent: proporciona los campos `name`, `slots`, `slotDetails` y `confirmationStatus` de la intención.

  `nluIntentConfidenceScore` es la confianza que Amazon Lex tiene en que la intención actual es la mejor para cumplir con la intención actual del usuario.

`slots` es una asignación de nombre de ranura, que se configura para la intención, para rellenar las ranuras con los valores que Amazon Lex ha reconocido en la conversación con el usuario. Un valor de slot es nulo hasta que el usuario proporciona un valor.

El valor de ranura del evento de entrada podría no coincidir con uno de los valores configurados para la ranura. Por ejemplo, si el usuario responde a la pregunta “¿Qué color de automóvil prefiere?”, con “pizza”, Amazon Lex devolverá “pizza” como valor de ranura. Su función debería validar los valores para asegurarse de que tienen sentido en el contexto.

`slotDetails` proporciona información adicional acerca de un valor de slot. La matriz `resolutions` contiene una lista de los valores adicionales que se reconocen para el slot. Cada slot puede tener un máximo de cinco valores.

El campo `originalValue` contiene el valor que el usuario ha introducido para el slot. Cuando el tipo de slot se configura para devolver el valor de resolución superior como valor de slot, `originalValue` podría ser diferente al valor del campo `slots`.

`confirmationStatus` proporciona la respuesta del usuario a un mensaje de confirmación, si existe. Por ejemplo, si Amazon Lex pregunta “¿Desea pedir una pizza de quesos grande?”, el valor de este campo puede ser `Confirmed` o `Denied` en función de la respuesta del usuario. De lo contrario, el valor de este campo es `None`.

Si el usuario confirma la intención, Amazon Lex establece el valor del campo en `Confirmed`. Si el usuario deniega la intención, Amazon Lex establece el valor del campo en `Denied`.

En la respuesta de confirmación, un enunciado del usuario puede proporcionar actualizaciones de slot. Por ejemplo, el usuario podría decir: "sí, cambiar el tamaño a mediana". En este caso, el evento de Lambda subsiguiente tendrá el valor de ranura actualizado y `PizzaSize` establecerá

el valor en `medium`. Amazon Lex establece `confirmationStatus` en `None` porque el usuario ha modificado algunos datos de ranura, de modo que la función de Lambda tiene que validar los datos del usuario.

- alternativeIntents: si ha habilitado las puntuaciones de confianza, Amazon Lex devuelve hasta cuatro intenciones alternativas. Cada intención incluye una puntuación que indica el nivel de confianza que Amazon Lex tiene en que la intención es la correcta, según el enunciado del usuario.

  El contenido de las intenciones alternativas es el mismo que el contenido del campo `currentIntent`. Para obtener más información, consulte Uso de puntuaciones de confianza.

- bot: información sobre el bot que ha procesado la solicitud.
  - `name`: nombre del bot que ha procesado la solicitud.
  - `alias`: alias de la versión del bot que ha procesado la solicitud.
  - `version`: versión del bot que ha procesado la solicitud.

- userId: este valor lo proporciona la aplicación cliente. Amazon Lex simplemente lo pasa a la función de Lambda.

- inputTranscript: texto que se utiliza para procesar la solicitud.

  Si la entrada era texto, el campo `inputTranscript` contiene el texto que ha introducido el usuario.

  Si la entrada era una transmisión de audio, el campo `inputTranscript` contiene el texto que se ha extraído de la transmisión de audio. Este es el texto que se procesa realmente para reconocer los valores de las intenciones y slot.

- invocationSource: para indicar por qué Amazon Lex llama la función de Lambda, se establece en uno de los siguientes valores:
  - `DialogCodeHook`: Amazon Lex configura este valor para que la función de Lambda inicialice la función y valide la entrada de datos del usuario.

    Si la intención está configurada para invocar una función de Lambda como enlace de código de validación e inicialización, Amazon Lex invoca la función de Lambda especificada en cada entrada del usuario (enunciado) después de que Amazon Lex haya reconocido la intención.

    > **Note**
    >
    > Si la intención no está clara, Amazon Lex no podrá llamar la función de Lambda.

  - `FulfillmentCodeHook`: Amazon Lex establece este valor para indicar a la función de Lambda que cumpla con una intención.

    Si la intención está configurada para invocar una función de Lambda como enlace de código de cumplimiento, Amazon Lex establece este valor en `invocationSource` después de disponer de todos los datos de ranura para cumplir con la intención.

  En la configuración de una intención puede haber dos funciones de Lambda distintas, una para inicializar y validar los datos de usuario y otra para cumplir con la intención. También puede utilizar una función de Lambda para ambas acciones. En ese caso, la función de Lambda puede utilizar el valor `invocationSource` para seguir la ruta de código correcta.

- outputDialogMode: para cada entrada de usuario, el cliente envía la solicitud a Amazon Lex mediante una de las operaciones de la API en tiempo de ejecución: PostContent o PostText. Amazon Lex utiliza los parámetros de la solicitud para determinar si la respuesta al cliente es texto o voz y define el valor de este campo según corresponda.

La función de Lambda puede utilizar esta información para generar un mensaje apropiado. Por ejemplo, si el cliente espera una respuesta de voz, la función de Lambda podría responder en formato SSML (Speech Synthesis Markup Language) en lugar de con texto.

- messageVersion: la versión del mensaje que identifica el formato de los datos de evento que se van a pasar a la función de Lambda y el formato que se espera en la respuesta de la función de Lambda.

  > **Note**
  >
  > Puede configurar este valor a la hora de definir una intención. En la implementación actual, solo se admite la versión 1.0 de los mensajes. Por lo tanto, la consola asume el valor predeterminado de 1.0 y no muestra el mensaje de la versión.

- sessionAttributes: atributos de la sesión específicos de la aplicación que el cliente envía en la solicitud. Si desea que Amazon Lex los incluya en la respuesta al cliente, la función de Lambda debe devolverlos a Amazon Lex en la respuesta. Para obtener más información, consultar Definición de atributos de la sesión

- requestAttributes: atributos específicos de la solicitud que el cliente envía en la solicitud. Utilice los atributos de solicitud para pasar información que no tiene por qué persistir durante toda la sesión. Si no hay atributos de solicitud, el valor será nulo. Para obtener más información, consultar Definición de los atributos de solicitud

- recentIntentSummaryView: información acerca del estado de una intención. Puede consultar información acerca de las tres últimas intenciones que se hayan utilizado. Puede utilizar esta información para establecer los valores de la intención o para volver a una intención anterior. Para obtener más información, consulte Administración de sesiones con la API de Amazon Lex.

- sentimentResponse: el resultado de un análisis de opiniones de Amazon Comprehend del último enunciado. Puede utilizar esta información para administrar el flujo de la conversación del bot en

función del sentimiento expresado por el usuario. Para obtener más información, consulte Análisis de opiniones.

- kendraResponse: el resultado de una consulta en un índice de Amazon Kendra. Solo está presente en la entrada de un enlace de código de cumplimentación y solo cuando la intención es más amplia que la intención de `AMAZON.KendraSearchIntent` incorporada. El campo contiene toda la respuesta de la búsqueda de Amazon Kendra. Para obtener más información, consulte AMAZON.KendraSearchIntent.

- activeContexts: uno o más contextos activos durante este turno de una conversación con el usuario.
  - timeToLive: la duración o el número de turnos de la conversación con el usuario en que el contexto permanece activo.
  - name: el nombre del contexto.
  - parameters: una lista de pares clave/valor que contiene el nombre y el valor de las ranuras de la intención que ha activado el contexto

  Para obtener más información, consulte Establecimiento del contexto de la intención.

Formato de respuesta

Amazon Lex espera una respuesta de una función de Lambda con el siguiente formato:

```
{
  "sessionAttributes": {
    "key1": "value1",
    "key2": "value2"
    ...
  },
  "recentIntentSummaryView": [
    {
       "intentName": "Name",
       "checkpointLabel": "Label",
       "slots": {
         "slot name": "value",
         "slot name": "value"
        },
```

```
      "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)",
       "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
 Close",
       "fulfillmentState": "Fulfilled or Failed",
       "slotToElicit": "Next slot to elicit"
    }
  ],
  "activeContexts": [
     {
       "timeToLive": {
          "timeToLiveInSeconds": seconds,
          "turnsToLive": turns
      },
      "name": "name",
      "parameters": {
        "key name": "value"
      }
    }
  ],
  "dialogAction": {
    "type": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or Close",
    Full structure based on the type field. See below for details.
  }
}
```

La respuesta se compone de cuatro campos. Los campos `sessionAttributes`, `recentIntentSummaryView` y `activeContexts` son opcionales, pero el campo `dialogAction` es obligatorio. El contenido del campo `dialogAction` depende del valor del campo `type`. Para obtener más información, consulte dialogAction.

sessionAttributes

Opcional. Si incluye el campo `sessionAttributes`, puede estar vacío. Si la función de Lambda no devuelve los atributos de la sesión, se pasa el último campo `sessionAttributes` conocido a través de la API o se mantiene la función de Lambda. Para obtener más información, consulte las operaciones PostContent y PostText.

```
  "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"
  }
```

recentIntentSummaryView

Opcional. Si se incluye, establece los valores de una o varias intenciones recientes. Puede incluir información de hasta tres intenciones. Por ejemplo, puede establecer los valores de intenciones anteriores en función de la información recopilada por la intención actual. La información del resumen debe ser válida para la intención. Por ejemplo, el nombre de la intención debe ser una intención en el bot. Si incluye un valor de slot en la vista de resumen, el slot debe existir en la intención. Si no incluye `recentIntentSummaryView` en la respuesta, todos los valores de las intenciones recientes permanecen sin cambios. Para obtener más información, consulte la operación PutSession o el tipo de datos 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. Si se incluye, establece el valor para uno o más contextos. Por ejemplo, puede incluir un contexto para que una o más intenciones que tengan ese contexto como entrada se puedan reconocer en el siguiente turno de la conversación.

Los valores de tiempo de vida de los contextos activos que no se han incluido en la respuesta se reducen, aunque los contextos pueden seguir activos en la siguiente solicitud.

Si especifica un tiempo de vida de 0 para un contexto incluido en el evento de entrada, estará inactivo en la próxima solicitud.

Para obtener más información, consulte Establecimiento del contexto de la intención.

dialogAction

Obligatorio. El campo `dialogAction` indica a Amazon Lex el siguiente paso y describe lo que cabe esperar del usuario después de que Amazon Lex devuelva una respuesta al cliente.

El campo `type` indica el siguiente paso. También determina el resto de los campos que tiene que proporcionar la función de Lambda como parte del valor `dialogAction`.

- `Close`: indica a Amazon Lex que no debe esperar una respuesta del usuario. Por ejemplo, "Your pizza order has been placed" no requiere una respuesta.

  El campo `fulfillmentState` es obligatorio. Amazon Lex utiliza este valor para definir el campo `dialogState` en la respuesta PostContent o PostText a la aplicación cliente. Los campos `message` y `responseCard` son opcionales. Si no especifica un mensaje, Amazon Lex usa el mensaje de despedida o el mensaje de seguimiento que se ha configurado para la intención.

```
"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 a Amazon Lex que se espera que el usuario dé una respuesta afirmativa o negativa para confirmar o denegar la intención actual.

  Debe incluir los campos `intentName` y `slots`. El campo `slots` debe incluir una entrada por cada uno de los slots de la intención especificada que se han rellenado. No es necesario que incluya en el campo `slots` una entrada para los slots sin rellenar. Debe incluir el campo `message` si el campo `confirmationPrompt` de intención es nulo. El contenido del campo `message` devuelto por la función de Lambda tiene prioridad sobre el objeto `confirmationPrompt` especificado en la intención. El campo `responseCard` es 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`: indica a Amazon Lex que elija el siguiente paso en función de la configuración del bot. Si la respuesta no incluye atributos de sesión, Amazon Lex mantiene los atributos de sesión existentes. Si desea que un valor de slot sea nulo, no debe incluir el campo de slot en la solicitud. Se producirá una excepción `DependencyFailedException` si la función de cumplimentación devuelve la acción de diálogo `Delegate` sin eliminar ningún slot.

  Los campos `kendraQueryFilterString` y `kendraQueryRequestPayload` son opcionales y solo se utilizan cuando la intención procede de la intención integrada en `AMAZON.KendraSearchIntent`. Para obtener más información, 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`: indica a Amazon Lex que se espera que el usuario responda con un enunciado que incluya una intención. Por ejemplo, "I want a large pizza", que indica la intención `OrderPizzaIntent`. Por otra parte, el enunciado “large” no es suficiente para que Amazon Lex deduzca la intención del usuario.

Los campos `message` y `responseCard` son opcionales. Si no proporciona un mensaje, Amazon Lex utiliza una de las preguntas aclaratorias del bot. Si no hay ninguna pregunta aclaratoria definida, Amazon Lex devuelve la excepción 400 de solicitud errónea.

```
{
  "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`: indica a Amazon Lex que se espera que el usuario proporcione un valor de ranura en la respuesta.

  Los campos `intentName`, `slotToElicit` y `slots` son obligatorios. Los campos `message` y `responseCard` son opcionales. Si no especifica un mensaje, Amazon Lex utiliza una de las solicitudes para obtener valores de ranura configuradas para la ranura.

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

Esquemas de Amazon Lex y AWS Lambda

La consola de Amazon Lex ofrece bots de ejemplo (denominados esquemas de bot) que están preconfigurados para que pueda crear y probar rápidamente un bot en la consola. Para cada uno de estos esquemas de bot, también hay disponibles esquemas de funciones de Lambda. Estos

proyectos proporcionan código de muestra que funciona con sus bots específicos. Puede utilizar estos esquemas para crear rápidamente un bot configurado con una función de Lambda como enlace de código y probar la configuración integral sin tener que escribir código.

Puede usar los siguientes esquemas de bot de Amazon Lex y los esquemas de funciones de AWS Lambda correspondientes como enlaces de código para los bots:

- Esquema de Amazon Lex: `OrderFlowers`
  - Esquema de AWS Lambda: `lex-order-flowers-python`
- Esquema de Amazon Lex: `ScheduleAppointment`
  - Esquema de AWS Lambda: `lex-make-appointment-python`
- Esquema de Amazon Lex: `BookTrip`
  - Esquema de AWS Lambda: `lex-book-trip-python`

Para crear un bot con un esquema y configurarlo de modo que utilice una función de Lambda como enlace de código, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola). Para ver un ejemplo de cómo utilizar otros proyectos, consulte Ejemplos adicionales: creación de bots de Amazon Lex.

Actualización de un esquema para una configuración regional específica

Si utiliza un esquema en una configuración regional que no sea Inglés (EE. UU.) (en-US), debe actualizar el nombre de todas las instancias para que incluyan la configuración regional. Por ejemplo, si utiliza el esquema `OrderFlowers`, debe hacer lo siguiente.

- Busque la función `dispatch` cerca del final del código de la función de Lambda.
- En la función `dispatch`, actualice el nombre de la intención para incluir la configuración regional que está utilizando. Por ejemplo, si utiliza la configuración regional en inglés (Australia) (en-AU), modifique la línea:

  ```
  if intent_name == 'OrderFlowers':
  ```

  a

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

Aunque los nombres de las intenciones sean distintos en otros esquemas, es necesario actualizarlos tal como se describe en el ejemplo anterior antes de utilizarlos.

Implementación de bots de Amazon Lex

En esta sección se proporcionan ejemplos de implementación de bots de Amazon Lex en diferentes plataformas de mensajería y en aplicaciones móviles.

Temas

Implementación de un bot de Amazon Lex en una plataforma de mensajería

En esta sección se explica cómo implementar bots de Amazon Lex en las plataformas de mensajería de Facebook, Slack y Twilio.

> **Note**
>
> Al almacenar las configuraciones de Facebook, Slack o Twilio, Amazon Lex utiliza claves administradas por el cliente de AWS Key Management Service para cifrar la información. La primera vez que crea un canal a una de estas plataformas de mensajería, Amazon Lex crea una clave administrada por el cliente predeterminada (`aws/lex`). También puede crear su propia clave administrada por el cliente con AWS KMS. Esto le da más flexibilidad, incluida la capacidad de crear, rotar y deshabilitar las claves. También puede definir controles de acceso y auditar las claves de cifrado que se utilizan para proteger sus datos. Para obtener más información, consulte AWS Key Management Service Developer Guide.

Cuando una plataforma de mensajería envía una solicitud a Amazon Lex, incluye información específica de la plataforma como atributo de la solicitud para la función de Lambda. Utilice estos atributos para personalizar el comportamiento del bot. Para obtener más información, consulte Definición de los atributos de solicitud.

Todos los atributos toman el espacio de nombres `x-amz-lex:` como prefijo. Por ejemplo, el atributo `user-id` se denomina `x-amz-lex:user-id`. Existen atributos comunes que se envían por todas las plataformas de mensajería, además de atributos específicos para cada plataforma. En las tablas

siguientes se muestran los atributos de solicitud que envían las plataformas de mensajería a la función de Lambda del bot.

Atributos de solicitud comunes

Atributo	Descripción
`channel-id`	El identificador del punto de conexión del canal de Amazon Lex.
`channel-name`	El nombre del canal de Amazon Lex.
`channel-type`	Uno de los valores siguientes: • `Facebook` • `Kik` • `Slack` • `Twilio-SMS`
`webhook-endpoint-url`	El punto de conexión de Amazon Lex para el canal.

Atributos de solicitud de Facebook

Atributo	Descripción
`user-id`	El identificador de Facebook del remitente. Consulte https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.
`facebook-page-id`	El identificador de la página de Facebook del remitente. Consulte https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.

Atributos de solicitud de Kik

Atributo	Descripción
`kik-chat-id`	Identificador de la conversación en la que está involucrado su bot. Para obtener más información, consulte https://dev.kik.com/#/docs/messaging#message-formats.
`kik-chat-type`	Tipo de conversación desde la que se originó el mensaje. Para obtener más información, consulte https://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-id`	El UUID que identifica el mensaje. Para obtener más información, consulte https://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-type`	Tipo de mensaje. Para obtener más información, consulte https://dev.kik.com/#/docs/messaging#message-types.

Atributos de solicitud de Twilio

Atributo	Descripción
`user-id`	El número de teléfono del remitente (propiedad "From"). Consulte https://www.twilio.com/docs/api/rest/message.
`twilio-target-phone-number`	El número de teléfono del destinatario (propiedad "To"). Consulte https://www.twilio.com/docs/api/rest/message.

Atributos de solicitud de Slack

Atributo	Descripción
`user-id`	El identificador de usuario de Slack. Consulte https://api.slack.com/types/user.
`slack-team-id`	El identificador del equipo que envió el mensaje. Consulte https://api.slack.com/methods/team.info.

Atributo	Descripción
`slack-bot-token`	El token del desarrollador que proporciona al bot acceso a las API de Slack. Consulte https://api.slack.com/docs/token-types.

Integración de un bot de Amazon Lex con Facebook Messenger

Este ejercicio muestra cómo integrar Facebook Messenger con un bot de Amazon Lex. Debe realizar los pasos siguientes:

1. crear un bot de Amazon Lex,
2. Cree una aplicación para Facebook.
3. integrar Facebook Messenger con el bot de Amazon Lex
4. Valide la integración.

Temas

Paso 1: creación de un bot de Amazon Lex

Si aún no tiene un bot de Amazon Lex, cree uno e impleméntelo. En este tema, supongamos que está utilizando el bot que ha creado en el ejercicio de introducción 1. También puede utilizar cualquiera de los ejemplos de bot que se proporcionan en esta guía. Para el Ejercicio de introducción 1, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).

1. Cree un bot de Amazon Lex. Para obtener instrucciones, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).
2. Implemente el bot y cree un alias. Para obtener instrucciones, consulte Ejercicio 3: publicación de una versión y creación de un alias.

Paso 2: creación de una aplicación para Facebook

En el portal de desarrolladores de Facebook, cree una aplicación para Facebook y una página de Facebook. Para obtener instrucciones, consulte la Guía de inicio rápido en la plataforma de documentación de Facebook Messenger. Anote lo siguiente:

- El valor de secreto de la aplicación (App Secret) de Facebook
- El token de acceso de Page Access Token de la página de Facebook

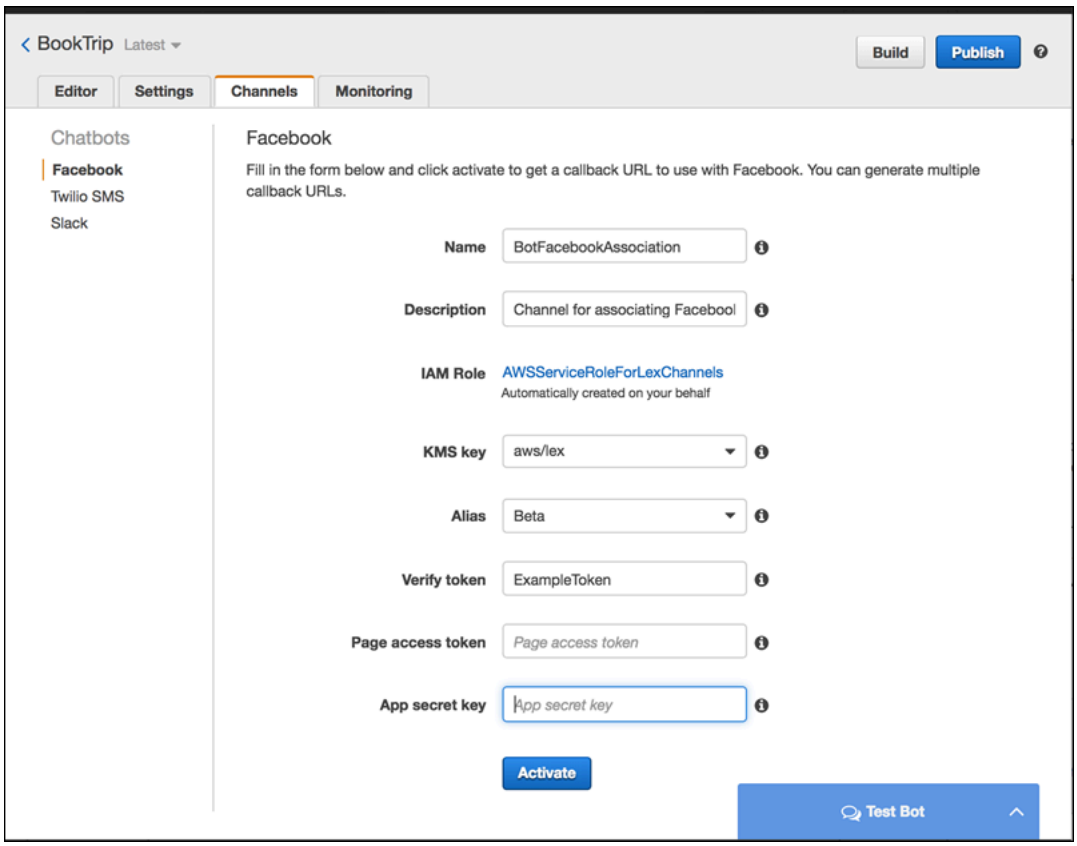
Paso 3: integración de Facebook Messenger con el bot de Amazon Lex

En esta sección, integrará Facebook Messenger con el bot de Amazon Lex.

Cuando acabe este paso, la consola ofrece una URL de devolución de llamada. Anote esta URL.

Para integrar Facebook Messenger con su bot

1. a. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
   b. Elija un bot de Amazon Lex.
   c. Elija Channels.
   d. Elija Facebook en Chatbots. La consola muestra la página de integración de Facebook.
   e. En la página de integración de Facebook, haga lo siguiente:
      - Escriba el siguiente nombre: `BotFacebookAssociation`.
      - En KMS key, elija aws/lex.
      - En Alias, elija el alias del bot.
      - En Verify token, escriba un token. Puede ser cualquier cadena que elija (por ejemplo, `ExampleToken`). Utilizará este mismo token más adelante en el portal de desarrolladores de Facebook cuando configure el webhook.
      - En Page access token, escriba el token que ha obtenido de Facebook en el paso 2.
      - En App secret key, escriba la clave que ha obtenido de Facebook en el paso 2.



f. Elija Activar.

La consola crea la asociación de canal de bot y devuelve una URL de devolución de llamada. Anote esta URL.

2. En el portal de desarrolladores de Facebook, elija su aplicación.
3. Elija el producto Messenger y luego Setup webhooks en la sección Webhooks de la página.

Para obtener instrucciones, consulte la Guía de inicio rápido en la plataforma de documentación de Facebook Messenger.

4. En la página webhook del asistente de suscripción, haga lo siguiente:

- En URL de respuesta, escriba la URL de respuesta que ha proporcionado en la consola de Amazon Lex anteriormente en el procedimiento.
- En Verificar token, escriba el mismo token que ha utilizado en Amazon Lex.
- Elija Subscription Fields (messages, messaging_postbacks y messaging_optins).
- Elija Verify and Save. De esta manera se inicia un protocolo de enlace entre Facebook y Amazon Lex.

5. Habilite la integración de webhooks. Elija la página que acaba de crear y, a continuación, elija subscribe.

> **Note**
>
> Si actualiza o vuelve a crear un webhook, deberá cancelar la suscripción y, a continuación, suscribirse a la página de nuevo.

Paso 4: comprobación de la integración

Ahora puede iniciar una conversación desde Facebook Messenger con el bot de Amazon Lex.

1. Abra su página de Facebook y elija Message.
2. En la ventana de Messenger, utilice los mismos enunciados de prueba que se facilitaron en Paso 1: creación de un bot de Amazon Lex (consola).

Integración de un bot de Amazon Lex con Kik

Este ejercicio proporciona instrucciones para la integración de un bot de Amazon Lex con la aplicación de mensajería Kik. Debe realizar los pasos siguientes:

1. Creación de un bot de Amazon Lex
2. Cree un bot de Kik mediante la aplicación y el sitio web de Kik.
3. Integre el bot de Amazon Lex con el bot de Kik mediante la consola de Amazon Lex.
4. Participe en una conversación con el bot de Amazon Lex con Kik para probar la asociación entre el bot de Amazon Lex y Kik.

Temas

Paso 1: creación de un bot de Amazon Lex

Si aún no tiene un bot de Amazon Lex, cree uno e impleméntelo. En este tema, supongamos que está utilizando el bot que ha creado en el ejercicio de introducción 1. También puede utilizar cualquiera de los ejemplos de bot que se proporcionan en esta guía. Para el ejercicio de introducción 1, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola)

1. Cree un bot de Amazon Lex. Para obtener instrucciones, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).
2. Implemente el bot y cree un alias. Para obtener instrucciones, consulte Ejercicio 3: publicación de una versión y creación de un alias.

Paso siguiente

Paso 2: creación de un bot de Kik

Paso 2: creación de un bot de Kik

En este paso, se utiliza la interfaz de usuario de Kik para crear un bot de Kik. La información generada al crear el bot se utiliza para conectarlo al bot de Amazon Lex.

1. Si aún no lo ha hecho, descargue e instale la aplicación de Kik e inscríbase en una cuenta de Kik. Si tiene una cuenta, inicie sesión.
2. Abra el sitio web de Kik en https://dev.kik.com/. Deje abierta la ventana del navegador.
3. En la aplicación de Kik, elija el icono de engranaje para abrir la configuración y, a continuación, elija Your Kik Code.
4. Escanee el código de Kik en el sitio web de Kik para abrir el chatbot de Botsworth. Elija Yes para abrir el panel Bot.
5. En la aplicación de Kik, elija Create a Bot. Siga las instrucciones para crear su bot de Kik.
6. Una vez creado el bot, elija Configuration en el navegador. Asegúrese de que está seleccionado el bot nuevo.
7. Anote el nombre del bot y la clave de API para la siguiente sección.

Paso siguiente

Paso 3: integración del bot de Kik con el bot de Amazon Lex

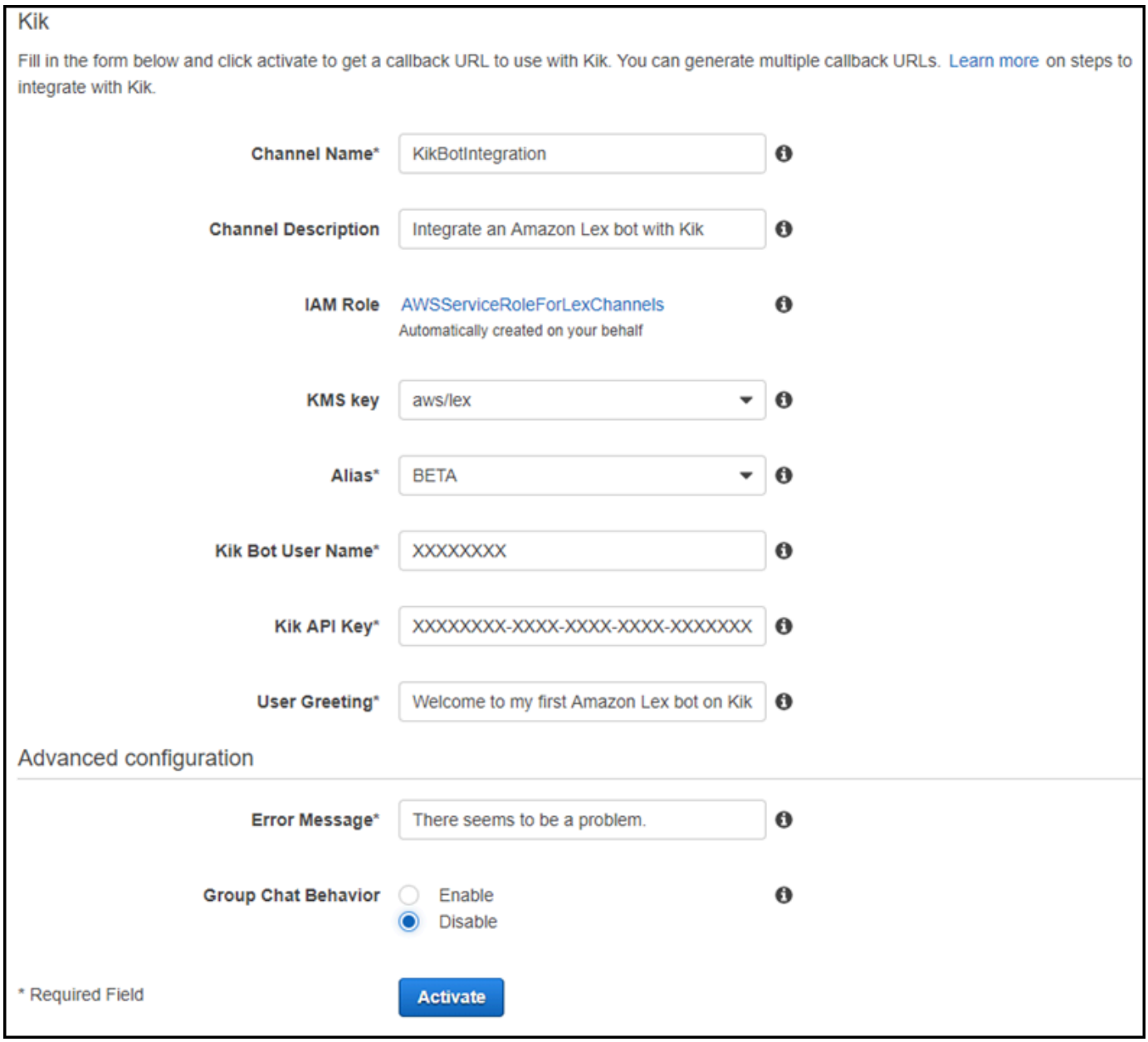
Paso 3: integración del bot de Kik con el bot de Amazon Lex

Ahora que ha creado un bot de Amazon Lex y un bot de Kik, puede crear una asociación de canal entre ellos en Amazon Lex. Cuando se activa la asociación, Amazon Lex configura automáticamente una dirección URL de devolución de llamada con Kik.

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon en https://console.aws.amazon.com/lex/.
2. Elija el bot de Amazon Lex que ha creado en el paso 1.
3. Elija la pestaña Channels.
4. En la sección Channels, elija Kik.
5. En la página de Kik, especifique lo siguiente:

   - Escriba un nombre. Por ejemplo, `BotKikIntegration`.
   - Escriba una descripción.
   - Elija "aws/lex" en el menú desplegable KMS key.
   - En Alias, elija un alias en la lista desplegable.
   - En Kik bot user name, escriba el nombre que asignó al bot en Kik.
   - En Kik API key, escriba la clave de API que asignó al bot en Kik.
   - En User greeting, escriba el saludo que desea que envíe el bot la primera vez que un usuario converse con él.
   - El Error message, introduzca un mensaje de error que se mostrará al usuario cuando no se entienda parte de la conversación.
   - En Group chat behavior, elija una de estas opciones:
     - Enable: permite que todo el grupo de chat interactúe con el bot en una única conversación.
     - Disable: restringe la conversación a un usuario del grupo de chat.

   - Elija Activate para crear la asociación y enlazarla al bot de Kik.



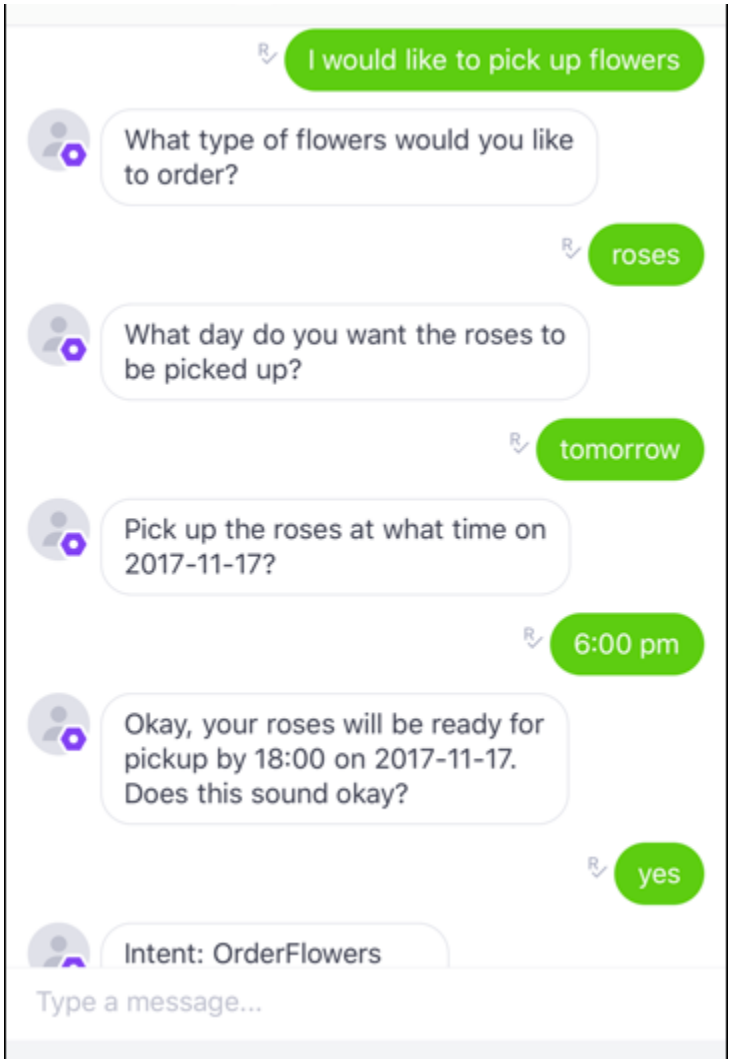
Paso siguiente

[Paso 4: comprobación de la integración](#)

Paso 4: comprobación de la integración

Ahora que ha creado una asociación entre el bot de Amazon Lex y Kik, puede utilizar la aplicación Kik para probar la asociación.

1. Inicie la aplicación de Kik e inicie sesión. Seleccione el bot que ha creado.
2. Puede probar el bot con la conversación siguiente:



A medida que introduzca cada frase, el bot de Amazon Lex responderá a través de Kik con la solicitud que haya creado para cada ranura.

Integración de un bot de Amazon Lex con Slack

Este ejercicio proporciona instrucciones para la integración de un bot de Amazon Lex con la aplicación de mensajería Slack. Debe realizar los pasos siguientes:

1. crear un bot de Amazon Lex,
2. crear una aplicación de mensajería Slack,
3. integrar la aplicación Slack con el bot de Amazon Lex,

4. Probar la integración entablando una conversación con el bot de Amazon Lex. Puede enviar mensajes con la aplicación Slack y realizar una prueba en una ventana del navegador.

Temas

Paso 1: creación de un bot de Amazon Lex

Si aún no tiene un bot de Amazon Lex, cree uno e impleméntelo. En este tema, supongamos que está utilizando el bot que ha creado en el ejercicio de introducción 1. También puede utilizar cualquiera de los ejemplos de bot que se proporcionan en esta guía. Para el ejercicio de introducción 1, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola)

1. Cree un bot de Amazon Lex. Para obtener instrucciones, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).
2. Implemente el bot y cree un alias. Para obtener instrucciones, consulte Ejercicio 3: publicación de una versión y creación de un alias.

Paso siguiente

Paso 2: inscripción en Slack y creación de un equipo de Slack

Paso 2: inscripción en Slack y creación de un equipo de Slack

Inscríbase para obtener una cuenta de Slack y crear un equipo. Para obtener instrucciones, consulte Uso de Slack. En la siguiente sección, debe crear una aplicación Slack, que pueda instalar cualquier equipo de Slack.

Paso siguiente

Paso 3: creación de una aplicación Slack

Paso 3: creación de una aplicación Slack

En esta sección, hará lo siguiente:

1. Cree una aplicación Slack en la consola API de Slack
2. Configure la aplicación para añadir mensajes interactivos a su bot:

Al final de esta sección, obtendrá las credenciales de la aplicación (ID de cliente, secreto de cliente y token de verificación). En la siguiente sección utilizará esta información para configurar la asociación de canal de bot en la consola de Amazon Lex.

1. Inicie sesión en la consola de la API de Slack en http://api.slack.com .
2. Cree una aplicación.

   Si ha creado la aplicación correctamente, Slack muestra la página Basic Information de la aplicación.
3. Configure las características de la aplicación de la siguiente manera:

   - En el menú de la izquierda, seleccione Interactividad y atajos.
     - Deslice el interruptor para activar los componentes interactivos.
     - En el cuadro Request URL, especifique cualquier URL válida. Por ejemplo, puede utilizar **`https://slack.com`**.

       > **Note**
       >
       > Por ahora, introduzca cualquier dirección URL válida para obtener el token de verificación que necesita en el siguiente paso. Deberá actualizar esta URL después de añadir la asociación de canal de bot en la consola de Amazon Lex.

     - Elija Guardar cambios.
4. En el menú izquierdo, en Settings, elija Basic Information. Registre las siguientes credenciales de la aplicación:

   - ID de cliente
   - Secreto del cliente
   - Token de verificación

Paso siguiente

Paso 4: integración de la aplicación Slack con el bot de Amazon Lex

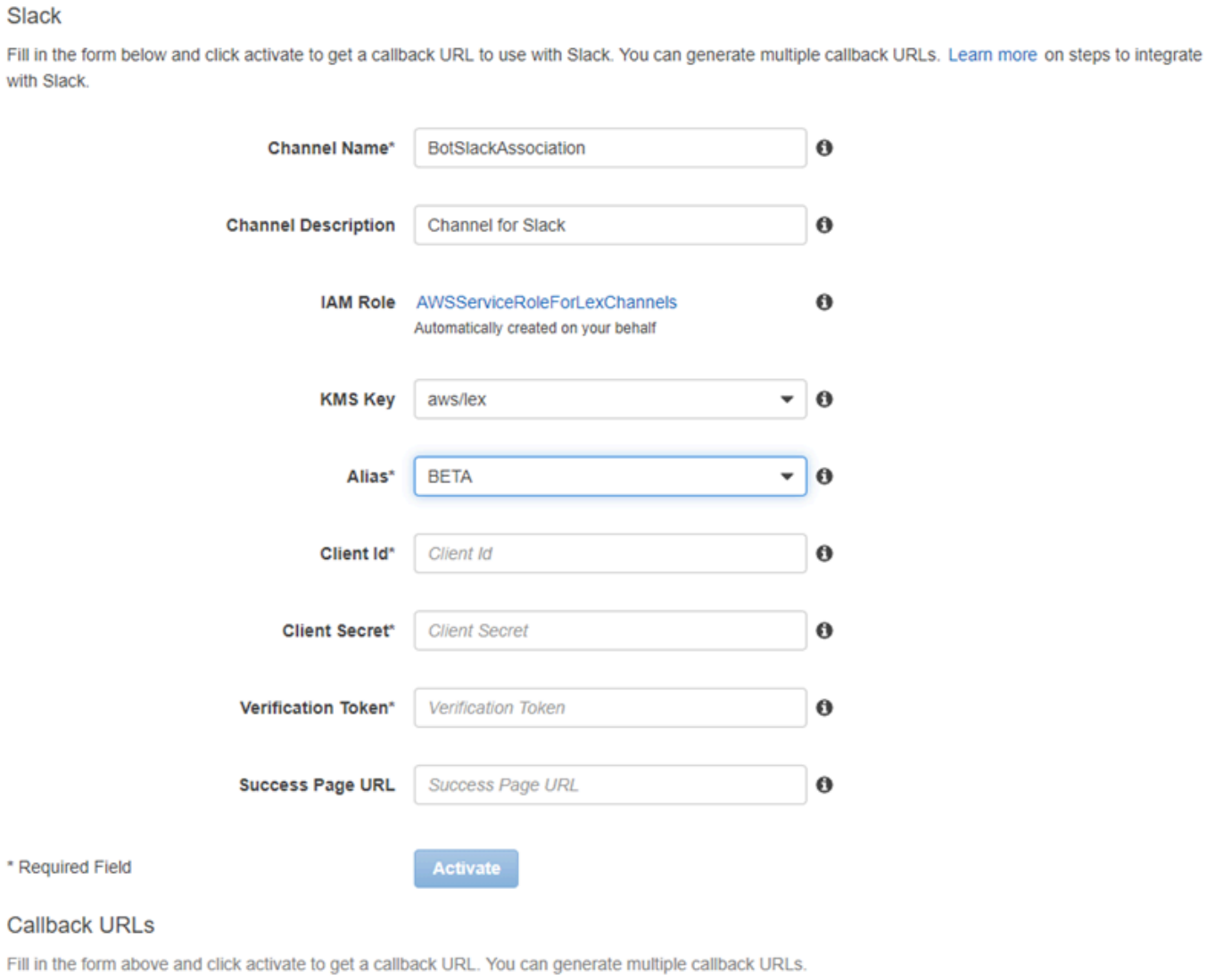
Paso 4: integración de la aplicación Slack con el bot de Amazon Lex

Ahora que ya tiene las credenciales de la aplicación Slack, puede integrar la aplicación con el bot de Amazon Lex. Para asociar la aplicación Slack con el bot, añada una asociación de canal de bot en Amazon Lex.

En la consola de Amazon Lex, active una asociación de canal de bot para asociar el bot con la aplicación Slack. Cuando la asociación del canal de bot se activa, Amazon Lex devuelve dos direcciones URL (URL de Postback y URL de OAuth). Registre estas direcciones URL, pues las necesitará más tarde.

Integración de la aplicación Slack con el bot de Amazon Lex

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon en https://console.aws.amazon.com/lex/.
2. Elija el bot de Amazon Lex que ha creado en el paso 1.
3. Elija la pestaña Channels.
4. En el menú de la izquierda, elija Slack.
5. En la página Slack, especifique lo siguiente:
   - Escriba un nombre. Por ejemplo, `BotSlackIntegration`.
   - Elija "aws/lex" en el menú desplegable KMS key.
   - En Alias, elija el alias del bot.
   - Rellene los campos Client Id, Client secret y Verification Token, cuyos datos registró en el paso anterior. Estas son las credenciales de la aplicación Slack.



6. Elija Activar.

   La consola crea la asociación del canal de bot y devuelve dos direcciones URL (URL de Postback y URL de OAuth). Regístrelas. En la siguiente sección, actualice la configuración de la aplicación Slack para utilizar estos puntos de conexión de la siguiente manera:

   - La dirección URL de Postback es el punto de conexión del bot de Amazon Lex que escucha los eventos de Slack. Utilice esta dirección URL:
     - Como dirección URL de solicitud en la función Event Subscriptions de la aplicación Slack.
     - Para sustituir el valor del marcador de posición de la dirección URL solicitada en la función Interactive Messages de la aplicación Slack.

- La dirección URL de OAuth es el punto de conexión del bot de Amazon Lex para un protocolo de enlace OAuth con Slack.

Paso siguiente

Paso 5: finalización de la integración con Slack

Paso 5: finalización de la integración con Slack

En esta sección, utilice la consola de la API de Slack para completar la integración de la aplicación Slack.

1. Inicie sesión en la consola de la API de Slack en http://api.slack.com. Elija la aplicación que ha creado en el Paso 3: creación de una aplicación Slack.
2. Actualice la función OAuth & Permissions de la siguiente manera:
   a. En el menú de la izquierda, elija OAuth & Permissions (OAuth y permisos).
   b. En la sección Redirigir URL, añada la URL de OAuth que Amazon Lex facilitó en el paso anterior. Elija Add a new Redirect URL y, a continuación, elija Save URLs.
   c. En la sección Ámbitos del token del bot, añada dos permisos con el botón Añadir un ámbito de OAuth. Filtre la lista con el siguiente texto:
      - **`chat:write`**
      - **`team:read`**
3. Actualice la característica Interactividad y atajos mediante la actualización del valor de Solicitar URL con la URL de Postback que Amazon Lex proporcionó en el paso anterior. Introduzca la dirección URL de postback que guardó en el paso 4 y, a continuación, elija Save Changes (Guardar cambios).
4. Suscríbase a la función Event Subscriptions de la siguiente manera:
   - Habilite los eventos mediante la opción On.
   - Establezca como valor de URL de solicitud la dirección URL de Postback que Amazon Lex ha facilitado en el paso anterior.
   - En la sección Subscribe to Bot Events, suscríbase al evento de bot `message.im` para permitir la mensajería directa entre el usuario final y el bot de Slack.
   - Guarde los cambios.

5. Habilite el envío de mensajes desde la pestaña de mensajes de la siguiente manera:

   - En el menú izquierdo, seleccione Inicio de la aplicación.
   - En la sección Mostrar pestañas, seleccione Permitir a los usuarios enviar comandos y mensajes desde la pestaña de mensajes.

Paso siguiente

Paso 6: comprobación de la integración

Paso 6: comprobación de la integración

Ahora utilice una ventana de navegador para probar la integración de Slack con el bot de Amazon Lex.

1. Elija Manage Distribution en Settings. Elija Add to Slack para instalar la aplicación. Autorice que el bot responda a mensajes.
2. Se le redirigirá a su equipo de Slack. En el menú de la izquierda, en la sección Direct Messages, elija su bot. Si no ve su bot, elija el icono más (+) junto a Direct Mensajes para buscarlo.
3. Entable un chat con la aplicación de Slack vinculada al bot de Amazon Lex. Ahora su bot responde a los mensajes.

   Si creó el bot en Ejercicio de introducción 1, puede utilizar los ejemplos de conversaciones proporcionados en dicho ejercicio. Para obtener más información, consulte Paso 4: adición de la función de Lambda como enlace de código (consola).

Integración de un bot de Amazon Lex con Twilio Programmable SMS

Este ejercicio proporciona instrucciones para integrar un bot de Amazon Lex con el servicio de mensajería simple (SMS) de Twilio. Debe realizar los pasos siguientes:

1. crear un bot de Amazon Lex,
2. integrar Twilio Programmable SMS con el bot de Amazon Lex,
3. participar en una interacción con el bot de Amazon Lex y probar la configuración con el servicio de SMS en el teléfono móvil,
4. probar la integración.

Temas

Paso 1: creación de un bot de Amazon Lex

Si aún no tiene un bot de Amazon Lex, cree uno e impleméntelo. En este tema, supongamos que está utilizando el bot que ha creado en el ejercicio de introducción 1. También puede utilizar cualquiera de los ejemplos de bot que se proporcionan en esta guía. Para el Ejercicio de introducción 1, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).

1. Cree un bot de Amazon Lex. Para obtener instrucciones, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).
2. Implemente el bot y cree un alias. Para obtener instrucciones, consulte Ejercicio 3: publicación de una versión y creación de un alias.

Paso 2: creación de una cuenta de Twilio SMS

Inscríbase para obtener una cuenta de Twilio y registre la siguiente información de la cuenta:

- ACCOUNT SID
- AUTH TOKEN

Para ver las instrucciones, consulte https://www.twilio.com/console.

Paso 3: integración del punto de conexión con el servicio de mensajería de Twilio con el bot de Amazon Lex

Integración de Twilio con el bot de Amazon Lex

1. Para asociar el bot de Amazon Lex con el punto de conexión de Twilio Programmable SMS, active la asociación del canal de bot en la consola de Amazon Lex. Cuando se active la

asociación del canal de bot, Amazon Lex devolverá una URL de devolución de llamada. Tome nota de esta URL de devolución de llamada porque la necesitará más adelante.

a. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.

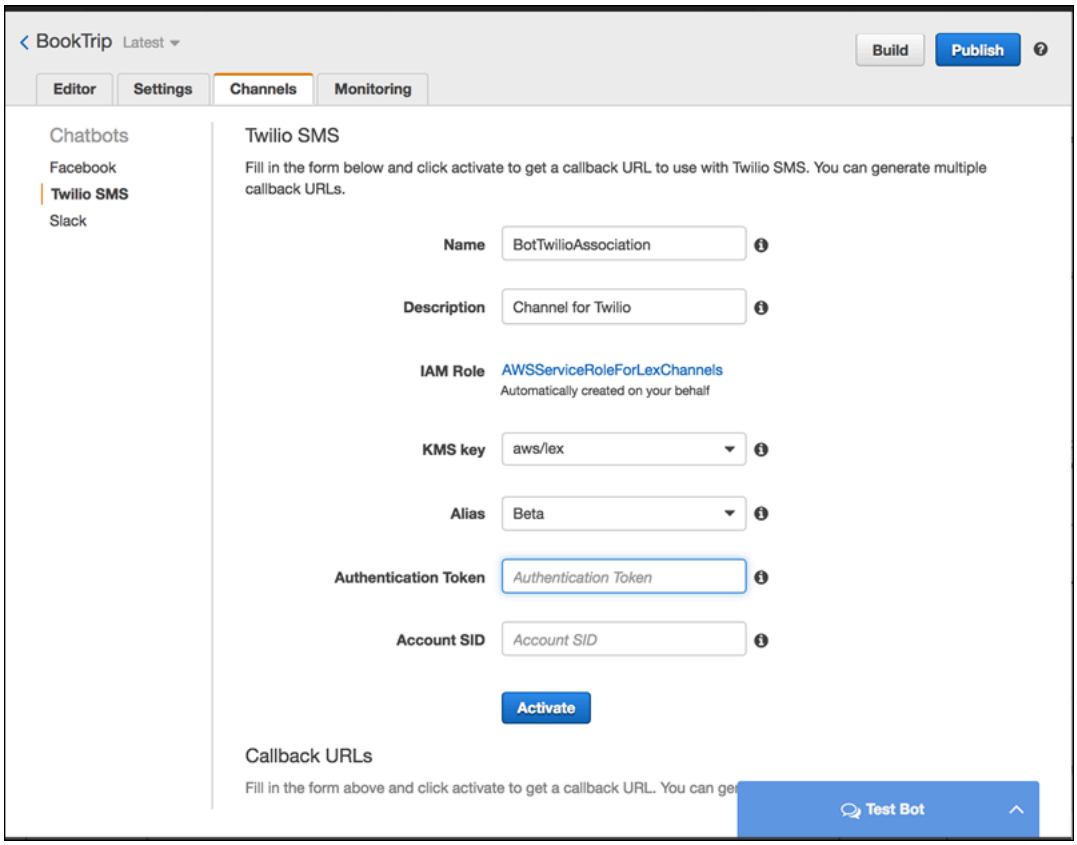
b. Elija el bot de Amazon Lex que ha creado en el paso 1.

c. Elija la pestaña Channels.

d. En la sección Chatbots, elija Twilio SMS.

e. En la página Twilio SMS, proporcione la siguiente información:

- Escriba un nombre. Por ejemplo, `BotTwilioAssociation`.
- Elija "aws/lex" en KMS key.
- En Alias, elija el alias del bot.
- En Authentication Token, escriba el AUTH TOKEN de su cuenta de Twilio.
- En Account SID, escriba el ACCOUNT SID de su cuenta de Twilio.



f. Elija Activar.

La consola crea la asociación de canal de bot y devuelve una URL de devolución de llamada. Registre esta URL.

2. En la consola de Twilio, conecte el punto de conexión de Twilio SMS al bot de Amazon Lex.

   a. Inicie sesión en la consola de Twilio en https://www.twilio.com/console.

   b. Si no tiene un punto de enlace de Twilio SMS, créelo.

   c. Actualice la configuración de entrada del servicio de mensajería. Para ello, especifique el valor de REQUEST URL en la URL de devolución de llamada que Amazon Lex ha proporcionado en el paso anterior.

Paso 4: comprobación de la integración

Utilice su teléfono móvil para probar la integración entre Twilio SMS y su bot.

Para probar la integración

1. Inicie sesión en la consola de Twilio en https://www.twilio.com/console y haga lo siguiente:

   a. Compruebe que tiene un número de Twilio asociado al servicio de mensajería en Manage Numbers.

      Puede enviar mensajes a este número y participar en una interacción de SMS con el bot de Amazon Lex desde su teléfono móvil.

   b. Compruebe que el teléfono móvil está habilitado como ID de intermediario verificado.

      Si no lo está, siga las instrucciones en la consola de Twilio para habilitar el teléfono móvil que quiere utilizar para las pruebas.

      Ahora puede utilizar su teléfono móvil para enviar mensajes al punto de conexión de Twilio SMS asignado al bot de Amazon Lex.

2. Use su teléfono móvil para enviar mensajes al número de Twilio.

   El bot de Amazon Lex responderá. Si creó el bot en Ejercicio de introducción 1, puede utilizar los ejemplos de conversaciones proporcionados en dicho ejercicio. Para obtener más información, consulte Paso 4: adición de la función de Lambda como enlace de código (consola).

Implementación de un bot de Amazon Lex en aplicaciones móviles

Con AWS Amplify, puede integrar bots de Amazon Lex en aplicaciones web y móviles. Para obtener más información, consulte Introducción a las interacciones en la documentación de AWS Amplify.

Importación y exportación de bots, intenciones y tipos de ranura de Amazon Lex

Puede importar o exportar un bot, una intención o un tipo de slot. Por ejemplo, si desea compartir un bot con un colega en otra cuenta de AWS, puede exportarla y, a continuación, enviarlo a ella. Si desea añadir varios enunciados a un bot, puede exportarlo, añadir los enunciados y, a continuación, volver a importarlos en su cuenta.

Puede exportar bots, intenciones y tipos de ranura en el formato de Amazon Lex (para compartirlos o modificarlos) o en un formato de habilidad de Alexa. Puede importar recursos únicamente en formato de Amazon Lex.

Los recursos deben exportarse en un formato compatible con el servicio al que exporta, ya sea Amazon Lex o Alexa Skills Kit. Si exporta un bot en formato de Amazon Lex, puede volver a importarlo en su cuenta o un usuario de Amazon Lex en otra cuenta puede importarlo en su cuenta. También puede exportar un bot en un formato compatible con una habilidad de Alexa. A continuación, puede importar el bot con el Alexa Skills Kit para hacer que su bot esté disponible con Alexa. Para obtener más información, consulte Exportación de una habilidad de Alexa.

Al exportar un bot, intención o tipo de slot, sus recursos se escriben en un archivo JSON. Para exportar un bot, una intención o un tipo de ranura, puede utilizar la consola de Amazon Lex o la operación GetExport. Importe un bot, una intención o un tipo de slot mediante el StartImport.

Temas

Importación y exportación en formato de Amazon Lex

Para exportar bots, intenciones y tipos de ranura desde Amazon Lex con el objeto de volver a importarlos en Amazon Lex, cree un archivo JSON en formato de Amazon Lex. Puede editar los recursos en este archivo y volver a importarlo en Amazon Lex. Por ejemplo, puede añadir enunciados a una intención y, a continuación, volver a importar la intención cambiada en su cuenta. También puede utilizar el formato JSON para compartir un recurso. Por ejemplo, puede exportar un

bot desde una región de AWS y, a continuación, importarlo en otra región. O bien, puede enviar el archivo JSON a un colega para compartir un bot.

Temas

Exportación en formato de Amazon Lex

Puede exportar los bots, intenciones y tipos de ranura de Amazon Lex con un formato que después puede importar en una cuenta de AWS. Puede exportar los siguientes recursos:

- Un bot, incluidos todas las intenciones y tipos de slot personalizados utilizados por el bot
- Una intención, incluidos todos los tipos de slot utilizados por la intención
- Un tipo de slot personalizado, incluidos todos los valores para el tipo de slot

Puede exportar solo una versión numerada de un recurso. No puede exportar la versión `$LATEST` de un recurso.

La exportación es un proceso asíncrono. Cuando se completa la exportación, se obtiene una URL prefirmada de Amazon S3. La dirección URL proporciona la ubicación de un archivo .zip que contiene el recurso exportado en formato JSON.

Puede utilizar la consola o la operación GetExport para exportar bots, intenciones y tipos de slot personalizados.

El proceso de exportación de un bot, una intención o un tipo de slot es el mismo. En los siguientes procedimientos, sustituya la intención o el tipo de slot del bot.

Exportación de un bot

Para exportar un bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon en https://console.aws.amazon.com/lex/.
2. Elija Bots y, a continuación, seleccione el bot que desea exportar.

3. En el menú Actions (Acciones), elija Export (Exportar).
4. En el cuadro de diálogo Export Bot (Exportar bot), elija la versión del bot que desea exportar. Para Platform (Plataforma), elija Amazon Lex.
5. Elija Export (Exportar).
6. Descargue y guarde el archivo .zip.

Amazon Lex exporta el bot a un archivo JSON incluido en el archivo .zip. Para actualizar el bot, modifique el texto del archivo JSON y, a continuación, vuelva a importarlo en Amazon Lex.

Paso siguiente

Importación en formato de Amazon Lex

Importación en formato de Amazon Lex

Una vez exportado un recurso a un archivo JSON en formato de Amazon Lex, puede importar el archivo JSON del recurso en una o varias cuentas de AWS. Por ejemplo, puede exportar un bot y, a continuación, importarlo en otra región de AWS. O puede enviar el bot a un colega para que pueda importarlo en su cuenta.

Al importar un bot, una intención o un tipo de slot, debe decidir si desea sobrescribir la versión `$LATEST` de un recurso, como una intención o un tipo de slot, durante la importación o si desea que la importación devuelva un error si desea conservar el recurso que se encuentra en su cuenta. Por ejemplo, si va a cargar una versión modificada de un recurso en su cuenta, podría elegir sobrescribir la versión `$LATEST`. Si está cargando un recurso que le ha enviado un colega, puede elegir que la importación devuelva un error si hay conflictos de recursos para que no se reemplacen sus propios recursos.

Al importar un recurso, se aplican los permisos asignados al usuario que realiza la solicitud de importación. El usuario debe tener permisos para todos los recursos de la cuenta a la que afecta a la importación. El usuario debe también tener permisos para ejecutar las operaciones GetBotPutBot, GetIntent PutIntent, GetSlotType, PutSlotType. Para obtener más información sobre los permisos, consulte Cómo funciona Amazon Lex con IAM.

Los errores de informes de importación que se producen durante el procesamiento. Algunos errores se notifican antes de que comience la importación, otros se notifican durante el proceso de importación. Por ejemplo, si la cuenta en la que se importa una intención no dispone de permisos

para llamar a una función de Lambda que la intención utiliza, la importación falla antes de que se realicen los cambios en los tipos de ranura o las intenciones. Si una importación devuelve un error durante el proceso de importación, se modifica la versión `$LATEST` de cualquier intención o tipo de slot importado antes del error del proceso. No se pueden revertir los cambios realizados en la versión `$LATEST`.

Al importar un recurso, todos los recursos dependientes se importan en la versión `$LATEST` del recurso y, a continuación, reciben una versión numerada. Por ejemplo, si un bot utiliza una intención, a la intención se le asigna una versión numerada. Si una intención utiliza un tipo de slot personalizado, al tipo de slot se le asigna una versión numerada.

Un recurso se importa solo una vez. Por ejemplo, si el bot contiene una intención `OrderPizza` y una intención `OrderDrink` que ambos se basan en el tipo de slot personalizado `Size`, el tipo de slot `Size` se importa una vez y se utiliza para ambas intenciones.

> **Note**
>
> Si ha exportado el bot con el parámetro `enableModelImprovements` establecido en `false`, debe abrir el archivo .zip que incluye la definición del bot y cambiar el parámetro `enableModelImprovements` a `true` en las siguientes regiones:
>
> - Asia Pacífico (Singapur) (ap-southeast-1)
> - Asia Pacífico (Tokio) (ap-northeast-1)
> - UE (Fráncfort) (eu-central-1)
> - UE (Londres) (eu-west-2)

El proceso de importación de un bot, una intención o un tipo de slot personalizado es el mismo. En los siguientes procedimientos, sustituya la intención o el tipo de slot, según corresponda.

Importación de un bot

Para importar un bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon en https://console.aws.amazon.com/lex/.
2. Elija Bots y, a continuación, seleccione el bot que desea importar. Para importar un nuevo bot, omita este paso.

3. Para Actions (Acciones), elija Import (Importar).
4. Para Import Bot (Importar bot), elija el archivo .zip que contiene el archivo JSON que contiene el bot que desea importar. Si desea ver conflictos de combinación antes de la misma, elija Notify me of merge conflicts (Notificarme conflictos de combinación). Si desactiva la comprobación de conflictos, se sobrescribe la versión $LATEST de todos los recursos utilizados por el bot.
5. Elija Importar. Si ha elegido que se le notifiquen los conflictos de combinación y hay conflictos, aparece un cuadro de diálogo que los enumera. Para sobrescribir la versión $LATEST de todos los recursos en conflicto, elija Overwrite y continue (Sobrescribir y continuar). Para detener la importación, seleccione Cancel (Cancelar).

Ahora puede probar el bot en su cuenta.

Formato JSON para importación y exportación

Los siguientes ejemplos muestran la estructura JSON para exportación e importación de tipos de ranura, intenciones y bots en formato de Amazon Lex.

Estructura de tipo de slot

A continuación, se muestra la estructura JSON para los tipos de slot personalizados. Utilice esta estructura al importar o exportar tipos de slot y cuando exporte intenciones que dependan 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"
  }
}
```

Estructura de intenciones

A continuación, se muestra la estructura JSON para intenciones. Utilice esta estructura al importar o exportar intenciones y bots que dependan de una intención.

```
{
  "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 Estructura de tipo de slot.
    ]
  }
}
```

Estructura de bots

A continuación, se muestra la estructura JSON para bots. Utilice esta estructura al importar o 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 Estructura de intenciones.
    ],
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Estructura de 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"
        }
      ]
    }
  }
}
```

Exportación de una habilidad de Alexa

Puede exportar su esquema bot en un formato compatible con una habilidad de Alexa. Después de exportar el bot a un archivo JSON, debe cargarlo en Alexa utilizando el constructor de habilidades.

Para exportar un bot y su esquema (modelo de interacción)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot que desee exportar.
3. Para Actions (Acciones), elija Export (Exportar).
4. Elija la versión del bot que desee exportar. Para el formato, elija Alexa Skills Kit, a continuación elija Export (Exportar).
5. Si aparece un cuadro de diálogo de descarga, elija la ubicación en la que quiere guardar el archivo y haga clic en Save (Guardar).

El archivo descargado es un archivo .zip que contiene un archivo que lleva el nombre del bot exportado. Contiene la información necesaria para importar el bot como una habilidad de Alexa.

> **Note**
>
> Amazon Lex y Alexa Skills Kit difieren de las siguientes formas:
>
> - Los atributos de sesión, que se identifican por corchetes ([]), no son compatibles con Alexa Skills Kit. Tiene que actualizar las preguntas que utilizan atributos de sesión.
> - Los signos de puntuación no son compatibles con Alexa Skills Kit. Tiene que actualizar los enunciados que utilizan signos de puntuación.

Para cargar el bot en una habilidad de Alexa

1. Inicie sesión en el portal para desarrolladores en https://developer.amazon.com/.
2. En la página Alexa Skills (Skills de Alexa), seleccione Create Skill (Crear skill).
3. En la página Create a new skill (Crear una nueva skill), especifique el nombre y el idioma predeterminado de la skill. Asegúrese de que la opción Custom (Personalizado) está seleccionada en el modelo de skill y haga clic en Create skill (Crear skill).
4. No olvide activar la casilla Start from scratch (Comenzar desde el principio). Después, haga clic en Choose (Elegir).
5. En el menú de la izquierda, elija JSON Editor (Editor de JSON). Arrastre el archivo JSON que ha exportado de Amazon Lex hasta el editor de JSON.
6. Seleccione Save Model (Guardar modelo) para guardar el modelo de interacción.

Después de cargar el esquema en una habilidad de Alexa, realice los cambios necesarios para ejecutar la habilidad con Alexa. Para obtener más información sobre cómo crear una habilidad de Alexa, consulte Use Skill Builder (Beta) en Alexa Skills Kit.

Ejemplos adicionales: creación de bots de Amazon Lex

En las siguientes secciones se ofrecen ejercicios de Amazon Lex adicionales con instrucciones paso a paso.

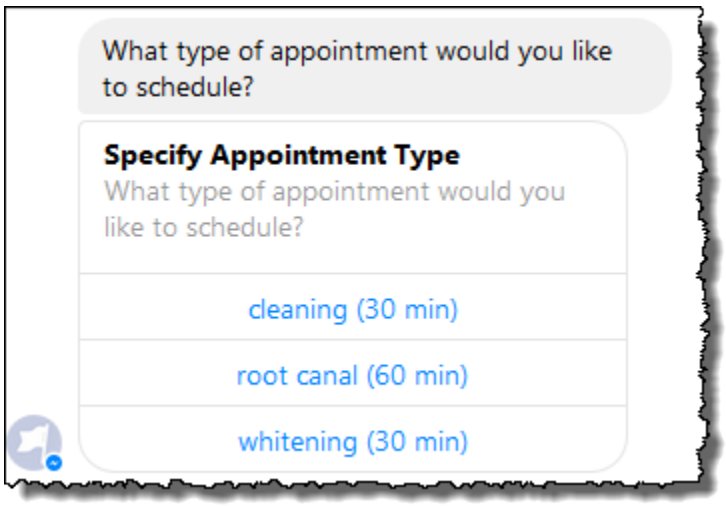
Temas

Programación de citas

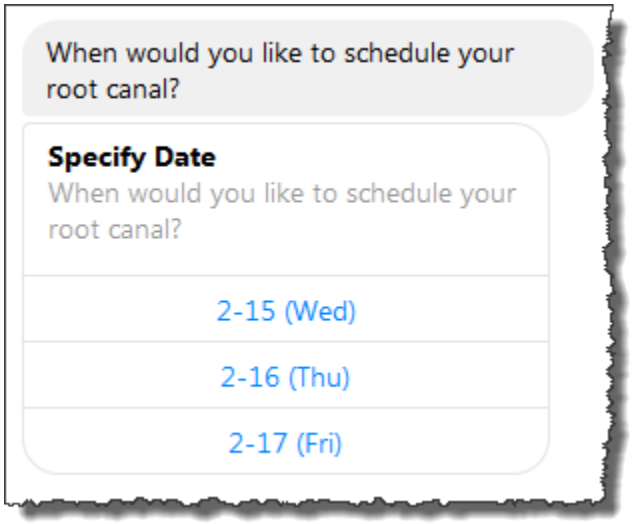
El bot de ejemplo en este ejercicio programa citas para un dentista. El ejemplo también muestra el uso de tarjetas de respuesta para obtener las entradas del usuario con botones. En concreto, el ejemplo ilustra cómo se generan tarjetas de respuesta de forma dinámica en el tiempo de ejecución.

Puede configurar tarjetas de respuesta en tiempo de creación (también denominadas tarjetas de respuesta estáticas) o generarlas de forma dinámica en una función AWS Lambda. En este ejemplo, el bot utiliza las siguientes tarjetas de respuesta:

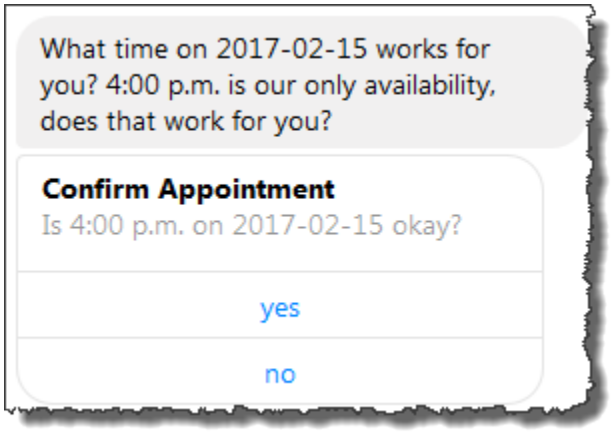
- Una tarjeta de respuesta que enumera los botones para el tipo de cita. Para ver un ejemplo, consulte la siguiente imagen:



- Una tarjeta de respuesta que enumera los botones para la fecha de la cita. Para ver un ejemplo, consulte la siguiente imagen:



- Una tarjeta de respuesta que enumera botones para confirmar una propuesta de hora para la cita. Para ver un ejemplo, consulte la siguiente imagen:



Las fechas y horas disponibles para la cita varían, por lo que debe generan tarjetas de respuesta en el tiempo de ejecución. Puede utilizar una función AWS Lambda para generar estas tarjetas de respuesta de forma dinámica. La función de Lambda devuelve tarjetas de respuesta en su respuesta a Amazon Lex. Amazon Lex incluye la tarjeta de respuesta en su respuesta al cliente.

Si un cliente (por ejemplo, Facebook Messenger) admite tarjetas de respuesta, el usuario puede seleccionarlas en la lista de botones o escribir la respuesta. Si no las admite, el usuario simplemente escribe la respuesta.

Además del botón que se muestra en el ejemplo anterior, también puede incluir imágenes, archivos adjuntos y otra información útil en las tarjetas de respuesta. Para obtener más información sobre las tarjetas de respuesta, consulte Tarjetas de respuesta.

En este ejercicio, hará lo siguiente:

- Crear y probar un bot (con el proyecto ScheduleAppointment). Para este ejercicio deberá utilizar un proyecto de bot para configurar y probar rápidamente el bot. Para ver una lista de los proyectos disponibles, consulte Esquemas de Amazon Lex y AWS Lambda. Este bot está preconfigurado con una intención (`MakeAppointment`).

- Crear y probar una función de Lambda (con el esquema lex-make-appointment-python proporcionado por Lambda). A continuación, deberá configurar la intención `MakeAppointment` para utilizar esta función de Lambda como enlace de código para realizar las tareas de inicialización, validación y cumplimiento.

  > **Note**
  >
  > La función de Lambda del ejemplo muestra una conversación dinámica basada en la disponibilidad simulada de una cita en un dentista. En una aplicación real, se suele utilizar un calendario para programar una cita.

- Actualizar la configuración de la intención `MakeAppointment` para utilizar la función de Lambda como enlace de código. A continuación, probará la experiencia integral.
- Publicar el bot de programación de citas en Facebook Messenger para ver las tarjetas de respuesta en acción (actualmente, el cliente en la consola de Amazon Lex no admite tarjetas de respuesta).

En las siguientes secciones se incluye información resumida sobre los proyectos que va a utilizar en este ejercicio.

Temas

Descripción general del proyecto de bot (ScheduleAppointment)

El proyecto ScheduleAppointment que utiliza para crear un bot para este ejercicio está preconfigurado con lo siguiente:

- Tipos de slot: un tipo de slot personalizado denominado `AppointmentTypeValue` con los valores de enumeración `root canal`, `cleaning` y `whitening`.
- Intención: una intención (`MakeAppointment`), que está preconfigurada de la siguiente manera:
  - Slots: la intención está configurada con los siguientes slots:
    - Slot `AppointmentType` del tipo personalizado `AppointmentTypes`.
    - Slot `Date` del tipo integrado `AMAZON.DATE`.
    - Slot `Time` del tipo integrado `AMAZON.TIME`.
  - Enunciados: la intención está preconfigurada con los siguientes enunciados:
    - "Me gustaría concertar una cita"
    - "Concertar una cita"
    - "Concertar una {AppointmentType}"

    Si el usuario utiliza uno de estos enunciados, Amazon Lex determina que `MakeAppointment` es la intención y, a continuación, utiliza las preguntas para obtener datos de ranura.
  - Preguntas: la intención está preconfigurada con las siguientes preguntas:
    - Pregunta para el slot `AppointmentType`: "¿Qué tipo de cita desea concertar?"

- Pregunta para el slot `Date`: "¿Cuándo debo concertar su {AppointmentType}?"
- Pregunta para el slot `Time`: “¿A qué hora desea concertar su {AppointmentType}?” Protección de los datos

  "¿A qué hora el {Date}?"
- Pregunta de confirmación: "{Time} está disponible. ¿Desea que continúe y programe la cita?"
- Mensaje de cancelación: "De acuerdo, no programaré ninguna cita".

Descripción general del esquema de la función de Lambda (lex-make-appointment-python)

El esquema de la función de Lambda (lex-make-appointment-python) es un enlace de código para bots que crea con el esquema de bot ScheduleAppointment.

El código del esquema de la función de Lambda puede realizar tanto la tarea de inicialización o validación como la de cumplimiento.

- El código de la función de Lambda muestra una conversación dinámica que se basa en un ejemplo de disponibilidad de una cita con el dentista (en aplicaciones reales, se suele utilizar un calendario). Para el día o la fecha que el usuario especifica, el código está configurado como sigue:
  - Si no hay citas disponibles, la función de Lambda devuelve una respuesta que indica a Amazon Lex que debe preguntar al usuario otro día o fecha (se establece el tipo `dialogAction` en `ElicitSlot`). Para obtener más información, consulte Formato de respuesta.
  - Si solo hay una cita disponible en el día o fecha especificada, la función de Lambda sugiere la hora disponible en la respuesta e indica a Amazon Lex que obtenga la confirmación del usuario fijando `dialogAction` en la respuesta a `ConfirmIntent`. Esto muestra cómo puede mejorar la experiencia del usuario al sugerir de forma proactiva la hora disponible para una cita.
  - Si hay varias citas disponibles, la función de Lambda devuelve una lista de horas disponibles en la respuesta a Amazon Lex. Amazon Lex devuelve una respuesta al cliente con el mensaje de la función de Lambda.
- Como enlace de código de cumplimiento, la función de Lambda devuelve un mensaje de resumen que indica que hay una cita programada (es decir, se ha cumplido la intención).

> **Note**
>
> En este ejemplo le mostramos cómo utilizar tarjetas de respuesta. La función de Lambda crea y devuelve una tarjeta de respuesta a Amazon Lex. La tarjeta de respuesta enumera los días y las horas disponibles como botones entre los que elegir. Al probar el bot con el cliente proporcionado por la consola de Amazon Lex, no es posible ver la tarjeta de respuesta. Para verla, debe integrar el bot en una plataforma de mensajería, como Facebook Messenger. Para obtener instrucciones, consulte Integración de un bot de Amazon Lex con Facebook Messenger. Para obtener más información sobre las tarjetas de respuesta, consulte Administración de mensajes .

Cuando Amazon Lex invoca la función de Lambda, le pasa los datos del evento como entrada. Uno de los campos de evento es `invocationSource`, que la función de Lambda utiliza para elegir entre la actividad de validación de la entrada y la de cumplimiento. Para obtener más información, consulte Formato del evento de entrada.

Paso siguiente

Paso 1: creación de un bot de Amazon Lex

Paso 1: creación de un bot de Amazon Lex

En esta sección va a crear un bot de Amazon Lex con el esquema ScheduleAppointment, que se proporciona en la consola de Amazon Lex.

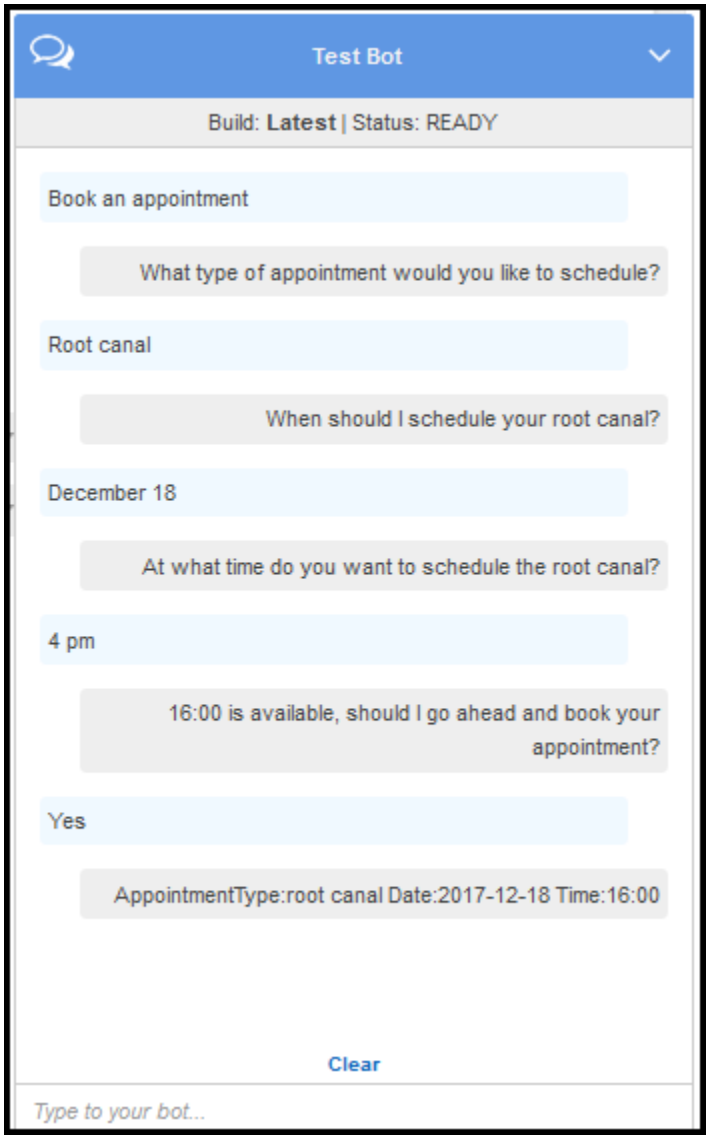
1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la página Bots, elija Create.
3. En la página Create your Lex bot, haga lo siguiente:
   - Elija el proyecto ScheduleAppointment.
   - Deje el nombre de bot predeterminado (ScheduleAppointment).
4. Seleccione Create (Crear).

   Este paso guarda y crea el bot. La consola envía las siguientes solicitudes a Amazon Lex durante el proceso de creación:

- Crear una nueva versión de los tipos de slot (a partir de la versión $LATEST). Para obtener más información sobre los tipos de slot en este proyecto de bot, consulte Descripción general del proyecto de bot (ScheduleAppointment).
- Crear una versión de la intención `MakeAppointment` (a partir de la versión $LATEST). En algunos casos, la consola envía una solicitud para la operación de la API `update` antes de crear una nueva versión.
- Actualizar la última versión $LATEST del bot.

  En ese momento, Amazon Lex crea un modelo de machine learning para el bot. Al probar el bot en la consola, esta utiliza la API en tiempo de ejecución para enviar las entradas del usuario de nuevo a Amazon Lex. A continuación, Amazon Lex utiliza el modelo de machine learning para interpretar las entradas del usuario.

5. La consola muestra el bot ScheduleAppointment. En la pestaña Editor, revise los detalles de la intención preconfigurada (`MakeAppointment`).
6. Pruebe el bot en la ventana de pruebas. Utilice la siguiente captura de pantalla para entablar una conversación de prueba con su bot:



Tenga en cuenta lo siguiente:

- El bot deduce la intención (`MakeAppointment`) a partir de la entrada inicial del usuario ("Concertar una cita").
- A continuación, el bot utiliza las preguntas configuradas para obtener datos de slot del usuario.
- El proyecto de bot tiene la intención `MakeAppointment` configurada con la siguiente pregunta de confirmación:

```
{Time} is available, should I go ahead and book your appointment?
```

Después de que el usuario proporcione todos los datos de ranura, Amazon Lex devuelve una respuesta al cliente con una pregunta de confirmación en forma de mensaje. El cliente muestra el mensaje al usuario:

```
16:00 is available, should I go ahead and book your appointment?
```

Observe que el bot acepta cualquier valor de fecha y hora para la cita, pues no dispone de ningún código para inicializar o validar los datos del usuario. En la siguiente sección, añadirá una función de Lambda para que lo haga.

Paso siguiente

Paso 2: creación de una función de Lambda

Paso 2: creación de una función de Lambda

En esta sección va a crear una función de Lambda con un esquema (lex-make-appointment-python) que se proporciona en la consola de Lambda. También invocará la función de Lambda para probarla con ejemplos de datos de evento de Amazon Lex de muestra proporcionados por la consola.

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Seleccione Create a Lambda function (Crear una función Lambda).
3. En Select blueprint, escriba **`lex`** para buscar el proyecto y luego elija el proyecto lex-make-appointment-python.
4. Configure la función de Lambda de la siguiente forma.
   - Escriba el nombre de la función de Lambda (`MakeAppointmentCodeHook`).
   - Para el rol, elija Create a new role from template(s) y luego escriba el nombre del rol.
   - Deje los demás valores predeterminados.
5. Elija Create Function (Crear función).
6. Si utiliza una configuración regional que no sea Inglés (EE. UU.) (en-US), actualice los nombres de las intenciones tal como se indica en Actualización de un esquema para una configuración regional específica.
7. Pruebe la función de Lambda.

a. Elija Actions y, a continuación, elija Configure test event.

b. En la lista Sample event template, elija Lex-Make Appointment (preview). Este ejemplo de evento sigue el modelo de solicitud/respuesta de Amazon Lex. Los valores se establecen para satisfacer una solicitud del bot de Amazon Lex. Para obtener información acerca del modelo de solicitud/respuesta de Amazon Lex, consulte Uso de funciones de Lambda.

c. Elija Save and test (Guardar y probar).

d. Compruebe que la función de Lambda se ha ejecutado correctamente. En este caso, la respuesta sigue el modelo de respuesta de Amazon Lex.

Paso siguiente

Paso 3: actualización de la intención - configuración de un enlace de código

Paso 3: actualización de la intención - configuración de un enlace de código

En esta sección, va a actualizar la configuración de la intención `MakeAppointment` para utilizar la función de Lambda como enlace de código para las actividades de validación y cumplimiento.

1. En la consola de Amazon Lex, seleccione el bot ScheduleAppointment. La consola muestra la intención MakeAppointment. Modifique la configuración de la intención de la siguiente manera.

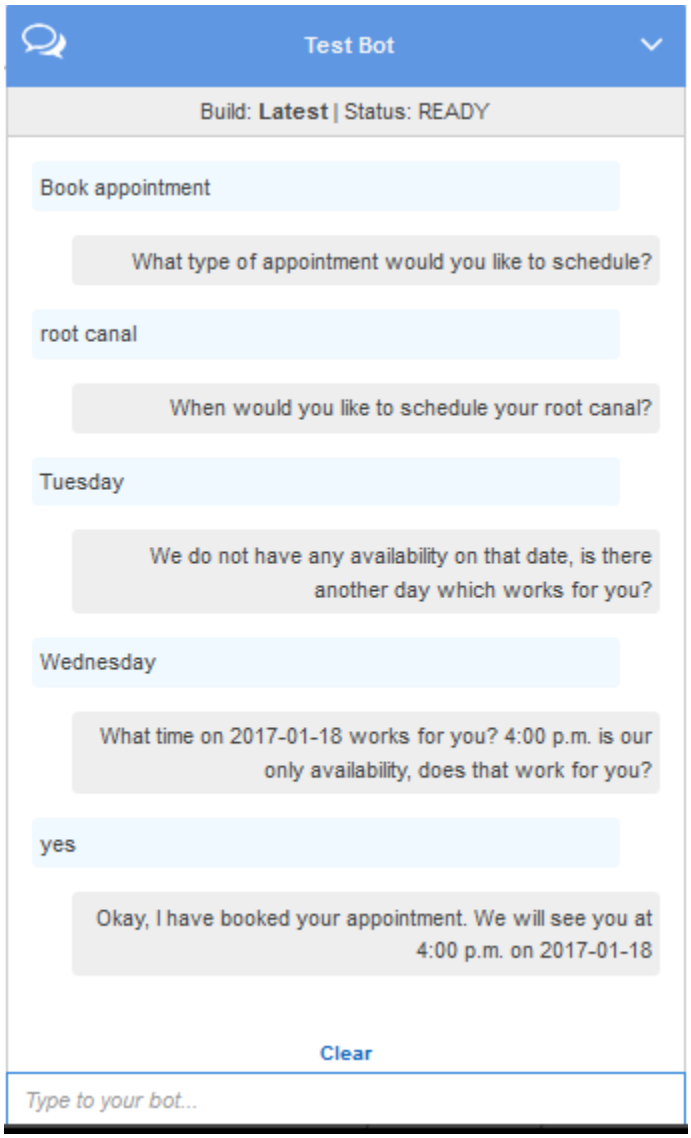
> **Note**
>
> Solo puede actualizar las versiones $LATEST de cualquiera de los recursos de Amazon Lex, incluidas las intenciones. Asegúrese de que la versión de la intención esté configurada en $LATEST. Aún no ha publicado ninguna versión de su bot, por lo que todavía debería ser la versión $LATEST de la consola.

a. En la sección Opciones, elija Enlace de código de inicialización y validación y, a continuación, seleccione la función de Lambda en la lista.

b. En la sección Cumplimiento, elija Función de AWS Lambda y, a continuación, seleccione la función de Lambda en la lista.

c. Elija Goodbye message y escriba un mensaje.

2. Elija Save y, a continuación, elija Build.
3. Pruebe el bot, tal como se indica en la siguiente imagen:



Paso siguiente

[Paso 4: implementación del bot en la plataforma Facebook Messenger]

Paso 4: implementación del bot en la plataforma Facebook Messenger

En la sección anterior ha probado el bot ScheduleAppointment con el cliente en la consola de Amazon Lex. En la actualidad, la consola de Amazon Lex no admite tarjetas de respuesta. Para probar las tarjetas de respuesta generadas dinámicamente que admite el bot, implemente el bot en la plataforma Facebook Messenger y pruébela.

Para obtener instrucciones, consulte Integración de un bot de Amazon Lex con Facebook Messenger.

Paso siguiente

Detalles del flujo de información

Detalles del flujo de información

El proyecto de bot `ScheduleAppointment` muestra principalmente el uso de tarjetas de respuesta generadas dinámicamente. En este ejercicio, la función de Lambda incluye tarjetas de respuesta en su respuesta a Amazon Lex. Amazon Lex incluye las tarjetas de respuesta en su respuesta al cliente. En esta sección se explican los dos flujos siguientes:

- El flujo de datos entre el cliente y Amazon Lex.

  En la sección se supone que el cliente envía solicitudes a Amazon Lex mediante la API en tiempo de ejecución `PostText` y que muestra los detalles de la solicitud y la respuesta. Para obtener más información acerca de la API en tiempo de ejecución `PostText`, consulte PostText.

  > **Note**
  >
  > Para ver un ejemplo de flujo de la información entre el cliente y Amazon Lex en el que el usuario usa la API `PostContent`, consulte Paso 2a (opcional): revisión de los detalles del flujo de información escrita (consola) .

- El flujo de datos entre Amazon Lex y la función de Lambda. Para obtener más información, consulte Formato del evento de entrada y de la respuesta de la función de Lambda.

> **Note**
>
> El ejemplo asume que está utilizando el cliente Facebook Messenger, que no transfiere los atributos de la sesión en la solicitud a Amazon Lex. En consecuencia, las solicitudes de ejemplo que aparecen en esta sección mostrarán los `sessionAttributes` vacíos. Si

prueba el bot con el cliente proporcionado en la consola de Amazon Lex, el cliente incluirá los atributos de la sesión.

En esta sección se describe lo que ocurre después de cada entrada del usuario.

1. Usuario: escribe **`Book an appointment`**.

   a. El cliente (la consola) envía la siguiente solicitud PostContent a Amazon Lex:

```
POST /bot/ScheduleAppointment/alias/$LATEST/
user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"book appointment",
   "sessionAttributes":{}
}
```

   Tanto la URI de la solicitud como el cuerpo proporcionan información a Amazon Lex:

   - URI de la solicitud: proporciona el nombre del bot (ScheduleAppointment), el alias del bot ($LATEST) y el ID del nombre de usuario. El `text` de cola indica que se trata de una solicitud de API `PostText` (no `PostContent`).
   - Cuerpo de la solicitud: incluye la entrada del usuario (`inputText`). No hay `sessionAttributes`.

   b. Amazon Lex deduce la intención (`MakeAppointment`) a partir de `inputText`. El servicio invoca la función de Lambda, que está configurada como enlace de código, para que lleve a cabo la inicialización y la validación al transferir el siguiente evento. Para obtener más información, consulte Formato del evento 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": {}
}
```

Además de la información que envía el cliente, Amazon Lex también incluye los siguientes datos:

- `currentIntent`: proporciona información sobre la intención actual.
- `invocationSource`: indica la finalidad de invocar la función de Lambda. En este caso, la finalidad consiste en inicializar y validar los datos del usuario. Amazon Lex sabe que el usuario no ha proporcionado todos los datos de ranura para cumplir la intención.
- `messageVersion`: actualmente Amazon Lex solo es compatible con la versión 1.0.

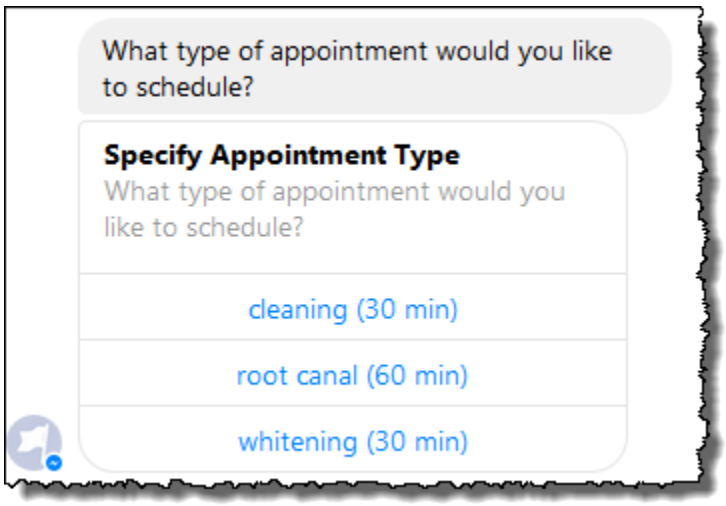
c. En este momento, todos los valores de slot son nulos (no hay nada que validar). La función de Lambda devuelve la siguiente respuesta a Amazon Lex para que el servicio obtenga información para la ranura `AppointmentType`. Para obtener más información sobre el formato de la respuesta, consulte Formato de respuesta.

```
{
    "dialogAction": {
        "slotToElicit": "AppointmentType",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "cleaning (30 min)",
                            "value": "cleaning"
                        },
                        {
                            "text": "root canal (60 min)",
```

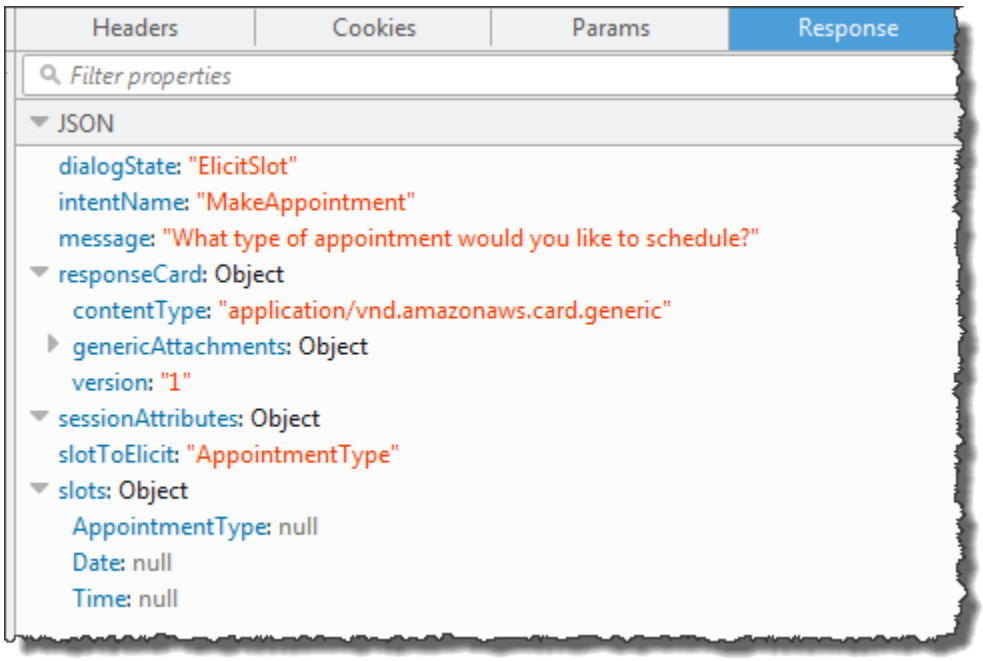
```
                            "value": "root canal"
                        },
                        {
                            "text": "whitening (30 min)",
                            "value": "whitening"
                        }
                    ],
                    "subTitle": "What type of appointment would you like to
 schedule?",
                    "title": "Specify Appointment Type"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

La respuesta incluye los campos `dialogAction` y `sessionAttributes`. Entre otras cosas, el campo `dialogAction` devuelve los siguientes campos:

- `type`: al configurar este campo con `ElicitSlot`, la función de Lambda indica a Amazon Lex que obtenga el valor de la ranura especificado en el campo `slotToElicit`. La función de Lambda también proporciona un valor `message` que transmitir al usuario.
- `responseCard`: identifica una lista de valores posibles para la ranura `AppointmentType`. Un cliente que admite tarjetas de respuesta (por ejemplo, Facebook Messenger) muestra una tarjeta de respuesta para permitir al usuario que elija un tipo de cita, tal como se muestra en la siguiente imagen:



d. Tal como indica `dialogAction.type` en la respuesta de la función de Lambda, Amazon Lex envía la siguiente respuesta al cliente:



El cliente lee la respuesta y, a continuación, muestra el mensaje: “¿Qué tipo de cita desea concertar?” y la tarjeta de respuesta (si el cliente admite tarjetas de respuesta).

2. Usuario: dependiendo del cliente, el usuario tiene dos opciones:

   - Si se muestra la tarjeta de respuesta, elija root canal (60 min) o escriba **`root canal`**.
   - Si el cliente no admite tarjetas de respuesta, escriba **`root canal`**.

a. El cliente envía la siguiente solicitud `PostText` a Amazon Lex (se han añadido saltos de línea para facilitar la lectura):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "root canal",
    "sessionAttributes": {}
}
```

b. Amazon Lex invoca la función de Lambda para que valide los datos del usuario mediante el envío del siguiente evento como parámetro:

```
{
    "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": {}
}
```

En los datos de evento, tenga en cuenta lo siguiente:

- `invocationSource` sigue siendo `DialogCodeHook`. En este paso, solo se validan los datos del usuario.
- Amazon Lex establece el campo `AppointmentType` de la ranura `currentIntent.slots` en `root canal`.
- Amazon Lex simplemente transfiere el campo `sessionAttributes` entre el cliente y la función de Lambda.

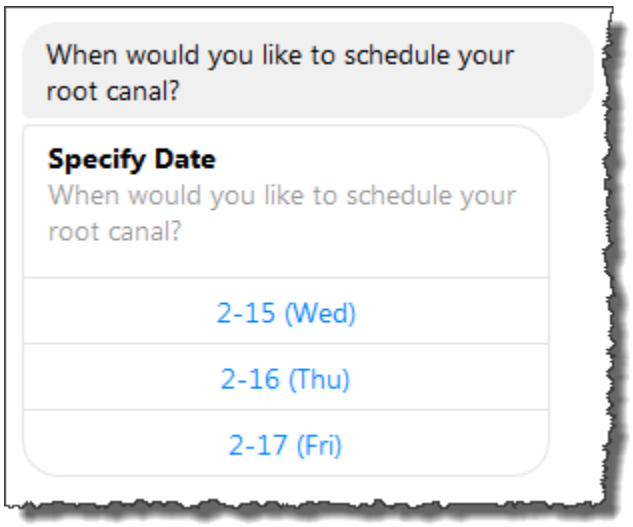
c. La función de Lambda valida la entrada del usuario y devuelve la siguiente respuesta a Amazon Lex para que el servicio obtenga un valor para la fecha de la cita.

```
{
    "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": {}
}
```

De nuevo, la respuesta incluye los campos `dialogAction` y `sessionAttributes`. Entre otras cosas, el campo `dialogAction` devuelve los siguientes campos:

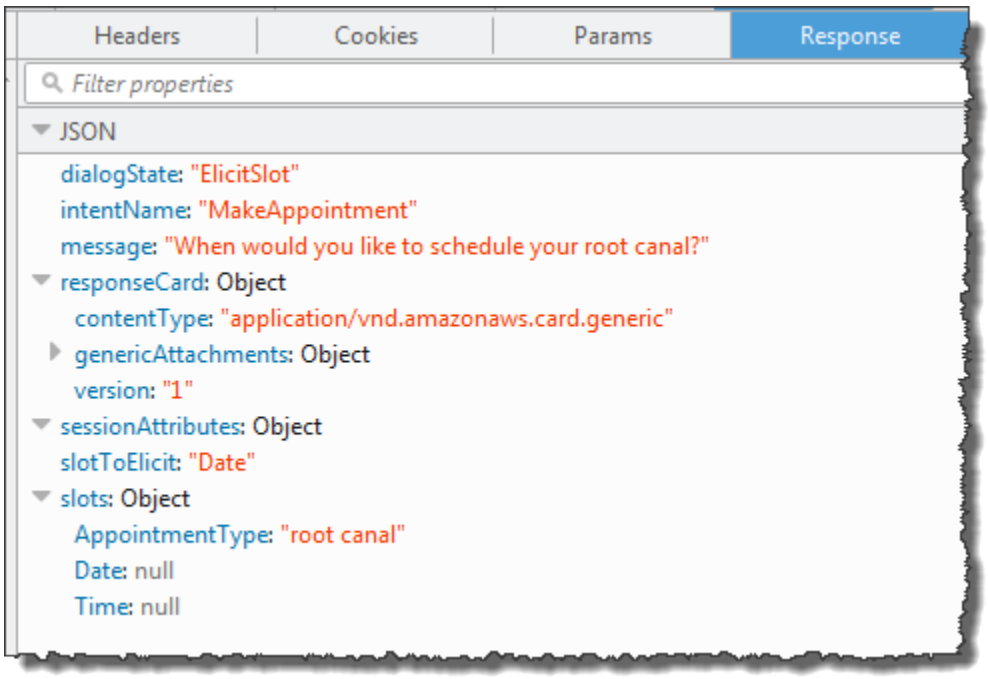
- `type`: al configurar este campo con `ElicitSlot`, la función de Lambda indica a Amazon Lex que obtenga el valor de la ranura especificado en el campo `slotToElicit`. La función de Lambda también proporciona un valor `message` que transmitir al usuario.
- `responseCard`: identifica una lista de valores posibles para la ranura `Date`. Un cliente que admite tarjetas de respuesta (por ejemplo, Facebook Messenger) muestra una tarjeta de respuesta para permitir al usuario que elija un tipo de cita, tal como se muestra en la siguiente imagen:



Aunque la función de Lambda devuelva cinco fechas, el cliente (Facebook Messenger) tiene un límite de tres botones por tarjeta de respuesta. Por lo tanto, en la captura de pantalla solo verá los primeros tres valores.

Estas fechas están codificadas de forma rígida en la función de Lambda. En una aplicación de producción, puede utilizar un calendario para ver las fechas disponibles en tiempo real. Dado que las fechas son dinámicas, debe generar la tarjeta de respuesta dinámicamente en la función de Lambda.

d. Amazon Lex detecta el valor de `dialogAction.type` y devuelve la siguiente respuesta al cliente que incluye información de la respuesta de la función de Lambda.



El cliente muestra el mensaje: ¿Cuándo desea concertar su endodoncia? y la tarjeta de respuesta (si el cliente admite tarjetas de respuesta).

3. Usuario: escribe **`Thursday`**.

   a. El cliente envía la siguiente solicitud `PostText` a Amazon Lex (se han añadido saltos de línea para facilitar la lectura):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   b. Amazon Lex invoca la función de Lambda para que valide los datos del usuario mediante el envío del siguiente evento como parámetro:

```
{
    "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": {}
}
```

En los datos de evento, tenga en cuenta lo siguiente:

- `invocationSource` sigue siendo `DialogCodeHook`. En este paso, solo se validan los datos del usuario.
- Amazon Lex establece el campo `Date` de la ranura `currentIntent.slots` en `2017-02-16`.
- Amazon Lex simplemente transfiere `sessionAttributes` entre el cliente y la función de Lambda.

c. La función de Lambda valida la entrada del usuario. Esta vez, la función de Lambda determina que no hay citas disponibles en la fecha especificada. Por ello, devuelve la siguiente respuesta a Amazon Lex para que el servicio obtenga un nuevo valor para la fecha de la cita.

```
{
    "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\": []}"
   }
}
```

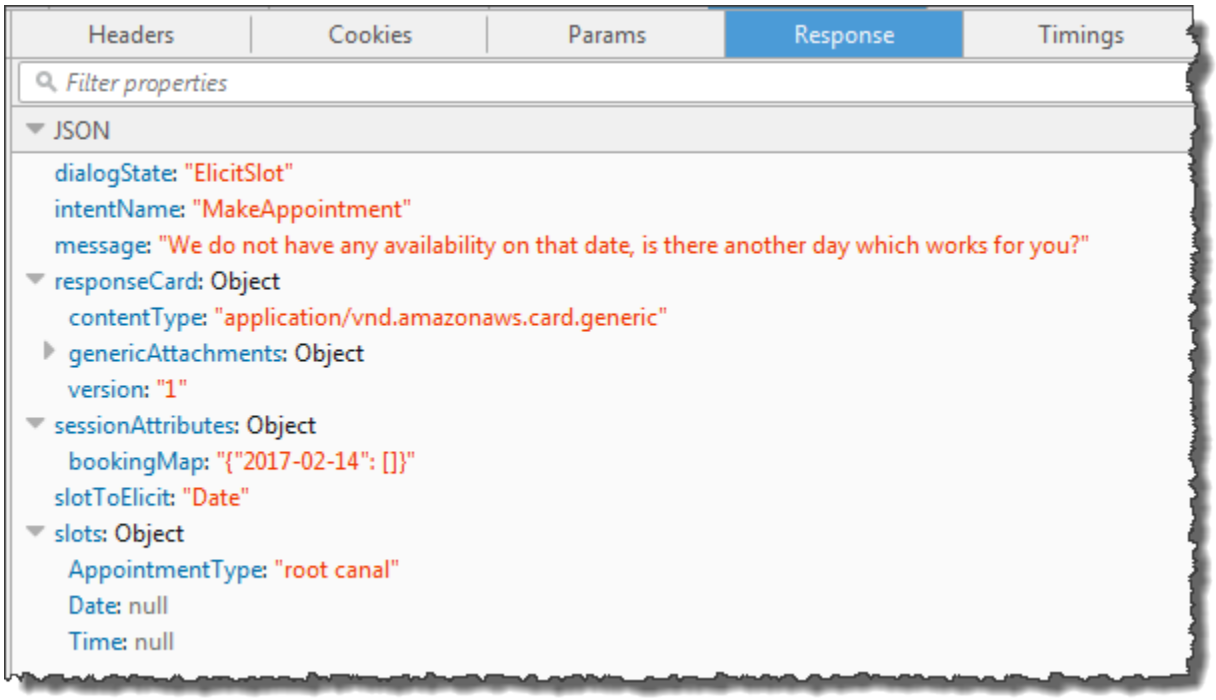
De nuevo, la respuesta incluye los campos `dialogAction` y `sessionAttributes`. Entre otras cosas, `dialogAction` devuelve los siguientes campos:

- `dialogAction` field:

- `type`: la función de Lambda establece este valor en `ElicitSlot` y restablece el campo `slotToElicit` con `Date`. La función de Lambda también proporciona un valor de `message` adecuado para el usuario.
- `responseCard`: devuelve una lista de valores para el slot `Date`.

- :`sessionAttributes`: esta vez, la función de Lambda incluye el atributo de la sesión `bookingMap`. Su valor es la fecha solicitada de la cita y las citas disponibles (un objeto vacío significa que no hay citas disponibles).

d. Amazon Lex detecta el valor de `dialogAction.type` y devuelve la siguiente respuesta al cliente que incluye información de la respuesta de la función de Lambda.



El cliente muestra el mensaje: We do not have any availability on that date, is there another day which works for you? y la tarjeta de respuesta (si el cliente admite tarjetas de respuesta).

4. Usuario: dependiendo del cliente, el usuario tiene dos opciones:

- Si se muestra la tarjeta de respuesta, elija 2-15 (Wed) o escriba **`Wednesday`**.
- Si el cliente no admite tarjetas de respuesta, escriba **`Wednesday`**.

a. El cliente envía la siguiente solicitud `PostText` a Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

> **Note**
>
> El cliente Facebook Messenger no establece atributos de la sesión. Si desea conservar los estados de sesión entre las solicitudes, debe hacerlo en la función de Lambda. En una aplicación real, es posible que tenga que almacenar estos atributos de la sesión en una base de datos back-end.

b. Amazon Lex invoca la función de Lambda para que valide los datos del usuario mediante el envío del siguiente evento como parámetro:

```
{
    "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 ha actualizado `currentIntent.slots` al configurar la ranura `Date` con `2017-02-15`.

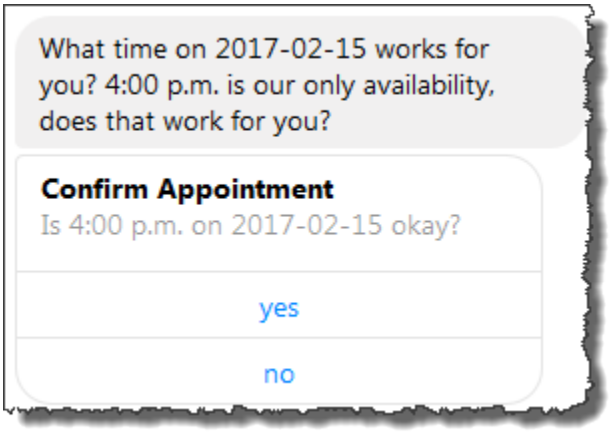
c. La función de Lambda valida la entrada del usuario y devuelve la siguiente respuesta a Amazon Lex para que obtenga un valor para la hora de la cita.

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

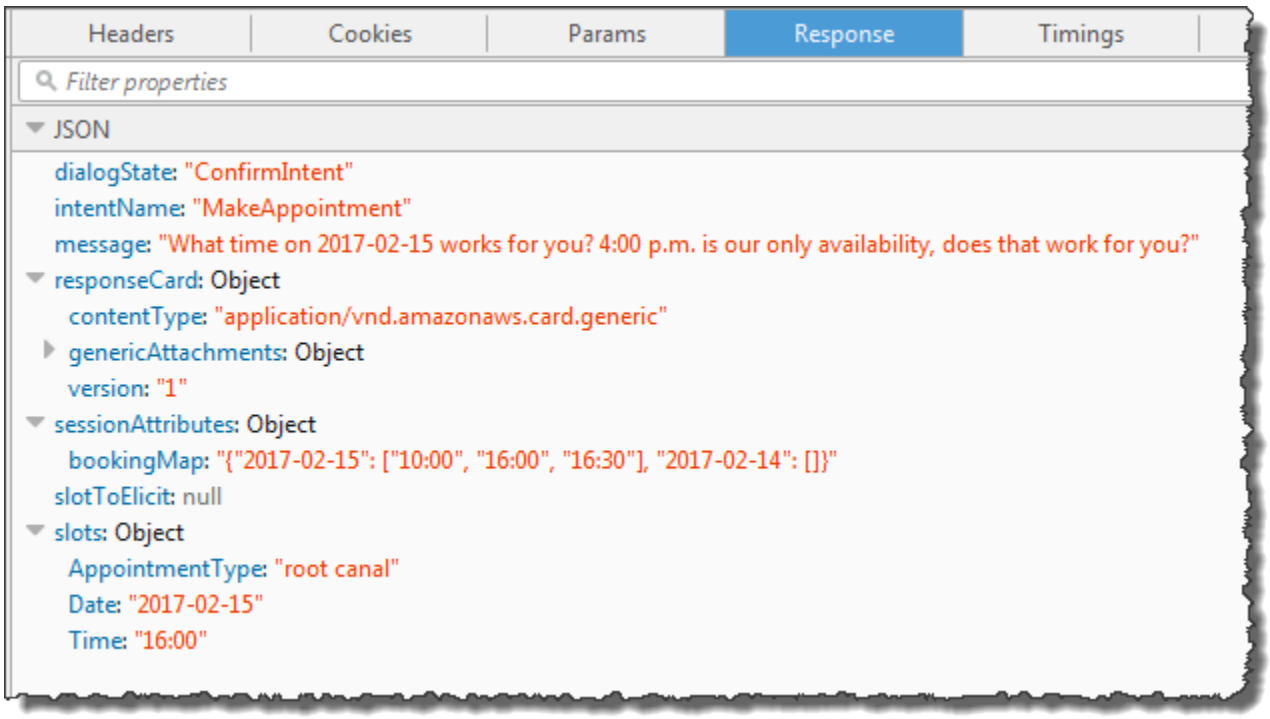
De nuevo, la respuesta incluye los campos `dialogAction` y `sessionAttributes`. Entre otras cosas, `dialogAction` devuelve los siguientes campos:

- `dialogAction` field:
  - `type`: la función `Lambda` establece este valor en `ConfirmIntent` para que Amazon Lex obtenga la confirmación por parte del usuario de la hora de la cita sugerida en `message`.
  - `responseCard`: devuelve una lista de valores sí/no para que el usuario elija. Si el cliente admite tarjetas de respuesta, mostrará la tarjeta de respuesta, tal y como aparece en el ejemplo siguiente:



- `sessionAttributes`: la función de Lambda establece como valor del atributo de sesión `bookingMap` la fecha de la cita y las citas disponibles en esa fecha. En este ejemplo, trabajamos con citas de 30 minutos. Para una endodoncia se necesita una hora y solo está disponible las 16:00 horas.

d. Como se indica en `dialogAction.type` en la respuesta de la función de Lambda, Amazon Lex devuelve la siguiente respuesta al cliente:



El cliente muestra el siguiente mensaje: What time on 2017-02-15 works for you? 4:00 p.m. is our only availability, does that work for you?

5. Usuario: elija **`yes`**.

   Amazon Lex invoca la función de Lambda con los siguientes datos del evento. Dado que el usuario ha contestado **`yes`**, Amazon Lex establece `confirmationStatus` en `Confirmed` y el campo `Time` en `currentIntent.slots` para `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": {
   }
}
```

Puesto que `confirmationStatus` está confirmado, la función de Lambda procesa la intención (programa una cita con el dentista) y devuelve la siguiente respuesta a 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\"]}"
    }
}
```

Tenga en cuenta lo siguiente:

- La función Lambda ha actualizado los `sessionAttributes`.
- `dialogAction.type` se establece en `Close`, por lo que Amazon Lex no esperará ninguna respuesta del usuario.
- `dialogAction.fulfillmentState` se establece en `Fulfilled`, lo que indica que se ha cumplido la intención correctamente.

El cliente muestra el mensaje: Okay, I have booked your appointment. We will see you at 4:00 p.m. on 2017-02-15.

Reserva de viaje

Este ejemplo muestra cómo se crea un bot configurado para respaldar múltiples intenciones. El ejemplo también ilustra cómo puede utilizar los atributos de la sesión para el intercambio de información entre intenciones. Después de crear el bot, debe utilizar un cliente de pruebas en la consola de Amazon Lex para probar el bot (BookTrip). El cliente utiliza la operación de la API en tiempo de ejecución PostText para enviar una solicitud a Amazon Lex por cada entrada del usuario.

El bot BookTrip en este ejemplo está configurado con dos intenciones (BookHotel y BookCar). Por ejemplo, suponga que un usuario reserva primero un hotel. Durante la interacción, el usuario proporciona información como, por ejemplo, la fecha de llegada, la ubicación y el número de noches. Una vez cumplida la intención, el cliente puede conservar esta información gracias a los atributos de la sesión. Para obtener más información acerca de los atributos de la sesión, consulte PostText.

Ahora suponga que el usuario reserva un automóvil. Al usar la información que el usuario ha especificado en la intención BookHotel anterior (es decir, ciudad de destino y fechas de llegada y de salida), el enlace de código (función Lambda), que ha configurado para inicializar y validar la intención BookCar, inicializa los datos de slot para la intención BookCar (es decir, destino, ciudad de recogida, fecha de recogida y fecha de devolución). Con ello se ilustra cómo el intercambio de información entre intenciones le permite crear bots que pueden participar en una conversación dinámica con el usuario.

En este ejemplo utilizamos los siguientes atributos de la sesión. Solo el cliente y la función de Lambda pueden definir y actualizar los atributos de la sesión. Amazon Lex solo los pasa entre el cliente y la función de Lambda. Amazon Lex no mantiene ni modifica los atributos de la sesión.

- `currentReservation`: contiene los datos de la ranura para una reserva en curso y demás información relevante. Por ejemplo, a continuación presentamos un ejemplo de solicitud del cliente a Amazon Lex. En él se muestra el atributo de la sesión `currentReservation` en el cuerpo de la solicitud.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Chicago",
```

```
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Moscow\",
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

- `lastConfirmedReservation`: contiene información similar de una intención anterior, si la hay. Por ejemplo, si el usuario reserva un hotel y, a continuación, reserva un automóvil, este atributo de la sesión almacena los datos de slot de la intención BookHotel anterior.

- `confirmationContext`: la función de Lambda lo fija en `AutoPopulate` cuando rellena previamente algunos de los datos de ranura basados en datos de ranura de la reserva anterior (si la hay). Esto permite el intercambio de información entre intenciones. Por ejemplo, si el usuario ya ha reservado un hotel y ahora quiere reservar un automóvil, Amazon Lex puede solicitar al usuario que confirme (o deniegue) que está reservado el automóvil para la misma ciudad y las mismas fechas que la reserva de hotel.

En este ejercicio utilizará esquemas para crear un bot de Amazon Lex y una función de Lambda. Para obtener más información acerca de los proyectos, consulte Esquemas de Amazon Lex y AWS Lambda.

Paso siguiente

Paso 1: revisión de los proyectos utilizados en este ejercicio

Paso 1: revisión de los proyectos utilizados en este ejercicio

Temas

- Descripción general sobre el proyecto de bot (BookTrip)
- Descripción general del esquema de funciones de Lambda (lex-book-trip-python)

Descripción general sobre el proyecto de bot (BookTrip)

El proyecto (BookTrip) que utiliza para crear un bot incluye la siguiente configuración previa:

- Tipos de slot: dos tipos de slot personalizados:
  - `RoomTypes` con valores de enumeración: `king`, `queen` y `deluxe` para usar con la intención `BookHotel`.
  - `CarTypes` con valores de enumeración: `economy`, `standard`, `midsize`, `full size`, `luxury` y `minivan` para usar con la intención `BookCar`.

- Intención 1 (BookHotel): está preconfigurada del siguiente modo:
  - Slots preconfigurados
    - `RoomType` del tipo de slot personalizado `RoomTypes`
    - `Location` del tipo de slot integrado `AMAZON.US_CITY`
    - `CheckInDate` del tipo de slot integrado `AMAZON.DATE`
    - `Nights` del tipo de slot integrado `AMAZON.NUMBER`
  - Enunciados preconfigurados
    - "Reservar un hotel"
    - "Quiero hacer reservas de hotel"
    - "Reservar una {Nights} en {Location}"

    Si el usuario utiliza uno de estos enunciados, Amazon Lex determina que `BookHotel` es la intención y, a continuación, solicita al usuario datos de ranura.
  - Preguntas preconfiguradas
    - Pregunta para el slot `Location`: "¿En qué ciudad va a pernoctar?"
    - Pregunta para el slot `CheckInDate`: "¿Qué día desea registrarse?"
    - Pregunta para el slot `Nights`: "¿Cuántas noches va a quedarse?"
    - Pregunta para el slot `RoomType`: "¿Qué tipo de habitación desea: queen, king o deluxe?"
    - Enunciado de confirmación: “De acuerdo, tengo anotado que busca alojamiento para {Nights} noches en {Location} a partir del {CheckInDate}. ¿Tramito la reserva?"
    - Rechazo: "De acuerdo, he cancelado la reserva en curso".

- Intención 2 (BookCar): está preconfigurada del siguiente modo:
  - Slots preconfigurados
    - `PickUpCity` del tipo integrado `AMAZON.US_CITY`
    - `PickUpDate` del tipo integrado `AMAZON.DATE`
    - `ReturnDate` del tipo integrado `AMAZON.DATE`
    - `DriverAge` del tipo integrado `AMAZON.NUMBER`
    - `CarType` del tipo personalizado `CarTypes`
  - Enunciados preconfigurados
    - "Reservar un automóvil"
    - "Reservar un automóvil"
    - "Realizar una reserva de automóvil"

    Si el usuario utiliza uno de estos enunciados, Amazon Lex determina que BookCar es la intención y, a continuación, solicita al usuario datos de ranura.
  - Preguntas preconfiguradas
    - Pregunta para el slot `PickUpCity`: "¿En qué ciudad desea alquilar un automóvil?"
    - Pregunta para el slot `PickUpDate`: "¿A partir de qué día desea alquilarlo?"
    - Pregunta para el slot `ReturnDate`: "¿Qué día desea devolver el automóvil?"
    - Pregunta para el slot `DriverAge`: "¿Qué edad tiene el conductor que va a alquilar el automóvil?"
    - Pregunta para el slot `CarType`: “¿Qué tipo de automóvil desea alquilar?” Nuestras opciones más populares son: económico, tamaño medio y lujo"
    - Enunciado de confirmación: “De acuerdo, tengo anotado que quiere alquilar un {CarType} en {PickUpCity} a partir de {PickUpDate} hasta {ReturnDate}. ¿Tramito la reserva?"
    - Rechazo: "De acuerdo, he cancelado la reserva en curso".

Descripción general del esquema de funciones de Lambda (lex-book-trip-python)

Además del proyecto de bot, AWS Lambda incluye un proyecto (lex-book-trip-python) que puede utilizar como enlace de código al proyecto de bot. Para obtener una lista de los esquemas de bot y los esquemas de funciones de Lambda correspondientes, consulte Esquemas de Amazon Lex y AWS Lambda.

Al crear un bot con el esquema BookTrip, debe actualizar la configuración de ambas intenciones (BookCar y BookHotel) mediante la adición de esta función de Lambda como enlace de código tanto para la inicialización o validación de la entrada de datos del usuario como para el cumplimiento de las intenciones.

Este código de la función Lambda muestra una conversación dinámica con información conocida de antemano (incluida en los atributos de la sesión) acerca de un usuario para inicializar valores de slot para una intención. Para obtener más información, consulte Gestión del contexto de la conversación.

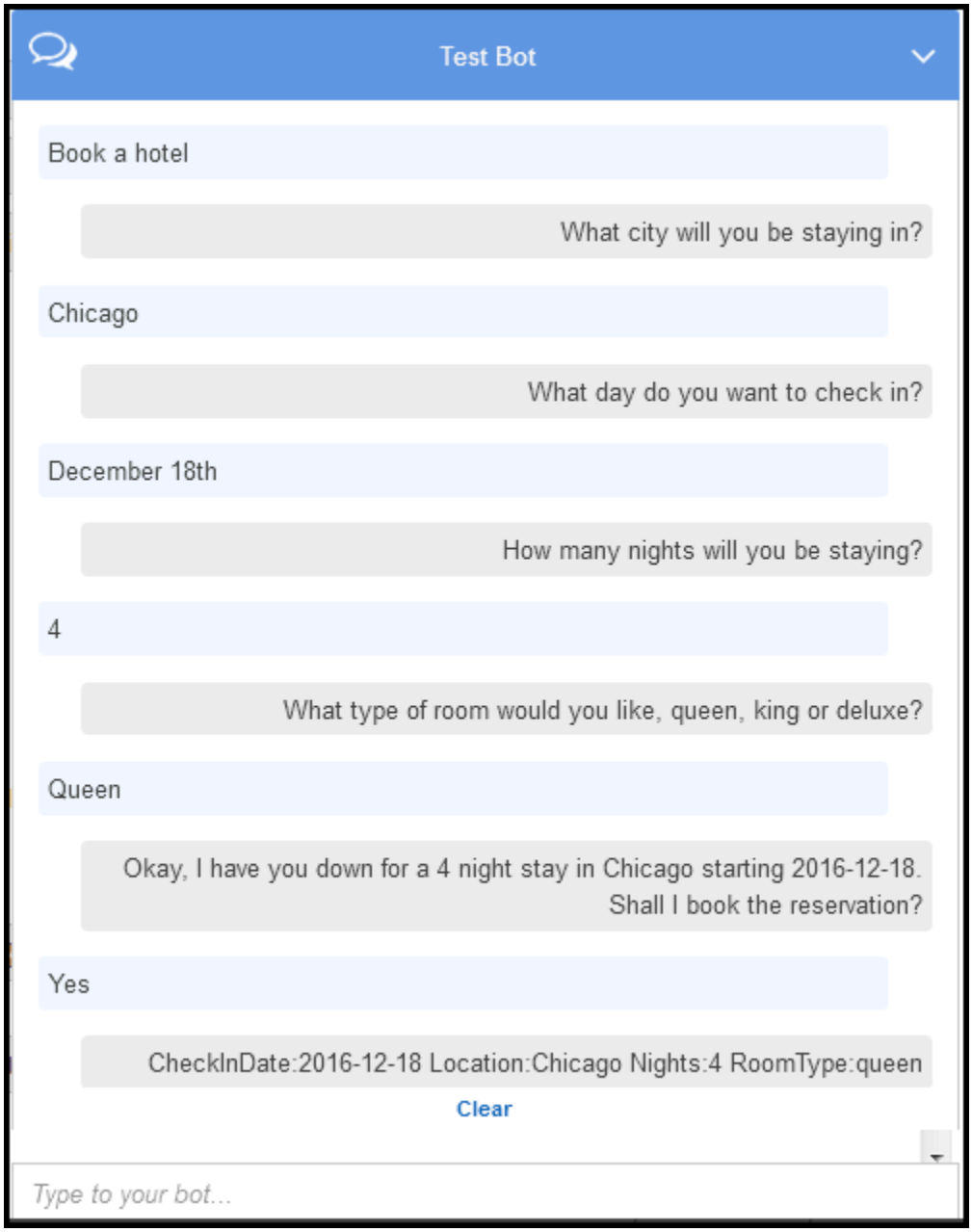
Paso siguiente

Paso 2: creación de un bot de Amazon Lex

Paso 2: creación de un bot de Amazon Lex

En esta sección, creará un bot de Amazon Lex (BookTrip).

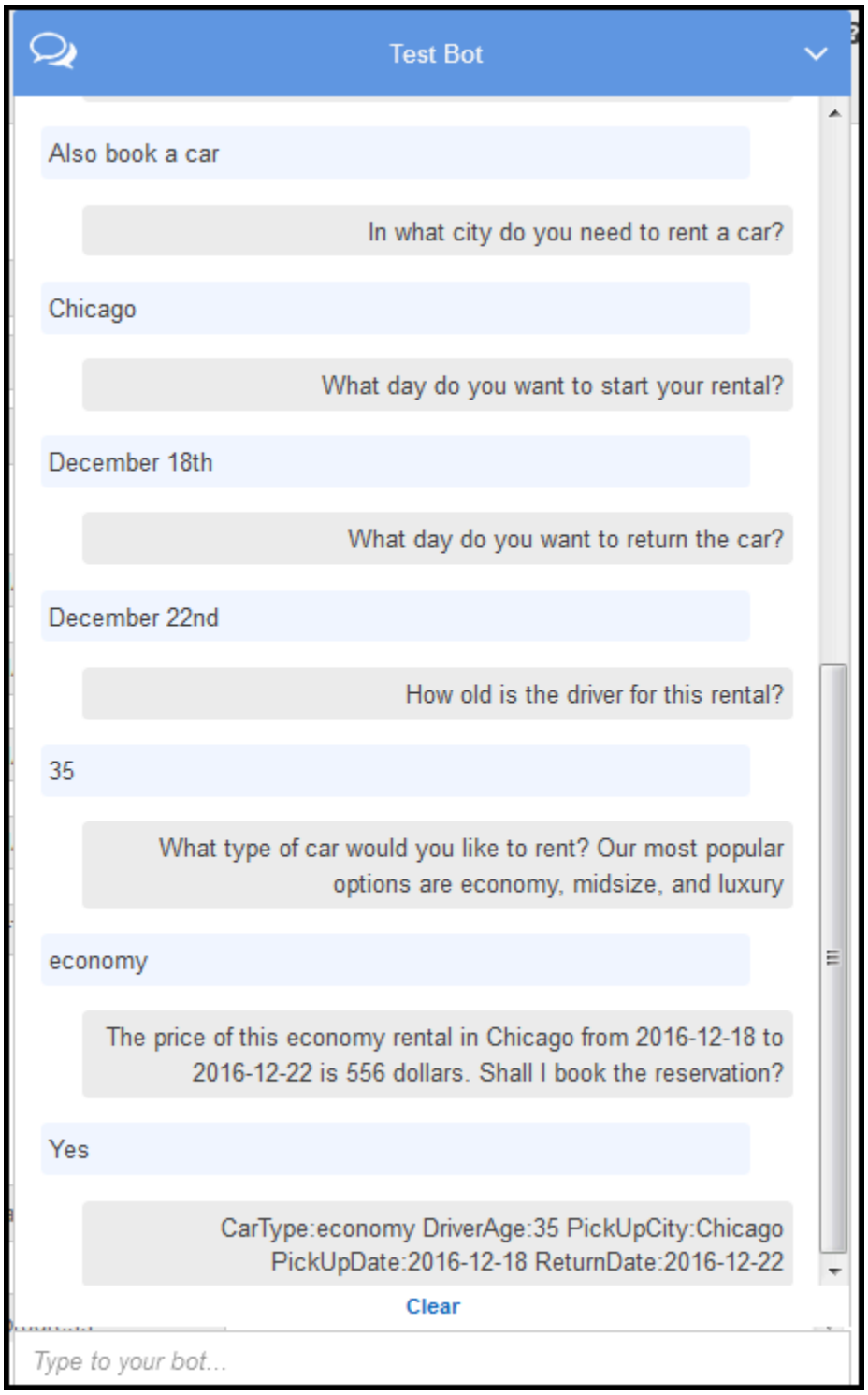
1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la página Bots, elija Create.
3. En la página Create your Lex bot:
   - Elija el proyecto BookTrip.
   - Deje el nombre de bot predeterminado (BookTrip).
4. Seleccione Create (Crear). La consola envía una serie de solicitudes a Amazon Lex para crear el bot. Tenga en cuenta lo siguiente:
5. La consola muestra el bot BookTrip. En la pestaña Editor, revise los detalles de las intenciones preconfiguradas (BookCar y BookHotel).
6. Pruebe el bot en la ventana de pruebas. Utilice lo siguiente para entablar una conversación de prueba con su bot:



Amazon Lex deduce la intención (BookHotel) a partir de la entrada inicial del usuario ("Book a hotel"). A continuación, el bot utiliza las preguntas preconfiguradas en esta intención para obtener datos de slot del usuario. Una vez que el usuario haya proporcionado todos los datos de ranura, Amazon Lex devuelve una respuesta al cliente con un mensaje que incluye todas las entradas del usuario en forma de mensaje. El cliente muestra el mensaje en la respuesta, tal como aparece a continuación.

```
CheckInDate:2016-12-18 Location:Chicago Nights:5 RoomType:queen
```

Ahora debe continuar con la conversación e intentar reservar un automóvil.



Tenga en cuenta que,

- En este momento no se validan los datos del usuario. Por ejemplo, puede indicar cualquier ciudad para reservar un hotel.
- Para reservar el automóvil está proporcionando parte de la misma información (destino, ciudad de recogida, fecha de recogida y fecha de devolución). En una conversación dinámica, su bot debería inicializar parte de esta información en función de las entradas anteriores del usuario para reservar un hotel.

En la siguiente sección debe crear una función Lambda para que valide ciertos datos del usuario e inicialice el intercambio de información entre intenciones mediante los atributos de la sesión. A continuación, debe actualizar la configuración de la intención y añadir la función Lambda como enlace de código para inicializar y validar la entrada del usuario y cumplir la intención.

Paso siguiente

Paso 3: creación de una función de Lambda

Paso 3: creación de una función de Lambda

En esta sección, debe crear una función de Lambda con un esquema (lex-book-trip-python) que se proporciona en la consola de AWS Lambda. También debe probar la función de Lambda con los ejemplos de datos de evento proporcionados por la consola.

Esta función de Lambda está escrita en Python.

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Crear función.
3. Seleccione Use a blueprint (Utilizar un esquema). Escriba **`lex`** para encontrar el proyecto `lex-book-trip-python` y selecciónelo.
4. Elija Configurar para configurar la función de Lambda de la siguiente forma.
   - Escriba el nombre de la función de Lambda (`BookTripCodeHook`).
   - Para el rol, elija Create a new role from template(s) y luego escriba el nombre del rol.
   - Deje los demás valores predeterminados.
5. Elija Crear función.

6. Si utiliza una configuración regional que no sea Inglés (EE. UU.) (en-US), actualice los nombres de las intenciones tal como se indica en Actualización de un esquema para una configuración regional específica.
7. Pruebe la función de Lambda. Invoque la función de Lambda dos veces con los datos de ejemplo, una para reservar un automóvil y otra para reservar un hotel.

    a. Elija Configure test event (Configurar evento de prueba) en el menú desplegable Select a test event (Seleccionar un evento de prueba).

    b. Elija Amazon Lex Book Hotel en la lista Sample event template (Plantilla de evento de muestra).

    Este ejemplo de evento sigue el modelo solicitud/respuesta de Amazon Lex. Para obtener más información, consulte Uso de funciones de Lambda.

    c. Elija Save and test (Guardar y probar).

    d. Compruebe que la función de Lambda se ha ejecutado correctamente. En este caso, la respuesta sigue el modelo de respuesta de Amazon Lex.

    e. Repita el paso. Esta vez elija Amazon Lex Book Car en la lista Sample event template (Plantilla de evento de muestra). La función de Lambda procesa la reserva del automóvil.

Paso siguiente

Paso 4: adición de la función de Lambda como enlace de código

Paso 4: adición de la función de Lambda como enlace de código

En esta sección debe actualizar la configuración de las intenciones BookCar y BookHotel mediante la adición de la función de Lambda como enlace de código para las actividades de inicialización o validación y de cumplimiento. Asegúrese de elegir la versión $LATEST de las intenciones, ya que solo es posible actualizar la versión $LATEST de los recursos de Amazon Lex.

1. En la consola de Amazon Lex, elija el bot BookTrip.
2. En la pestaña Editor, elija la intención BookHotel. Actualice la configuración de la intención de la siguiente manera:

    a. Asegúrese de que la versión de la intención (junto al nombre de la intención) es $LATEST.

b. Añada la función de Lambda como enlace de código de inicialización y validación de la siguiente manera:

   - En Options, elija Initialization and validation code hook.
   - Elija la función de Lambda en la lista.

c. Añada la función de Lambda como enlace de código de cumplimiento de la siguiente manera:

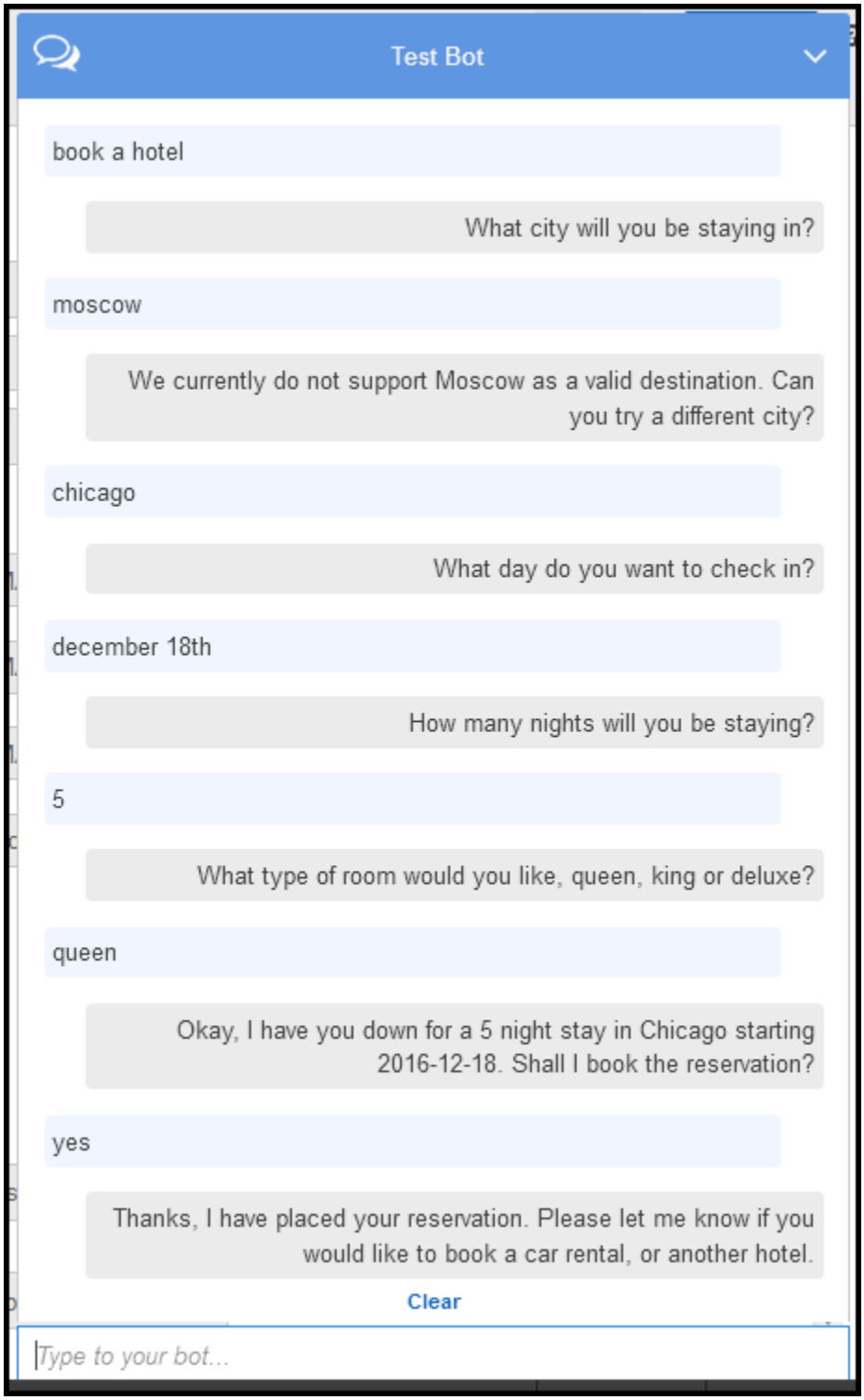
   - En Fulfillment, elija AWS Lambda function.
   - Elija la función de Lambda en la lista.
   - Elija Goodbye message y escriba un mensaje.

d. Seleccione Save.

3. En la pestaña Editor, elija la intención BookCar. Siga los pasos anteriores para añadir la función Lambda como enlace de código de validación y cumplimiento.

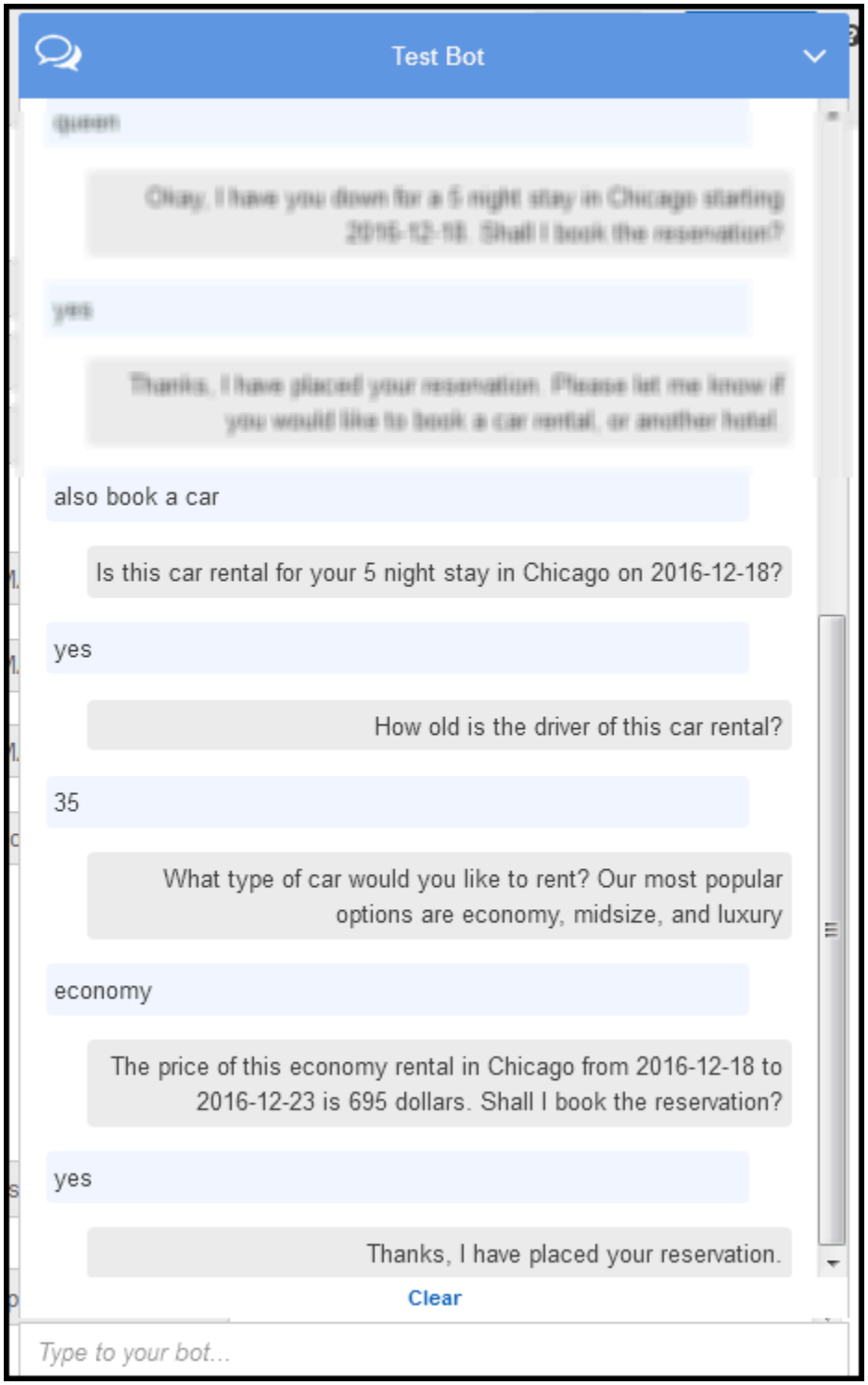
4. Elija Compilar. La consola envía una serie de solicitudes a Amazon Lex para guardar las configuraciones.

5. Pruebe el bot. Ahora que ya tiene una función de Lambda que realiza la inicialización, la validación de los datos del usuario y el cumplimiento, podrá ver la diferencia en la interacción con el usuario en la siguiente conversación:



Para obtener más información sobre el flujo de datos desde el cliente (consola) a Amazon Lex y desde Amazon Lex a la función de Lambda, consulte Flujo de datos: intención de reservar un hotel.

6. Siga con la conversación y reserve un automóvil, tal y como se muestra en la siguiente imagen:



Cuando decide reservar un automóvil, el cliente (la consola) envía una solicitud a Amazon Lex, que incluye los atributos de la sesión (de la conversación anterior, BookHotel). Amazon Lex pasa esta información a la función de Lambda, que, a continuación, inicializa (es decir,

rellena previamente) algunos de los datos de ranura de BookCar (PickUpDate, ReturnDate y PickUpCity).

> **Note**
>
> Aquí se muestra cómo se pueden utilizar los atributos de la sesión para conservar el contexto entre diferentes intenciones. La consola cliente proporciona el enlace Clear en la ventana de prueba que un usuario puede utilizar para borrar cualquier atributo de la sesión anterior.

Para obtener más información sobre el flujo de datos desde el cliente (consola) a Amazon Lex y desde Amazon Lex a la función de Lambda, consulte Flujo de datos: intención de reservar un automóvil.

Detalles del flujo de información

En este ejercicio ha iniciado una conversación con el bot BookTrip de Amazon Lex a través de la ventana de pruebas del cliente proporcionada en la consola de Amazon Lex. En esta sección se explica lo siguiente:

- El flujo de datos entre el cliente y Amazon Lex.

En esta sección se supone que el cliente envía solicitudes a Amazon Lex mediante la API en tiempo de ejecución `PostText` y se muestran los detalles correspondientes de la solicitud y la respuesta. Para obtener más información acerca de la API en tiempo de ejecución `PostText`, consulte PostText.

> **Note**
>
> Para ver un ejemplo de flujo de información entre el cliente y Amazon Lex en el que el usuario usa la API `PostContent`, consulte Paso 2a (opcional): revisión de los detalles del flujo de información escrita (consola) .

- El flujo de datos entre Amazon Lex y la función de Lambda. Para obtener más información, consulte Formato del evento de entrada y de la respuesta de la función de Lambda.

Temas

- Flujo de datos: intención de reservar un hotel
- Flujo de datos: intención de reservar un automóvil

Flujo de datos: intención de reservar un hotel

En esta sección se explica lo que ocurre después de cada entrada del usuario.

1. Usuario: "reservar un hotel"

   a. El cliente (la consola) envía la siguiente solicitud PostText a 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 la URI de la solicitud como el cuerpo proporcionan información a Amazon Lex:

   - URI de la solicitud: proporciona el nombre del bot (BookTrip), el alias del bot ($LATEST) y el nombre de usuario. El `text` de cola indica que se trata de una solicitud de API `PostText` (no `PostContent`).
   - Cuerpo de la solicitud: incluye la entrada del usuario (`inputText`). No hay `sessionAttributes`. Inicialmente, se trata de un objeto vacío y la función de Lambda establece primero los atributos de sesión.

   b. Amazon Lex deduce la intención (BookHotel) a partir de `inputText`. Esta intención está configurada con una función de Lambda como enlace de código para las tareas de inicialización y validación de los datos del usuario. Por ello, Amazon Lex invoca la función

de Lambda pasando la siguiente información como parámetro de evento (consulte Formato del evento 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"
   }
}
```

Además de la información que envía el cliente, Amazon Lex también incluye los siguientes datos adicionales:

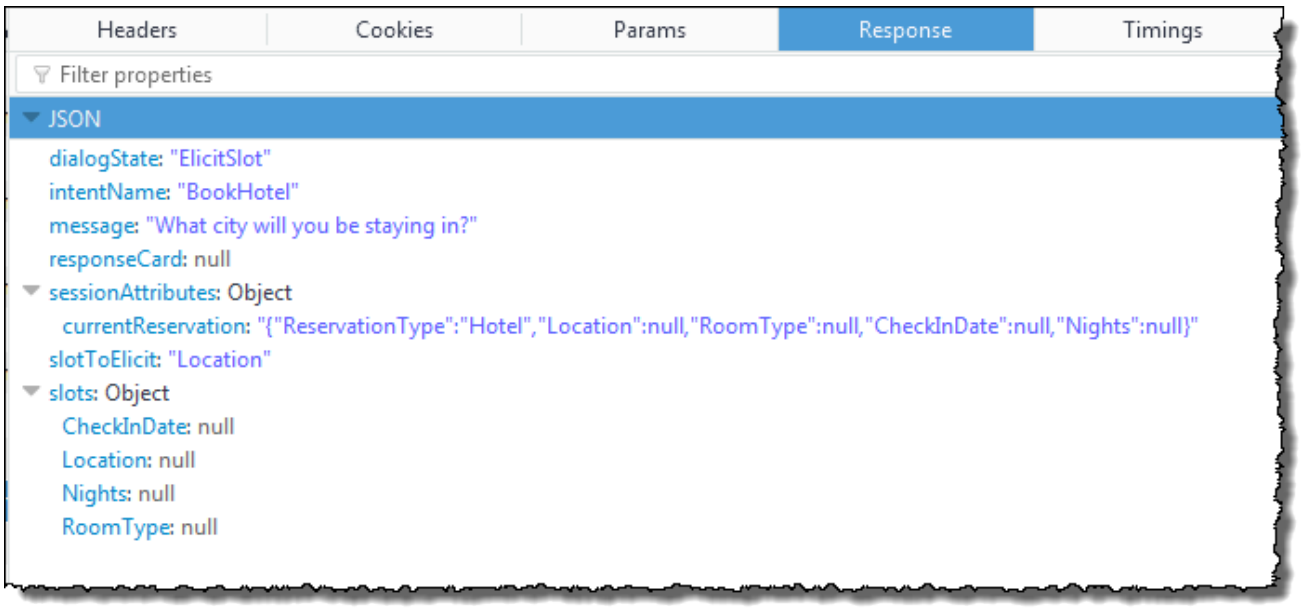
- `messageVersion`: actualmente Amazon Lex solo es compatible con la versión 1.0.
- `invocationSource`: indica la finalidad de invocar la función de Lambda. En este caso, se trata de inicializar y validar los datos del usuario (en este punto Amazon Lex sabe que el usuario aún no ha proporcionado todos los datos de ranura para llevar a cabo la intención).
- `currentIntent`: todos los valores de ranura se configuran como nulos.

c. En este momento, todos los valores de slot son nulos. La función de Lambda no tiene que validar nada y devuelve la siguiente respuesta a Amazon Lex. Para obtener más información sobre el formato de la respuesta, consulte Formato de respuesta.

```
{
   "sessionAttributes":{
      "currentReservation":"{\"ReservationType\":\"Hotel\",\"Location\":null,
\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
   },
   "dialogAction":{
      "type":"Delegate",
      "slots":{
         "RoomType":null,
         "CheckInDate":null,
         "Nights":null,
         "Location":null
      }
   }
}
```

> **Note**
>
> - `currentReservation`: la función de Lambda incluye este atributo de la sesión. Su valor es una copia de la información actual de slot y el tipo de reserva.
>
> Solo el cliente y la función de Lambda pueden actualizar estos atributos de la sesión. Amazon Lex simplemente pasa estos valores.
> - `dialogAction.type`: al configurar este valor como `Delegate`, la función de Lambda delega la responsabilidad del siguiente procedimiento en Amazon Lex.
>
> Si la función de Lambda detecta algo en la validación de los datos del usuario, le indica a Amazon Lex qué debe hacer a continuación.

d. Según `dialogAction.type`, Amazon Lex decide el siguiente paso: obtener datos del usuario para la ranura `Location`. Selecciona uno de los mensajes de pregunta ("¿En qué ciudad va a pernoctar?") para este slot, según la configuración de la intención, y luego envía esta respuesta al usuario:



Los atributos de la sesión se transfieren al cliente.

El cliente lee la respuesta y, a continuación, muestra el mensaje: "¿En qué ciudad va a pernoctar?"

2. Usuario: "Moscú"

   a. El cliente envía la siguiente solicitud `PostText` a Amazon Lex (se añaden saltos de línea para facilitar la lectura):

```
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}"
   }
}
```

Además de `inputText`, el cliente incluye los mismos atributos de la sesión `currentReservation` que ha recibido.

b. En primer lugar, Amazon Lex interpreta `inputText` en el contexto de la intención actual (el servicio recuerda que ha solicitado al usuario información sobre la ranura `Location`). Actualiza el valor de la ranura para la intención actual e invoca la función de Lambda con el siguiente evento:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":null,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Moscow"
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `invocationSource` sigue siendo `DialogCodeHook`. En este paso, solo se validan los datos del usuario.
> - Amazon Lex simplemente transfiere el atributo de la sesión a la función de Lambda.

- Para `currentIntent.slots`, Amazon Lex ha actualizado la ranura `Location` con `Moscow`.

c. La función de Lambda valida los datos del usuario y determina que `Moscow` es una ubicación no válida.

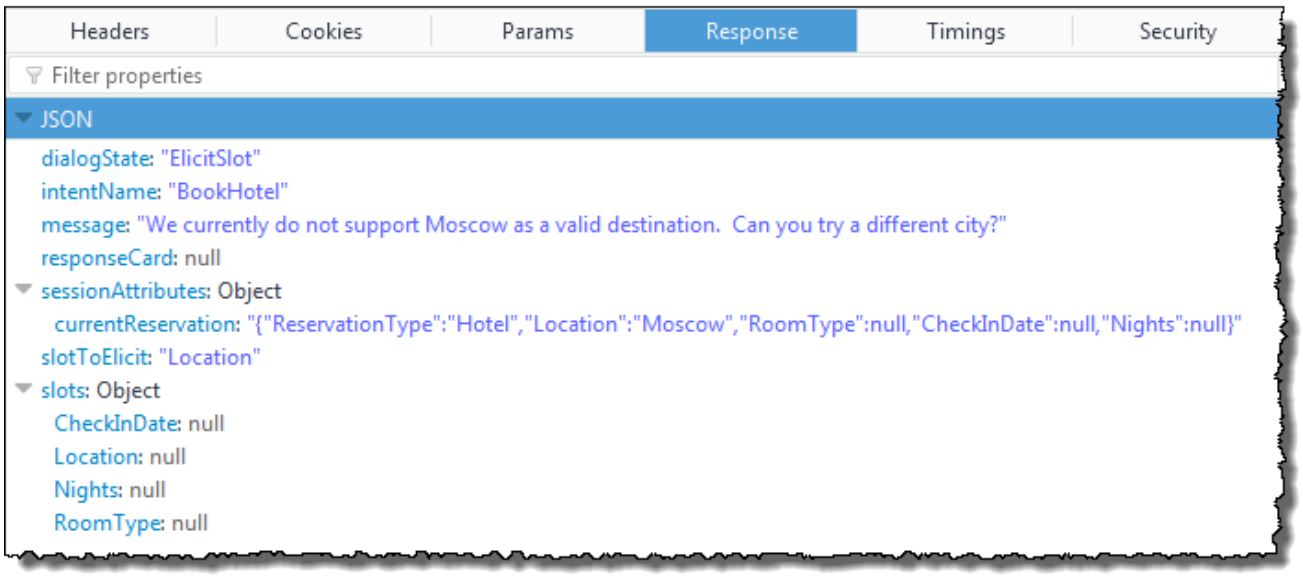
> **Note**
>
> En este ejercicio, la función de Lambda cuenta con una lista de ciudades muy simple y `Moscow` no aparece en ella. En una aplicación de producción, es posible que pueda utilizar una base de datos back-end para obtener esta información.

Restablece el valor de la ranura a nulo e indica a Amazon Lex que solicite al usuario otro valor enviando la siguiente respuesta:

```
{
    "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` se vuelve a fijar como nulo.
> - `dialogAction.type` se establece en `ElicitSlot`, por lo que Amazon Lex vuelve a preguntar al usuario proporcionando lo siguiente:
>   - `dialogAction.slotToElicit`: slot para el que se obtienen datos del usuario.
>   - `dialogAction.message`: un `message` que se transmite al usuario.

d. Amazon Lex tiene en cuenta `dialogAction.type` y pasa la información al cliente con la siguiente respuesta:



El cliente simplemente muestra el mensaje: "En la actualidad, no se admite Moscú como destino válido. ¿Puede probar con otra ciudad?"

3. Usuario: "Chicago"

a. El cliente envía la siguiente solicitud `PostText` a 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 conoce el contexto, es decir, que estaba obteniendo datos para la ranura `Location`. En este contexto, sabe que el valor de `inputText` es para el slot `Location`. A continuación, invoca la función de Lambda mediante el envío del siguiente evento:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":Moscow,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        },
        "confirmationStatus": "None"
    }
}
```

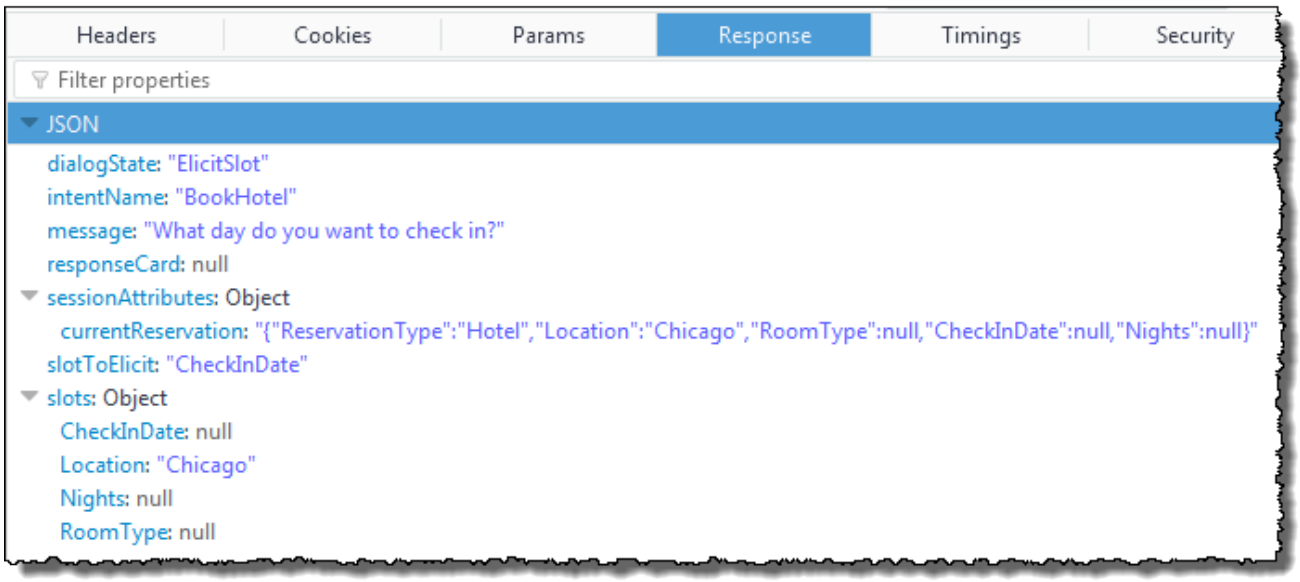
Amazon Lex ha actualizado `currentIntent.slots` al configurar la ranura `Location` con `Chicago`.

c. Al establecer el valor `invocationSource` como `DialogCodeHook`, la función de Lambda valida los datos del usuario. Reconoce `Chicago` como un valor de ranura válido, actualiza el atributo de sesión según corresponda y devuelve la siguiente respuesta a Amazon Lex.

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        }
    }
}
```

> **Note**
>
> - `currentReservation`: la función de Lambda actualiza este atributo de la sesión al configurar `Location` con `Chicago`.
> - `dialogAction.type`: se establece en `Delegate`. Los datos del usuario son válidos, por lo que la función de Lambda indica a Amazon Lex que decida el siguiente paso.

d. Amazon Lex elige el siguiente paso en función de `dialogAction.type`. Amazon Lex sabe que necesita más datos de ranura, de modo que elige la siguiente ranura vacía (`CheckInDate`) con la mayor prioridad, según la configuración de la intención. Selecciona uno de los mensajes de pregunta ("¿Qué día desea registrarse?") para este slot, según la configuración de la intención, y entonces envía esta respuesta al cliente:

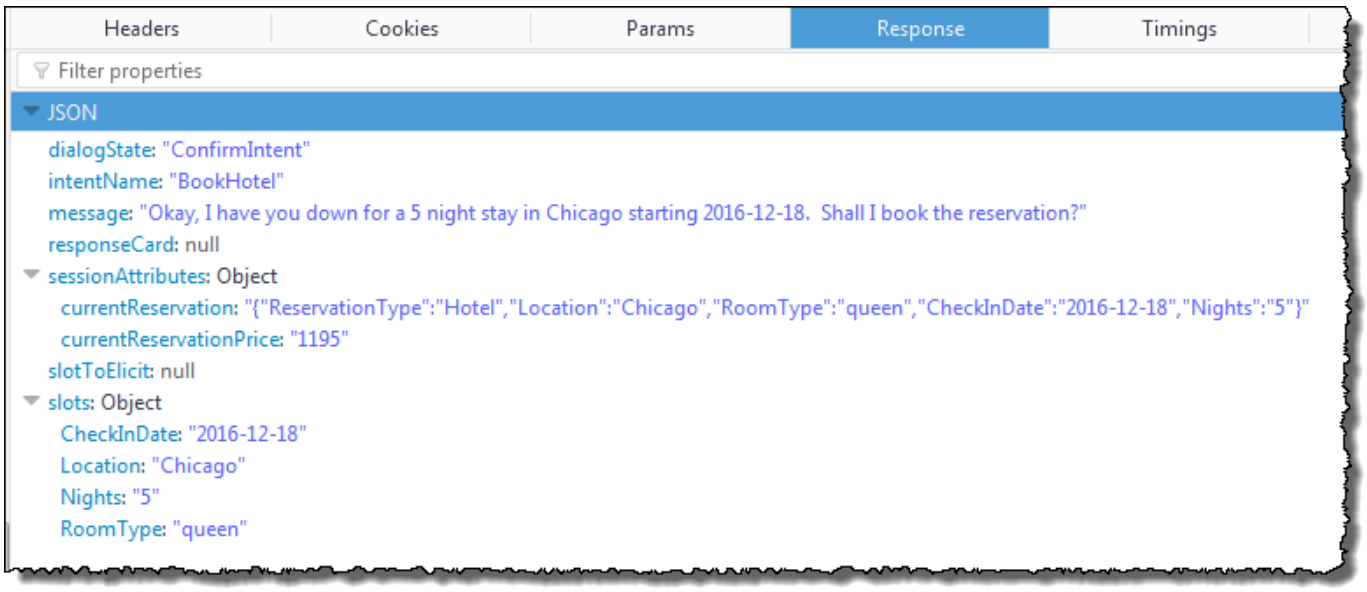


El cliente muestra el mensaje: "¿Qué día desea registrarse?"

4. La interacción con el usuario continúa: el usuario proporciona los datos, la función de Lambda los valida y delega el siguiente paso en Amazon Lex. Al final, el usuario proporciona todos los datos de ranura, la función de Lambda valida todas las entradas del usuario y Amazon Lex reconoce que cuenta con todos los datos de ranura.

> **Note**
>
> En este ejercicio, después de que el usuario proporcione todos los datos de ranura, la función de Lambda calcula el precio de la reserva de hotel y lo devuelve como otro atributo de sesión (`currentReservationPrice`).

En este momento, la intención ya se puede cumplir, pero BookHotel se ha configurado con una pregunta para requerir la confirmación del usuario antes de que Amazon Lex la cumpla. Por este motivo, Amazon Lex envía el siguiente mensaje al cliente para solicitar su confirmación antes de reservar el hotel:



El cliente muestra el mensaje: "De acuerdo, tengo anotado que busca alojamiento para 5 noches en Chicago a partir del 18/12/2016. ¿Tramito la reserva?"

5. Usuario: "sí"

   a. El cliente envía la siguiente solicitud `PostText` a Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Yes",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Chicago\",
                              \"RoomType\":\"queen\",
                              \"CheckInDate\":\"2016-12-18\",
                              \"Nights\":\"5\"}",
      "currentReservationPrice":"1195"
   }
}
```

   b. Amazon Lex interpreta `inputText` en el contexto de la confirmación de la intención actual. Amazon Lex entiende que el usuario quiere continuar con la reserva. Esta vez Amazon Lex llama a la función de Lambda para que cumpla con la intención mediante el envío del siguiente evento. Al configurar `invocationSource` como

`FulfillmentCodeHook` en el evento, lo envía a la función de Lambda. Amazon Lex también establece `confirmationStatus` en `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"
    }
}
```

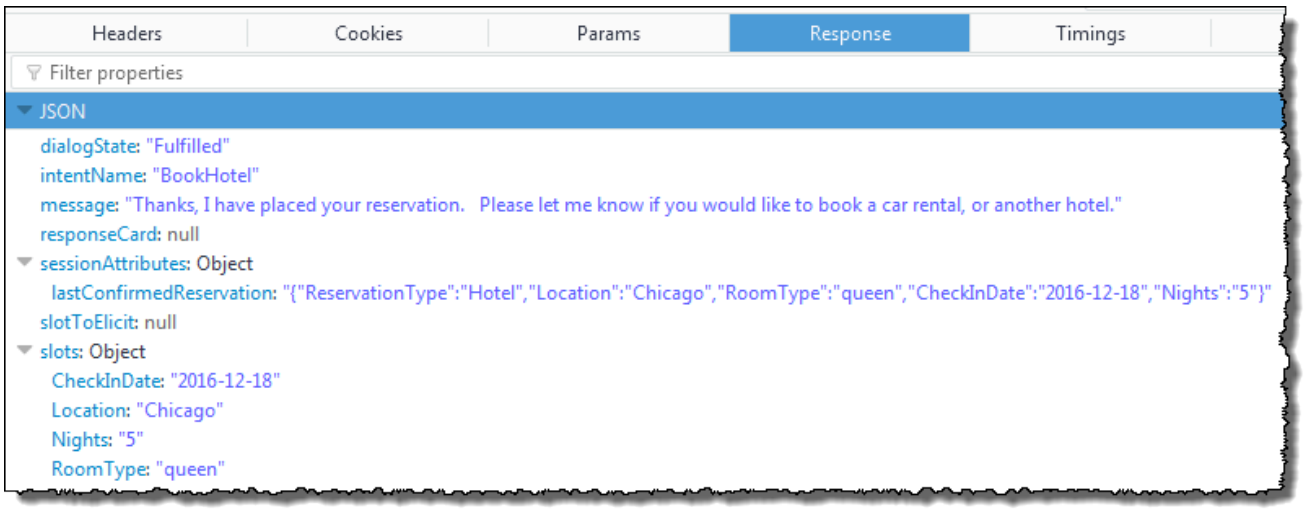
> **Note**
>
> - `invocationSource`: esta vez, Amazon Lex configura este valor como `FulfillmentCodeHook`, por lo que indica a la función de Lambda que cumpla con la intención.
> - `confirmationStatus`: se establece en `Confirmed`.

c. Esta vez, la función de Lambda lleva a cabo la intención BookHotel, Amazon Lex efectúa la reserva y, a continuación, devuelve la siguiente respuesta:

```
{
    "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`: es un nuevo atributo de la sesión que ha añadido la función de Lambda (en lugar de `currentReservation`, `currentReservationPrice`).
> - `dialogAction.type`: la función de Lambda establece este valor en `Close`, lo que indica que Amazon Lex no esperará ninguna respuesta del usuario.
> - `dialogAction.fulfillmentState`: se establece en `Fulfilled` e incluye un `message` adecuado para transmitir al usuario.

d. Amazon Lex analiza `fulfillmentState` y envía la siguiente respuesta al cliente:



> **Note**
>
> - `dialogState`: Amazon Lex establece este valor en `Fulfilled`.
> - `message`: es el mismo mensaje que ha proporcionado la función de Lambda.

El cliente muestra el mensaje.

Flujo de datos: intención de reservar un automóvil

En este ejercicio, el bot BookTrip admite dos intenciones (BookHotel y BookCar). Después de reservar un hotel, el usuario puede continuar la conversación para reservar un automóvil. Mientras la sesión no haya caducado, en cada solicitud subsiguiente el cliente sigue enviando los atributos de la sesión (en este ejemplo, la `lastConfirmedReservation`). La función de Lambda puede utilizar esta información para inicializar los datos de ranura de la intención BookCar. Aquí se muestra cómo puede utilizar los atributos de la sesión en el uso compartido de datos entre intenciones.

En concreto, cuando el usuario elige la intención BookCar, la función de Lambda utiliza la información pertinente del atributo de sesión para rellenar previamente las ranuras (PickUpDate, ReturnDate y PickUpCity) para la intención BookCar.

> **Note**
>
> La consola de Amazon Lex proporciona el enlace Borrar, que puede utilizar para borrar los atributos de sesión anteriores.

Siga los pasos descritos en este procedimiento para continuar con la conversación.

1. Usuario: "reservar también un automóvil"

   a. El cliente envía la siguiente solicitud `PostText` a 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\"}"
      }
   }
   ```

   El cliente incluye el atributo de la sesión `lastConfirmedReservation`.

   b. Amazon Lex deduce la intención (BookHotel) a partir de `inputText`. Esta intención también está configurada para invocar la función de Lambda, para que realice las tareas de inicialización y validación de los datos del usuario. Amazon Lex invoca la función de Lambda con los siguientes 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`: actualmente Amazon Lex solo es compatible con la versión 1.0.
> - `invocationSource`: indica que el objetivo de la invocación es inicializar y validar los datos del usuario.
> - `currentIntent`: incluye el nombre de la intención y las ranuras. En este momento, todos los valores de slot son nulos.

c. La función de Lambda observa que todos los valores de ranura son nulos y no hay nada que validar. Sin embargo, utiliza los atributos de la sesión para inicializar algunos de los valores de slot (`PickUpDate`, `ReturnDate` y `PickUpCity`) y, a continuación, devuelve la siguiente respuesta:

```
{
    "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?"
        }
    }
}
```

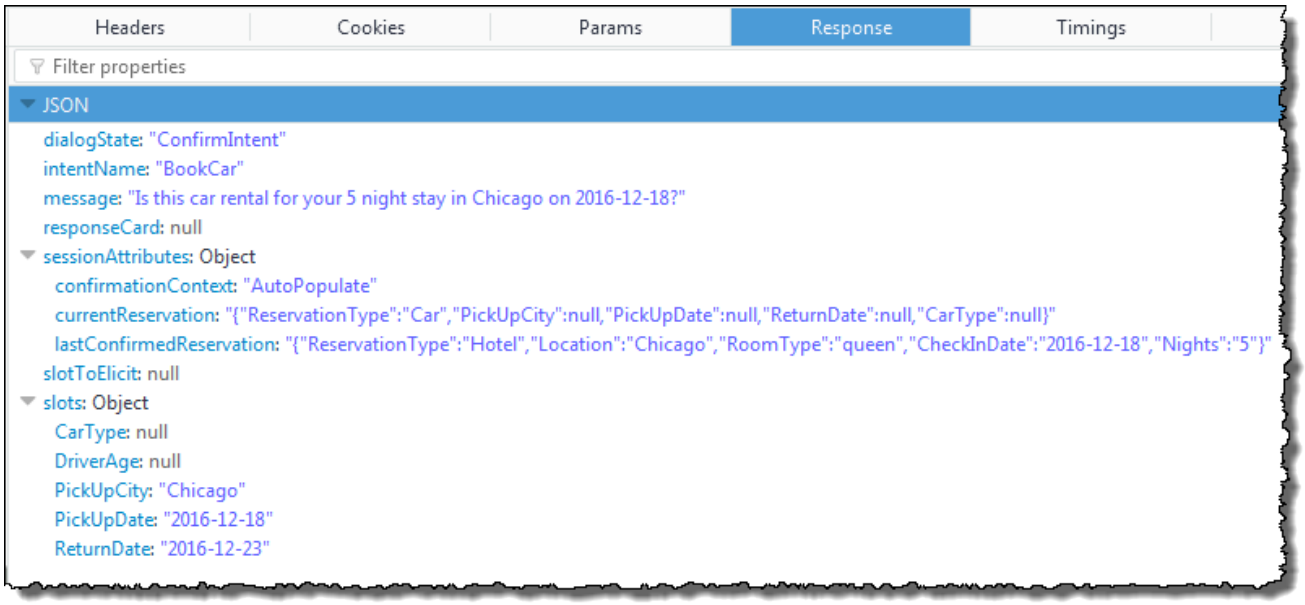
Note

- Además de `lastConfirmedReservation`, la función de Lambda incluye más atributos de la sesión (`currentReservation` y `confirmationContext`).
- `dialogAction.type` toma el valor `ConfirmIntent`, lo que indica a Amazon Lex que se espera una respuesta sí/no del usuario (si el valor de confirmationContext es AutoPopulate, la función de Lambda sabe que la respuesta sí/no del usuario sirve para obtener su confirmación de la inicialización que la función de Lambda ha realizado (de los datos de ranura rellenados automáticamente)).

  La función de Lambda también incluye en la respuesta un mensaje informativo en `dialogAction.message` para que Amazon Lex se lo envíe al cliente.

> **Note**
>
> El término `ConfirmIntent` (valor del `dialogAction.type`) no está relacionado con ninguna intención del bot. En el ejemplo, la función de Lambda utiliza este término para indicar a Amazon Lex que obtenga una respuesta sí/no del usuario.

d. Amazon Lex devuelve la siguiente respuesta al cliente en función de `dialogAction.type`:



El cliente muestra el mensaje: "¿El alquiler de este automóvil es para su estancia de 5 noches en Chicago a partir del 18/12/2016?"

2. Usuario: "sí"

a. El cliente envía la siguiente solicitud `PostText` a Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"yes",
   "sessionAttributes":{
      "confirmationContext":"AutoPopulate",
```

```
      "currentReservation":"{\"ReservationType\":\"Car\",
                             \"PickUpCity\":null,
                             \"PickUpDate\":null,
                             \"ReturnDate\":null,
                             \"CarType\":null}",
      "lastConfirmedReservation":"{\"ReservationType\":\"Hotel\",
                                   \"Location\":\"Chicago\",
                                   \"RoomType\":\"queen\",
                                   \"CheckInDate\":\"2016-12-18\",
                                   \"Nights\":\"5\"}"
   }
}
```

b. Amazon Lex lee `inputText` y conoce el contexto (pedirá al usuario que confirme los datos rellenados de forma automática). Amazon Lex invoca la función de Lambda al enviar el siguiente 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"
    }
}
```

Dado que el usuario ha respondido con un Sí, Amazon Lex establece `confirmationStatus` en `Confirmed`.

c. Por el valor de `confirmationStatus`, la función de Lambda sabe que los valores rellenados previamente son correctos. La función de Lambda hace lo siguiente:

   - Actualiza el atributo de la sesión `currentReservation` con el valor de slot rellenado previamente.
   - Establece el `dialogAction.type` en `ElicitSlot`
   - Establece el valor de `slotToElicit` en `DriverAge`.

   Se envía la siguiente respuesta:

```
{
    "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 devuelve la siguiente respuesta:

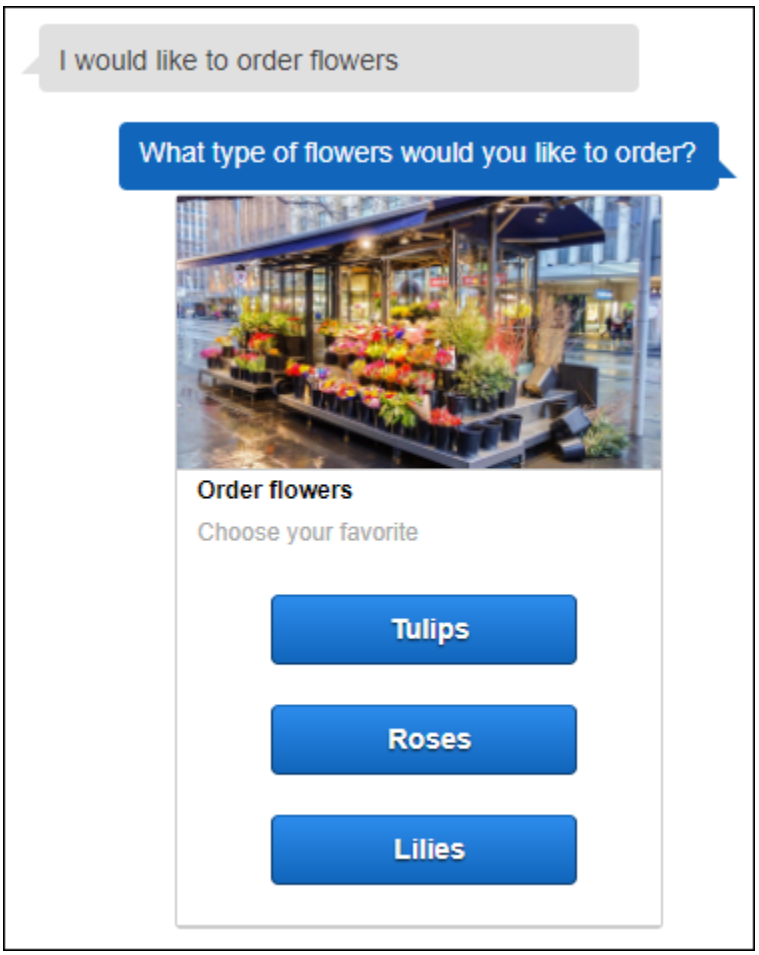
El cliente muestra el mensaje “¿Qué edad tiene el conductor del automóvil de alquiler?” y la conversación continúa.

Uso de una tarjeta de respuesta

En este ejercicio, va a ampliar el ejercicio de introducción 1 añadiendo una tarjeta de respuesta. Puede crear un bot que admita la intención OrderFlowers y, a continuación, actualizar la intención añadiendo una tarjeta de respuesta para el slot `FlowerType`. Además de la siguiente pregunta para el slot `FlowerType`, el usuario puede elegir el tipo de flores de la tarjeta de respuesta:

```
What type of flowers would you like to order?
```

A continuación se presenta la respuesta de la tarjeta:

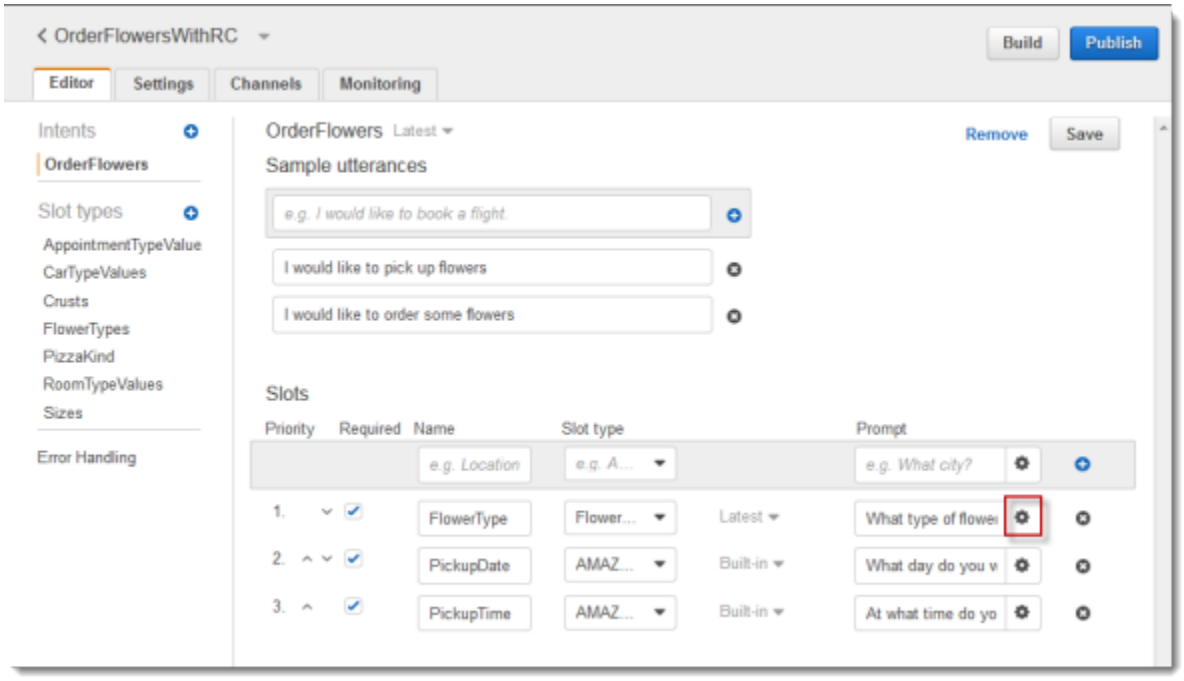


El usuario del bot puede escribir el texto o elegirlo de la lista de tipos de flores. Esta tarjeta de respuesta está configurada con una imagen, que aparece en el cliente tal y como se muestra. Para obtener más información sobre las tarjetas de respuesta, consulte Tarjetas de respuesta.

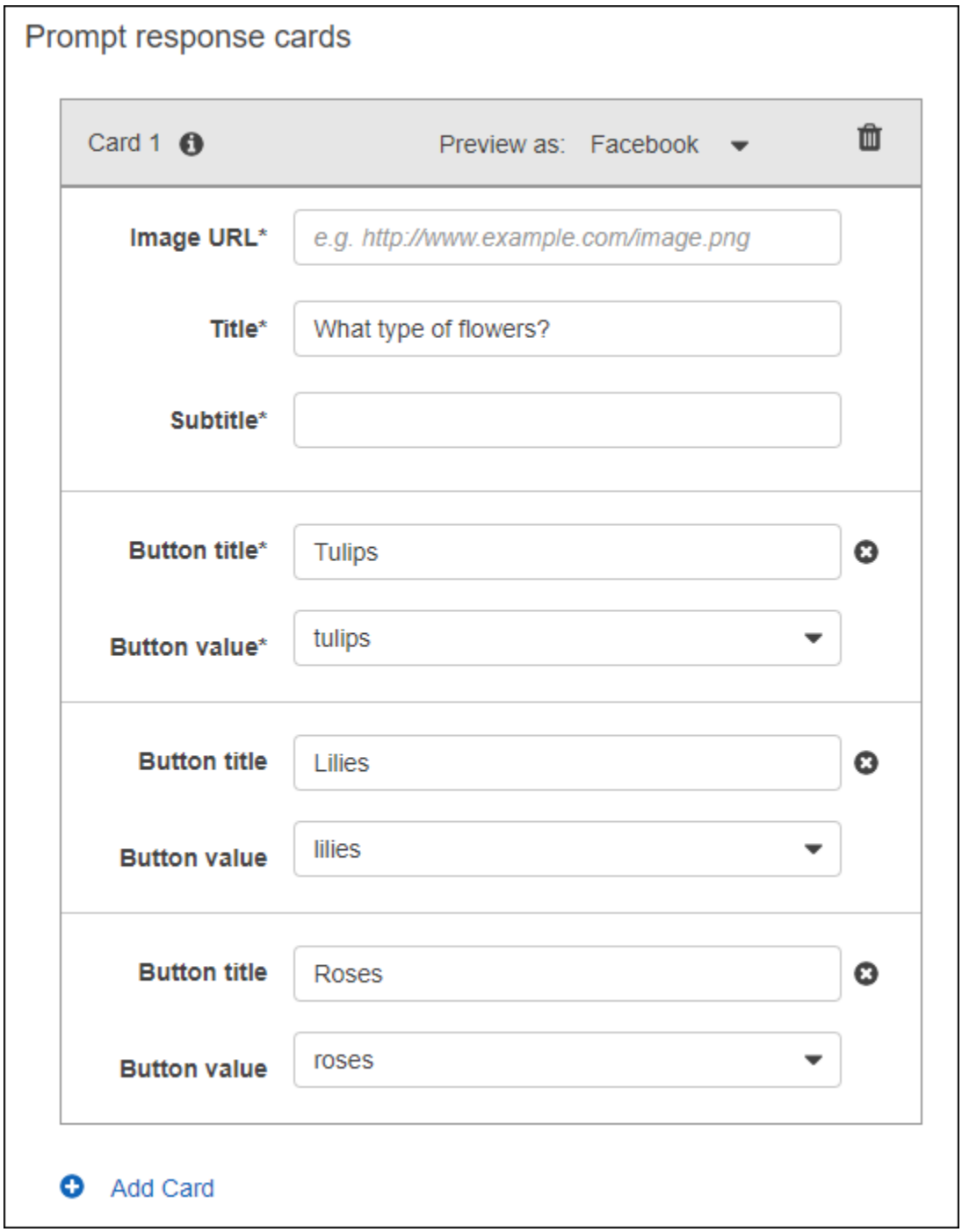
Para crear y probar un bot con una tarjeta de respuesta:

1. Siga el ejercicio de introducción 1 para crear y probar un bot OrderFlowers. Deberá completar los pasos 1, 2 y 3. No es necesario añadir una función de Lambda para probar la tarjeta de respuesta. Para obtener instrucciones, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).
2. Actualice el bot añadiendo la tarjeta de respuesta y, a continuación, publique una versión. Al publicar una versión, especifique un alias (BETA) que apunte hacia ella.
    a. En la consola de Amazon Lex, elija el bot.
    b. Elija la intención `OrderFlowers`.

c. Elija el icono de la rueda de configuración al lado de la pregunta “¿Qué tipo de flores?” para configurar una tarjeta de respuesta para `FlowerType`, tal como se muestra en la siguiente imagen.



d. Escriba un título para la tarjeta y configure tres botones, tal y como se muestra en la siguiente captura de pantalla. Si lo desea, puede añadir una imagen a la tarjeta de respuesta, siempre que tenga una imagen URL. Si va a implementar su bot mediante Twilio SMS, debe proporcionar una URL de imagen.



e. Seleccione Save (Guardar) para guardar la tarjeta de respuesta.

f. Elija Save intent (Guardar intención) para guardar la configuración de la intención.

g. Para crear el bot, elija Build.

h. Para publicar una versión del bot, elija Publish. Especifique BETA como alias que apunte a la versión de bot. Para obtener más información acerca del control de versiones, consulte Control de versiones y alias.

3. Implementar el bot en una plataforma de mensajería:

   - Implemente el bot en la plataforma Facebook Messenger y pruebe la integración. Para obtener instrucciones, consulte Integración de un bot de Amazon Lex con Facebook Messenger. Al comprar las flores, la ventana de mensaje muestra la tarjeta de respuesta para que pueda elegir un tipo de flor.
   - Implemente el bot en la plataforma Slack y pruebe la integración. Para obtener instrucciones, consulte Integración de un bot de Amazon Lex con Slack . Al comprar las flores, la ventana de mensaje muestra la tarjeta de respuesta para que pueda elegir un tipo de flor.
   - Implemente el bot en la plataforma Twilio SMS. Para obtener instrucciones, consulte Integración de un bot de Amazon Lex con Twilio Programmable SMS . Al comprar flores, el mensaje de Twilio muestra la imagen de la tarjeta de respuesta. Twilio SMS no admite botones en la respuesta.

Actualización de enunciados

En este ejercicio, añadirá enunciados adicionales a los que ha creado en el ejercicio 1 de introducción. Utilice la pestaña Supervisión de la consola de Amazon Lex para ver los enunciados que el bot no ha reconocido. Para mejorar la experiencia para los usuarios, hay que añadir esos enunciados al bot.

Las estadísticas de los enunciados no se generan en las siguientes condiciones:

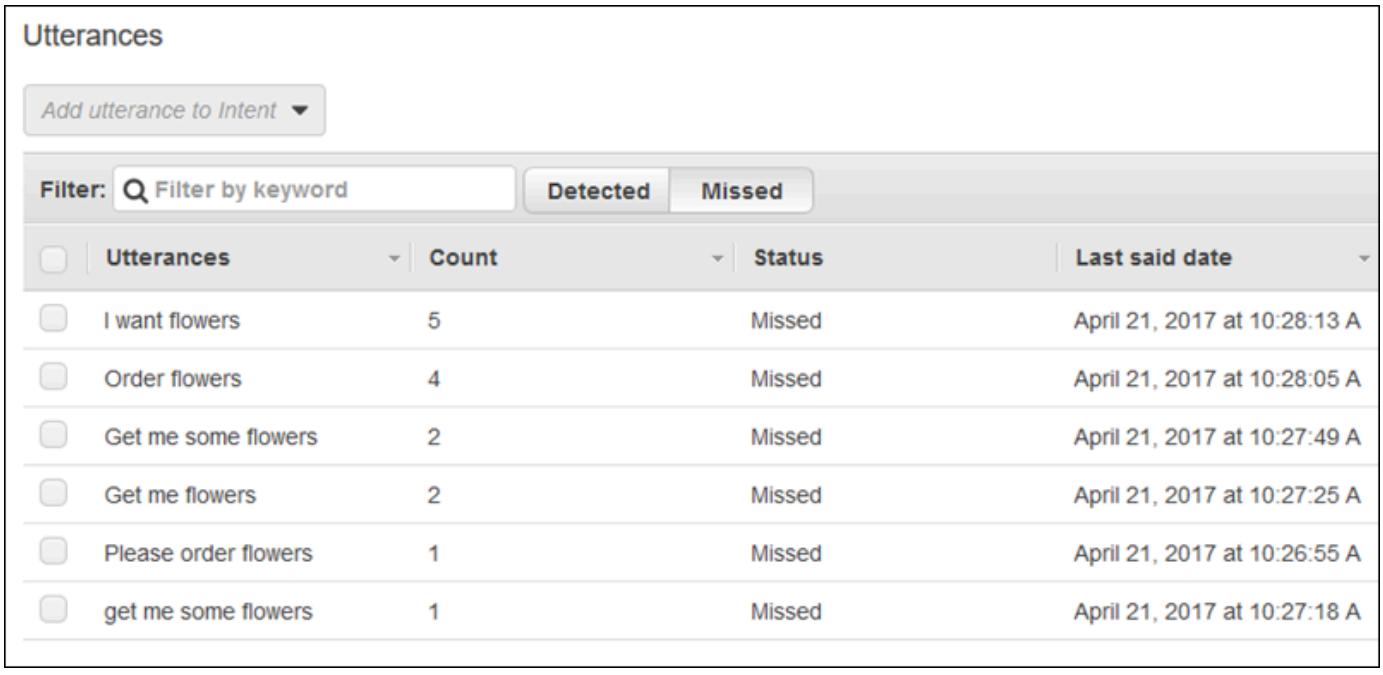
- El `childDirected` campo estaba establecido en verdadero cuando se creó el bot.
- Está utilizando la ofuscación de ranuras con una o más ranuras.
- Ha optado por no participar en la mejora de Amazon Lex.

> **Note**
>
> Las estadísticas de enunciados se generan una vez al día. Puede consultar el enunciado que no se ha reconocido, cuántas veces se ha escuchado y la última fecha y hora en las que se escuchó. Los enunciados perdidos pueden tardar hasta 24 horas en aparecer en la consola.

Puede ver expresiones de las diferentes versiones de su bot. Para cambiar la versión del bot cuyas expresiones está viendo, elija otra versión del menú desplegable situado junto al nombre de bot.

Para ver y añadir enunciados perdidos a un bot:

1. Siga el primer paso del ejercicio de introducción 1 para crear y probar un bot `OrderFlowers`. Para obtener instrucciones, consulte Ejercicio 1: creación de un bot de Amazon Lex mediante un esquema (consola).
2. Compruebe el bot escribiendo los siguientes enunciados en la ventana Test Bot. Escriba cada enunciado varias veces. El bot de ejemplo no reconoce los siguientes enunciados:
   - Order flowers
   - Get me flowers
   - Please order flowers
   - Get me some flowers
3. Espere a que Amazon Lex recopile datos de uso sobre los enunciados no reconocidos. Los datos de los enunciados se generan una vez al día, por lo general, por la noche.
4. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
5. Elija el bot `OrderFlowers`.
6. Elija la pestaña Monitoring, luego elija Utterances en el menú de la izquierda y, a continuación, seleccione el botón Missed. En el siguiente panel se muestra un máximo de 100 enunciados no reconocidos.



7. Para elegir los enunciados perdidos que desea añadir al bot, seleccione la casilla de verificación que hay junto a ellos. Para añadir el enunciado a la versión `$LATEST` de la intención, elija la flecha hacia abajo junto a la lista desplegable Add utterance to intent y, a continuación, elija la intención.
8. Para reconstruir el bot, elija Build y luego Build de nuevo para reconstruirlo.
9. Para verificar que el bot reconoce los nuevos enunciados, utilice el panel Test Bot.

Integración con un sitio web

En este ejemplo, va a integrar un bot con un sitio web con texto y voz. Con JavaScript y los servicios de y AWS puede crear una experiencia interactiva para los visitantes de su sitio web. Puede elegir entre estos ejemplos documentados en el blog de inteligencia artificial de AWS:

- Deploy a Web UI for Your Chatbot (en inglés): muestra una interfaz de usuario web con todas las características, que se puede usar como cliente web para chatbots de Amazon Lex. Puede utilizarla para conocer los clientes web o como componente básico para sus propias aplicaciones.

- "Greetings, visitor!"—Engage Your Web Users with Amazon Lex (en inglés): explica cómo crear una experiencia de conversación en un sitio web con Amazon Lex, AWS SDK para JavaScript en el navegador y Amazon Cognito.

- Capturing Voice Input in a Browser and Sending it to Amazon Lex (en inglés): demuestra cómo se integra un chatbot basado en voz en un sitio web mediante el SDK para JavaScript en el navegador. La aplicación graba audio, lo envía a Amazon Lex y, a continuación, reproduce la respuesta.

Asistente para agentes de centros de llamadas

En este tutorial, utilizará Amazon Lex con Amazon Kendra para compilar un bot de asistencia para agentes de atención al cliente y publicarlo como aplicación web. Amazon Kendra es un servicio de búsqueda empresarial que utiliza machine learning para encontrar respuestas en documentos. Para obtener más información acerca de Amazon Kendra, consulte la Guía para desarrolladores de Amazon Kendra.

El uso de bots de Amazon Lex está muy extendido en los centros de llamadas como primer punto de contacto para los clientes. En muchas ocasiones, un bot basta para responder a las preguntas de los clientes. Cuando un bot no puede proporcionar una respuesta válida, transfiere la conversación a un miembro del equipo de atención al cliente.

En este tutorial, crearemos un bot de Amazon Lex para que los agentes lo empleen para responder a las consultas de los clientes en tiempo real. El bot proporciona respuestas a los agentes para que tan solo tengan que leerlas y no hayan de buscarlas manualmente.

El bot y la aplicación web que creará a lo largo de este tutorial ayudarán a los agentes a atender a los clientes con mayor eficiencia y precisión, ya que podrán dirigirlos a los recursos adecuados. El siguiente diagrama muestra cómo funciona la aplicación web.

Tal como muestra el diagrama, el índice de documentos de Amazon Kendra se almacena en un bucket de Amazon Simple Storage Service (Amazon S3). Si todavía no cuenta con un bucket de S3, puede configurarlo cuando cree el índice de Amazon Kendra. En este tutorial utilizará Amazon Cognito, además de Amazon S3. Amazon Cognito administra los permisos de implementación del bot como aplicación web.

En este tutorial, creará un índice de Amazon Kendra que proporcionará respuestas a las preguntas de los clientes, creará un bot al que agregará intenciones que le permitirán sugerir respuestas basadas en la conversación mantenida con el cliente, configurará Amazon Cognito para administrar los permisos de acceso e implementará el bot como aplicación web.

Tiempo estimado: 75 minutos

Coste estimado: 2,50 USD por hora para un índice de Amazon Kendra y 0,75 USD para 1000 solicitudes de Amazon Lex. El índice de Amazon Kendra se seguirá ejecutando una vez haya finalizado con este ejercicio. Asegúrese de eliminarlo para evitar costes innecesarios.

Nota: Asegúrese de elegir la misma Región de AWS para todos los servicios utilizados en este tutorial.

Temas

Paso 1: creación de un índice de Amazon Kendra

El primer paso es crear un índice de documentos de Amazon Kendra que permita responder a las preguntas de los clientes. El índice proporciona una API de búsqueda para las consultas de los clientes y se crea a partir de documentos fuente. Amazon Kendra devuelve las respuestas que encuentra en los documentos indexados al bot, que se las muestra al agente.

La calidad y precisión de las respuestas sugeridas por Amazon Kendra dependen de los documentos que se indexen. Los documentos deben incluir archivos a los que el agente acceda con frecuencia y deben almacenarse en un bucket de S3. Puede indexar datos no estructurados y semiestructurados en formato .html, Microsoft Office (.doc o .ppt), PDF y texto.

Para crear un índice de Amazon Kendra, consulte Introducción a los buckets de S3 (consola) en la Guía del desarrollador de Amazon Kendra.

Para añadir preguntas y respuestas (FAQ) que respondan a las consultas de los clientes, consulte Adición de preguntas y respuestas en la Guía del desarrollador de Amazon Kendra. Para este tutorial, utilice el archivo ML_FAQ.csv disponible en GitHub.

Paso siguiente

Paso 2: creación de un bot de Amazon Lex

Paso 2: creación de un bot de Amazon Lex

Amazon Lex proporciona una interfaz entre el agente del centro de llamadas y el índice de Amazon Kendra. Realiza un seguimiento de la conversación entre el agente y el cliente y llama a la intención

`AMAZON.KendraSearchIntent` en función de las preguntas que haga el cliente. Una intención es la acción que el usuario desea realizar.

Amazon Kendra busca en los documentos indexados y devuelve una respuesta a Amazon Lex que se muestra en el bot. Solo el agente puede ver esta respuesta.

Creación de un bot de asistencia para agentes

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el panel de navegación, elija Bots.
3. Seleccione Create (Crear).
4. Elija Bot personalizado y configure el bot.
    a. Nombre del bot: introduzca un nombre que indique la finalidad del bot, como **`AgentAssistBot`**.
    b. Voz de salida: elija Ninguna.
    c. Tiempo de espera de la sesión: introduzca **5**.
    d. COPPA: elija No.
5. Seleccione Create (Crear). Después de crear el bot, Amazon Lex muestra la pestaña del editor del bot.

Paso siguiente

Paso 3: adición de intenciones integradas y personalizadas

Paso 3: adición de intenciones integradas y personalizadas

Una intención representa una acción que el agente del centro de llamadas quiere que realice el bot. En este caso, el agente quiere que el bot sugiera respuestas y recursos útiles basados en la conversación entre el agente y el cliente

Amazon Lex cuenta con dos tipos de intenciones; personalizadas e integradas. `AMAZON.KendraSearchIntent`, por ejemplo, es una intención integrada. El bot utiliza la intención `AMAZON.KendraSearchIntent` para consultar el índice y mostrar las sugerencias de respuesta de Amazon Kendra.

El bot de este ejemplo no necesita una intención personalizada. No obstante, para compilar el bot, debe crear al menos una intención personalizada que tenga, como mínimo, un enunciado de ejemplo. Esta intención solo es necesaria para compilar el bot de asistencia para agentes. No realiza ninguna otra función. El enunciado de la intención no debe responder a ninguna de las preguntas que haga el cliente. De esta forma, se evitan llamadas a `AMAZON.KendraSearchIntent` para responder a consultas de los clientes. Para obtener más información, consulte AMAZON.KendraSearchIntent.

Creación de la intención personalizada necesaria

1. En la página Introducción al bot, elija Crear intención.
2. En Agregar intención, elija Crear intención.
3. En el cuadro de diálogo Crear intención, introduzca un nombre que describa la intención, como **`RequiredIntent`**.
4. En Ejemplos de enunciados, introduzca un enunciado descriptivo, como **`Required utterance`**.
5. Elija Guardar intención.

Adición de la intención AMAZON.KendraSearchIntent y del mensaje de respuesta

1. En el panel de navegación, elija el signo más (+) situado junto a Intenciones.
2. Elija Buscar intenciones existentes.
3. En el cuadro de búsqueda Buscar intenciones, introduzca **`AMAZON.KendraSearchIntent`** y seleccione esta intención en la lista.
4. Asigne un nombre descriptivo a la intención, como **`AgentAssistSearchIntent`**, y elija Agregar.
5. En el editor de intenciones, seleccione la Consulta de Amazon Kendra para abrir las opciones de consulta.
6. Seleccione el índice en el que quiera que busque la intención.
7. En la sección Respuesta, agregue los tres mensajes siguientes a un grupo de mensajes.

```
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. Elija Guardar intención.
9. Para crear el bot, elija Crear.

Paso siguiente

Paso 4: configuración de Amazon Cognito

Paso 4: configuración de Amazon Cognito

Para administrar los permisos y los usuarios de la aplicación web, es necesario configurar Amazon Cognito. Amazon Cognito garantiza que la aplicación web sea segura y disponga de control de accesos. Amazon Cognito utiliza grupos de identidades para proporcionar las credenciales de AWS que otorgan acceso a los usuarios a otros servicios de AWS. En el contexto de este tutorial, proporciona acceso a Amazon Lex.

Al crear un grupo de identidades, Amazon Cognito le proporciona los roles de AWS Identity and Access Management (IAM) necesarios para usuarios autenticados y sin autenticar. Para modificar los roles de IAM, agregue políticas que otorguen acceso a Amazon Lex.

Configuración de Amazon Cognito

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Cognito en https://console.aws.amazon.com/cognito/.
2. Seleccione Manage Identity Pools (Administrar grupos de identidades).
3. Elija Create new identity pool.
4. Configure el grupo de identidades.
    a. Nombre del grupo de identidades: introduzca un nombre que indique la finalidad del bot, como **`BotPool`**.
    b. En la sección Identidades sin autenticar, elija Habilitar el acceso a identidades sin autenticar.
5. Elija Create Pool (Crear grupo).
6. En la página Identifique los roles de IAM que se utilizarán en el nuevo grupo de identidades, elija Ver detalles.
7. Anote los nombres de los roles de IAM. Los modificará más adelante.

8. Elija Allow.
9. En la página Introducción a Amazon Cognito, para Plataforma, elija JavaScript.
10. En la sección Obtener credenciales de AWS, busque el ID del grupo de identidades y anótelo.
11. Para permitir el acceso a Amazon Lex, modifique los roles de IAM autenticados y sin autenticar.

    a. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.

    b. En el panel de navegación, en Administración del acceso, elija Roles.

    c. En el cuadro de búsqueda, introduzca el nombre del rol de IAM autenticado y marque la casilla situada al lado del nombre.

        i. Seleccione Attach policies (Asociar políticas).

        ii. En el cuadro de búsqueda, escriba **`AmazonLexRunBotsOnly`** y marque la casilla situada al lado del rol.

        iii. Elija Attach policy (Asociar política).

    d. En el cuadro de búsqueda, introduzca el nombre del rol de IAM sin autenticar y marque la casilla situada al lado del nombre.

        i. Seleccione Attach policies (Asociar políticas).

        ii. En el cuadro de búsqueda, escriba **`AmazonLexRunBotsOnly`** y marque la casilla situada al lado del rol.

        iii. Elija Attach policy (Asociar política).

Paso siguiente

Paso 5: implementación del bot como aplicación web

Paso 5: implementación del bot como aplicación web

Implementación del bot como aplicación web

1. Descargue el repositorio de https://github.com/awsdocs/amazon-lex-developer-guide/blob/master/example_apps/agent_assistance_bot/ en su equipo.
2. Navegue al repositorio descargado y abra el archivo index.html en un editor.
3. Realice los siguientes cambios.

a. En la sección `AWS.config.credentials`, introduzca el nombre de la región y el ID del grupo de identidades.

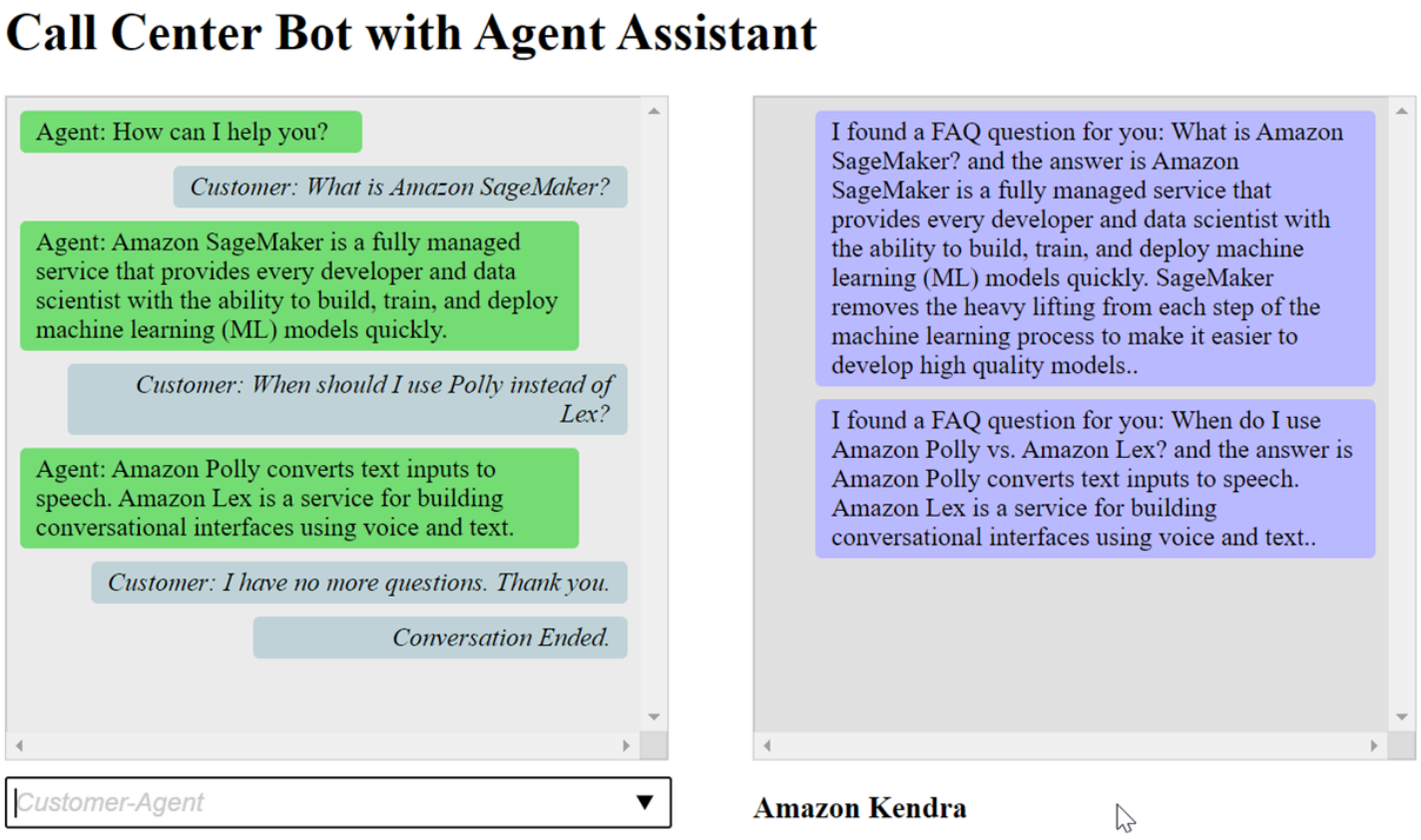
b. En la sección `Amazon Lex runtime parameters`, introduzca el nombre del bot.

c. Guarde el archivo.

Paso 6: uso del bot

Con fines de demostración, tendrá que agregar entradas al bot como cliente y como agente. Para diferenciar entre ambos tipos de entradas, las preguntas que haga el cliente empezarán por “Client:” y las respuestas proporcionadas por el agente, por “Agent:”. Puede elegir varias opciones del menú de sugerencias de entrada.

Abra `index.html` para ejecutar la aplicación web y entablar una conversación con el bot parecida a la de la siguiente imagen:



La función `pushChat()` del archivo index.html se explica a continuación.

```
            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;
```

Cuando proporciona una entrada como cliente, la API en tiempo de ejecución de Amazon Lex la envía a Amazon Lex.

La función `showMessage(daText, senderRequest, displayWindow)` muestra la conversación entre el agente y el cliente en la ventana del chat. Las respuestas que sugiere Amazon Kendra se muestran en una ventana adyacente. La conversación finaliza cuando el cliente dice: **“I have no more questions. Thank you.”**

Nota: Elimine el índice de Amazon Kendra si no lo va a utilizar.

Migración de un bot

La API de Amazon Lex V2 utiliza una arquitectura de información actualizada que permite simplificar el control de versiones de los recursos y admite varios idiomas en un bot. Para obtener más información, consulte la Guía de migración en la Guía para desarrolladores de Amazon Lex V2.

Es necesario migrar un bot para disfrutar de estas nuevas características. El proceso de migración de un bot de Amazon Lex incluye lo siguiente:

- La migración copia las intenciones y los tipos de ranura personalizados al bot de Amazon Lex V2.
- Puede agregar varios idiomas a un mismo bot de Amazon Lex V2. En Amazon Lex V1, es necesario crear un bot para cada idioma. Puede migrar varios bots de Amazon Lex V1, cada uno de ellos en un idioma distinto, a un solo bot de Amazon Lex V2.
- Amazon Lex asigna las intenciones y los tipos de ranura integrados de Amazon Lex V1 a las intenciones y los tipos de ranura integrados de Amazon Lex V2. Si no se puede migrar un recurso integrado, Amazon Lex le muestra un mensaje con los siguientes pasos.

Los siguientes recursos quedan excluidos del proceso de migración:

- Alias
- Índices de Amazon Kendra
- Funciones de AWS Lambda
- La configuración del registro de conversaciones
- Canales de mensajería como Slack
- Etiquetas

Para migrar un bot, el usuario o el rol deben tener permisos de IAM para Amazon Lex y las operaciones de la API de Amazon Lex V2. Para conocer los permisos necesarios, consulte Permitir a un usuario migrar un bot a las API de Amazon Lex V2.

Migración de un bot (consola)

Utilice la consola de Amazon Lex V1 para migrar la estructura de un bot a un bot de Amazon Lex V2.

Uso de la consola para migrar un bot a la API de Amazon Lex V2

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el menú de la izquierda, elija Herramienta de migración.
3. Seleccione el bot que desea migrar en la lista de bots y, a continuación, elija Migrar.
4. Elija la versión del bot que desea migrar e introduzca el nombre del bot destinatario de la migración. Si introduce el nombre de un bot de Amazon Lex V2 existente, el bot de Amazon Lex V1 se migra en el idioma que se muestra en los detalles y sobreescribe la versión de borrador del idioma.
5. Elija Next (Siguiente).
6. Elija el rol de IAM que Amazon Lex utilizará para ejecutar la versión de la API de Amazon Lex V2 del bot. Puede o bien crear un nuevo rol con los permisos estrictamente necesarios para ejecutar el bot o bien elegir un rol de IAM existente.
7. Elija Next (Siguiente).
8. Revise la configuración de la migración. Si todo está correcto, elija Comenzar la migración.

Cuando empiece el proceso de migración, vuelve a aparecer la página de inicio de la herramienta de migración. Puede supervisar el progreso de la migración desde la tabla Historial. Si en la columna Estado de migración aparece Completada, eso significa que la migración ha finalizado.

Amazon Lex utiliza la operación `StartImport` de la API de Amazon Lex V2 para importar el bot migrado. Se mostrará una entrada en la tabla del historial de importación de la consola Amazon Lex V2 para cada migración.

Durante la migración, es posible que Amazon Lex encuentre recursos del bot que no se pueden migrar. Verá un mensaje de error o de advertencia para cada recurso que no se pueda migrar. Cada mensaje incluye un enlace a la documentación con información sobre cómo solucionar el problema.

Migración de una función de Lambda

Las funciones de Lambda se definen de forma distinta para un bot en Amazon Lex V2. Solo se permite una función de Lambda en un alias para cada idioma de un bot. Para obtener más información acerca de la migración de funciones de Lambda, consulte Migración de una función de Lambda de Amazon Lex V1 a Amazon Lex V2.

Mensajes de migración

Durante la migración, Amazon Lex encontrará recursos como tipos de ranura integrados que no puede migrar al recurso equivalente de Amazon Lex V2. Cuando ocurre esto, Amazon Lex devuelve un mensaje de migración que describe lo sucedido e incluye un enlace a la documentación, en el que se explica cómo corregir el problema de migración. En las siguientes secciones se describen los problemas que pueden surgir mientras se migra un bot y cómo solucionarlos.

Temas

Intención integrada

Cuando utiliza una intención integrada que no se admite en Amazon Lex V2, la intención se asigna a una intención personalizada en el bot de Amazon Lex V2. La intención personalizada no contiene enunciados. Deberá agregar ejemplos de enunciados para seguir usando la intención.

Tipo de ranura integrado

No se asignará ningún valor de tipo de ranura a las ranuras migradas que utilicen un tipo de ranura no admitido en Amazon Lex V2. Para utilizar esta ranura:

- cree un tipo de ranura personalizado,
- agregue los valores de tipo de ranura que se esperan para ese tipo de ranura,
- actualice la ranura para utilizar el nuevo tipo de ranura personalizado.

Registros de conversaciones

La migración no actualiza la configuración del registro de conversaciones del bot de Amazon Lex V2.

Configuración de registros de conversaciones

1. Abra la consola de Amazon Lex V2 en https://console.aws.amazon.com/lexv2.
2. En la lista de bots, seleccione el bot para el que desea configurar los registros de conversaciones.
3. En el menú de la izquierda, seleccione Alias y, a continuación, seleccione un alias de la lista.
4. En la sección Registros de conversaciones, elija Administrar registros de conversación para configurar los registros de conversaciones del alias del bot.

Grupos de mensajes

Amazon Lex V2 solo admite un mensaje y dos mensajes alternativos por cada grupo de mensajes. Si un bot de Amazon Lex V1 cuenta con más de tres mensajes por grupo de mensajes, solo se migrarán los tres primeros mensajes. Si desea incluir más mensajes en un grupo de mensajes, utilice una función de Lambda para generar varios mensajes.

Solicitudes y frases

Amazon Lex V2 utiliza un mecanismo diferente para las preguntas de seguimiento, aclaratorias y de cierre.

En el caso de las preguntas de seguimiento, se utiliza el contexto acumulado para cambiar a una intención diferente una vez cumplida la intención.

Por ejemplo, supongamos que tiene la intención de reservar un coche de alquiler que está configurado para devolver un contexto de salida denominado `book_car_fulfilled`. Cuando se cumple la intención, Amazon Lex establece la variable del contexto de salida en `book_car_fulfilled`. Dado que `book_car_fulfilled` es un contexto activo, se tiene en cuenta el reconocimiento de una intención con el contexto `book_car_fulfilled` establecido como contexto de entrada, siempre y cuando el enunciado del usuario se reconozca como un intento de obtener esa intención. Puede usarlo para fines que solo tengan sentido después de reservar un vehículo, como enviar un recibo por correo electrónico o modificar una reserva.

Amazon Lex V2 no admite preguntas aclaratorias y frases de cierre (afirmaciones de anulación). Los bots de Amazon Lex V2 contienen una intención alternativa predeterminada que se invoca si no se asocia ninguna intención. Para enviar una pregunta aclaratoria con reintentos, configure una función de Lambda y habilite el enlace al código de diálogo en la intención alternativa. La función de Lambda

puede generar una pregunta aclaratoria como respuesta y el valor de reintento en un atributo de la sesión. Si el valor de reintento supera el número máximo de reintentos, puede enviar una frase de cierre para acabar la conversación.

Otras características de Amazon Lex V1

La herramienta de migración solo admite la migración de bots de Amazon Lex V1 y sus intenciones, tipos de ranura y ranuras subyacentes. Para obtener información acerca de otras características, consulte los siguientes temas de la documentación de Amazon Lex V2.

- Alias de bots: Alias
- Canales de bots: Implementación de un bot de Amazon Lex V2 en una plataforma de mensajería
- Configuración del registro de conversaciones: Supervisión con registros de conversaciones
- Índices de Amazon Kendra: AMAZON.KendraSearchIntent
- Funciones de Lambda: Uso de una función de AWS Lambda
- Etiquetas: Etiquetado de recursos

Migración de una función de Lambda de Amazon Lex V1 a Amazon Lex V2

Amazon Lex V2 solo admite una función de Lambda para cada idioma en un bot. La función de Lambda y sus valores se configuran para el alias del bot que se utiliza en tiempo de ejecución.

La función de Lambda se invoca para todas las intenciones en el idioma correspondiente si los enlaces de código de diálogo y de cumplimiento están habilitados para la intención.

Las funciones de Lambda de Amazon Lex V2 tienen un formato de mensaje de entrada y de salida diferente al de Amazon Lex V1. A continuación se indican las diferencias en lo que respecta al formato de entrada de las funciones de Lambda.

- Amazon Lex V2 reemplaza las estructuras `currentIntent` y `alternativeIntents` por la estructura `interpretations`. Cada interpretación contiene una intención, la puntuación de confianza de NLU para dicha intención y un análisis de opiniones opcional.
- Amazon Lex V2 mueve los objetos `activeContexts` y `sessionAttributes` de Amazon Lex V1 a la estructura `sessionState` unificada. Esta estructura proporciona información acerca del estado actual de la conversación, incluido el ID de la solicitud de origen.

- Amazon Lex V2 no devuelve `recentIntentSummaryView`. En su lugar, utilice la información de la estructura `sessionState`.
- La entrada de Amazon Lex V2 proporciona `botId` y `localeId` en el atributo `bot`.
- La estructura de entrada contiene un atributo `inputMode` que proporciona información acerca del tipo de entrada, que puede ser texto, voz o DTMF.

A continuación se indican las diferencias en lo que respecta al formato de salida de las funciones de Lambda.

- Las estructuras `activeContexts` y `sessionAttributes` de Amazon Lex V1 se reemplazan por la estructura `sessionState` en Amazon Lex V2.
- `recentIntentSummaryView` no se incluye en el resultado.
- La estructura `dialogAction` de Amazon Lex V1 se divide en dos estructuras: `dialogAction`, que forma parte de la estructura `sessionState`, y `messages`, que se requiere cuando `dialogAction.type` es `ElicitIntent`. Amazon Lex selecciona los mensajes de esta estructura para mostrárselos al usuario.

Cuando crea un bot con las API de Amazon Lex V2, solo puede haber una función de Lambda en cada alias de bot por idioma, en vez de una función de Lambda para cada intención. Si desea continuar usando funciones distintas, puede crear una función de enrutamiento que active una función distinta para cada intención. A continuación se muestra un ejemplo de función de enrutamiento que puede utilizar o modificar en su aplicación.

```
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 actualizados

En las siguientes tablas se incluye información detallada sobre los campos actualizados en el formato de solicitud y respuesta de Lambda para Amazon Lex V2. Utilice las tablas para asignar campos entre versiones.

Solicitud

Los campos siguientes se han actualizado en el formato de solicitud de las funciones de Lambda.

Contextos activos

La estructura `activeContexts` ahora forma parte de la estructura `sessionState`.

Estructura de V1	Estructura de V2
activeContexts	sessionState.activeContexts
activeContexts[*].timeToLive	sessionState.activeContexts[*].timeToLive
activeContexts[*].timeToLive.timeToLiveInSeconds	sessionState.activeContexts[*].timeToLive.timeToLiveInSeconds
activeContexts[*].timeToLive.turnsToLive	sessionState.activeContexts[*].timeToLive.turnsToLive
activeContexts[*].name	sessionState.activeContexts[*].name
activeContexts[*].parameters	sessionState.activeContexts[*].contextAttributes

Intenciones alternativas

La lista de interpretaciones del índice 1 al N contiene la lista de intenciones alternativas que Amazon Lex V2 ha predicho, junto con las puntuaciones de confianza correspondientes. `recentIntentSummaryView` se ha eliminado de la estructura de solicitud en Amazon Lex V2. Para obtener información acerca de `recentIntentSummaryView`, utilice la operación GetSession.

Estructura de V1	Estructura de V2
alternativeIntents	interpretations[1:*]
recentIntentSummaryView	N/A

Bot

En Amazon Lex V2, los bots y los alias cuentan con identificadores. El ID del bot forma parte de la entrada del enlace de código. Se incluye el ID del alias, pero no su nombre. Amazon Lex V2 admite varias configuraciones regionales para el mismo bot, por lo que se incluye el ID de la configuración regional.

Estructura de V1	Estructura de V2
bot	bot
bot.name	bot.name
N/A	bot.id
bot.alias	N/A
N/A	bot.aliasId
bot.version	bot.version
N/A	bot.localeId

Intención actual

La estructura `sessionState.intent` contiene los detalles de la intención activa. Amazon Lex V2 también devuelve una lista con todas las intenciones, incluidas intenciones alternativas, en la estructura `interpretations`. El primer elemento de la lista de interpretaciones coincide con el primer elemento de `sessionState.intent`.

Estructura de V1	Estructura de V2
currentIntent	sessionState.intent O interpretations[0].intent
currentIntent.name	sessionState.intent.name O interpretations[0] .intent.name
currentIntent.nluConfidenceScore	interpretations[0].nluConfidence.score

Acción de diálogo

El campo `confirmationStatus` ahora forma parte de la estructura `sessionState`.

Estructura de V1	Estructura de V2
currentIntent.confirmationStatus	sessionState.intent.confirmationState O interpretations[0].intent.confirmationState
N/A	sessionState.intent.state O interpretations[*] .intent.state

Amazon Kendra

El campo `kendraResponse` ahora forma parte de las estructuras `sessionState` y `interpretations`.

Estructura de V1	Estructura de V2
kendraResponse	sessionState.intent.kendraResponse O interpretations[0].intent.kendraResponse

Opinión

La estructura `sentimentResponse` se mueve a la nueva estructura `interpretations`.

Estructura de V1	Estructura de V2
sentimentResponse	interpretations[0].sentimentResponse
sentimentResponse.sentimentLabel	interpretations[0].sentimentResponse.sentiment
sentimentResponse.sentimentScore	interpretations[0].sentimentResponse.sentimentScore

Slots

Amazon Lex V2 proporciona un solo objeto `slots` en el interior de la estructura `sessionState.intent` que contiene los valores resueltos, el valor interpretado y el valor original de lo que ha dicho el usuario. Amazon Lex V2 también admite ranuras con múltiples valores si se establece `slotShape` como `List` y si se configura la lista `values`. El campo `value` admite ranuras con un solo valor y se asume que su forma es `Scalar`.

Estructura de V1	Estructura de V2
currentIntent.slots	sessionState.intent.slots O interpretations[0].intent.slots
currentIntent.slots[*].value	sessionState.intent.slots[*].value.interpretedValue O interpretations[0].intent.slots[*].value.interpretedValue
N/A	sessionState.intent.slots[*].value.shape O interpretations[0].intent.slots[*].shape
N/A	sessionState.intent.slots[*].values O interpretations[0].intent.slots[*].values
currentIntent.slotDetails	sessionState.intent.slots O interpretations[0].intent.slots

Estructura de V1	Estructura de V2
currentIntent.slotDetails[*].resolutions	sessionState.intent.slots[*].resolvedValues O interpretations[0].intent.slots[*].resolvedValues
currentIntent.slotDetails[*].originalValue	sessionState.intent.slots[*].originalValue O interpretations[0].intent.slots[*].originalValue

Otros

El campo `sessionId` de Amazon Lex V2 es el mismo que el campo `userId` de Amazon Lex V1. Amazon Lex V2 también envía el objeto `inputMode` de la persona que llama: texto, DTMF o voz.

Estructura de V1	Estructura de V2
userId	sessionId
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion
sessionAttributes	sessionState.sessionAttributes
requestAttributes	requestAttributes
N/A	inputMode
N/A	originatingRequestId

Respuesta

Los campos siguientes se han modificado en el formato de mensaje de respuesta de las funciones de Lambda.

Contextos activos

La estructura `activeContexts` se mueve a la estructura `sessionState`.

Estructura de V1	Estructura de V2
activeContexts	sessionState.activeContexts
activeContexts[*].timeToLive	sessionState.activeContexts[*].timeToLive
activeContexts[*].timeToLive.timeToLiveInSeconds	sessionState.activeContexts[*].timeToLive.timeToLiveInSeconds
activeContexts[*].timeToLive.turnsToLive	sessionState.activeContexts[*].timeToLive.turnsToLive
activeContexts[*].name	sessionState.activeContexts[*].name
activeContexts[*].parameters	sessionState.activeContexts[*].contextAttributes

Acción de diálogo

La estructura `dialogAction` se mueve a la estructura `sessionState`. Ahora puede especificar varios mensajes en una acción de diálogo. La estructura `genericAttachments` es ahora la estructura `imageResponseCard`.

Estructura de V1	Estructura de 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

Estructura de V1	Estructura de V2
dialogAction.message.content	messages[*].content
dialogAction.responseCard	messages[*].imageResponseCard
dialogAction.responseCard.version	N/A
dialogAction.responseCard.contentType	messages[*].contentType
dialogAction.responseCard.genericAttachments	N/A
dialogAction.responseCard.genericAttachments[*].title	messages[*].imageResponseCard.title
dialogAction.responseCard.genericAttachments[*].subTitle	messages[*].imageResponseCard.subtitle
dialogAction.responseCard.genericAttachments[*].imageUrl	messages[*].imageResponseCard.imageUrl
dialogAction.responseCard.genericAttachments[*].buttons	messages[*].imageResponseCard.buttons
dialogAction.responseCard.genericAttachments[*].buttons[*].value	messages[*].imageResponseCard.buttons[*].value
dialogAction.responseCard.genericAttachments[*].buttons[*].text	messages[*].imageResponseCard.buttons[*].text
dialogAction.kendraQueryRequestPayload	dialogAction.kendraQueryRequestPayload
dialogAction.kendraQueryFilterString	dialogAction.kendraQueryFilterString

Intenciones y ranuras

Los campos de intención y ranura que formaban parte de la estructura `dialogAction` ahora forman parte de la estructura `sessionState`.

Estructura de V1	Estructura de V2
dialogAction.intentName	sessionState.intent.name
dialogAction.slots	sessionState.intent.slots
dialogAction.slots[*].key	sessionState.intent.slots[*].key
dialogAction.slots[*].value	sessionState.intent.slots[*].value.interpretedValue
N/A	sessionState.intent.slots[*].value.shape
N/A	sessionState.intent.slots[*].values

Otros

La estructura `sessionAttributes` ahora forma parte de la estructura `sessionState`. La estructura `recentIntentSummaryReview` se ha eliminado.

Estructura de V1	Estructura de V2
sessionAttributes	sessionState.sessionAttributes
recentIntentSummaryView	N/A

Seguridad en Amazon Lex

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información sobre los programas de conformidad que se aplican a Amazon Lex, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación lo ayudará a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon Lex. En los siguientes temas, se mostrará cómo configurar Amazon Lex para satisfacer sus objetivos de seguridad y conformidad. También puede aprender a utilizar otros servicios de AWS que le ayudan a monitorizar y proteger sus recursos de Amazon Lex.

Temas

Protección de los datos en Amazon Lex

Amazon Lex recopila el contenido del cliente para la resolución de problemas y ayuda a mejorar el servicio. El contenido del cliente está protegido de forma predeterminada. Puede eliminar el contenido de clientes individuales mediante la API de Amazon Lex.

Amazon Lex almacena cuatro tipos de contenido:

- Enunciados de muestra, que se utilizan para crear y capacitar a un bot
- Enunciados de los clientes, procedentes de los usuarios que interactúan con el bot
- Atributos de sesión, que ofrecen información específica de la aplicación durante la interacción de un usuario con un bot
- Atributos de solicitud, que contienen información que se aplica a una única solicitud realizada a un bot

Cualquier bot de Amazon Lex que esté diseñado para su uso por menores se rige por la Ley de Protección de la Privacidad Online de menores (COPPA, por sus siglas en inglés). Puede indicar a Amazon Lex que el bot está sujeto a COPPA mediante la consola o la API de Amazon Lex para establecer el campo `childDirected` en `true`. Cuando el campo `childDirected` se establece en `true`, no se almacena ningún enunciado de usuario.

Temas

Cifrado en reposo

Amazon Lex cifra los enunciados de usuario que almacena.

Temas

Enunciados de muestra

Cuando desarrolla un bot, puede proporcionar enunciados de muestra para cada intención y slot. También puede proporcionar sinónimos y valores personalizados para los slots. Esta información se utiliza para compilar el bot y crear la experiencia de usuario.

Enunciados de los clientes

Amazon Lex cifra los enunciados que los usuarios envían al bot a menos que el campo `childDirected` se establezca en `true`.

Cuando el campo `childDirected` se establece en `true`, no se almacena ningún enunciado de usuario.

Cuando el campo `childDirected` se establece en `false` (valor predeterminado), los enunciados de usuario se cifran y almacenan durante 15 días para utilizarlos con la operación GetUtterancesView. Para eliminar los enunciados almacenados de un usuario específico, utilice la operación DeleteUtterances.

Cuando el bot acepta entrada de voz, la entrada se almacena de forma indefinida. Amazon Lex la utiliza para mejorar la capacidad de respuesta del bot a las entradas del usuario.

Utilice la operación DeleteUtterances para eliminar los enunciados almacenados de un usuario específico.

Atributos de sesión

Los atributos de la sesión contienen información específica de la aplicación que se transfiere entre Amazon Lex y las aplicaciones cliente. Amazon Lex transfiere los atributos de sesión a todas AWS Lambda las funciones configuradas para un bot. Si una función de Lambda añade o actualiza los atributos de sesión, Amazon Lex devuelve la nueva información a la aplicación cliente.

Los atributos de sesión se mantienen en un almacén cifrado durante la sesión. Puede configurar la sesión para que permanezca activa desde un mínimo de 1 minuto hasta un máximo de 24 horas después del último enunciado de usuario. La duración predeterminada de la sesión es de 5 minutos.

Atributos de solicitud

Los atributos de solicitud contienen información específica de solicitud y se aplican únicamente a la solicitud actual. Una aplicación cliente utiliza los atributos de solicitud para enviar información a Amazon Lex en tiempo de ejecución.

Utilice los atributos de solicitud para pasar información que no tiene por qué persistir durante toda la sesión. Dado que los atributos de solicitud no se mantienen entre solicitudes, no se almacenan.

Cifrado en tránsito

Amazon Lex utiliza el protocolo HTTPS para comunicarse con la aplicación cliente. Utiliza firmas HTTPS y AWS para comunicarse con otros servicios, como Amazon Polly, y AWS Lambda en nombre de su aplicación.

Administración de claves

Amazon Lex protege su contenido del uso no autorizado con claves internas.

Identity and Access Management para Amazon Lex

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon Lex. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que realice en Amazon Lex.

Usuario de servicio: si utiliza el servicio Amazon Lex para realizar el trabajo, el administrador proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Amazon Lex para realizar el trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica de Amazon Lex, consulte Solución de problemas de identidad y acceso de Amazon Lex.

Administrador de servicio: si está a cargo de los recursos de Amazon Lex de su empresa, probablemente tenga acceso completo a Amazon Lex. Su trabajo consiste en determinar a qué características y recursos de Amazon Lex deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información acerca de cómo la empresa puede utilizar IAM con Amazon Lex, consulte Cómo funciona Amazon Lex con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que desee obtener información sobre cómo escribir políticas para administrar el acceso a Amazon Lex. Para consultar ejemplos de políticas de Amazon Lex basadas en identidades que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidades para Amazon Lex.

Autenticación con identidades

La autenticación es la forma de iniciar sesión para AWS usar sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus

credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWSen la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué

pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .

- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, en algunos casos Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para conocer la diferencia entre las funciones y las políticas basadas en recursos para el acceso entre cuentas, consulte el tema sobre el acceso a los recursos entre cuentas en IAM en la Guía del usuario de IAM.
- Acceso entre servicios: algunos utilizan funciones en otros. Servicios de AWS Servicios de AWS Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations .
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon Lex con IAM

Antes de utilizar IAM para administrar el acceso a Amazon Lex, obtenga información sobre qué características de IAM se encuentran disponibles con Amazon Lex.

Características de IAM que puede utilizar con Amazon Lex

Característica de IAM	Soporte de Amazon Lex V2
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACL	No
ABAC (etiquetas en políticas)	Parcial
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	Sí
Roles vinculados al servicio	Sí

Para obtener una visión general de cómo funcionan Amazon Lex y otros AWS servicios con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Políticas de Amazon Lex basadas en identidades

Compatibilidad con las políticas basadas en identidad	Sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está adjunto. Para más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades para Amazon Lex

Para ver ejemplos de políticas basadas en identidad de Amazon Lex, consulte Ejemplos de políticas basadas en identidades para Amazon Lex.

Políticas de Amazon Lex basadas en recursos

Compatibilidad con las políticas basadas en recursos	No

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los directores pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para obtener más información, consulte el tema Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Acciones de políticas para Amazon Lex

Admite acciones de política	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de las acciones de Amazon Lex, consulte Acciones definidas por Amazon Lex en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Amazon Lex utilizan el siguiente prefijo antes de la acción:

```
lex
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "lex:action1",
      "lex:action2"
```

```
        ]
```

Puede utilizar caracteres comodín (*) para especificar varias acciones . Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "lex:Describe*"
```

Recursos de políticas para Amazon Lex

Admite recursos de políticas	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Un ARN de un recurso de bot de Amazon Lex tiene el siguiente formato.

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

Para obtener más información sobre el formato de los ARN, consulte Nombres de recursos de Amazon (ARN) y espacios de nombres de AWS servicio.

Por ejemplo, para especificar el bot de `OrderFlowers` en su instrucción, utilice el siguiente ARN.

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:OrderFlowers"
```

Para especificar todos los bots que pertenecen a una cuenta específica, utilice el carácter comodín (*).

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:*"
```

Algunas acciones de Amazon Lex, como las que se utilizan para crear recursos, no se pueden llevar a cabo en un recurso específico. En dichos casos, debe utilizar el carácter comodín, (*).

```
"Resource": "*"
```

Para ver una lista de tipos de recursos de Amazon Lex y sus ARN, consulte Tipos de recurso definidos por Amazon Lex en la Referencia de autorizaciones de servicio. Para obtener información acerca de las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por Amazon Lex.

Claves de condiciones de políticas para Amazon Lex

Admite claves de condición de políticas específicas del servicio	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado

con su nombre de usuario de IAM. Para más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de las claves de condición de Amazon Lex, consulte Claves de condición para Amazon Lex en la Referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por Amazon Lex.

En la siguiente tabla se enumeran las claves de condición de Amazon Lex que se aplican a los recursos de Amazon Lex. Puede incluir estas claves en los elementos `Condition` de una política de permisos de IAM.

ACL en Amazon Lex

Admite las ACL	No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Amazon Lex

Admite ABAC (etiquetas en las políticas)	Parcial

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte ¿Qué es ABAC? en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Puede asociar etiquetas con determinados tipos de recursos de Amazon Lex para la autorización. Para controlar el acceso utilizando etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `lex:ResourceTag/${TagKey}`, `aws:RequestTag/${TagKey}` o `aws:TagKeys`.

Para obtener más información acerca del etiquetado de recursos de Amazon Lex, consulte Etiquetado de los recursos de Amazon Lex.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Usar una etiqueta para acceder a un recurso.

En la tabla siguiente se enumeran las acciones y los tipos de recursos correspondientes para el control de acceso basado en etiquetas. Cada acción se autoriza en función de las etiquetas asociadas al tipo de recurso correspondiente.

Acción	Tipo de recurso	Claves de condición	Notas
CreateBotVersion	bot	`lex:ResourceTag`	
DeleteBot	bot	`lex:ResourceTag`	
DeleteBotAlias	alias	`lex:ResourceTag`	
DeleteBotChannelAssociation	channel	`lex:ResourceTag`	
DeleteBotVersion	bot	`lex:ResourceTag`	

Acción	Tipo de recurso	Claves de condición	Notas
DeleteSession	bot o alias	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando el alias se define en `$LATEST`. Utiliza etiquetas asociadas al alias especificado cuando se utiliza con otros alias.
DeleteUtterances	bot	`lex:ResourceTag`	
GetBot	bot o alias	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando `versionOrAlias` se define en `$LATEST` o versión numérica. Utiliza etiquetas asociadas al alias especificado cuando se utiliza con alias
GetBotAlias	alias	`lex:ResourceTag`	
GetBotChannelAssociation	canal	`lex:ResourceTag`	
GetBotChannelAssociations	canal	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando el alias se define en «-». Utiliza etiquetas asociadas al alias especificado cuando se especifica un alias bot

Acción	Tipo de recurso	Claves de condición	Notas
GetBotVersions	bot	`lex:ResourceTag`	
GetExport	bot	`lex:ResourceTag`	
GetSession	bot o alias	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando el alias se define en `$LATEST`. Utiliza etiquetas asociadas al alias especificado cuando se utiliza con otros alias.
GetUtterancesView	bot	`lex:ResourceTag`	
ListTagsForResource	bot, alias o canal	`lex:ResourceTag`	
PostContent	bot o alias	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando el alias se define en `$LATEST`. Utiliza etiquetas asociadas al alias especificado cuando se utiliza con otros alias.

Acción	Tipo de recurso	Claves de condición	Notas
PostText	bot o alias	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando el alias se define en `$LATEST`. Utiliza etiquetas asociadas al alias especificado cuando se utiliza con otros alias.
PutBot	bot	`lex:ResourceTag, aws:RequestTag, aws:TagKeys`	
PutBotAlias	alias	`lex:ResourceTag, aws:RequestTag, aws:TagKeys`	
PutSession	bot o alias	`lex:ResourceTag`	Utiliza etiquetas asociadas con el bot cuando el alias se define en `$LATEST`. Utiliza etiquetas asociadas al alias especificado cuando se utiliza con otros alias.

Acción	Tipo de recurso	Claves de condición	Notas
StartImport	bot	lex:ResourceTag	Se basa en la política de acceso para la operación PutBot. Se omiten las etiquetas y los permisos específicos de la operación StartImport .
TagResource	bot, alias o canal	lex:ResourceTag, aws:RequestTag, aws:TagKeys	
UntagResource	bot, alias o canal	lex:ResourceTag, aws:RequestTag, aws:TagKeys	

Uso de credenciales temporales con Amazon Lex

Compatible con el uso de credenciales temporales	Sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluida la información sobre cuáles Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para más

información sobre el cambio de roles, consulte Cambio a un rol (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para más información, consulte Credenciales de seguridad temporales en IAM.

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Las credenciales de seguridad temporales se obtienen llamando a operaciones de la AWS STS API, como AssumeRoleo GetFederationToken.

Permisos de entidades principales entre servicios de Amazon Lex

Admite Forward access sessions (FAS)	Sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en AWSél, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Roles de servicio para Amazon Lex

Compatible con roles de servicio	Sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Amazon Lex. Edite los roles de servicio solo cuando Amazon Lex proporcione orientación para hacerlo.

Elección de un rol de IAM en Amazon Lex

Amazon Lex utiliza roles vinculados a servicios para llamar a Amazon Comprehend y a Amazon Polly. Utiliza permisos a nivel de recursos en sus AWS Lambda funciones para invocarlos.

Debe proporcionar un rol de IAM para habilitar el etiquetado de conversaciones. Para obtener más información, consulte Creación de un rol de IAM y políticas para registros de conversación.

Roles vinculados a servicios para Amazon Lex

Compatible con roles vinculados al servicio	Sí

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para obtener más información sobre cómo crear o administrar roles vinculados a servicios de Amazon Lex, consulte Uso de roles vinculados a servicios para Amazon Lex.

Ejemplos de políticas basadas en identidades para Amazon Lex

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Amazon Lex. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS la API. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

A fin de obtener más información sobre las acciones y los tipos de recursos definidos por Amazon Lex, incluido el formato de los ARN para cada tipo de recurso, consulte Acciones, recursos y claves de condición para Amazon Lex en la Referencia de autorizaciones de servicio.

Temas

- Prácticas recomendadas sobre las políticas
- Uso de la consola de Amazon Lex
- Cómo permitir a los usuarios consultar sus propios permisos
- Eliminar todos los bots de Amazon Lex
- Permitir a un usuario migrar un bot a las API de Amazon Lex V2
- Usar una etiqueta para acceder a un recurso

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon Lex de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS

CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utilice el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Política de validación de Analizador de acceso de IAM en la Guía de usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para más información, consulte Configuración del acceso a una API protegido por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de Amazon Lex

Para acceder a la consola de Amazon Lex, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de Amazon Lex que tiene en su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No necesita conceder permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

AWS aborda muchos casos de uso comunes al proporcionar políticas de IAM independientes que son creadas y administradas por. AWS Estas políticas se denominan políticas administradas por AWS. Con las políticas administradas por AWS podrá asignar de forma más sencilla los permisos adecuados a los usuarios, grupos y roles que si tuviera que escribir políticas. Para más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

Las siguientes políticas AWS gestionadas, que puede adjuntar a los grupos y funciones de su cuenta, son específicas de Amazon Lex:

- AmazonLexReadOnly— Otorga acceso de solo lectura a los recursos de Amazon Lex.
- AmazonLexRunBotsSolo: otorga acceso para ejecutar bots conversacionales de Amazon Lex.
- AmazonLexFullAccess— Otorga acceso completo para crear, leer, actualizar, eliminar y ejecutar todos los recursos de Amazon Lex. También concede la capacidad de asociar las funciones de Lambda cuyo nombre comience por `AmazonLex` a intenciones de Amazon Lex.

> **Note**
>
> Para consultar estas políticas de permisos, inicie sesión en la consola de IAM y busque las políticas específicas.

La AmazonLexFullAccesspolítica no concede al usuario permiso para utilizar la `KendraSearchIntent` intención de consultar un índice de Amazon Kendra. Para consultar un índice, debe agregar permisos adicionales a la política. Para conocer los permisos necesarios, consulte Política de IAM para Amazon Kendra Search.

También puede crear sus propias políticas de IAM personalizadas con el fin de conceder permisos para realizar acciones de la API de Amazon Lex. Puede asociar estas políticas personalizadas a los roles de IAM o grupos que requieran esos permisos.

Para obtener información acerca de las políticas administradas de AWS para Amazon Lex, consulte Políticas administradas de AWS para Amazon Lex.

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la AWS CLI API o. 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": "*"
        }
    ]
}
```

Eliminar todos los bots de Amazon Lex

Esta política de ejemplo concede a un usuario de su cuenta de AWS permiso para eliminar cualquier bot de la cuenta.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

Permitir a un usuario migrar un bot a las API de Amazon Lex V2

La siguiente política de permisos de IAM permite a un usuario empezar a migrar un bot de Amazon Lex a las API de Amazon Lex V2 y ver la lista de migraciones y su progreso.

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

Usar una etiqueta para acceder a un recurso

Esta política de ejemplo concede permiso a un usuario o rol de su cuenta de AWS para utilizar la operación `PostText` con cualquier recurso etiquetado con la clave **`Department`** y el valor **`Support`**.

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

```
    "Statement": [
        {
            "Action": "lex:PostText",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "lex:ResourceTag/Department": "Support"
                }
            }
        }
    ]
}
```

Políticas administradas de AWS para Amazon Lex

Una política administrada de AWS es una política independiente que AWS crea y administra. Las políticas administradas de AWS se diseñan para ofrecer permisos para muchos casos de uso comunes, por lo que puede empezar a asignar permisos a los usuarios, grupos y roles.

Tenga presente que es posible que las políticas administradas de AWS no concedan permisos de privilegio mínimo para los casos de uso concretos, ya que están disponibles para que las utilicen todos los clientes de AWS. Se recomienda definir políticas administradas por el cliente para los casos de uso a fin de reducir aún más los permisos.

No puede cambiar los permisos definidos en las políticas administradas por AWS. Si AWS actualiza los permisos definidos en una política administrada de AWS, la actualización afecta a todas las identidades de entidades principales (usuarios, grupos y roles) a las que está adjunta la política. Lo más probable es que AWS actualice una política administrada de AWS cuando se lance un nuevo Servicio de AWS o las operaciones de la API nuevas estén disponibles para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Política administrada de AWS: AmazonLexReady Only

Puede adjuntar la política `AmazonLexReadOnly` a las identidades de IAM.

Esta política concede permisos de solo lectura que permiten a los usuarios ver todas las acciones del servicio de creación de modelos Amazon Lex V2 y Amazon Lex.

Detalles sobre los permisos

Esta política incluye los siguientes permisos:

- `lex`: acceso de solo lectura a los recursos de Amazon Lex V2 y Amazon Lex en el servicio de creación 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 administrada de AWS: AmazonlexRbotsOnly

Puede adjuntar la política `AmazonLexRunBotsOnly` a las identidades de IAM.

Esta política concede permisos de solo lectura que permiten el acceso para ejecutar los bots conversacionales de Amazon Lex V2 y Amazon Lex.

Detalles sobre los permisos

Esta política incluye los siguientes permisos:

- `lex`: acceso de solo lectura a todas las acciones de tiempo de ejecución de Amazon Lex V2 y Amazon Lex.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

Política administrada de AWS: AmazonLexFullAccess

Puede adjuntar la política `AmazonLexFullAccess` a las identidades de IAM.

Esta política concede permisos administrativos que permiten al usuario crear, leer, actualizar y eliminar recursos de Amazon Lex V2 y Amazon Lex, y ejecutar bots conversacionales de Amazon Lex V2 y Amazon Lex.

Detalles sobre los permisos

Esta política incluye los siguientes permisos:

- `lex`: permite a las entidades principales obtener acceso de lectura y escritura a todas las acciones de los servicios de creación de modelos y tiempo de ejecución de Amazon Lex V2 y Amazon Lex.
- `cloudwatch`: permite que las entidades principales vean las alarmas y métricas de Amazon CloudWatch.
- `iam`: permite a las entidades principales crear y eliminar funciones vinculadas a servicios, transferir funciones y adjuntar y desvincular políticas a un rol. Los permisos están restringidos a «lex.amazonaws.com» para las operaciones de Amazon Lex y a «lexv2.amazonaws.com» para las operaciones de Amazon Lex V2.
- `kendra`: permite a las entidades principales enumerar índices de Amazon Kendra.
- `kms`: permite a las entidades principales describir claves y alias de AWS KMS.

- `lambda`: permite a las entidades principales enumerar las funciones de AWS Lambda y gestionar los permisos asociados a cualquier función de Lambda.
- `polly`: permite a las entidades principales describir las voces de Amazon Polly y sintetizar el habla.

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

Actualizaciones de Amazon Lex a políticas administradas por AWS

Es posible consultar los detalles sobre las actualizaciones de las políticas administradas por AWS para Amazon Lex desde que este servicio comenzó a hacer un seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página Historial de documentos de Amazon Lex de Amazon Lex.

Cambio	Descripción	Fecha
AmazonLexFullAccess: actualización de una política existente	Amazon Lex agregó nuevos permisos para permitir el acceso de solo lectura a las operaciones del servicio de creación de modelos de Amazon Lex V2.	18 de agosto de 2021
AmazonLexreadOnly: actualización de una política existente	Amazon Lex agregó nuevos permisos para permitir el acceso de solo lectura a	18 de agosto de 2021

Cambio	Descripción	Fecha
	las operaciones del servicio de creación de modelos de Amazon Lex V2.	
AmazonLexRunbotsOnly: actualización de una política existente	Amazon Lex agregó nuevos permisos para permitir el acceso de solo lectura a las operaciones del servicio de tiempo de ejecución de Amazon Lex V2.	18 de agosto de 2021
Amazon Lex ha comenzado a hacer un seguimiento de los cambios	Amazon Lex comenzó a realizar el seguimiento de los cambios de las políticas administradas por AWS.	18 de agosto de 2021

Uso de roles vinculados a servicios para Amazon Lex

Amazon Lex utiliza roles vinculados a servicios de AWS Identity and Access Management (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que se encuentra vinculado directamente a Amazon Lex. Los roles vinculados a servicios se encuentran predefinidos por Amazon Lex e incluyen todos los permisos que el servicio requiere para llamar a otros servicios de AWS en su nombre.

Un rol vinculado a servicios simplifica la configuración de Amazon Lex porque ya no tendrá que agregar manualmente los permisos requeridos. Amazon Lex define los permisos de sus roles vinculados a servicios y, a menos que esté definido de otra manera, solo Amazon Lex puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos y que la política de permisos no se pueda adjuntar a ninguna otra entidad de IAM.

Solo puede eliminar un rol vinculado a servicios después de eliminar sus recursos relacionados. De esta forma, se protegen los recursos de Amazon Lex, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Permisos de roles vinculados a servicios para Amazon Lex

Amazon Lex utiliza dos roles vinculados a servicios:

- AWSServiceRoleForLexBots: Amazon Lex utiliza este rol vinculado a servicios para invocar Amazon Polly, que sintetiza las respuestas por voz para el bot; para llamar a Amazon Comprehend, con fines de análisis de opiniones; y, opcionalmente, para invocar Amazon Kendra, con fines de búsqueda de índices.
- AWSServiceRoleForLexChannels: Amazon Lex usa este rol vinculado a servicios para publicar texto en el bot cuando se administran los canales.

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a servicios para Amazon Lex

No necesita crear manualmente un rol vinculado a servicios. Cuando se crea un bot, un canal de bots o una intención de búsqueda de Amazon Kendra en la AWS Management Console, Amazon Lex crea el rol vinculado al servicio por usted.

Si elimina este rol vinculado al servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Si crea un nuevo bot, una asociación de canal o una intención de búsqueda de Amazon Kendra, Amazon Lex volverá a crear el rol vinculado al servicio.

También puede usar AWS CLI para crear un rol vinculado al servicio con el caso de uso AWSServiceRoleForLexBots . En la AWS CLI, cree un rol vinculado a servicios con el nombre de servicio de Amazon Lex, `lex.amazonaws.com`. Para obtener más información, consulte Paso 1: Crear un rol vinculado a servicio (AWS CLI). Si elimina este rol vinculado al servicio, puede utilizar este mismo proceso para volver a crear el rol.

Edición de un rol vinculado a servicios para Amazon Lex

Amazon Lex no le permite editar roles vinculados a servicios de Amazon Lex. Después de crear un rol vinculado a servicios, no puede cambiarle el nombre, ya que varias entidades pueden hacer referencia al mismo. Sin embargo, puede editar la descripción del rol mediante IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM..

Eliminación de un rol vinculado a servicios para Amazon Lex

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a un servicio, recomendamos que elimine dicho rol. De esta forma no tiene una entidad no utilizada que no se

monitorice ni mantenga de forma activa. Sin embargo, debe limpiar los recursos del rol vinculado al servicio antes de eliminarlo manualmente.

> **Note**
>
> Si el servicio de Amazon Lex está usando el rol cuando intenta eliminar los recursos, es posible que la eliminación falle. En tal caso, espere unos minutos e intente de nuevo la operación.

Para eliminar los recursos de Amazon Lex que utilizan los roles vinculados a servicios:

1. Elimine los canales de bots que esté utilizando.
2. Elimine los bots de su cuenta.

Para eliminar manualmente el rol vinculado a servicios mediante IAM

Utilice la consola de IAM, la AWS CLI o la API de AWS para eliminar los roles vinculados a servicios de Amazon Lex. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Regiones admitidas para los roles vinculados a servicios de Amazon Lex

Amazon Lex admite el uso de roles vinculados a servicios en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte Cuotas y puntos de conexión de Amazon Lex.

Solución de problemas de identidad y acceso de Amazon Lex

Utilice la siguiente información para diagnosticar y solucionar los problemas habituales que pueden surgir cuando se trabaja con Amazon Lex e IAM.

Temas

- No tengo autorización para realizar una acción en Amazon Lex
- No estoy autorizado a realizar tareas como: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Amazon Lex

No tengo autorización para realizar una acción en Amazon Lex

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `lex:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `lex:`*`GetWidget`*.

Si necesitas ayuda, ponte en contacto con tu administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibe un error que indica que no tiene autorización para llevar a cabo la acción `iam:PassRole`, las políticas se deben actualizar para permitirle pasar un rol a Amazon Lex.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Amazon Lex. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Amazon Lex

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si Amazon Lex admite estas características, consulte Cómo funciona Amazon Lex con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer la diferencia entre usar roles y políticas basadas en recursos para el acceso entre cuentas, consulte el tema Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Supervisión en Amazon Lex

La supervisión es importante para mantener la fiabilidad, disponibilidad y desempeño de los bots de chat de Amazon Lex. En esta sección se describe cómo utilizar Registros de Amazon CloudWatch y AWS CloudTrail para supervisar Amazon Lex y se describe el tiempo de ejecución de Amazon Lex y las métricas de asociación de canal.

Temas

- Monitoreo de Amazon Lex con Amazon CloudWatch
- Supervisión de las llamadas a la API de Amazon Lex con Registros de AWS CloudTrail

Monitoreo de Amazon Lex con Amazon CloudWatch

Para realizar un seguimiento del estado de los bots de Amazon Lex, utilice Amazon CloudWatch. Con CloudWatch, puede obtener métricas de las operaciones de Amazon Lex individuales o globales de su cuenta. También puede configurar alarmas de CloudWatch para recibir una notificación cuando una o varias métricas superen el umbral que defina. Por ejemplo, puede monitorizar el número de solicitudes realizadas a un bot durante un periodo de tiempo concreto, ver la latencia de las solicitudes correctas o crear una alarma para cuando los errores superen un umbral.

Métricas de CloudWatch para Amazon Lex

Para obtener métricas de las operaciones de Amazon Lex, debe especificar la siguiente información:

- La dimensión de la métrica. Una dimensión es un conjunto de pares nombre-valor que se emplea para identificar una métrica. Amazon Lex tiene tres dimensiones:
  - `BotAlias, BotName, Operation`
  - `BotAlias, BotName, InputMode, Operation`
  - `BotName, BotVersion, InputMode, Operation`
- El nombre de la métrica, como `MissedUtteranceCount` o `RuntimeRequestCount`.

Puede obtener métricas de Amazon Lex con la AWS Management Console, la AWS CLI o la API de CloudWatch. Puede utilizar la API de CloudWatch mediante uno de los kits de desarrollo de software (SDK) de Amazon AWS o las herramientas de la API de Amazon CloudWatch. La consola de Amazon Lex muestra una serie de gráficos basados en los datos sin procesar de la API de CloudWatch.

Debe tener los permisos de CloudWatch adecuados para supervisar Amazon Lex con CloudWatch. Para obtener más información, consulte Autenticación y control de acceso para Amazon CloudWatch en la Guía del usuario de Amazon CloudWatch.

Pasos para ver las métricas de Amazon Lex

Vea métricas de Amazon Lex mediante la consola de Amazon Lex o de CloudWatch.

Visualización de métricas (consola de Amazon Lex)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.

2. En la lista de bots, elija el bot cuyas métricas quiere ver.
3. Elija Monitoring (Monitorización). Las métricas se muestran en gráficos.

Pasos para ver las métricas (consola de CloudWatch)

1. Inicie sesión en la AWS Management Console y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Metrics, elija All Metrics y después AWS/Lex.
3. Elija la dimensión, un nombre de métrica y, a continuación, Add to graph (Añadir al gráfico).
4. Elija un valor para el intervalo de fechas. El recuento de las métricas del intervalo de fechas seleccionado se muestra en el gráfico.

Creación de una alarma

Las alarmas de CloudWatch controlan una única métrica durante el periodo de tiempo especificado y realizan una o más acciones: enviar notificaciones a un tema de Amazon Simple Notification Service (Amazon SNS) o a una política escalado automático. Las acciones se basan en el valor de la métrica con respecto a un umbral determinado durante los períodos de tiempo que se especifiquen. CloudWatch también puede enviar mensajes de Amazon SNS cuando la alarma cambia de estado.

Las alarmas de CloudWatch solo invocan acciones cuando el estado cambia y se mantiene durante el período especificado.

Para configurar una alarma

1. Inicie sesión en la AWS Management Console y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Alarms y, a continuación, seleccione Create Alarm.
3. Elija AWS/Lex Metrics y después elija una métrica.
4. En Time Range, elija un intervalo de tiempo para monitorizar y, a continuación, elija Next.
5. Introduzca un Name (Nombre) y una Description (Descripción).
6. En Whenever, elija >= y escriba un valor máximo.
7. Si desea que CloudWatch envíe un correo electrónico cuando se alcance el estado de alarma, dentro de la sección Acciones, en Cuando este estado de alarma sea…, elija ALARMA. En Send notification to, elija una lista de correo o elija New list y cree una nueva lista de correo.

8. Obtenga una vista previa de la alarma en la sección Alarm Preview. Si está satisfecho con la alarma, elija Create Alarm.

Métricas de CloudWatch para el tiempo de ejecución de Amazon Lex

En la siguiente tabla, se describen las métricas del tiempo de ejecución de Amazon Lex.

Métrica	Descripción
`KendraIndexAccessError`	El número de veces que Amazon Lex no ha podido acceder al índice de Amazon Kendra. Dimensión válida para la operación `PostContent` con el `InputMode` `Text` o `Speech`: • `BotName, BotAlias, Operation, InputMode` Dimensión válida para la operación `PostText`: • `BotName, BotAlias, Operation` Unidad: recuento
`KendraLatency`	El tiempo que tarda Amazon Kendra en responder a una solicitud del `AMAZON.KendraSearchIntent` . Dimensiones válidas para la operación `PostContent` con el `InputMode` `Text` o `Speech`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensiones válidas para la operación `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation`

Métrica	Descripción
	Unidad: milisegundos
KendraSuccess	El número de solicitudes con éxito de AMAZON.KendraSearchIntent para el índice de Amazon Kendra. Dimensiones válidas para la operación PostContent con el InputMode Text o Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensiones válidas para la operación PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Unidad: recuento
KendraSystemErrors	El número de veces que Amazon Lex no pudo consultar el índice de Amazon Kendra. Dimensión válida para la operación PostContent con el InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación PostText: • BotName, BotAlias, Operation Unidad: recuento

Métrica	Descripción
KendraThrottledEvents	La cantidad de veces que Amazon Kendra limitó las solicitudes de AMAZON.KendraSearchIntent . Dimensión válida para la operación PostContent con el InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación PostText: • BotName, BotAlias, Operation Unidad: recuento
MissedUtteranceCount	El número de declaraciones no reconocidas durante el período especificado. Dimensiones válidas para la operación PostContent con el InputMode Text o Speech: • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensiones válidas para la operación PostText: • BotName, BotVersion, Operation • BotName, BotAlias, Operation

Métrica	Descripción
RuntimeConcurrency	El número de conexiones simultáneas en el período especificado. RuntimeConcurrency se informa como StatisticSet . Dimensiones válidas para la operación PostContent con el InputMode Text o Speech: • Operation, BotName, BotVersion, InputMode • Operation, BotName, BotAlias, InputMode Dimensiones válidas para otras operaciones: • Operation, BotName, BotVersion • Operation, BotName, BotAlias Unidad: recuento
RuntimeInvalidLambdaResponses	El número de respuestas de AWS Lambda (Lambda) no válidas en el período especificado. Dimensión válida para la operación PostContent con el InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación PostText: • BotName, BotAlias, Operation

Métrica	Descripción
RuntimeLambdaErrors	El número de errores de tiempo de ejecución de Lambda en el período especificado. Dimensión válida para la operación PostContent con el InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación PostText: • BotName, BotAlias, Operation
RuntimePollyErrors	El número de respuestas de Amazon Polly no válidas durante el período especificado. Dimensión válida para la operación PostContent con el InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación PostText: • BotName, BotAlias, Operation

Métrica	Descripción
`RuntimeRequestCount`	El número de solicitudes en tiempo de ejecución durante el periodo especificado. Dimensiones válidas para la operación `PostContent` con el `InputMode` `Text` o `Speech`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensiones válidas para la operación `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unidad: recuento
`RuntimeSucessfulRequestLatency` **Important** Esta métrica es `RuntimeSuccessfulRequestLatency` y no `RuntimeSuccessfulRequestLatency`.	La latencia de las solicitudes correctas entre el momento en que se realizó la solicitud y se pasó la respuesta de vuelta. Dimensiones válidas para la operación `PostContent` con el `InputMode` `Text` o `Speech`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensiones válidas para la operación `PostText`: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unidad: milisegundos

Métrica	Descripción
`RuntimeSystemErrors`	El número de errores del sistema en el período especificado. El intervalo de códigos de respuesta de un error del sistema es de 500 a 599. Dimensión válida para la operación `PostContent` con el `InputMode` `Text` o `Speech`: • `BotName, BotAlias, Operation, InputMode` Dimensión válida para la operación `PostText`: • `BotName, BotAlias, Operation` Unidad: recuento
`RuntimeThrottledEvents`	El número de solicitudes restringidas. Amazon Lex limita una solicitud cuando recibe más solicitudes que el límite de transacciones por segundo de su cuenta. Si el límite establecido para su cuenta se supera con frecuencia, puede solicitar un aumento del límite. Para solicitar un aumento, consulte Límites de los servicios de AWS. Dimensión válida para la operación `PostContent` con el `InputMode` `Text` o `Speech`: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación `PostText`: • `BotName, BotAlias, Operation` Unidad: recuento

Métrica	Descripción
RuntimeUserErrors	El número de errores del usuario en el período especificado. El intervalo de códigos de respuesta de un error de usuario comprende de 400 a 499. Dimensión válida para la operación PostContent con el InputMode Text o Speech: • BotName, BotAlias, Operation, InputMode Dimensión válida para la operación PostText: • BotName, BotAlias, Operation Unidad: recuento

Las métricas del tiempo de ejecución de Amazon Lex utilizan el espacio de nombres AWS/Lex y proporcionan métricas en las siguientes dimensiones. Puede agrupar las métricas por dimensiones en la consola de CloudWatch:

Dimensión	Descripción
BotName, BotAlias, Operation, InputMode	Agrupa las métricas por el alias del bot, el nombre del bot, la operación (PostContent) y si la entrada era texto o voz.
BotName, BotVersion, Operation, InputMode	Agrupa las métricas por el nombre del bot, la versión del bot, la operación (PostContent) y si la entrada era texto o voz.
BotName, BotVersion, Operation	Agrupa las métricas por el nombre del bot, la versión del bot y la operación, PostText.
BotName, BotAlias, Operation	Agrupa las métricas por el nombre del bot, el alias del bot y la operación, PostText.

Métricas de CloudWatch para asociaciones de canal de Amazon Lex

Una asociación de canal es la asociación entre Amazon Lex y un canal de mensajería como, por ejemplo, Facebook. En la siguiente tabla se describen las métricas de asociación de canal de Amazon Lex.

Métrica	Descripción
`BotChannelAuthErrors`	El número de errores de autenticación que devuelve el canal de mensajería en el periodo de tiempo especificado. Un error de autenticación indica que el token secreto proporcionado durante la creación del canal no es válido o ha caducado.
`BotChannelConfigurationErrors`	El número de errores de configuración en el periodo especificado. Un error de configuración indica que una o varias entradas de configuración del canal no son válidas.
`BotChannelInboundThrottledEvents`	El número de veces que Amazon Lex limitó los mensajes enviados por el canal de mensajería durante el periodo especificado.
`BotChannelOutboundThrottledEvents`	El número de veces que se limitaron los eventos de salida de Amazon Lex al canal de mensajería durante el periodo especificado.
`BotChannelRequestCount`	El número de solicitudes realizadas en un canal durante el periodo especificado.
`BotChannelResponseCardErrors`	El número de veces que Amazon Lex no ha podido publicar tarjetas de respuesta en el periodo especificado.
`BotChannelSystemErrors`	El número de errores internos que se produjeron en Amazon Lex para un canal durante el periodo especificado.

Las métricas de asociación de canal de Amazon Lex utilizan el espacio de nombres `AWS/Lex` y proporcionan métricas para la siguiente dimensión. Puede agrupar las métricas por dimensiones en la consola de CloudWatch:

Dimensión	Descripción
`BotAlias, BotChannelName, BotName, Source`	Agrupe las métricas por el alias del bot, el nombre del canal, el nombre del bot y el origen del tráfico.

Métricas de CloudWatch para registros de conversaciones

Amazon Lex utiliza las siguientes métricas para el registro de conversaciones:

Métrica	Descripción
`ConversationLogsAudioDeliverySuccess`	El número de registros de audio entregados correctamente al bucket de S3 en el período de tiempo especificado. Unidades: recuento
`ConversationLogsAudioDeliveryFailure`	El número de registros de audio que no se pudieron entregar al bucket de S3 en el período de tiempo especificado. Un error de entrega indica un error con los recursos configurados para los registros de conversación. Los errores pueden incluir permisos de IAM insuficientes, una clave de AWS KMS inaccesible o un bucket de S3 inaccesible. Unidades: recuento
`ConversationLogsTextDeliverySuccess`	El número de registros de texto entregados correctamente a Registros de CloudWatch en el período de tiempo especificado. Unidades: recuento
`ConversationLogsTextDeliveryFailure`	El número de registros de texto que no se pudieron entregar a Registros de CloudWatch en el período de tiempo especificado. Un error

Métrica	Descripción
	de entrega indica un error con los recursos configurados para los registros de conversación. Los errores pueden incluir permisos de IAM insuficientes, una clave de AWS KMS inaccesible o un grupo de registro de Registros de CloudWatch inaccesible. Unidades: recuento

Las métricas de registro de conversaciones de Amazon Lex utilizan el espacio de nombres `AWS/Lex` y proporcionan métricas para las siguientes dimensiones. Puede agrupar las métricas por dimensión en la consola de CloudWatch.

Dimensión	Descripción
`BotAlias`	Agrupar métricas por el alias del bot.
`BotName`	Agrupar métricas por el nombre del bot.
`BotVersion`	Agrupar métricas por la versión del bot.

Supervisión de las llamadas a la API de Amazon Lex con Registros de AWS CloudTrail

Amazon Lex se integra con AWS CloudTrail, un servicio que proporciona un registro de las acciones hechas por un usuario, un rol o un servicio de AWS en Amazon Lex. CloudTrail obtiene un subconjunto de llamadas a la API para Amazon Lex como eventos, incluidas las llamadas procedentes de la consola de Amazon Lex y las llamadas de código a las API de Amazon Lex. Si crea un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, incluidos los eventos de Amazon Lex. Si no configura un registro de seguimiento, puede ver los eventos más recientes de la consola de CloudTrail en el Historial de eventos. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon Lex, la dirección IP de origen desde la que se realizó la solicitud, quién realizó la solicitud, cuándo se realizó y otros detalles adicionales.

Para obtener más información acerca de CloudTrail, incluso cómo configurarlo y habilitarlo, consulte la Guía del usuario de AWS CloudTrail.

Información de Amazon Lex en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce una actividad de eventos compatible en Amazon Lex, la actividad se registra en un evento de CloudTrail junto con otros eventos de servicios de AWS en Historial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte Ver eventos con el historial de eventos de CloudTrail.

Para mantener un registro continuo de los eventos de la cuenta de AWS, incluidos los eventos de Amazon Lex, cree un registro de seguimiento. Un seguimiento habilita a CloudTrail a enviar archivos de registro a un bucket de Amazon Simple Storage Service (Amazon S3). De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de S3 especificado. También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte:

- Introducción a la creación de registros de seguimiento
- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de registro de CloudTrail de varias regiones y Recibir archivos de registro de CloudTrail de varias cuentas

Amazon Lex admite el registro de las siguientes operaciones como eventos en archivos de registros de CloudTrail:

- 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 registro o evento contiene información sobre quién generó la solicitud. Esta información le ayuda a determinar lo siguiente:

- si la solicitud se realizó con las credenciales del nodo raíz o del usuario
- si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado
- si la solicitud la realizó otro servicio de AWS

Para obtener más información, consulte el Elemento userIdentity de CloudTrail.

Para obtener información sobre las acciones de Amazon Lex que se registran en los registros de CloudTrail, consulte Amazon Lex Model Building Service. Por ejemplo, las llamadas a las operaciones PutBot, GetBot y DeleteBot generan entradas en el registro de CloudTrail. Las acciones documentadas en Amazon Lex Runtime Service, PostContent y PostText no se registran.

Ejemplo: entradas del archivo de registro de Amazon Lex

Un registro de seguimiento es una configuración que permite la entrega de eventos como archivos de log al bucket de S3 que se especifique. Los archivos log de CloudTrail pueden contener una o varias entradas de log. Un evento representa una única solicitud de cualquier origen e incluye información acerca de la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etcétera. Los archivos de registro de CloudTrail no son un rastro de pila ordenado de las llamadas a las API públicas, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo se muestra una entrada de registro de CloudTrail que muestra el resultado de una llamada a la operación `PutBot`.

```
   {
               "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"
            }
        }
```

Validación de la conformidad en Amazon Lex

Los auditores externos evalúan la seguridad y el cumplimiento de Amazon Lex como parte de varios programas de AWS cumplimiento. Amazon Lex es un servicio compatible con HIPAA. Es compatible con PCI, SOC e ISO. Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Amazon Lex se determina en función de la sensibilidad de los datos, los objetivos de conformidad de la organización, así como de la legislación y los reglamentos aplicables. Si su uso de Amazon Lex está sujeto a conformidad con normas tales como PCI, AWS le ofrece los siguientes recursos para ayudarle:

- Guías de inicio rápido sobre seguridad y conformidad: guías de implementación en las que se analizan las consideraciones arquitectónicas y se proporcionan los pasos necesarios para implementar entornos básicos centrados en la seguridad y el cumplimiento en AWS
- Documento técnico sobre cómo diseñar arquitecturas que cumplan los requisitos de seguridad y conformidad de HIPAA: en este documento técnico, se explica cómo las empresas pueden utilizar AWS para crear aplicaciones conformes con HIPAA.
- Recursos de conformidad de AWS: un conjunto de manuales y guías que podría aplicarse a su sector y ubicación.
- AWS Config: un servicio que evalúa en qué medida las configuraciones de los recursos cumplen las prácticas internas, las directrices del sector y la normativa.
- AWS Security Hub— Una visión integral del estado de su seguridad AWS que le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad

Para ver una lista de AWS los servicios incluidos en el ámbito de los programas de conformidad específicos, consulte Servicios de AWS incluidos en el ámbito de aplicación por programa de conformidad. Para obtener información general, consulte Programas de conformidad de AWS.

Resiliencia en Amazon Lex

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global. AWS

Además de la infraestructura AWS global, Amazon Lex ofrece varias funciones que ayudan a respaldar sus necesidades de respaldo y resiliencia de datos.

Seguridad de la infraestructura en Amazon Lex

Como se trata de un servicio administrado, Amazon Lex está protegido por los procedimientos de seguridad de red globales de AWS , que se describen en el documento técnico Amazon Web Services: Información general sobre los procesos de seguridad.

Utiliza las llamadas a la AWS API publicadas para acceder a Amazon Lex a través de la red. Los clientes deben ser compatibles con TLS (Transport Layer Security 1.0. Le recomendamos TLS 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS), como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos, como Java 7 y posteriores, son compatibles con estos modos. Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de seguridad de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Puede llamar a estas operaciones de la API desde cualquier ubicación de red, pero Amazon Lex admite políticas de acceso en el nivel de recursos, que pueden incluir restricciones en función de la dirección IP de origen. También puede utilizar políticas de Amazon Lex para controlar el acceso desde puntos de enlace específicos de Amazon Virtual Private Cloud (Amazon VPC) o VPC específicas. De hecho, esto aísla el acceso a la red a un recurso de Amazon Lex determinado únicamente de la VPC específica de la red. AWS

Directrices y cuotas de Amazon Lex

Las siguientes secciones indican las directrices y cuotas de uso de Amazon Lex.

Temas

Regiones admitidas

Para ver una lista de las regiones de AWS en las que Amazon Lex está disponible, consulte Regiones y puntos de conexión de AWS en la Referencia general de Amazon Web Services.

Directrices generales

En esta sección se describen las directrices generales del uso de Amazon Lex.

- Firma de solicitudes: todas las operaciones de la API en tiempo de ejecución y de desarrollo del modelo de Amazon Lex en Referencia de la API utilizan Signature Version 4 para autenticar solicitudes. Para obtener más información acerca de la autenticación de solicitudes, consulte Proceso de firma Signature Version 4 en la Referencia general de Amazon Web Services.

  Para PostContent, Amazon Lex utiliza la opción de carga sin firma que se describe en Cálculos de firma para el encabezado de autorización: transferencia de cargas en un solo fragmento (AWS Signature Version 4) en la Referencia de la API de Amazon Simple Storage Service (S3).

  Al utilizar la opción de carga sin firma, no incluya el hash de la carga en la solicitud canónica. En su lugar, utilice la cadena literal "UNSIGNED-PAYLOAD" como hash de la carga. Incluya también un encabezado con el nombre `x-amz-content-sha256` y el valor `UNSIGNED-PAYLOAD` en la solicitud `PostContent`.

- Tenga en cuenta lo siguiente sobre el modo en que Amazon Lex captura los valores de ranura de los enunciados del usuario:

  Amazon Lex utiliza los valores de enumeración facilitados en una definición de tipo de ranura para entrenar a sus modelos de machine learning. Suponga que define una intención denominada `GetPredictionIntent` con el siguiente enunciado de muestra:

  ```
  "Tell me the prediction for {Sign}"
  ```

  Donde `{Sign}` es un slot de tipo personalizado `ZodiacSign` y Tiene 12 valores de enumeración, de `Aries` a `Pisces`. Por el enunciado del usuario “Tell me the prediction for…”, Amazon Lex entiende que lo que sigue es un signo del zodíaco.

  Cuando en el campo `valueSelectionStrategy` se ha establecido `ORIGINAL_VALUE` con la operación PutSlotType o se ha seleccionado Expandir valores en la consola, si el usuario dice “Tell me the prediction for earth”, Amazon Lex deduce que “earth” es un `ZodiacSign` y se lo pasa a la aplicación cliente o a las funciones de Lambda. Debe comprobar que los valores de slot sean válidos antes de usarlos en su actividad de cumplimiento.

  Si establece el campo `valueSelectionStrategy` en `TOP_RESOLUTION` utilizando la operación PutSlotType o si se ha seleccionado Restrict to slot values and synonyms en la consola, los valores devueltos se limitan a los valores definidos para el tipo de slot. Por ejemplo, si el usuario dice "Tell me the prediction for earth", el valor no se reconocería porque no es uno de los valores definidos para el tipo de slot. Cuando se definen sinónimos para los valores slot, se reconocen como si fueran un valor de slot, pero se devuelve el valor de slot en lugar del sinónimo.

  Cuando Amazon Lex llama a una función de Lambda o devuelve el resultado de una interacción de voz con la aplicación cliente, no se garantiza el uso de mayúsculas y minúsculas para los valores de ranura. Por ejemplo, si obtiene valores para el tipo de ranura integrado AMAZON.Movie y un usuario dice o escribe “Lo que el viento se llevó”, podría devolver “Lo Que El Viento Se Llevó”, “lo que el viento se llevó” o “Lo que el Viento se Llevó”. En el caso de las interacciones de texto, el

uso de mayúsculas y minúsculas en los valores de slot coincide con el texto introducido o con el valor de slot, en función del valor del campo `valueResolutionStrategy`.

- Al definir los valores de los slots que contienen acrónimos, utilice los siguientes patrones:
  - Letras mayúsculas separadas por puntos (D.V.D.)
  - Letras mayúsculas separadas por espacios (D V D)

- Amazon Lex no admite el tipo de ranura integrado AMAZON.LITERAL que sí admite Alexa Skills Kit. Sin embargo, Amazon Lex admite la creación de tipos de ranura personalizados que puede utilizar para implementar esta funcionalidad. Como se mencionó anteriormente, puede capturar valores fuera de la definición del tipo de slot personalizado. Añada otros valores de enumeración diferentes para aumentar la precisión del reconocimiento automático de voz (ASR) y la comprensión del lenguaje natural (NLU).

- Los tipos de slot integrados AMAZON.DATE y AMAZON.TIME capturan fechas y horas relativas y absolutas. Las fechas y horas relativas se resuelven para la región donde Amazon Lex procesa la solicitud.

  Para el tipo de ranura integrado `AMAZON.TIME`, si el usuario no especifica si una hora es antes o después del mediodía, esa hora es ambigua y Amazon Lex volverá a preguntar al usuario. Recomendamos utilizar preguntas que permitan obtener la hora absoluta. Por ejemplo, para una pregunta como «¿Cuándo quieres recibir la pizza?» Puede decir "6 PM" o "6 in the evening".

- Si proporciona datos de aprendizaje confusos en el bot, se reducirá la capacidad de Amazon Lex para comprender la entrada del usuario. Estudie estos ejemplos:

  Imagine que dispone de dos intenciones (`OrderPizza` y `OrderDrink`) en su bot y ambas están configuradas con un enunciado "I want to order". Este enunciado no corresponde a una intención específica que Amazon Lex pueda aprender durante el desarrollo del modelo de lenguaje para el

bot en tiempo de compilación. Como resultado, cuando un usuario dice este enunciado en tiempo de ejecución, Amazon Lex no puede elegir una intención con un alto nivel de confianza.

Otro ejemplo podría ser cuando define una intención personalizada para obtener una confirmación del usuario (por ejemplo, `MyCustomConfirmationIntent`) y configura la intención con los enunciados "Yes" y "No". Tenga en cuenta que Amazon Lex también cuenta con un modelo de lenguaje para comprender las confirmaciones del usuario. Esto puede generar situaciones conflictivas. Cuando el usuario responde afirmativamente, ¿significa que se trata de una confirmación de la intención en curso o que el usuario solicita la intención personalizada que usted ha creado?

En general, los enunciados de muestra que proporcione deben corresponder a una intención específica y, de forma opcional, a valores de slot específicos.

- Las operaciones de la API en tiempo de ejecución PostContent y PostText toman un ID de usuario como el parámetro obligatorio. Los desarrolladores lo pueden establecer en cualquier valor que cumpla las limitaciones descritas en la API. Recomendamos no usar este parámetro para enviar cualquier información confidencial, por ejemplo, inicios de sesión de los usuarios, mensajes de correo o números de la seguridad social. Este ID se utiliza principalmente para identificar de forma exclusiva la conversación con un bot (puede haber varios usuarios que pidan pizza).

- Si la aplicación cliente utiliza Amazon Cognito para la autenticación, podría utilizar el ID de usuario de Amazon Cognito como ID de usuario de Amazon Lex. Tenga en cuenta que cualquier función de Lambda configurada para el bot debe tener su propio mecanismo de autenticación para identificar al usuario en cuyo nombre Amazon Lex invoca la función de Lambda.

- Le animamos a que defina una intención que capture la intención de un usuario de interrumpir la conversación. Por ejemplo, puede definir una intención (`NothingIntent`) con enunciados de ejemplo (“No quiero nada”, “salir”, “adiós”), sin ranuras y sin ninguna función de Lambda configurada como enlace de código. Esto permite a los usuarios cerrar una conversación con fluidez.

Cuotas

En esta sección se describen las cuotas actuales de Amazon Lex. Estas cuotas se agrupan por categorías.

Algunas Service Quotas se pueden ajustar o aumentar. Póngase en contacto con el servicio de atención al cliente de AWS para solicitar un aumento de cuota. Puede tardar varios días en aumentar una cuota de servicio. Si va a aumentar su cuota como parte de un proyecto más grande, asegúrese de añadir este tiempo a su plan.

Temas

Cuotas de servicio en tiempo de ejecución

Además de las cuotas descritas en la referencia de la API, tenga en cuenta lo siguiente:

Cuotas de API

- La entrada de voz para la operación PostContent puede tener una duración de hasta 15 segundos.

- Tanto en las operaciones de API en tiempo de ejecución PostContent como PostText, el texto de entrada puede tener un tamaño máximo de 1 024 caracteres Unicode.

- El tamaño máximo de los encabezados `PostContent` es 16 kB. El tamaño máximo combinado de los encabezados de solicitud y sesión es de 12 kB.

- Al utilizar las operaciones `PostContent` o `PostText` en modo texto, el número máximo de conversaciones concurrentes con un bot es 2 para el alias `$LATEST` y 50 para el resto de alias. La cuota se aplica por separado para cada API.

- Al utilizar la operación `PostContent` en modo texto, el número máximo de conversaciones concurrentes con un bot es 2 para el alias `$LATEST` y 125 para el resto de alias. La cuota se aplica por separado para cada API.

- El número máximo de llamadas de administración de sesión concurrentes (PutSession, GetSession y DeleteSession) es 2 para el alias `$LATEST` de un bot y 50 para el resto de alias.

- El tamaño de entrada máximo a función de Lambda es de 12 kB. El tamaño de salida máximo es de 25 kB, de los cuales 12 kB pueden ser atributos de la sesión.

Uso de la versión **$LATEST**

- La versión `$LATEST` del bot solo debería utilizarse para llevar a cabo pruebas manuales. Amazon Lex limita el número de solicitudes en tiempo de ejecución que puede realizar a la versión `$LATEST` del bot.

- Al actualizar la versión `$LATEST` del bot, Amazon Lex finaliza cualquier conversación en curso de cualquier aplicación cliente que utilice la versión `$LATEST`. Por lo general, no debería usar la versión `$LATEST` de un bot en producción porque la versión `$LATEST` se puede actualizar. En su lugar, debe publicar una versión y utilizarla.

- Al actualizar un alias, Amazon Lex tarda varios minutos en aplicar los cambios. Si modifica la versión `$LATEST` del bot, el cambio se aplica inmediatamente.

Tiempo de espera de la sesión

- El tiempo de espera de sesión establecido al crear el bot determina el tiempo durante el cual el bot conserva el contexto de la conversación, como la intención actual del usuario y los datos del slot.

- Una vez que un usuario comienza una conversación con el bot, y hasta que la sesión termine, Amazon Lex utiliza la misma versión del bot, incluso si se actualiza el alias del bot para que apunte a otra versión.

Cuotas de creación de modelos

Con desarrollo del modelo se hace referencia a la creación y gestión de bots. Esto incluye la creación y la administración de bots, las intenciones, los tipos de slot y las asociaciones de canal de bot.

Temas

Cuotas de bots

- Puede configurar preguntas e instrucciones a lo largo de la API del modelo de desarrollo. Cada una de estas preguntas o instrucciones puede tener hasta cinco mensajes y cada mensaje puede contener desde 1 hasta 1 000 caracteres UTF-8.

- Al utilizar grupos de mensajes, puede definir hasta cinco grupos para cada mensaje. Cada grupo de mensajes puede contener un máximo de cinco mensajes y tiene una limitación de 15 mensajes en todos los grupos de mensajes.

- Puede definir enunciados de ejemplo para las intenciones y los slots. Puede utilizar un máximo de 200 000 caracteres para todos los enunciados.

- Cada tipo de slot puede definir un máximo de 10 000 valores y sinónimos. Cada bot puede contender un máximo de 50.000 valores de tipo de slot y sinónimos.

- Los nombres de bot, de alias y de asociación del canal de bot no distinguen entre mayúsculas y minúsculas en el momento de su creación. Si crea `PizzaBot` y, a continuación, vuelve a crear `pizzaBot`, obtendrá un error. Sin embargo, cuando se accede un recurso, los nombres del recurso distinguen entre mayúsculas y minúsculas, por lo que debe especificar `PizzaBot` y no `pizzaBot`. Estos nombres deben tener entre 2 y 50 caracteres ASCII.

- El número máximo de versiones que puede publicar para todos los tipos de recursos es 100. Tenga en cuenta que no existe control de versiones para los alias.

- Dentro de un bot, los nombres de las intenciones y de los slot deben ser únicos, no puede tener una intención y un slot con el mismo nombre.

- Puede crear un bot que esté configurado para que admita múltiples intenciones. Si dos intenciones tienen un slot con el mismo nombre, entonces el tipo de slot correspondiente debe ser el mismo.

  Por ejemplo, suponga que crea un bot que admite dos intenciones (`OrderPizza` y `OrderDrink`). Si ambas intenciones tienen el slot `size`, entonces el tipo de slot debe ser igual en los dos sitios.

  Además, los enunciados de ejemplo que proporcione para un slot en una de las intenciones se aplicarán a un slot con el mismo nombre en otras intenciones.

- Puede asociar un máximo de 250 intenciones con un bot.

- Al crear un bot, tendrá que especificar un tiempo de espera de la sesión. El tiempo de espera de la sesión puede oscilar entre un minuto y un día. El valor predeterminado es de cinco minutos.

- Puede crear hasta cinco alias para un bot.

- Puede crear hasta 250 bots para cada cuenta de AWS.

- No puede crear varias intenciones que surjan de la misma intención integrada.

Cuotas de intención

- Al crearlos, los nombres de las intenciones y de los slots no distinguen entre mayúsculas y minúsculas. Es decir, si crea la intención `0rderPizza` y, a continuación, intenta crear otra intención `orderPizza`, obtendrá un error. Sin embargo, cuando se accede a estos recursos, los nombres del recurso distinguen entre mayúsculas y minúsculas, por lo que debe especificar `0rderPizza` y no `orderPizza`. Estos nombres deben tener entre 1 y 100 caracteres ASCII.

- Una intención puede tener hasta 1,500 enunciados de muestra. Se necesita un enunciado de ejemplo como mínimo. Cada enunciado de ejemplo puede tener hasta 200 caracteres UTF-8. Puede utilizar un máximo de 200 000 caracteres para todos los enunciados de intenciones y slot en un bot. Enunciado de ejemplo para una intención:
  - Puede hacer referencia a cero o más nombres de slot.
  - Puede hacer referencia a un nombre de slot una única vez.

  Por ejemplo:

```
I want a pizza
I want a {pizzaSize} pizza
I want a {pizzaSize} {pizzaTopping} pizza
```

- Aunque cada intención admite hasta 1500 enunciados, si utiliza menos Amazon Lex podrá reconocer mejor las entradas no incluidas en el conjunto proporcionado.

- Puede crear hasta cinco grupos de mensajes para cada mensaje en una intención. Puede haber un total de 15 mensajes en todos los grupos correspondientes a un mensaje.

- La consola solo puede crear grupos para los mensajes `conclusionStatement` y `followUpPrompt`. Puede crear grupos de mensajes para cualquier otro mensaje mediante la API de Amazon Lex.

- Cada slot puede tener hasta 10 enunciados de muestra. Cada enunciado de muestra debe hacer referencia al nombre de slot una única vez. Por ejemplo:

  ```
  {pizzaSize} please
  ```

- Cada bot puede tener un máximo de 200 000 caracteres para la combinación de intenciones y enunciados de slot.

- No puede proporcionar enunciados para intenciones que surjan de las intenciones integradas. Para todas las demás intenciones deberá proporcionar al menos un enunciado de muestra. Las intenciones contienen slots, pero los enunciados de muestra del nivel de slot son opcionales.

- Intenciones integradas
  - En la actualidad, Amazon Lex no admite la obtención de ranuras para intenciones integradas. No es posible crear funciones de Lambda para devolver la directiva `ElicitSlot` en la respuesta cuando la intención se deriva de las intenciones integradas. Para obtener más información, consulte Formato de respuesta.
  - El servicio no admite añadir enunciados de muestra a intenciones integradas. Del mismo modo, no se pueden añadir o eliminar slots en intenciones integradas.

- Puede crear hasta 1 000 intenciones para cada cuenta de AWS. Puede crear hasta 100 slots en una intención.

Cuotas de tipo de slot

- Al crearlos, los nombres del tipo de slot no distinguen entre minúsculas y mayúsculas. Si crea el tipo de slot `PizzaSize` y, a continuación, intenta crear de nuevo el tipo de slot `pizzaSize`, obtendrá un error. Sin embargo, cuando se accede a estos recursos, los nombres del recurso sí distinguen entre mayúsculas y minúsculas (debe especificar `PizzaSize` y no `pizzaSize`). Los nombres deben tener entre 1 y 100 caracteres ASCII.

- El tipo de slot personalizado que cree puede tener un máximo de 10 000 valores de enumeración y sinónimos. Cada valor puede tener una longitud de hasta 140 caracteres UTF-8. Los valores de enumeración y sinónimos no pueden contener duplicados.

- Para obtener un valor de tipo de slot, cuando sea necesario, especifique las mayúsculas y las minúsculas. Por ejemplo, para un tipo de slot llamado `Procedure`, si el valor es `MRI`, especifique los valores "MRI" y "mri".

- Tipos de ranura integrados: en la actualidad, Amazon Lex no permite añadir valores de enumeración o sinónimos para los tipos de ranura integrados.

Referencia de la API

Esta sección proporciona documentación sobre las operaciones de la API de Amazon Lex. Para ver una lista de las regiones de AWS en las que Amazon Lex está disponible, consulte Regiones y puntos de conexión en la Referencia general de Amazon Web Services.

Temas

Acciones

Amazon Lex Model Building Service admite las siguientes acciones:

- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents
- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

Amazon Lex Runtime Service admite las siguientes acciones:

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Amazon Lex Model Building Service

Amazon Lex Model Building Service admite las siguientes acciones:

- 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

Servicio: Amazon Lex Model Building Service

Crea una nueva versión del bot basada en la versión `$LATEST`. Si la versión `$LATEST` de este recurso no ha cambiado desde que creó la última versión, Amazon Lex no crea una nueva versión. Devuelve la última versión creada.

> **Note**
>
> Puede actualizar únicamente la versión `$LATEST` del bot. No puede actualizar las versiones numeradas que ha creado con la operación `CreateBotVersion`.

Al crear la primera versión de un bot, Amazon Lex establece la versión en 1. Las versiones subsiguientes se incrementan en 1. Para obtener más información, consulte Control de versiones.

Esta operación necesita permiso para la acción `lex:CreateBotVersion`.

Sintaxis de la solicitud

```
POST /bots/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del bot a partir del cual desea crear una nueva versión. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

checksum

Identifica una revisión especifica de la versión `$LATEST` del bot. Si especifica una suma de comprobación y la versión `$LATEST` del bot tiene una suma de comprobación diferente, se devuelve una excepción `PreconditionFailedException` y Amazon Lex no publica una nueva versión. Si no especifica una suma de comprobación, Amazon Lex publica la versión `$LATEST`.

Tipo: cadena

Requerido: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 201.

El servicio devuelve los datos siguientes en formato JSON.

abortStatement

El mensaje que Amazon Lex utiliza para cancelar una conversación. Para obtener más información, consulte PutBot.

Tipo: objeto Statement

checksum

La suma de comprobación que identifica la versión del bot que se ha creado.

Tipo: cadena

childDirected

Para cada bot de Amazon Lex creado con el Servicio de creación de modelos de Amazon Lex, debe especificar si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro

tipo de aplicación que esté dirigido, total o parcialmente, a niños menores de 13 años y esté sujeto a la Ley de protección de la privacidad infantil en línea (COPPA, por sus siglas en inglés). Para ello, especifique `true` o `false` en el campo `childDirected`. Al especificar `true` en el campo `childDirected`, confirma que el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. Al especificar `false` en el campo `childDirected`, confirma que el uso de Amazon Lex no está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. No puede especificar un valor predeterminado en el campo `childDirected` que no indique de forma precisa si el uso de Amazon Lex está relacionado o no con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA.

Si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años, debe obtener un consentimiento parental verificable, obligatorio en virtud de la COPPA. Para obtener información acerca del uso de Amazon Lex con relación a sitios web, programas u otras aplicaciones dirigidos, total o parcialmente, a niños menores de 13 años, consulte las Preguntas frecuentes de Amazon Lex.

Tipo: Booleano

clarificationPrompt

El mensaje que Amazon Lex utiliza cuando no comprende la solicitud del usuario. Para obtener más información, consulte PutBot.

Tipo: objeto Prompt

createdDate

La fecha en que se creó la versión del bot.

Tipo: marca temporal

description

La descripción del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

detectSentiment

Indica si los enunciados que ha introducido el usuario deben enviarse a Amazon Comprehend con fines de análisis de opiniones.

Tipo: Booleano

enableModelImprovements

Indica si el bot utiliza mejoras en la precisión. `true` indica que el bot utiliza las mejoras y `false`, lo contrario.

Tipo: Booleano

failureReason

Si `status` es `FAILED`, Amazon Lex proporciona el motivo por el que no se ha podido crear el bot.

Tipo: cadena

idleSessionTTLInSeconds

El tiempo máximo que Amazon Lex retiene los datos recopilados en una conversación, en segundos. Para obtener más información, consulte PutBot.

Tipo: entero

Rango válido: valor mínimo de 60. Valor máximo de 86 400.

intents

Una matriz de objetos `Intent`. Para obtener más información, consulte PutBot.

Tipo: matriz de objetos Intent

lastUpdatedDate

La fecha en la que se actualizó la versión `$LATEST` de este bot.

Tipo: marca temporal

locale

Especifica la configuración regional de destino para el bot.

Tipo: cadena

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

El nombre del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

status

Cuando envía una solicitud para crear o actualizar un bot, Amazon Lex establece el elemento de respuesta `status` en `BUILDING`. Una vez que Amazon Lex ha creado el bot, se establece `status` en `READY`. Si Amazon Lex no puede crear el bot, se establece `status` en `FAILED`. Amazon Lex devuelve el motivo del error en el elemento de respuesta `failureReason`.

Tipo: cadena

Valores válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

La versión del bot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

voiceId

El ID de voz de Amazon Polly que Amazon Lex utiliza para las interacciones de voz con el usuario.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

CreateIntentVersion

Servicio: Amazon Lex Model Building Service

Crea una nueva versión de una intención basada en la versión `$LATEST` de la intención. Si la versión `$LATEST` de esta intención no ha cambiado desde que se actualizó por última vez, Amazon Lex no crea una nueva versión. Devuelve la última versión creada.

> **Note**
>
> Puede actualizar únicamente la versión `$LATEST` de la intención. No puede actualizar las versiones numeradas que ha creado con la operación `CreateIntentVersion`.

Al crear una versión de una intención, Amazon Lex establece la versión en 1. Las versiones subsiguientes se incrementan en 1. Para obtener más información, consulte Control de versiones.

Esto operación requiere permisos para realizar la acción `lex:CreateIntentVersion`.

Sintaxis de la solicitud

```
POST /intents/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de la intención a partir del cual desea crear una nueva versión. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

checksum

La suma de comprobación de la versión $LATEST de la intención que se debe utilizar para crear la nueva versión. Si especifica una suma de comprobación y la versión $LATEST de la intención tiene una suma de comprobación diferente, Amazon Lex devuelve una excepción PreconditionFailedException y no publica una nueva versión. Si no especifica una suma de comprobación, Amazon Lex publica la versión $LATEST.

Tipo: cadena

Requerido: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 201.

El servicio devuelve los datos siguientes en formato JSON.

checksum

La suma de comprobación de la versión de la intención creada.

Tipo: cadena

conclusionStatement

Una vez que la función de Lambda especificada en el campo `fulfillmentActivity` cumple con la intención, Amazon Lex transmite esta afirmación al usuario.

Tipo: objeto Statement

confirmationPrompt

Si se define, la solicitud que Amazon Lex utiliza para confirmar la intención del usuario antes de cumplirla.

Tipo: objeto Prompt

createdDate

La fecha en que se creó la intención.

Tipo: marca temporal

description

Una descripción de la intención.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

dialogCodeHook

Si está definida, Amazon Lex invoca esta función de Lambda para cada entrada del usuario.

Tipo: objeto CodeHook

followUpPrompt

Si se define, Amazon Lex utiliza esta solicitud para solicitar actividad adicional del usuario una vez que se ha cumplido con la intención.

Tipo: objeto FollowUpPrompt

fulfillmentActivity

Describe cómo se cumple con la intención.

Tipo: objeto FulfillmentActivity

inputContexts

Una matriz de objetos `InputContext` que enumera los contextos que deben estar activos para que Amazon Lex elija la intención en una conversación con el usuario.

Tipo: matriz de objetos InputContext

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 5 artículos.

kendraConfiguration

Información sobre la configuración necesaria, si procede, para conectar un índice de Amazon Kendra con la intención `AMAZON.KendraSearchIntent`.

Tipo: objeto KendraConfiguration

lastUpdatedDate

La fecha de actualización de la intención.

Tipo: marca temporal

name

El nombre de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

outputContexts

Una matriz de objetos `OutputContext` que enumera los contextos en los que la intención se activa cuando esta se cumple.

Tipo: matriz de objetos OutputContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

parentIntentSignature

Un identificador único para una intención integrada.

Tipo: cadena

rejectionStatement

Si el usuario responde “no” a la pregunta definida en `confirmationPrompt`, Amazon Lex responde con esta afirmación para confirmar que la intención se ha cancelado.

Tipo: objeto Statement

sampleUtterances

Una matriz de ejemplos de enunciados configurados para la intención.

Tipo: matriz de cadenas

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 1500 elementos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 200 caracteres.

slots

Una matriz de tipos de ranura que define la información necesaria para cumplir con la intención.

Tipo: matriz de objetos Slot

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 100 artículos.

version

El número de versión asignado a la nueva versión de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

CreateSlotTypeVersion

Servicio: Amazon Lex Model Building Service

Crea una nueva versión de un tipo de ranura basada en la versión `$LATEST` del tipo de ranura especificado. Si la versión `$LATEST` de este recurso no ha cambiado desde que creó la última versión, Amazon Lex no crea una nueva versión. Devuelve la última versión creada.

> **Note**
>
> Solo puede actualizar la versión `$LATEST` de un tipo de ranura. No puede actualizar las versiones numeradas que ha creado con la operación `CreateSlotTypeVersion`.

Al crear una versión de un tipo de ranura, Amazon Lex establece la versión en 1. Las versiones subsiguientes se incrementan en 1. Para obtener más información, consulte Control de versiones.

Esta operación necesita permisos para la acción `lex:CreateSlotTypeVersion`.

Sintaxis de la solicitud

```
POST /slottypes/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del tipo de ranura a partir del cual desea crear una nueva versión. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

checksum

La suma de comprobación de la versión $LATEST del tipo de ranura que desea publicar. Si especifica una suma de comprobación y la versión $LATEST del tipo de ranura tiene una suma de comprobación diferente, Amazon Lex devuelve una excepción PreconditionFailedException y no publica la nueva versión. Si no especifica una suma de comprobación, Amazon Lex publica la versión $LATEST.

Tipo: cadena

Requerido: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 201.

El servicio devuelve los datos siguientes en formato JSON.

checksum

La suma de comprobación de la versión `$LATEST` del tipo de ranura.

Tipo: cadena

createdDate

La fecha de creación del tipo de ranura.

Tipo: marca temporal

description

Una descripción del tipo de slot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

enumerationValues

Una lista de objetos `EnumerationValue` que define los valores que puede tomar el tipo de ranura.

Tipo: matriz de objetos EnumerationValue

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 10 000 elementos.

lastUpdatedDate

La fecha de actualización del tipo de ranura. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre del tipo de slot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

El tipo de ranura integrado que se utiliza como principal para este tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Información sobre la configuración que amplía el tipo de ranura integrado principal.

Tipo: matriz de objetos SlotTypeConfiguration

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

valueSelectionStrategy

La estrategia que Amazon Lex utiliza para determinar el valor de la ranura. Para obtener más información, consulte PutSlotType.

Tipo: cadena

Valores válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

La versión asignada a la nueva versión del tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBot

Servicio: Amazon Lex Model Building Service

Elimina todas las versiones del bot, incluida la versión `$LATEST`. Para eliminar una versión específica del bot, utilice la operación DeleteBotVersion. La operación `DeleteBot` no elimina de inmediato el esquema de bots. Lo que hace es marcarlos para su eliminación y los elimina más adelante.

Amazon Lex almacena los enunciados de forma indefinida para mejorar la capacidad de respuesta a entradas del usuario del bot. Estos enunciados no se eliminan cuando se elimina el bot. Para eliminar los enunciados, utilice la operación DeleteUtterances.

Si un bot tiene un alias, no se puede eliminar. En su lugar, la operación `DeleteBot` devuelve una excepción `ResourceInUseException` que incluye una referencia al alias que, a su vez, hace referencia al bot. Para eliminar la referencia al bot, elimine el alias. Si vuelve a aparecer la misma excepción, elimine el alias de referencia hasta que la operación `DeleteBot` se realice correctamente.

Esta operación necesita permisos para la acción `lex:DeleteBot`.

Sintaxis de la solicitud

```
DELETE /bots/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del bot. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBotAlias

Servicio: Amazon Lex Model Building Service

Elimina un alias para el bot especificado.

No puede eliminar un alias que se utiliza en la asociación entre un bot y un canal de mensajería. Si se utiliza un alias en una asociación de canal, la operación `DeleteBot` devuelve una excepción `ResourceInUseException` que incluye una referencia a la asociación de canal que, a su vez, hace referencia al bot. Para eliminar la referencia al alias, elimine la asociación de canal. Si vuelve a aparecer la misma excepción, elimine la asociación de la referencia hasta que la operación `DeleteBotAlias` se realice correctamente.

Sintaxis de la solicitud

```
DELETE /bots/botName/aliases/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botName

El nombre del bot al que apunta el alias.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

name

El nombre del alias que se va a eliminar. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBotChannelAssociation

Servicio: Amazon Lex Model Building Service

Elimina la asociación entre un bot de Amazon Lex y una plataforma de mensajería.

Esta operación necesita permiso para la acción `lex:DeleteBotChannelAssociation`.

Sintaxis de la solicitud

```
DELETE /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

aliasName

Un alias que apunta a la versión exacta del bot de Amazon Lex al que se vincula esta asociación.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

botName

El nombre del bot de Amazon Lex.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

name

El nombre de la asociación. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteBotVersion

Servicio: Amazon Lex Model Building Service

Elimina una versión específica de un bot. Para eliminar todas las versiones de un bot, utilice la operación DeleteBot.

Esta operación necesita permisos para la acción `lex:DeleteBotVersion`.

Sintaxis de la solicitud

```
DELETE /bots/name/versions/version HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

version

La versión del bot que se va a eliminar. No puede eliminar la versión `$LATEST` del bot. Para eliminar la versión `$LATEST`, utilice la operación DeleteBot.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteIntent

Servicio: Amazon Lex Model Building Service

Elimina todas las versiones de la intención, incluida la versión `$LATEST`. Para eliminar una versión específica de la intención, utilice la operación DeleteIntentVersion.

Puede eliminar una versión de una intención solo si no tiene referencias. Para eliminar una intención a la que se hace referencia en uno o más bots (consulte Funcionamiento de Amazon Lex), debe eliminar primero las referencias.

> **Note**
>
> Si se muestra la excepción `ResourceInUseException`, proporciona un ejemplo de referencia que indica dónde se hace referencia a la intención. Para eliminar la referencia a la intención, actualice el bot o elimínelo. Si se muestra la misma excepción cuando intenta eliminar la intención, repita la acción hasta que la intención no tenga referencias y la llamada a `DeleteIntent` sea correcta.

Esta operación necesita permiso para la acción `lex:DeleteIntent`.

Sintaxis de la solicitud

```
DELETE /intents/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de la intención. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteIntentVersion

Servicio: Amazon Lex Model Building Service

Elimina una versión específica de una intención. Para eliminar todas las versiones de una intención, utilice la operación DeleteIntent.

Esta operación necesita permisos para la acción `lex:DeleteIntentVersion`.

Sintaxis de la solicitud

```
DELETE /intents/name/versions/version HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de la intención.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

version

La versión de la intención que se va a eliminar. No puede eliminar la versión `$LATEST` de la intención. Para eliminar la versión `$LATEST`, utilice la operación DeleteIntent.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteSlotType

Servicio: Amazon Lex Model Building Service

Elimina todas las versiones del tipo de ranura, incluida la versión `$LATEST`. Para eliminar una versión específica del tipo de ranura, utilice la operación DeleteSlotTypeVersion.

Puede eliminar una versión de un tipo de ranura solo si no tiene referencias. Para eliminar un tipo de ranura al que se hace referencia en una o más intenciones, debe eliminar primero las referencias.

> **Note**
>
> Si se muestra la excepción `ResourceInUseException`, la excepción proporciona un ejemplo de referencia que indica la intención en la que se hace referencia al tipo de ranura. Para eliminar la referencia al tipo de ranura, actualice la intención o elimínela. Si se muestra la misma excepción cuando intenta eliminar el tipo de ranura, repita la acción hasta que la intención no tenga referencias y la llamada a `DeleteSlotType` sea correcta.

Esta operación necesita permiso para la acción `lex:DeleteSlotType`.

Sintaxis de la solicitud

```
DELETE /slottypes/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del tipo de slot. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteSlotTypeVersion

Servicio: Amazon Lex Model Building Service

Elimina una versión específica de un tipo de ranura. Para eliminar todas las versiones de un tipo de ranura, utilice la operación DeleteSlotType.

Esta operación necesita permisos para la acción `lex:DeleteSlotTypeVersion`.

Sintaxis de la solicitud

```
DELETE /slottypes/name/version/version HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del tipo de slot.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

version

La versión del tipo de ranura que se va a eliminar. No puede eliminar la versión `$LATEST` del tipo de ranura. Para eliminar la versión `$LATEST`, utilice la operación DeleteSlotType.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

ResourceInUseException

Otro recurso hace referencia al recurso que intenta eliminar. Utilice esta información para eliminar referencias al recurso que intenta eliminar.

El cuerpo de la excepción contiene un objeto JSON que describe el recurso.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {

"name": string, "version": string } }
```

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteUtterances

Servicio: Amazon Lex Model Building Service

Elimina los enunciados almacenados.

Amazon Lex almacena los enunciados que los usuarios envían al bot. Los enunciados se almacenan durante 15 días para su uso con la operación GetUtterancesView y, a continuación, se almacenan indefinidamente con el fin de mejorar la capacidad del bot de responder a entradas del usuario.

Utilice la operación `DeleteUtterances` para eliminar manualmente los enunciados almacenados de un usuario específico. Al utilizar la operación `DeleteUtterances`, los enunciados almacenados con el fin de mejorar la capacidad del bot de responder a entradas del usuario se eliminan de inmediato. Los enunciados almacenados para su uso con la operación `GetUtterancesView` se eliminan una vez transcurridos 15 días.

Esta operación necesita permisos para la acción `lex:DeleteUtterances`.

Sintaxis de la solicitud

```
DELETE /bots/botName/utterances/userId HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botName

El nombre del bot que ha almacenado los enunciados.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

userId

El identificador único del usuario que ha hecho los enunciados. Este es el ID de usuario que se envió en la solicitud de PostTextoperación PostContento que contenía el enunciado.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBot

Servicio: Amazon Lex Model Building Service

Devuelve información sobre los metadatos de un bot específico. Debe proporcionar el nombre del bot y la versión del bot o el alias.

Esta operación necesita permisos para la acción `lex:GetBot`.

Sintaxis de la solicitud

```
GET /bots/name/versions/versionoralias HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del bot. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

versionoralias

La versión o el alias del bot.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

abortStatement

El mensaje que Amazon Lex devuelve cuando el usuario elige acabar con la conversación sin completarla. Para obtener más información, consulte PutBot.

Tipo: objeto Statement

checksum

La suma de comprobación del bot que se utiliza para identificar una revisión específica de la versión `$LATEST` del bot.

Tipo: cadena

childDirected

Para cada bot de Amazon Lex creado con el Servicio de creación de modelos de Amazon Lex, debe especificar si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que esté dirigido, total o parcialmente, a niños menores de 13 años y esté sujeto a la Ley de protección de la privacidad infantil en línea (COPPA, por sus siglas en inglés). Para ello, especifique `true` o `false` en el campo `childDirected`. Al especificar `true` en el campo `childDirected`, confirma que el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. Al especificar `false` en el campo `childDirected`, confirma que el uso de Amazon Lex no está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. No puede especificar un valor predeterminado en el campo `childDirected` que no indique de forma precisa si el uso de Amazon Lex está relacionado o no con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA.

Si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años, debe obtener un consentimiento parental verificable, obligatorio en virtud de la COPPA. Para obtener información acerca del uso de Amazon Lex con relación a sitios web, programas u otras aplicaciones

dirigidos, total o parcialmente, a niños menores de 13 años, consulte las Preguntas frecuentes de Amazon Lex.

Tipo: Booleano

clarificationPrompt

El mensaje que utiliza Amazon Lex cuando no entiende la solicitud del usuario. Para obtener más información, consulte PutBot.

Tipo: objeto Prompt

createdDate

La fecha de creación del bot.

Tipo: marca temporal

description

La descripción del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

detectSentiment

Indica si los enunciados del usuario deben enviarse a Amazon Comprehend con fines de análisis de opiniones.

Tipo: Booleano

enableModelImprovements

Indica si el bot utiliza mejoras en la precisión. `true` indica que el bot utiliza las mejoras y `false`, lo contrario.

Tipo: Booleano

failureReason

Si `status` es `FAILED`, Amazon Lex explica por qué no se ha podido crear el bot.

Tipo: cadena

idleSessionTTLInSeconds

El tiempo máximo que Amazon Lex retiene los datos recopilados en una conversación, en segundos. Para obtener más información, consulte PutBot.

Tipo: entero

Rango válido: valor mínimo de 60. Valor máximo de 86 400.

intents

Una matriz de objetos `intent`. Para obtener más información, consulte PutBot.

Tipo: matriz de objetos Intent

lastUpdatedDate

La fecha de actualización del bot. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

locale

La configuración regional de destino para el bot.

Tipo: cadena

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

El nombre del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

La puntuación que determina dónde inserta Amazon Lex `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`, o ambas, al devolver intenciones

alternativas en una PostTextrespuesta PostContento. AMAZON.FallbackIntentse inserta si la puntuación de confianza de todos los intentos está por debajo de este valor. AMAZON.KendraSearchIntentsolo se inserta si está configurado para el bot.

Tipo: Doble

Rango válido: valor mínimo de 0. El valor máximo es de 1.

status

El estado del bot.

Cuando el estado es BUILDING, Amazon Lex está creando el bot para probarlo y usarlo.

Si el estado del bot es READY_BASIC_TESTING, puede probar el bot con los enunciados exactos que se especifican en las intenciones del bot. Cuando el bot está listo para probarse definitivamente o para ejecutarse, el estado es READY.

Si ha habido un problema al crear el bot, el estado es FAILED y en el campo failureReason se explica por qué no se ha compilado el bot.

Si el bot se ha guardado sin compilar, el estado es NOT_BUILT.

Tipo: cadena

Valores válidos: BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

version

La versión del bot. En el caso de un bot nuevo, la versión es siempre $LATEST.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: \$LATEST|[0-9]+

voiceId

El ID de voz de Amazon Polly que Amazon Lex utiliza para la interacción de voz con el usuario. Para obtener más información, consulte PutBot.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetBotAlias

Servicio: Amazon Lex Model Building Service

Devuelve información sobre un alias de bot de Amazon Lex. Para obtener más información acerca de los alias, consulte Control de versiones y alias.

Esta operación necesita permisos para la acción `lex:GetBotAlias`.

Sintaxis de la solicitud

```
GET /bots/botName/aliases/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botName

El nombre del bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

name

El nombre del alias del bot. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

botName

El nombre del bot al que apunta el alias.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

botVersion

La versión del bot a la que apunta el alias.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

checksum

La suma de comprobación del alias del bot.

Tipo: cadena

conversationLogs

La configuración que determina cómo Amazon Lex utiliza los registros de conversaciones para el alias.

Tipo: objeto ConversationLogsResponse

createdDate

La fecha de creación del alias del bot.

Tipo: marca temporal

description

Una descripción del alias del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

lastUpdatedDate

La fecha de actualización del alias del bot. Al crear un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre del alias del bot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetBotAliases

Servicio: Amazon Lex Model Building Service

Devuelve una lista de alias para un bot especificado de Amazon Lex.

Esta operación necesita permisos para la acción `lex:GetBotAliases`.

Sintaxis de la solicitud

```
GET /bots/botName/aliases/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botName

El nombre del bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

maxResults

El número máximo de alias que se devuelven en una respuesta. El valor predeterminado es 50.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nameContains

La subcadena que debe coincidir con los nombres de alias del bot. Se devolverá un alias si alguna parte del nombre coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nextToken

Un token de paginación para obtener la siguiente página de alias. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de alias, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

BotAliases

Una matriz de objetos `BotAliasMetadata` en la que cada uno describe un alias de bot.

Tipo: matriz de objetos BotAliasMetadata

nextToken

Un token de paginación para obtener la siguiente página de alias. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de alias, especifique el token de paginación en la siguiente solicitud.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBotChannelAssociation

Servicio: Amazon Lex Model Building Service

Devuelve información sobre la asociación entre un bot de Amazon Lex y una plataforma de mensajería.

Esta operación necesita permisos para la acción `lex:GetBotChannelAssociation`.

Sintaxis de la solicitud

```
GET /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

aliasName

Un alias que apunta a la versión exacta del bot de Amazon Lex al que se vincula esta asociación.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

botName

El nombre del bot de Amazon Lex.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

name

El nombre de la asociación entre el bot y el canal. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: ^([A-Za-z]_?)+$

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

botAlias

Un alias que apunta a la versión exacta del bot de Amazon Lex al que se vincula esta asociación.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: ^([A-Za-z]_?)+$

botConfiguration

Proporciona la información que necesita la plataforma de mensajería para comunicarse con el bot de Amazon Lex.

Tipo: mapa de cadena a cadena

Entradas de mapa: número máximo de 10 elementos.

botName

El nombre del bot de Amazon Lex.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

createdDate

La fecha en que se creó la asociación entre el bot y el canal.

Tipo: marca temporal

description

Una descripción de la asociación entre el bot y el canal.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

failureReason

Si `status` es `FAILED`, Amazon Lex proporciona el motivo por el que no se ha podido crear la asociación.

Tipo: cadena

name

El nombre de la asociación entre el bot y el canal.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

status

El estado del canal del bot.

- `CREATED`: el canal se ha creado y está listo para usarse.
- `IN_PROGRESS`: el canal se está creando.
- `FAILED`: se ha producido un error al crear el canal. Para obtener información acerca del motivo del error, consulte el campo `failureReason`.

Tipo: cadena

Valores válidos: `IN_PROGRESS | CREATED | FAILED`

type

El tipo de plataforma de mensajería.

Tipo: cadena

Valores válidos: `Facebook | Slack | Twilio-Sms | Kik`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBotChannelAssociations

Servicio: Amazon Lex Model Building Service

Devuelve una lista de todos los canales asociados con el bot especificado.

La operación `GetBotChannelAssociations` necesita permisos para la acción `lex:GetBotChannelAssociations`.

Sintaxis de la solicitud

```
GET /bots/botName/aliases/aliasName/channels/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

aliasName

Un alias que apunta a la versión exacta del bot de Amazon Lex al que se vincula esta asociación.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^(-|^([A-Za-z]_?)+$)$`

Obligatorio: sí

botName

El nombre del bot de Amazon Lex en la asociación.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

maxResults

El número máximo de asociaciones que se devuelven en una respuesta. El valor predeterminado es 50.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nameContains

La subcadena que debe coincidir con los nombres de asociaciones de canal. Se devolverá una asociación si alguna parte del nombre coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”. Para mostrar todas las asociaciones de canales de bots, utilice un guion (“-”) como parámetro `nameContains`.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nextToken

Un token de paginación para obtener la siguiente página de asociaciones. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de asociaciones, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

botChannelAssociations

Una matriz de objetos, uno para cada asociación, que proporciona información acerca del bot de Amazon Lex y su asociación con el canal.

Tipo: matriz de objetos BotChannelAssociation

nextToken

Un token de paginación que obtiene la siguiente página de asociaciones. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de asociaciones, especifique el token de paginación en la siguiente solicitud.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBots

Servicio: Amazon Lex Model Building Service

Devuelve información del bot de la siguiente forma:

- Si proporciona el campo `nameContains`, la respuesta incluye información sobre la versión `$LATEST` de todos los bots cuyo nombre contiene la cadena especificada.
- Si no especifica el campo `nameContains`, la operación devuelve información sobre la versión `$LATEST` de todos los bots.

Esta operación necesita permiso para la acción `lex:GetBots`.

Sintaxis de la solicitud

```
GET /bots/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de bots que se devuelve en la respuesta de la solicitud. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nameContains

La subcadena que debe coincidir con los nombres de bots. Se devolverá un bot si alguna parte del nombre coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nextToken

Un token de paginación que obtiene la siguiente página de bots. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de bots, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

bots

Una matriz de objetos `botMetadata`, con una entrada para cada bot.

Tipo: matriz de objetos BotMetadata

nextToken

Si la respuesta está truncada, incluye un token de paginación que puede especificar en su próxima solicitud para obtener la siguiente página de bots.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetBotVersions

Servicio: Amazon Lex Model Building Service

Obtiene información sobre todas las versiones de un bot.

La operación `GetBotVersions` devuelve un objeto `BotMetadata` para cada versión de un bot. Por ejemplo, si un bot tiene tres versiones numeradas, la operación `GetBotVersions` devuelve cuatro objetos `BotMetadata` en la respuesta, uno para cada versión numerada y uno para la versión `$LATEST`.

La operación `GetBotVersions` siempre devuelve al menos una versión, la versión `$LATEST`.

Esta operación necesita permisos para la acción `lex:GetBotVersions`.

Sintaxis de la solicitud

```
GET /bots/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de versiones del bot que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

name

El nombre del bot del que deben devolverse las versiones.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

nextToken

Un token de paginación para obtener la siguiente página de versiones del bot. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta.

Para obtener la siguiente página de versiones, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

bots

Una matriz de objetos `BotMetadata`, uno para cada versión numerada del bot, así como uno para la versión `$LATEST`.

Tipo: matriz de objetos BotMetadata

nextToken

Un token de paginación para obtener la siguiente página de versiones del bot. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta.

Para obtener la siguiente página de versiones, especifique el token de paginación en la siguiente solicitud.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBuiltinIntent

Servicio: Amazon Lex Model Building Service

Devuelve información sobre una intención integrada.

Esta operación necesita permiso para la acción `lex:GetBuiltinIntent`.

Sintaxis de la solicitud

```
GET /builtins/intents/signature HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

signature

Un identificador único para una intención integrada. Para encontrar la firma de una intención, consulte Intenciones integradas estándar en Alexa Skills Kit.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "signature": "string",
   "slots": [
      {
         "name": "string"
      }
   ],
   "supportedLocales": [ "string" ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

signature

Un identificador único para una intención integrada.

Tipo: cadena

slots

Una matriz de objetos `BuiltinIntentSlot`, con una entrada para cada tipo de ranura en la intención.

Tipo: matriz de objetos BuiltinIntentSlot

supportedLocales

Una lista con las configuraciones regionales que admite la intención.

Tipo: matriz de cadenas

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`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBuiltinIntents

Servicio: Amazon Lex Model Building Service

Obtiene una lista de las intenciones integradas que cumplen los criterios especificados.

Esta operación necesita permiso para la acción `lex:GetBuiltinIntents`.

Sintaxis de la solicitud

```
GET /builtins/intents/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

locale

Una lista con las configuraciones regionales que admite la intención.

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

El número máximo de intenciones que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nextToken

Un token de paginación que obtiene la siguiente página de intenciones. Si esta llamada a la API está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de intenciones, utilice el token de paginación en la siguiente solicitud.

signatureContains

La subcadena que debe coincidir con las firmas de intenciones integradas. Se devolverá una intención si alguna parte de su firma coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”. Para encontrar la firma de una intención, consulte Intenciones integradas estándar en Alexa Skills Kit.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ],
   "nextToken": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

intents

Una matriz de objetos `builtinIntentMetadata`, uno para cada intención en la respuesta.

Tipo: matriz de objetos BuiltinIntentMetadata

nextToken

Un token de paginación que obtiene la siguiente página de intenciones. Si la respuesta a esta llamada a la API está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de intenciones, especifique el token de paginación en la siguiente solicitud.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBuiltinSlotTypes

Servicio: Amazon Lex Model Building Service

Obtiene una lista de tipos de ranura integrados que cumplen los criterios especificados.

Para obtener una lista con los tipos de ranura integrados, consulte Referencia del tipo de ranura en Alexa Skills Kit.

Esta operación necesita permiso para la acción `lex:GetBuiltInSlotTypes`.

Sintaxis de la solicitud

```
GET /builtins/slottypes/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

locale

Una lista con las configuraciones regionales que admite el tipo de ranura.

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

El número máximo de tipos de ranura que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nextToken

Un token de paginación que obtiene la siguiente página de tipos de ranura. Si la respuesta a esta llamada a la API está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de tipos de ranura, especifique el token de paginación en la siguiente solicitud.

signatureContains

La subcadena que debe coincidir con las firmas de tipos de ranura integrados. Se devolverá un tipo de ranura si alguna parte de su firma coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

nextToken

Si la respuesta está truncada, la respuesta incluye un token de paginación que puede utilizar en su próxima solicitud para obtener la siguiente página de tipos de ranura.

Tipo: cadena

slotTypes

Una matriz de objetos `BuiltInSlotTypeMetadata`, con una entrada para cada tipo de ranura devuelto.

Tipo: matriz de objetos BuiltinSlotTypeMetadata

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetExport

Servicio: Amazon Lex Model Building Service

Exporta el contenido de un recurso de Amazon Lex en un formato específico.

Sintaxis de la solicitud

```
GET /exports/?exportType=exportType&name=name&resourceType=resourceType&version=version
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

exportType

El formato de los datos exportados.

Valores válidos: `ALEXA_SKILLS_KIT | LEX`

Obligatorio: sí

name

El nombre del bot que se va a exportar.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `[a-zA-Z_]+`

Obligatorio: sí

resourceType

El tipo de recurso que se va a exportar.

Valores válidos: `BOT | INTENT | SLOT_TYPE`

Obligatorio: sí

version

La versión del bot que se va a exportar.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "exportStatus": "string",
   "exportType": "string",
   "failureReason": "string",
   "name": "string",
   "resourceType": "string",
   "url": "string",
   "version": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

exportStatus

El estado de la exportación.

- `IN_PROGRESS`: la exportación está en curso.
- `READY`: la exportación ha finalizado.
- `FAILED`: no se ha podido completar la exportación.

Tipo: cadena

Valores válidos: `IN_PROGRESS | READY | FAILED`

exportType

El formato de los datos exportados.

Tipo: cadena

Valores válidos: `ALEXA_SKILLS_KIT | LEX`

failureReason

Si `status` es `FAILED`, Amazon Lex proporciona el motivo por el que no se ha podido exportar el recurso.

Tipo: cadena

name

El nombre del bot que se está exportando.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `[a-zA-Z_]+`

resourceType

El tipo del recurso exportado.

Tipo: cadena

Valores válidos: `BOT | INTENT | SLOT_TYPE`

url

Una dirección URL prefirmada de S3 que proporciona la ubicación del recurso exportado. El recurso exportado es un archivo ZIP que contiene el recurso exportado en formato JSON. La estructura del archivo puede cambiar. El código no tiene que depender de la estructura del archivo.

Tipo: cadena

version

La versión del bot que se está exportando.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetImport

Servicio: Amazon Lex Model Building Service

Obtiene información sobre un trabajo de importación iniciado con la operación `StartImport`.

Sintaxis de la solicitud

```
GET /imports/importId HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

importId

El identificador de la información del trabajo de importación que se va a devolver.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "createdDate": number,
   "failureReason": [ "string" ],
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

createdDate

Una marca de tiempo para la fecha y hora en la que se creó el trabajo de importación.

Tipo: marca temporal

failureReason

Una cadena que describe el motivo por el que no se ha completado un trabajo de importación.

Tipo: matriz de cadenas

importId

El identificador de un trabajo de importación específico.

Tipo: cadena

importStatus

El estado del trabajo de importación. Si el estado es `FAILED`, puede consultar el motivo del fallo en el campo `failureReason`.

Tipo: cadena

Valores válidos: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

La acción que se llevó a cabo cuando se produjo un conflicto entre un recurso existente y un recurso del archivo de importación.

Tipo: cadena

Valores válidos: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

El nombre asignado al trabajo de importación.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `[a-zA-Z_]+`

resourceType

El tipo de recurso importado.

Tipo: cadena

Valores válidos: `BOT | INTENT | SLOT_TYPE`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetIntent

Servicio: Amazon Lex Model Building Service

Devuelve información sobre una intención. Además del nombre de la intención, debe especificar la versión de la intención.

Esto operación requiere permisos para realizar la acción `lex:GetIntent`.

Sintaxis de la solicitud

```
GET /intents/name/versions/version HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de la intención. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

version

La versión de la intención.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

checksum

La suma de comprobación de la intención.

Tipo: cadena

conclusionStatement

Una vez que la función de Lambda especificada en el elemento `fulfillmentActivity` cumple con la intención, Amazon Lex transmite esta afirmación al usuario.

Tipo: objeto Statement

confirmationPrompt

Si se define en el bot, la solicitud que Amazon Lex utiliza para confirmar la intención del usuario antes de cumplirla. Para obtener más información, consulte PutIntent.

Tipo: objeto Prompt

createdDate

La fecha en que se creó la intención.

Tipo: marca temporal

description

Una descripción de la intención.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

dialogCodeHook

Si se define en el bot, Amazon Lex invoca esta función de Lambda para cada entrada del usuario. Para obtener más información, consulte PutIntent.

Tipo: objeto CodeHook

followUpPrompt

Si se define en el bot, Amazon Lex utiliza esta solicitud para solicitar actividad adicional del usuario una vez que se ha cumplido con la intención. Para obtener más información, consulte PutIntent.

Tipo: objeto FollowUpPrompt

fulfillmentActivity

Describe cómo se cumple con la intención. Para obtener más información, consulte PutIntent.

Tipo: objeto FulfillmentActivity

inputContexts

Una matriz de objetos `InputContext` que enumera los contextos que deben estar activos para que Amazon Lex elija la intención en una conversación con el usuario.

Tipo: matriz de objetos InputContext

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 5 artículos.

kendraConfiguration

Información sobre la configuración necesaria, si procede, para conectar un índice de Amazon Kendra con la intención `AMAZON.KendraSearchIntent`.

Tipo: objeto KendraConfiguration

lastUpdatedDate

La fecha de actualización de la intención. Al crear un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

outputContexts

Una matriz de objetos `OutputContext` que enumera los contextos en los que la intención se activa cuando esta se cumple.

Tipo: matriz de objetos OutputContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

parentIntentSignature

Un identificador único para una intención integrada.

Tipo: cadena

rejectionStatement

Si el usuario responde “no” a la pregunta definida en `confirmationPrompt`, Amazon Lex responde con esta afirmación para confirmar que la intención se ha cancelado.

Tipo: objeto Statement

sampleUtterances

Una matriz de ejemplos de enunciados configurados para la intención.

Tipo: matriz de cadenas

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 1500 elementos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 200 caracteres.

slots

Una matriz de ranuras de intención configuradas para la intención.

Tipo: matriz de objetos Slot

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 100 artículos.

version

La versión de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetIntents

Servicio: Amazon Lex Model Building Service

Devuelve información de la intención de la siguiente forma:

- Si especifica el campo `nameContains`, devuelve la versión `$LATEST` de todas las intenciones que contienen la cadena especificada.
- Si no especifica el campo `nameContains`, devuelve información sobre la versión `$LATEST` de todas las intenciones.

La operación necesita permisos para la acción `lex:GetIntents`.

Sintaxis de la solicitud

```
GET /intents/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de intenciones que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nameContains

La subcadena que debe coincidir con los nombres de intenciones. Se devolverá una intención si alguna parte del nombre coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nextToken

Un token de paginación que obtiene la siguiente página de intenciones. Si la respuesta a esta llamada a la API está truncada, Amazon Lex devuelve un token de paginación en la respuesta.

Para obtener la siguiente página de intenciones, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

intents

Una matriz de objetos `Intent`. Para obtener más información, consulte PutBot.

Tipo: matriz de objetos IntentMetadata

nextToken

Si la respuesta está truncada, la respuesta incluye un token de paginación que puede especificar en su próxima solicitud para obtener la siguiente página de intenciones.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetIntentVersions

Servicio: Amazon Lex Model Building Service

Obtiene información sobre todas las versiones de una intención.

La operación `GetIntentVersions` devuelve un objeto `IntentMetadata` para cada versión de una intención. Por ejemplo, si una intención tiene tres versiones numeradas, la operación `GetIntentVersions` devuelve cuatro objetos `IntentMetadata` en la respuesta, uno para cada versión numerada y uno para la versión `$LATEST`.

La operación `GetIntentVersions` siempre devuelve al menos una versión, la versión `$LATEST`.

Esta operación necesita permisos para la acción `lex:GetIntentVersions`.

Sintaxis de la solicitud

```
GET /intents/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de versiones de la intención que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

name

El nombre de la intención de la que deben devolverse las versiones.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

nextToken

Un token de paginación para obtener la siguiente página de versiones de la intención. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la

respuesta. Para obtener la siguiente página de versiones, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

intents

Una matriz de objetos `IntentMetadata`, uno para cada versión numerada de la intención, así como uno para la versión `$LATEST`.

Tipo: matriz de objetos IntentMetadata

nextToken

Un token de paginación para obtener la siguiente página de versiones de la intención. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la

respuesta. Para obtener la siguiente página de versiones, especifique el token de paginación en la siguiente solicitud.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetMigration

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de una migración en curso o finalizada de un bot de Amazon Lex V1 a un bot de Amazon Lex V2. Utilice esta operación para ver las alertas y los avisos relacionados con la migración.

Sintaxis de la solicitud

```
GET /migrations/migrationId HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

migrationId

El identificador único de la migración que se consulta. `migrationID` lo devuelve la operación StartMigration.

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

alerts

Una lista de alertas y advertencias que indican problemas con la migración de un bot de Amazon Lex V1 a Amazon Lex V2. Recibirá una advertencia cuando una característica de Amazon Lex V1 se implemente de forma distinta en Amazon Lex V2.

Para obtener más información, consulte Migración de un bot en la Guía para desarrolladores de Amazon Lex V2.

Tipo: matriz de objetos MigrationAlert

migrationId

El identificador único de la migración. Es el mismo identificador que se utiliza al llamar a la operación `GetMigration`.

Tipo: cadena

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

migrationStatus

Indica el estado de la migración. Cuando el estado es `COMPLETE`, la migración ha finalizado y el bot está disponible en Amazon Lex V2. Es posible que haya alertas y advertencias que deban resolverse antes de completar la migración.

Tipo: cadena

Valores válidos: `IN_PROGRESS | COMPLETED | FAILED`

migrationStrategy

La estrategia empleada para llevar a cabo la migración.

- `CREATE_NEW`: crea un nuevo bot de Amazon Lex V2 y migra el bot de Amazon Lex V1 al nuevo bot.
- `UPDATE_EXISTING`: sobrescribe los metadatos de un bot de Amazon Lex V2 y la configuración regional que se van a migrar. No cambia ninguna otra configuración regional del bot de Amazon Lex V2. Si la configuración regional no existe, se crea una nueva configuración regional en el bot de Amazon Lex V2.

Tipo: cadena

Valores válidos: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

La fecha y hora en que comenzó la migración.

Tipo: marca temporal

v1BotLocale

La configuración regional del bot de Amazon Lex V1 migrado a Amazon Lex V2.

Tipo: cadena

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

El nombre del bot de Amazon Lex V1 migrado a Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

v1BotVersion

La versión del bot de Amazon Lex V1 migrado a Amazon Lex V2.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

v2BotId

El identificador único del bot de Amazon Lex V2 al que se migra desde Amazon Lex V1.

Tipo: cadena

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

v2BotRole

El rol de IAM que Amazon Lex utiliza para ejecutar el bot de Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetMigrations

Servicio: Amazon Lex Model Building Service

Obtiene una lista de las migraciones entre Amazon Lex V1 y Amazon Lex V2.

Sintaxis de la solicitud

```
GET /migrations?
maxResults=maxResults&migrationStatusEquals=migrationStatusEquals&nextToken=nextToken&sortByAtt
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de migraciones que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

migrationStatusEquals

Filtra la lista para que muestre solo las migraciones con el estado especificado.

Valores válidos: `IN_PROGRESS | COMPLETED | FAILED`

nextToken

Un token de paginación que obtiene la siguiente página de migraciones. Si la respuesta a esta operación está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de migraciones, especifique el token de paginación en la solicitud.

sortByAttribute

El campo de referencia para ordenar la lista de migraciones. Puede ordenar la lista por nombre de bot de Amazon Lex V1 o por la fecha y hora en que se inició la migración.

Valores válidos: `V1_BOT_NAME | MIGRATION_DATE_TIME`

sortByOrder

El orden en el que se ordena la lista.

Valores válidos: `ASCENDING | DESCENDING`

v1BotNameContains

Filtra la lista para que muestre solo los bots cuyo nombre contiene la cadena especificada. La cadena debe coincidir con cualquier parte del nombre del bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

migrationSummaries

Una matriz de resúmenes para migraciones de Amazon Lex V1 a Amazon Lex V2. Para ver los detalles de la migración, consulte `migrationId` en el resumen de una llamada a la operación GetMigration.

Tipo: matriz de objetos MigrationSummary

nextToken

Si la respuesta está truncada, incluye un token de paginación que puede especificar en su próxima solicitud para obtener la siguiente página de migraciones.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetSlotType

Servicio: Amazon Lex Model Building Service

Devuelve información sobre una versión específica de un tipo de ranura. Además de especificar el nombre del tipo de ranura, debe especificar la versión del tipo de ranura.

Esta operación necesita permisos para la acción `lex:GetSlotType`.

Sintaxis de la solicitud

```
GET /slottypes/name/versions/version HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del tipo de slot. El nombre distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

version

La versión del tipo de ranura.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

checksum

La suma de comprobación de la versión $LATEST del tipo de ranura.

Tipo: cadena

createdDate

La fecha de creación del tipo de ranura.

Tipo: marca temporal

description

Una descripción del tipo de slot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

enumerationValues

Una lista de objetos `EnumerationValue` que define los valores que puede tomar el tipo de ranura.

Tipo: matriz de objetos EnumerationValue

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 10 000 elementos.

lastUpdatedDate

La fecha de actualización del tipo de ranura. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre del tipo de slot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

El tipo de ranura integrado que se utiliza como principal para este tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Información sobre la configuración que amplía el tipo de ranura integrado principal.

Tipo: matriz de objetos SlotTypeConfiguration

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

valueSelectionStrategy

La estrategia que Amazon Lex utiliza para determinar el valor de la ranura. Para obtener más información, consulte PutSlotType.

Tipo: cadena

Valores válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

La versión del tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetSlotTypes

Servicio: Amazon Lex Model Building Service

Devuelve información del tipo de ranura de la siguiente forma:

- Si especifica el campo `nameContains`, devuelve la versión `$LATEST` de todos los tipos de ranura que contienen la cadena especificada.
- Si no especifica el campo `nameContains`, devuelve información sobre la versión `$LATEST` de todos los tipos de ranura.

La operación necesita permisos para la acción `lex:GetSlotTypes`.

Sintaxis de la solicitud

```
GET /slottypes/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de tipos de ranura que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

nameContains

La subcadena que debe coincidir con los nombres de tipos de ranura. Se devolverá un tipo de ranura si alguna parte del nombre coincide con la subcadena. Por ejemplo, “xyz” coincide con “xyzabc” y “abcxyz”.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nextToken

Un token de paginación que obtiene la siguiente página de tipos de ranura. Si la respuesta a esta llamada a la API está truncada, Amazon Lex devuelve un token de paginación en la respuesta.

Para obtener la siguiente página de tipos de ranura, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

nextToken

Si la respuesta está truncada, incluye un token de paginación que puede especificar en su próxima solicitud para obtener la siguiente página de tipos de ranura.

Tipo: cadena

slotTypes

Una matriz de objetos, uno para cada tipo de ranura, que proporciona datos como el nombre del tipo de ranura, la versión y una descripción.

Tipo: matriz de objetos SlotTypeMetadata

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetSlotTypeVersions

Servicio: Amazon Lex Model Building Service

Obtiene información sobre todas las versiones de un tipo de ranura.

La operación `GetSlotTypeVersions` devuelve un objeto `SlotTypeMetadata` para cada versión de un tipo de ranura. Por ejemplo, si un tipo de ranura tiene tres versiones numeradas, la operación `GetSlotTypeVersions` devuelve cuatro objetos `SlotTypeMetadata` en la respuesta, uno para cada versión numerada y uno para la versión `$LATEST`.

La operación `GetSlotTypeVersions` siempre devuelve al menos una versión, la versión `$LATEST`.

Esta operación necesita permisos para la acción `lex:GetSlotTypeVersions`.

Sintaxis de la solicitud

```
GET /slottypes/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

maxResults

El número máximo de versiones de tipo de ranura que se devuelven en la respuesta. El valor predeterminado es 10.

Rango válido: valor mínimo de 1. Valor máximo de 50.

name

El nombre del tipo de ranura del que deben devolverse las versiones.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

nextToken

Un token de paginación para obtener la siguiente página de versiones del tipo de ranura. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de versiones, especifique el token de paginación en la siguiente solicitud.

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

nextToken

Un token de paginación para obtener la siguiente página de versiones del tipo de ranura. Si la respuesta a esta llamada está truncada, Amazon Lex devuelve un token de paginación en la respuesta. Para obtener la siguiente página de versiones, especifique el token de paginación en la siguiente solicitud.

Tipo: cadena

slotTypes

Una matriz de objetos `SlotTypeMetadata`, uno para cada versión numerada del tipo de ranura, así como uno para la versión `$LATEST`.

Tipo: matriz de objetos SlotTypeMetadata

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetUtterancesView

Servicio: Amazon Lex Model Building Service

Utilice la operación `GetUtterancesView` para obtener información acerca de los enunciados que los usuarios han hecho al bot. Puede utilizar esta lista para ajustar los enunciados a los que responde el bot.

Por ejemplo, supongamos que ha creado un bot para pedir flores. Una vez haya transcurrido cierto tiempo y los usuarios hayan utilizado bastante el bot, utilice la operación `GetUtterancesView` para ver las solicitudes que han realizado y si el resultado ha sido satisfactorio. Es posible que el enunciado “Quiero flores” no se reconozca. Puede añadir este enunciado a la intención `OrderFlowers` para que el bot lo reconozca.

Después de publicar una nueva versión de un bot, puede obtener información acerca de la versión anterior y la nueva para comparar el rendimiento de ambas versiones.

Las estadísticas de enunciados se generan una vez al día. Hay datos disponibles para los últimos 15 días. Puede solicitar información para un máximo de 5 versiones del bot en cada solicitud. Amazon Lex devuelve los enunciados más frecuentes que ha recibido el bot a lo largo de los últimos 15 días. La respuesta contiene información acerca de un máximo de 100 enunciados para cada versión.

Las estadísticas de los enunciados no se generan en las siguientes condiciones:

- El `childDirected` campo estaba establecido en verdadero cuando se creó el bot.
- Está utilizando la ofuscación de ranuras con una o más ranuras.
- Ha optado por no participar en la mejora de Amazon Lex.

Esta operación necesita permisos para la acción `lex:GetUtterancesView`.

Sintaxis de la solicitud

```
GET /bots/botname/utterances?
view=aggregation&bot_versions=botVersions&status_type=statusType HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botname

El nombre del bot del que se devuelve información sobre enunciados.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

botVersions

La matriz de las versiones del bot de las que se devuelve información sobre enunciados. El límite es de 5 versiones por solicitud.

Miembros de la matriz: número mínimo de 1 elemento. La cantidad máxima es de 5 artículos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: sí

statusType

Para devolver enunciados reconocidos y gestionados, utilice `Detected`. Para devolver enunciados no reconocidos, utilice `Missed`.

Valores válidos: `Detected | Missed`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

botName

El nombre del bot del que se ha devuelto información sobre enunciados.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

utterances

Una matriz de objetos UtteranceList, en la que cada objeto contiene una lista de objetos UtteranceData que describen los enunciados que ha procesado el bot. La respuesta contiene un máximo de 100 objetos `UtteranceData` para cada versión. Amazon Lex devuelve los enunciados más frecuentes que ha recibido el bot a lo largo de los últimos 15 días.

Tipo: matriz de objetos UtteranceList

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListTagsForResource

Servicio: Amazon Lex Model Building Service

Obtiene una lista de las etiquetas asociadas a un recurso especificado. Solo bots, alias de bots y canales de bots tienen etiquetas asociadas.

Sintaxis de la solicitud

```
GET /tags/resourceArn HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

resourceArn

El nombre de recurso de Amazon (ARN) del recurso del que se obtiene una lista de etiquetas.

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1011.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

tags

Las etiquetas asociadas a un recurso.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS

- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutBot

Servicio: Amazon Lex Model Building Service

Crea un bot de conversación de Amazon Lex o sustituye un bot existente. Cuando crea o actualiza un bot, solo debe especificar un nombre, una configuración regional y si el bot está dirigido a niños menores de 13 años. Puede utilizar esta opción para agregar intenciones más adelante o para eliminarlas de un bot existente. Cuando crea un bot con la información mínima necesaria, el bot se crea o actualiza, pero Amazon Lex devuelve la respuesta `FAILED`. Puede compilar el bot una vez que haya agregado uno o más intentos. Para obtener más información acerca de los bots de Amazon Lex, consulte Funcionamiento de Amazon Lex.

Si especifica el nombre de un bot existente, los campos de la solicitud reemplazan los valores existentes en la versión `$LATEST` del bot. Amazon Lex elimina todos los campos para los que no proporciona valores en la solicitud, excepto para los campos `idleTTLInSeconds` y `privacySettings`, que se establecen en sus valores predeterminados. Si no especifica valores para los campos obligatorios, Amazon Lex devuelve una excepción.

Esta operación necesita permisos para la acción `lex:PutBot`. Para obtener más información, consulte Identity and Access Management para Amazon Lex.

Sintaxis de la solicitud

```
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"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del bot. El nombre no distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

abortStatement

Cuando Amazon Lex no comprende la entrada del usuario en el contexto de la conversación, intenta obtener la información varias veces. A continuación, Amazon Lex envía el mensaje definido en `abortStatement` al usuario y cancela la conversación. Para establecer el número de reintentos, utilice el campo `valueElicitationPrompt` del tipo de ranura.

Por ejemplo, en un bot de pedidos de pizza, Amazon Lex puede realizar una pregunta del estilo "¿Qué tipo de masa desea?". Si la respuesta del usuario no es una de las respuestas esperadas (por ejemplo, "masa fina", "base gruesa", etc.), Amazon Lex intenta obtener una respuesta correcta varias veces.

Por ejemplo, en una aplicación de pedidos de pizza, `OrderPizza` puede ser una de las intenciones. Esta intención puede requerir la ranura `CrustType`. El campo `valueElicitationPrompt` se especifica al crear la ranura `CrustType`.

Si ha definido una intención alternativa, la afirmación de cancelación no se enviará al usuario y se utilizará la intención alternativa. Para obtener más información, consulte AMAZON. FallbackIntent.

Tipo: objeto Statement

Obligatorio: no

checksum

Identifica una revisión específica de la versión `$LATEST`.

Si crea un nuevo bot, deje el campo `checksum` vacío. Si especifica una suma de comprobación, obtendrá una excepción `BadRequestException`.

Cuando quiera actualizar el bot, establezca el campo `checksum` en la suma de comprobación de la revisión más reciente de la versión `$LATEST`. Si no especifica el campo `checksum` o si la suma de comprobación no coincide con la versión `$LATEST`, obtendrá una excepción `PreconditionFailedException`.

Tipo: cadena

Requerido: no

childDirected

Para cada bot de Amazon Lex creado con el Servicio de creación de modelos de Amazon Lex, debe especificar si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que esté dirigido, total o parcialmente, a niños menores de 13 años y esté sujeto a la Ley de protección de la privacidad infantil en línea (COPPA, por sus siglas en inglés). Para ello, especifique `true` o `false` en el campo `childDirected`. Al especificar `true` en el campo `childDirected`, confirma que el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. Al especificar `false` en el campo `childDirected`, confirma que el uso de Amazon Lex no está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. No puede especificar un valor predeterminado en el campo `childDirected` que no indique de forma precisa si el uso de Amazon Lex está relacionado o no con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA.

Si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años, debe obtener un consentimiento parental verificable, obligatorio en virtud de la COPPA. Para obtener información acerca del uso de Amazon Lex con relación a sitios web, programas u otras aplicaciones dirigidos, total o parcialmente, a niños menores de 13 años, consulte las Preguntas frecuentes de Amazon Lex.

Tipo: Booleano

Obligatorio: sí

clarificationPrompt

Cuando Amazon Lex no comprende la intención del usuario, utiliza este mensaje para obtener una aclaración. Para especificar cuantas veces Amazon Lex debe repetir la pregunta aclaratoria, utilice el campo `maxAttempts`. Si Amazon Lex sigue sin comprender la intención, envía el mensaje del campo `abortStatement`.

Cuando cree una pregunta aclaratoria, asegúrese de que sugiere una respuesta correcta por parte del usuario. Por ejemplo, si el bot pregunta sobre pizzas y bebidas, una buena pregunta aclaratoria podría ser “¿Qué quiere hacer?”. Puede decir “Pedir una pizza” o “Pedir bebida”.

Si ha definido una intención alternativa, se invocará si la pregunta aclaratoria se repite el número de veces definido en el campo `maxAttempts`. Para obtener más información, consulte AMAZON.FallbackIntent.

Si no define una pregunta aclaratoria, en tiempo de ejecución, Amazon Lex devolverá una excepción 400 de solicitud errónea en estos tres casos:

- Pregunta aclaratoria: cuando el usuario responde a una pregunta de seguimiento, pero no proporciona una intención. Por ejemplo, como respuesta a una pregunta de seguimiento "¿Desea algo más hoy?", el usuario dice "Sí". Amazon Lex devolverá una excepción 400 de solicitud errónea porque no tiene una pregunta aclaratoria que pueda enviar al usuario para obtener una intención.
- Función de Lambda: al utilizar una función de Lambda, se devuelve un tipo de diálogo `ElicitIntent`. Dado que Amazon Lex no tiene una pregunta aclaratoria para obtener una intención del usuario, devuelve la excepción 400 de solicitud errónea.
- PutSession operación: al utilizar la `PutSession` operación, se envía un tipo `ElicitIntent` de diálogo. Dado que Amazon Lex no tiene una pregunta aclaratoria para obtener una intención del usuario, devuelve la excepción 400 de solicitud errónea.

Tipo: objeto Prompt

Obligatorio: no

createVersion

Cuando se establece en `true`, se crea una nueva versión numerada del bot. Se produce lo mismo que al llamar a la operación `CreateBotVersion`. Si no especifica `createVersion`, el valor predeterminado es `false`.

Tipo: Booleano

Obligatorio: no

description

La descripción del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

detectSentiment

Cuando se establece en `true`, los enunciados del usuario se envían a Amazon Comprehend con fines de análisis de opiniones. Si no especifica `detectSentiment`, el valor predeterminado es `false`.

Tipo: Booleano

Obligatorio: no

enableModelImprovements

Se establece en `true` para habilitar el acceso a mejoras de la comprensión del lenguaje natural.

Cuando establece el parámetro `enableModelImprovements` en `true`, puede utilizar el parámetro `nluIntentConfidenceThreshold` para configurar puntuaciones de confianza. Para obtener más información, consulte Puntuaciones de confianza.

Solo puede establecer el parámetro `enableModelImprovements` en ciertas regiones. Si establece el parámetro en `true`, el bot tendrá acceso a mejoras en la precisión.

Las regiones en las que puede establecer el parámetro `enableModelImprovements` en `false` para la configuración regional en-US son las siguientes:

- Este de EE. UU. (Norte de Virginia) (us-east-1)
- Oeste de EE. UU. (Oregón) (us-west-2)
- Asia-Pacífico (Sídney) (ap-southeast-2)
- UE (Irlanda) (eu-west-1)

En otras regiones y configuraciones regionales, el parámetro `enableModelImprovements` se establece en `true` de forma predeterminada. En estas regiones y configuraciones regionales, configurar el parámetro `false` da lugar a una excepción `ValidationException`.

Tipo: Booleano

Obligatorio: no

idleSessionTTLInSeconds

El tiempo máximo que Amazon Lex retiene los datos recopilados en una conversación, en segundos.

Una sesión de interacciones del usuario permanece activa durante el tiempo especificado. Si no se produce ninguna conversación durante este tiempo, la sesión caduca y Amazon Lex elimina los datos proporcionados antes de que se agote el tiempo de espera.

Por ejemplo, supongamos que un usuario elige la OrderPizza intención, pero se desvía a la mitad del proceso de realizar un pedido. Si el usuario no completa el pedido dentro de un plazo especificado, Amazon Lex descarta la información de la ranura que ha recopilado, por lo que el usuario tendrá que volver a iniciar el pedido.

Si no se incluye el elemento `idleSessionTTLInSeconds` en una solicitud de la operación `PutBot`, Amazon Lex utiliza el valor predeterminado. También se da esta situación si la solicitud reemplaza un bot existente.

El valor predeterminado es 300 segundos (5 minutos).

Tipo: entero

Rango válido: valor mínimo de 60. Valor máximo de 86 400.

Obligatorio: no

intents

Una matriz de objetos `Intent`. Cada intención representa una orden que puede expresar el usuario. Por ejemplo, un robot de pedidos de pizza podría respaldar una OrderPizza intención. Para obtener más información, consulte Funcionamiento de Amazon Lex.

Tipo: matriz de objetos Intent

Obligatorio: no

locale

Especifica la configuración regional de destino para el bot. Todas las intenciones que se utilicen en el bot deben ser compatibles con la configuración regional del bot.

El valor predeterminado es `en-US`.

Tipo: cadena

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`

Obligatorio: sí

nluIntentConfidenceThreshold

Determina el umbral en el que Amazon Lex insertará el `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`, o ambos, al devolver intenciones alternativas en una PostTextrespuesta PostContento. `AMAZON.FallbackIntenty` solo `AMAZON.KendraSearchIntent` se insertan si están configurados para el bot.

Debe establecer el parámetro `enableModelImprovements` en `true` para utilizar las puntuaciones de confianza en las siguientes regiones:

- Este de EE. UU. (Norte de Virginia) (us-east-1)
- Oeste de EE. UU. (Oregón) (us-west-2)
- Asia-Pacífico (Sídney) (ap-southeast-2)
- UE (Irlanda) (eu-west-1)

En otras regiones, el parámetro `enableModelImprovements` se establece en `true` de forma predeterminada.

Por ejemplo, supongamos que un bot se ha configurado con un umbral de confianza de 0,80 y `AMAZON.FallbackIntent`. Amazon Lex devuelve tres intenciones alternativas con las siguientes puntuaciones de confianza: IntentA (0,70), IntentB (0,60) e IntentC (0,50). La respuesta de la operación `PostText` sería:

- AMAZON. FallbackIntent
- IntentA
- IntentB
- IntentC

Tipo: Doble

Rango válido: valor mínimo de 0. El valor máximo es de 1.

Obligatorio: no

processBehavior

Si establece el elemento `processBehavior` en `BUILD`, Amazon Lex compila el bot para que se pueda ejecutar. Si establece el elemento en `SAVE`, Amazon Lex guarda el bot sin compilarlo.

Si no especifica este valor, el valor predeterminado es `BUILD`.

Tipo: cadena

Valores válidos: `SAVE | BUILD`

Obligatorio: no

tags

Una lista de las etiquetas que se agregarán al bot. Solo puede agregar etiquetas al crear un bot. No puede utilizar la operación `PutBot` para actualizar las etiquetas de un bot. Para actualizar las etiquetas, use la operación `TagResource`.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Obligatorio: no

voiceId

El ID de voz de Amazon Polly que quiera que Amazon Lex utilice para las interacciones de voz con el usuario. La configuración regional de la voz debe coincidir con la configuración regional del bot. Para obtener más información, consulte Voces en Amazon Polly en la Guía para desarrolladores de Amazon Polly.

Tipo: cadena

Requerido: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

abortStatement

El mensaje que Amazon Lex utiliza para cancelar una conversación. Para obtener más información, consulte PutBot.

Tipo: objeto Statement

checksum

La suma de comprobación del bot que ha creado.

Tipo: cadena

childDirected

Para cada bot de Amazon Lex creado con el Servicio de creación de modelos de Amazon Lex, debe especificar si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que esté dirigido, total o parcialmente, a niños menores de 13 años y esté sujeto a la Ley de protección de la privacidad infantil en línea (COPPA, por sus siglas en inglés). Para ello, especifique `true` o `false` en el campo `childDirected`. Al especificar `true` en el campo `childDirected`, confirma que el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. Al especificar `false` en el campo `childDirected`, confirma que el uso de Amazon Lex no está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA. No puede especificar un valor predeterminado en el campo `childDirected` que no indique de forma precisa si el uso de Amazon Lex está relacionado o no con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años y está sujeto a la COPPA.

Si el uso de Amazon Lex está relacionado con un sitio web, un programa u otro tipo de aplicación que está dirigido, total o parcialmente, a niños menores de 13 años, debe obtener un consentimiento parental verificable, obligatorio en virtud de la COPPA. Para obtener información acerca del uso de Amazon Lex con relación a sitios web, programas u otras aplicaciones dirigidos, total o parcialmente, a niños menores de 13 años, consulte las Preguntas frecuentes de Amazon Lex.

Tipo: Booleano

clarificationPrompt

Las solicitudes que Amazon Lex utiliza cuando no comprende la intención del usuario. Para obtener más información, consulte PutBot.

Tipo: objeto Prompt

createdDate

La fecha de creación del bot.

Tipo: marca temporal

createVersion

Es `True` si se ha creado una nueva versión del bot. Si el campo `createVersion` no se ha especificado en la solicitud, el campo `createVersion` se establece en “false” en la respuesta.

Tipo: Booleano

description

La descripción del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

detectSentiment

Es `true` si el bot está configurado para enviar enunciados del usuario a Amazon Comprehend con fines de análisis de opiniones. Si el campo `detectSentiment` no se ha especificado en la solicitud, el campo `detectSentiment` es `false` en la respuesta.

Tipo: Booleano

enableModelImprovements

Indica si el bot utiliza mejoras en la precisión. `true` indica que el bot utiliza las mejoras y `false`, lo contrario.

Tipo: Booleano

failureReason

Si `status` es `FAILED`, Amazon Lex proporciona el motivo por el que no se ha podido crear el bot.

Tipo: cadena

idleSessionTTLInSeconds

La duración máxima que Amazon Lex retiene los datos recopilados en una conversación. Para obtener más información, consulte PutBot.

Tipo: entero

Rango válido: valor mínimo de 60. Valor máximo de 86 400.

intents

Una matriz de objetos `Intent`. Para obtener más información, consulte PutBot.

Tipo: matriz de objetos Intent

lastUpdatedDate

La fecha de actualización del bot. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

locale

La configuración regional de destino para el bot.

Tipo: cadena

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

El nombre del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

La puntuación que determina dónde inserta Amazon Lex `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`, o ambas, al devolver intenciones alternativas en una PostTextrespuesta PostContento. `AMAZON.FallbackIntentse`

inserta si la puntuación de confianza de todos los intentos está por debajo de este valor. `AMAZON.KendraSearchIntent`solo se inserta si está configurado para el bot.

Tipo: Doble

Rango válido: valor mínimo de 0. Valor máximo de 1.

status

Cuando envía una solicitud para crear un bot y `processBehavior` se ha establecido en `BUILD`, Amazon Lex establece el elemento de respuesta `status` en `BUILDING`.

En el `READY_BASIC_TESTING` estado, puede probar el bot con entradas del usuario que coincidan exactamente con los enunciados configurados para las intenciones y los valores del bot en los tipos de ranura.

Si Amazon Lex no puede compilar el bot, Amazon Lex establece `status` en `FAILED`. Amazon Lex devuelve el motivo del error en el elemento de respuesta `failureReason`.

Cuando se establece `processBehavior` en `SAVE`, Amazon Lex establece el código de estado en `NOT BUILT`.

Cuando el bot se encuentra en el estado `READY`, se puede probar y publicar.

Tipo: cadena

Valores válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

tags

Una lista de etiquetas asociadas al bot.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

version

La versión del bot. En el caso de un bot nuevo, la versión es siempre `$LATEST`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

voiceId

El ID de voz de Amazon Polly que Amazon Lex utiliza para la interacción de voz con el usuario. Para obtener más información, consulte PutBot.

Tipo: cadena

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutBotAlias

Servicio: Amazon Lex Model Building Service

Crea un alias de la versión especificada del bot o reemplaza un alias del bot especificado. Para cambiar la versión del bot al que apunta el alias, reemplace el alias. Para obtener más información acerca de los alias, consulte Control de versiones y alias.

Esta operación necesita permisos para la acción `lex:PutBotAlias`.

Sintaxis de la solicitud

```
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"
      }
   ]
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botName

El nombre del bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

name

El nombre del alias. El nombre no distingue entre mayúsculas y minúsculas.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

botVersion

La versión del bot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: sí

checksum

Identifica una revisión específica de la versión `$LATEST`.

Si crea un nuevo alias de bot, deje el campo `checksum` vacío. Si especifica una suma de comprobación, obtendrá una excepción `BadRequestException`.

Cuando quiera actualizar un alias de bot, establezca el campo `checksum` en la suma de comprobación de la revisión más reciente de la versión `$LATEST`. Si no especifica el campo

`checksum` o si la suma de comprobación no coincide con la versión `$LATEST`, obtendrá una excepción `PreconditionFailedException`.

Tipo: cadena

Requerido: no

conversationLogs

La configuración de los registros de conversaciones para el alias.

Tipo: objeto ConversationLogsRequest

Obligatorio: no

description

Una descripción del alias.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

tags

Una lista de las etiquetas que se agregarán al alias del bot. Solo puede agregar etiquetas al crear un alias. No puede utilizar la operación `PutBotAlias` para actualizar las etiquetas de un alias de bot. Para actualizar las etiquetas, use la operación `TagResource`.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Obligatorio: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

botName

El nombre del bot al que apunta el alias.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

botVersion

La versión del bot a la que apunta el alias.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

checksum

La suma de comprobación para la versión actual del alias.

Tipo: cadena

conversationLogs

La configuración que determina cómo Amazon Lex utiliza los registros de conversaciones para el alias.

Tipo: objeto ConversationLogsResponse

createdDate

La fecha de creación del alias del bot.

Tipo: marca temporal

description

Una descripción del alias.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

lastUpdatedDate

La fecha de actualización del alias del bot. Al crear un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre del alias.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

tags

Una lista de etiquetas asociadas a un bot.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutIntent

Servicio: Amazon Lex Model Building Service

Crea una intención o sustituye una intención existente.

Para definir la interacción entre el usuario y el bot, utiliza una o varias intenciones. Por ejemplo, para un bot de pedidos de pizza, crearía una intención `OrderPizza`.

Para crear una intención o reemplazar una intención existente, debe proporcionar lo siguiente:

- El nombre de la intención. Por ejemplo, `OrderPizza`.
- Enunciados de muestra. Por ejemplo, “¿Puedo pedir una pizza?” y “Quiero pedir una pizza”.
- La información que debe recopilarse. Es necesario especificar los tipos de ranura para la información que el bot solicitará del usuario. Puede especificar tipos de ranura estándar, como una fecha o una hora, o tipos de ranura personalizados, como el tamaño y la masa de una pizza.
- Cómo se cumplirá la intención. Puede proporcionar una función de Lambda o configurar la intención para que devuelva información de la intención a la aplicación cliente. Si utiliza una función de Lambda, en el caso de que toda la información de la intención esté disponible, Amazon Lex invoca la función de Lambda. Si configura la intención para que devuelva la información de la intención a la aplicación cliente.

Puede especificar información opcional en la solicitud como la siguiente:

- Una pregunta de confirmación para solicitar al usuario que confirme una intención. Por ejemplo, “¿Debo pedir su pizza?”.
- Una afirmación de cierre que se envía al usuario después de cumplir con la intención. Por ejemplo, “He pedido su pizza”.
- Una pregunta de seguimiento para solicitar actividad adicional al usuario. Por ejemplo, puede preguntar “¿Quiere acompañar su pizza con una bebida?”.

Si especifica un nombre de intención existente para actualizar la intención, Amazon Lex reemplaza los valores de la versión `$LATEST` de la intención con los valores de la solicitud. Amazon Lex elimina los campos que no ha proporcionado en la solicitud. Si no especifica los campos obligatorios, Amazon Lex devuelve una excepción. Al actualizar la versión `$LATEST` de una intención, el campo `status` de cualquier bot que utilice la versión `$LATEST` de la intención se establece en `NOT_BUILT`.

Para obtener más información, consulte Funcionamiento de Amazon Lex.

Esta operación necesita permisos para la acción `lex:PutIntent`.

Sintaxis de la solicitud

```
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"
         }
      }
   ]
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de la intención. El nombre no distingue entre mayúsculas y minúsculas.

El nombre no puede coincidir con el nombre de una intención integrada o con el nombre de una intención integrada con la parte “AMAZON”. eliminada. Por ejemplo, como hay una intención integrada con el nombre `AMAZON.HelpIntent`, no puede crear una intención personalizada con el nombre `HelpIntent`.

Para obtener una lista de las intenciones integradas, consulte Intenciones estándar integradas en Alexa Skills Kit.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

checksum

Identifica una revisión específica de la versión `$LATEST`.

Si crea una nueva intención, deje el campo `checksum` vacío. Si especifica una suma de comprobación, obtendrá una excepción `BadRequestException`.

Cuando quiera actualizar una intención, establezca el campo `checksum` en la suma de comprobación de la revisión más reciente de la versión `$LATEST`. Si no especifica el campo `checksum` o si la suma de comprobación no coincide con la versión `$LATEST`, obtendrá una excepción `PreconditionFailedException`.

Tipo: cadena

Requerido: no

conclusionStatement

La afirmación que quiere que Amazon Lex transmita al usuario después de que la función de Lambda haya cumplido correctamente la intención.

Este elemento solo es relevante si proporciona una función de Lambda en `fulfillmentActivity`. Si devuelve la intención a la aplicación cliente, no puede especificar este elemento.

> **Note**
>
> `followUpPrompt` y `conclusionStatement` son mutuamente excluyentes. Puede especificar solo un valor.

Tipo: objeto Statement

Obligatorio: no

confirmationPrompt

Pide al usuario que confirme la intención. Esta pregunta debe tener una respuesta afirmativa o negativa.

Amazon Lex utiliza esta solicitud para confirmar que el usuario sabe que la intención puede cumplirse. Por ejemplo, con la intención `OrderPizza`, es posible que quiera confirmar que los detalles del pedido son correctos antes de procesarlo. En el caso de otras intenciones, como las intenciones que sirven únicamente para responder a preguntar del usuario, es posible que no quiera solicitar confirmación al usuario antes de proporcionar la información.

> **Note**
>
> Debe proporcionar `rejectionStatement` y `confirmationPrompt` o ningún valor.

Tipo: objeto Prompt

Obligatorio: no

createVersion

Cuando se establece en `true`, se crea una nueva versión numerada de la intención. Se produce lo mismo que al llamar a la operación `CreateIntentVersion`. Si no especifica `createVersion`, el valor predeterminado es `false`.

Tipo: Booleano

Obligatorio: no

description

Una descripción de la intención.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

dialogCodeHook

Especifica una función de Lambda para que se invoque para cada entrada del usuario. Puede invocar esta función de Lambda para personalizar la interacción del usuario.

Por ejemplo, supongamos que el bot determina que el usuario es John. La función de Lambda podría recuperar la información de John de una base de datos de back-end y rellenar previamente algunos de los valores. Por ejemplo, si descubre que John es intolerante al gluten, puede establecer la ranura de la intención correspondiente, `GlutenIntolerant`, en verdadero. Si encuentra el número de teléfono de John, puede configurar el atributo de la sesión correspondiente.

Tipo: objeto CodeHook

Obligatorio: no

followUpPrompt

Amazon Lex utiliza esta solicitud para solicitar actividad adicional una vez que se ha cumplido con una intención. Por ejemplo, si se cumple la intención `OrderPizza`, puede enviar una solicitud al usuario para saber si quiere pedir bebida.

La acción que Amazon Lex realiza depende de la respuesta del usuario, tal como se indica a continuación:

- Si el usuario dice “sí”, responde con la pregunta aclaratoria que se ha configurado para el bot.
- Si el usuario dice “sí” y continua con un enunciado que activa una intención, inicia una conversación para cumplir con la intención.
- Si el usuario dice “no”, responde con la afirmación de rechazo configurada para la pregunta de seguimiento.
- Si no reconoce el enunciado, repite la pregunta de seguimiento.

Los campos `followUpPrompt` y `conclusionStatement` son mutuamente excluyentes. Puede especificar solo un valor.

Tipo: objeto FollowUpPrompt

Obligatorio: no

fulfillmentActivity

Obligatorio. Describe cómo se cumple con la intención. Por ejemplo, cuando un usuario ha proporcionado toda la información necesaria para pedir una pizza, `fulfillmentActivity` define cómo el bot realiza el pedido en una pizzería local.

Puede configurar Amazon Lex para que devuelva toda la información de la intención a la aplicación cliente o indicar a Amazon Lex que invoque una función de Lambda que pueda procesar la intención (por ejemplo, realizar un pedido en una pizzería).

Tipo: objeto FulfillmentActivity

Obligatorio: no

inputContexts

Una matriz de objetos `InputContext` que enumera los contextos que deben estar activos para que Amazon Lex elija la intención en una conversación con el usuario.

Tipo: matriz de objetos InputContext

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 5 artículos.

Obligatorio: no

kendraConfiguration

Información de configuración necesaria para utilizar la intención `AMAZON.KendraSearchIntent` con la finalidad de conectarse a un índice de Amazon Kendra. Para obtener más información, consulte AMAZON. KendraSearchIntent.

Tipo: objeto KendraConfiguration

Obligatorio: no

outputContexts

Una matriz de objetos `OutputContext` que enumera los contextos en los que la intención se activa cuando esta se cumple.

Tipo: matriz de objetos OutputContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

Obligatorio: no

parentIntentSignature

Un identificador único para la intención integrada en la cual se debe basar esta intención. Para encontrar la firma de una intención, consulte Intenciones integradas estándar en Alexa Skills Kit.

Tipo: cadena

Requerido: no

rejectionStatement

Cuando el usuario responde “no” a la pregunta definida en `confirmationPrompt`, Amazon Lex responde con esta afirmación para confirmar que la intención se ha cancelado.

> **Note**
>
> Debe proporcionar `rejectionStatement` y `confirmationPrompt` o ningún valor.

Tipo: objeto Statement

Obligatorio: no

sampleUtterances

Una matriz de enunciados (cadenas) que un usuario podría decir para indicar la intención. Por ejemplo, «Quiero {PizzaSize} pizza», «Pedir {cantidad} {PizzaSize} pizzas».

En cada enunciado se incluye un nombre de ranura entre llaves.

Tipo: matriz de cadenas

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 1500 elementos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 200 caracteres.

Obligatorio: no

slots

Una matriz de ranuras de intención. En tiempo de ejecución, Amazon Lex obtiene los valores de ranura del usuario a través de las solicitudes definidas en las ranuras. Para obtener más información, consulte Funcionamiento de Amazon Lex.

Tipo: matriz de objetos Slot

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 100 artículos.

Obligatorio: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

checksum

La suma de comprobación de la versión `$LATEST` de la intención creada o actualizada.

Tipo: cadena

conclusionStatement

Una vez que la función de Lambda especificada en la intención `fulfillmentActivity` cumple con la intención, Amazon Lex transmite esta afirmación al usuario.

Tipo: objeto Statement

confirmationPrompt

Si se define en la intención, Amazon Lex solicita al usuario que confirme la intención antes de cumplirla.

Tipo: objeto Prompt

createdDate

La fecha en que se creó la intención.

Tipo: marca temporal

createVersion

Es `True` si se ha creado una nueva versión de la intención. Si el campo `createVersion` no se ha especificado en la solicitud, el campo `createVersion` se establece en “false” en la respuesta.

Tipo: Booleano

description

Una descripción de la intención.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

dialogCodeHook

Si se define en la intención, Amazon Lex invoca esta función de Lambda para cada entrada del usuario.

Tipo: objeto CodeHook

followUpPrompt

Si se define en la intención, Amazon Lex utiliza esta solicitud para solicitar actividad adicional del usuario una vez que se ha cumplido con la intención.

Tipo: objeto FollowUpPrompt

fulfillmentActivity

Si se define en la intención, Amazon Lex invoca esta función de Lambda para cumplir con la intención una vez que el usuario haya proporcionado toda la información que requiere la intención.

Tipo: objeto FulfillmentActivity

inputContexts

Una matriz de objetos `InputContext` que enumera los contextos que deben estar activos para que Amazon Lex elija la intención en una conversación con el usuario.

Tipo: matriz de objetos InputContext

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 5 artículos.

kendraConfiguration

Información de configuración, si procede, para conectarse a un índice de Amazon Kendra y utilizar la intención `AMAZON.KendraSearchIntent`.

Tipo: objeto KendraConfiguration

lastUpdatedDate

La fecha de actualización de la intención. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

outputContexts

Una matriz de objetos `OutputContext` que enumera los contextos en los que la intención se activa cuando esta se cumple.

Tipo: matriz de objetos OutputContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

parentIntentSignature

Un identificador único para la intención integrada en la cual se basa esta intención.

Tipo: cadena

rejectionStatement

Si el usuario responde “no” a la pregunta definida en `confirmationPrompt`, Amazon Lex responde con esta afirmación para confirmar que la intención se ha cancelado.

Tipo: objeto Statement

sampleUtterances

Una matriz de enunciados de muestra configurados para la intención.

Tipo: matriz de cadenas

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 1500 elementos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 200 caracteres.

slots

Una matriz de ranuras de intención configuradas para la intención.

Tipo: matriz de objetos Slot

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 100 artículos.

version

La versión de la intención. En el caso de una intención nueva, la versión es siempre `$LATEST`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutSlotType

Servicio: Amazon Lex Model Building Service

Crea un tipo de ranura personalizado o sustituye un tipo de ranura personalizado existente.

Para crear un tipo de ranura personalizado, especifique un nombre para el tipo de ranura y un conjunto de valores de enumeración, que son los valores que puede asumir una ranura de este tipo. Para obtener más información, consulte Funcionamiento de Amazon Lex.

Si especifica el nombre de un tipo de ranura existente, los campos de la solicitud reemplazan los valores existentes en la versión `$LATEST` del tipo de ranura. Amazon Lex elimina los campos que no ha proporcionado en la solicitud. Si no especifica los campos obligatorios, Amazon Lex devuelve una excepción. Al actualizar la versión `$LATEST` de un tipo de ranura, si el bot utiliza la versión `$LATEST` de una intención que contiene el tipo de ranura, el campo `status` del bot se establece en `NOT_BUILT`.

Esta operación necesita permisos para la acción `lex:PutSlotType`.

Sintaxis de la solicitud

```
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"
```

```
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del tipo de slot. El nombre no distingue entre mayúsculas y minúsculas.

El nombre no puede coincidir con el nombre de un tipo de ranura integrado o con el nombre de un tipo de ranura integrado con la parte “AMAZON.” eliminada. Por ejemplo, como hay un tipo de ranura integrado con el nombre `AMAZON.DATE`, no puede crear un tipo de ranura personalizado con el nombre `DATE`.

Para obtener una lista con los tipos de ranura integrados, consulte Referencia del tipo de ranura en Alexa Skills Kit.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

checksum

Identifica una revisión específica de la versión `$LATEST`.

Si crea un nuevo tipo de ranura, deje el campo `checksum` vacío. Si especifica una suma de comprobación, obtendrá una excepción `BadRequestException`.

Cuando quiera actualizar un tipo de ranura, establezca el campo `checksum` en la suma de comprobación de la revisión más reciente de la versión `$LATEST`. Si no especifica el campo `checksum` o si la suma de comprobación no coincide con la versión `$LATEST`, obtendrá una excepción `PreconditionFailedException`.

Tipo: cadena

Requerido: no

createVersion

Cuando se establece en `true`, se crea una nueva versión numerada del tipo de ranura. Se produce lo mismo que al llamar a la operación `CreateSlotTypeVersion`. Si no especifica `createVersion`, el valor predeterminado es `false`.

Tipo: Booleano

Obligatorio: no

description

Una descripción del tipo de slot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

enumerationValues

Una lista de objetos `EnumerationValue` que define los valores que puede tomar el tipo de ranura. Cada valor puede tener una lista de `synonyms`, que son valores adicionales que ayudan a entrenar el modelo de machine learning sobre los valores que resuelve para una ranura.

Un tipo de ranura de expresión regular no requiere valores de enumeración. Todos los demás tipos de ranuras requieren una lista de valores de enumeración.

Cuando Amazon Lex resuelve el valor de una ranura, genera una lista de resoluciones que contiene hasta cinco valores posibles para la ranura. Si utiliza una función de Lambda, esta lista de resoluciones se pasa a la función. Si no utiliza una función de Lambda, puede elegir que devuelva el valor que el usuario ha introducido o el primer valor de la lista de resoluciones como el valor de la ranura. El campo `valueSelectionStrategy` indica la opción que se utiliza.

Tipo: matriz de objetos EnumerationValue

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 10 000 elementos.

Obligatorio: no

parentSlotTypeSignature

El tipo de ranura integrado que se utiliza como principal para este tipo de ranura. Al definir un tipo de ranura principal, el nuevo tipo de ranura tiene la misma configuración que el tipo de ranura principal.

Solo se admite `AMAZON.AlphaNumeric`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Obligatorio: no

slotTypeConfigurations

Información sobre la configuración que amplía el tipo de ranura integrado principal. La configuración se agrega a la configuración del tipo de ranura principal.

Tipo: matriz de objetos SlotTypeConfiguration

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

Obligatorio: no

valueSelectionStrategy

Determina la estrategia de resolución de ranura que Amazon Lex utiliza para devolver valores de tipo de ranura. El campo se puede establecer en uno de los siguientes valores:

- `ORIGINAL_VALUE`: devuelve el valor que ha introducido el usuario, si el valor del usuario es similar a un valor de ranura.
- `TOP_RESOLUTION`: si hay una lista de resoluciones para la ranura, devuelve el primer valor de la lista de resoluciones como el valor del tipo de ranura. Si no hay lista de resoluciones, se devuelve un valor null (nulo).

Si no especifica `valueSelectionStrategy`, el valor predeterminado es `ORIGINAL_VALUE`.

Tipo: cadena

Valores válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

Obligatorio: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

checksum

La suma de comprobación de la versión $LATEST del tipo de ranura.

Tipo: cadena

createdDate

La fecha de creación del tipo de ranura.

Tipo: marca temporal

createVersion

Es `True` si se ha creado una nueva versión del tipo de ranura. Si el campo `createVersion` no se ha especificado en la solicitud, el campo `createVersion` se establece en “false” en la respuesta.

Tipo: Booleano

description

Una descripción del tipo de slot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

enumerationValues

Una lista de objetos `EnumerationValue` que define los valores que puede tomar el tipo de ranura.

Tipo: matriz de objetos EnumerationValue

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 10 000 elementos.

lastUpdatedDate

La fecha de actualización del tipo de ranura. Cuando se crea un tipo de recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

name

El nombre del tipo de slot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

El tipo de ranura integrado que se utiliza como principal para este tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Información sobre la configuración que amplía el tipo de ranura integrado principal.

Tipo: matriz de objetos SlotTypeConfiguration

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

valueSelectionStrategy

La estrategia de resolución de ranura que Amazon Lex utiliza para determinar el valor de la ranura. Para obtener más información, consulte PutSlotType.

Tipo: cadena

Valores válidos: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

La versión del tipo de ranura. En el caso de un tipo de ranura nuevo, la versión es siempre `$LATEST`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

PreconditionFailedException

La suma de comprobación del recurso que intenta modificar no coincide con la suma de comprobación de la solicitud. Revise la suma de comprobación del recurso e inténtelo de nuevo.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

StartImport

Servicio: Amazon Lex Model Building Service

Comienza un trabajo para importar un recurso a Amazon Lex.

Sintaxis de la solicitud

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
   "mergeStrategy": "string",
   "payload": blob,
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parámetros de solicitud del URI

La solicitud no utiliza ningún parámetro de URI.

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

mergeStrategy

Especifica la acción que debe llevar a cabo la operación `StartImport` cuando hay un recurso con el mismo nombre.

- FAIL_ON_CONFLICT: la operación de importación se detiene al detectarse el primer conflicto entre un recurso del archivo de importación y un recurso existente. El nombre del recurso que provoca el conflicto se encuentra en el campo `failureReason` de la respuesta a la operación `GetImport`.

  OVERWRITE_LATEST: la operación de importación continua aunque haya un conflicto con un recurso existente. La versión $LASTEST del recurso existente se sobreescribe con los datos del archivo de importación.

Tipo: cadena

Valores válidos: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

Obligatorio: sí

payload

Un archivo ZIP en formato binario. El archivo debe contener un archivo JSON que incluya el recurso que se va a importar. El recurso debe coincidir con el tipo especificado en el campo `resourceType`.

Tipo: objeto de datos binarios codificados en Base64

Obligatorio: sí

resourceType

Especifica el tipo de recurso que se va a exportar. Cada recurso exporta a su vez los recursos de los que depende.

- Un bot exporta intenciones dependientes.
- Una intención exporta los tipos de ranura dependientes.

Tipo: cadena

Valores válidos: `BOT | INTENT | SLOT_TYPE`

Obligatorio: sí

tags

Una lista de las etiquetas que se agregarán al bot importado. Solo puede agregar etiquetas cuando importa un bot. No puede agregar etiquetas a una intención o un tipo de ranura.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Obligatorio: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 201.

El servicio devuelve los datos siguientes en formato JSON.

createdDate

Una marca de tiempo para la fecha y hora en la que se solicitó el trabajo de importación.

Tipo: marca temporal

importId

El identificador de un trabajo de importación específico.

Tipo: cadena

importStatus

El estado del trabajo de importación. Si el estado es `FAILED`, puede consultar el motivo del fallo con la operación `GetImport`.

Tipo: cadena

Valores válidos: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

La acción que se debe llevar a cabo cuando se produce un conflicto de fusión.

Tipo: cadena

Valores válidos: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

El nombre asignado al trabajo de importación.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `[a-zA-Z_]+`

resourceType

El tipo de recurso que se va a importar.

Tipo: cadena

Valores válidos: `BOT | INTENT | SLOT_TYPE`

tags

Una lista de las etiquetas agregadas al bot importado.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

StartMigration

Servicio: Amazon Lex Model Building Service

Comienza a migrar un bot de Amazon Lex V1 a Amazon Lex V2. Migre su bot cuando desee aprovechar las nuevas características de Amazon Lex V2.

Para obtener más información, consulte Migración de un bot en la Guía para desarrolladores de Amazon Lex.

Sintaxis de la solicitud

```
POST /migrations HTTP/1.1
Content-type: application/json

{
   "migrationStrategy": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotName": "string",
   "v2BotRole": "string"
}
```

Parámetros de solicitud del URI

La solicitud no utiliza ningún parámetro de URI.

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

migrationStrategy

La estrategia empleada para llevar a cabo la migración.

- `CREATE_NEW`: crea un nuevo bot de Amazon Lex V2 y migra el bot de Amazon Lex V1 al nuevo bot.
- `UPDATE_EXISTING`: sobrescribe los metadatos de un bot de Amazon Lex V2 y la configuración regional que se van a migrar. No cambia ninguna otra configuración regional del bot de Amazon Lex V2. Si la configuración regional no existe, se crea una nueva configuración regional en el bot de Amazon Lex V2.

Tipo: cadena

Valores válidos: `CREATE_NEW | UPDATE_EXISTING`

Obligatorio: sí

v1BotName

El nombre del bot de Amazon Lex V1 que está migrando a Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

v1BotVersion

La versión del bot que se va a migrar a Amazon Lex V2. Puede migrar la versión `$LATEST`, así como cualquier versión numerada.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: sí

v2BotName

El nombre del bot de Amazon Lex V2 al que está migrando el bot de Amazon Lex V1.

- Si el bot de Amazon Lex V2 no existe, debe utilizar la estrategia de migración `CREATE_NEW`.
- Si el bot de Amazon Lex V2 existe, debe utilizar la estrategia de migración `UPDATE_EXISTING` para modificar el contenido del bot de Amazon Lex V2.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([0-9a-zA-Z][_-]?)+$`

Obligatorio: sí

v2BotRole

El rol de IAM que Amazon Lex utiliza para ejecutar el bot de Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Obligatorio: sí

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 202.

El servicio devuelve los datos siguientes en formato JSON.

migrationId

El identificador único que Amazon Lex ha asignado a la migración.

Tipo: cadena

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

migrationStrategy

La estrategia empleada para llevar a cabo la migración.

Tipo: cadena

Valores válidos: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

La fecha y hora en que comenzó la migración.

Tipo: marca temporal

v1BotLocale

La configuración regional utilizada para el bot de Amazon Lex V1.

Tipo: cadena

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

El nombre del bot de Amazon Lex V1 que está migrando a Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

v1BotVersion

La versión del bot que se va a migrar a Amazon Lex V2.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

v2BotId

El identificador único del bot de Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

v2BotRole

El rol de IAM que Amazon Lex utiliza para ejecutar el bot de Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Errores

AccessDeniedException

El usuario de IAM o rol no tiene permiso para llamar a las API de Amazon Lex V2 necesarias para migrar el bot.

Código de estado HTTP: 403

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

TagResource

Servicio: Amazon Lex Model Building Service

Agrega las etiquetas especificadas al recurso especificado. Si ya existe una clave de etiqueta, el valor existente se sustituye por el nuevo valor.

Sintaxis de la solicitud

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

resourceArn

El nombre de recurso de Amazon (ARN) del bot, alias de bot o canal de bot que se va a etiquetar.

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1011.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

tags

La lista de las claves de etiquetas que se quieren añadir al recurso. Si ya existe una clave de etiqueta, el valor existente se sustituye por el nuevo valor.

Tipo: matriz de objetos Tag

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Obligatorio: sí

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UntagResource

Servicio: Amazon Lex Model Building Service

Elimina las etiquetas de un bot, alias de bot o canal de bot.

Sintaxis de la solicitud

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

resourceArn

El nombre de recurso de Amazon (ARN) del recurso del que se van a eliminar las etiquetas.

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1011.

Obligatorio: sí

tagKeys

Una lista de las claves de etiqueta que se eliminarán del recurso. Si una clave de etiqueta no existe en el recurso, se ignora.

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 128 caracteres.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 204
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 204 con un cuerpo HTTP vacío.

Errores

BadRequestException

La solicitud no está bien formulada. Es posible que un valor no sea válido o que falte un campo obligatorio. Revise los valores del campo e inténtelo de nuevo.

Código de estado HTTP: 400

ConflictException

Se ha producido un conflicto al procesar la solicitud. Intente volver a realizar la solicitud.

Código de estado HTTP: 409

InternalFailureException

Se ha producido un error interno de Amazon Lex. Intente volver a realizar la solicitud.

Código de estado HTTP: 500

LimitExceededException

La solicitud ha superado un límite. Intente volver a realizar la solicitud.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso especificado en la solicitud. Compruebe si el recurso existe e inténtelo de nuevo.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

Amazon Lex Runtime Service

Amazon Lex Runtime Service admite las siguientes acciones:

- DeleteSession
- GetSession
- PostContent
- PostText
- PutSession

DeleteSession

Servicio: Amazon Lex Runtime Service

Elimina la información de sesión de un bot, alias e ID de usuario especificados.

Sintaxis de la solicitud

```
DELETE /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botAlias

El alias en uso para el bot que contiene los datos de la sesión.

Obligatorio: sí

botName

El nombre del bot que contiene los datos de la sesión.

Obligatorio: sí

userId

El identificador del usuario que está asociado a los datos de la sesión.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Patrón: `[0-9a-zA-Z._:-]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "botAlias": "string",
    "botName": "string",
    "sessionId": "string",
    "userId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

botAlias

El alias en uso para el bot asociado a los datos de la sesión.

Tipo: cadena

botName

El nombre del bot asociado a los datos de la sesión.

Tipo: cadena

sessionId

El identificador único de la sesión.

Tipo: cadena

userId

El ID del usuario de la aplicación cliente.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Patrón: `[0-9a-zA-Z._:-]+`

Errores

BadRequestException

Se ha producido un error al validar la solicitud, no hay mensajes útiles en el contexto o la compilación del bot ha fallado, está en curso o contiene cambios sin compilar.

Código de estado HTTP: 400

ConflictException

Dos clientes utilizan la misma cuenta de AWS, el mismo bot de Amazon Lex y el mismo ID de usuario.

Código de estado HTTP: 409

InternalFailureException

Error de servicio interno. Vuelva a intentar la llamada.

Código de estado HTTP: 500

LimitExceededException

Se ha superado un límite.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso (como el bot o un alias de Amazon Lex) al que se hace referencia.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

GetSession

Servicio: Amazon Lex Runtime Service

Devuelve la información de sesión de un bot, alias e ID de usuario especificados.

Sintaxis de la solicitud

```
GET /bot/botName/alias/botAlias/user/userId/session/?
checkpointLabelFilter=checkpointLabelFilter HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botAlias

El alias en uso para el bot que contiene los datos de la sesión.

Obligatorio: sí

botName

El nombre del bot que contiene los datos de la sesión.

Obligatorio: sí

checkpointLabelFilter

La cadena que se utiliza para filtrar las intenciones devueltas en la estructura `recentIntentSummaryView`.

Al especificar un filtro, solo se devuelven las intenciones cuyo campo `checkpointLabel` esté establecido para esa cadena.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `[a-zA-Z0-9-]+`

userId

El ID del usuario de la aplicación cliente. Amazon Lex lo utiliza para identificar una conversación del usuario con el bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Patrón: `[0-9a-zA-Z._:-]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

activeContexts

Una lista de los contextos activos para la sesión. Se puede establecer un contexto cuando se cumple una intención o mediante una llamada a la operación `PostContent`, `PostText` o `PutSession`.

Puede utilizar un contexto para controlar las intenciones que pueden acompañar una intención o para modificar la operación de la aplicación.

Tipo: matriz de objetos ActiveContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 20 artículos.

dialogAction

Describe el estado actual del bot.

Tipo: objeto DialogAction

recentIntentSummaryView

Una matriz de información sobre las intenciones que se utilizan en la sesión. La matriz puede contener hasta tres resúmenes. Si se utilizan más de tres intenciones en la sesión, la operación `recentIntentSummaryView` contiene información sobre los tres últimos intentos empleados.

Si establece el parámetro `checkpointLabelFilter` en la solicitud, la matriz contiene únicamente las intenciones con la etiqueta especificada.

Tipo: matriz de objetos IntentSummary

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 3 elementos.

sessionAttributes

Un mapa de pares clave/valor que representa la información de contexto específica de la sesión. Contiene información de la aplicación pasada entre Amazon Lex y una aplicación cliente.

Tipo: mapa de cadena a cadena

sessionId

Un identificador único de la sesión.

Tipo: cadena

Errores

BadRequestException

Se ha producido un error al validar la solicitud, no hay mensajes útiles en el contexto o la compilación del bot ha fallado, está en curso o contiene cambios sin compilar.

Código de estado HTTP: 400

InternalFailureException

Error de servicio interno. Vuelva a intentar la llamada.

Código de estado HTTP: 500

LimitExceededException

Se ha superado un límite.

Código de estado HTTP: 429

NotFoundException

No se ha encontrado el recurso (como el bot o un alias de Amazon Lex) al que se hace referencia.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PostContent

Servicio: Amazon Lex Runtime Service

Envía entradas de usuarios (texto o voz) a Amazon Lex. Los clientes utilizan esta API para enviar solicitudes de texto y audio a Amazon Lex en tiempo de ejecución. Amazon Lex interpreta la entrada del usuario con el modelo de machine learning que ha compilado para el bot.

La operación `PostContent` es compatible con la entrada de audio a 8 kHz y 16 kHz. Puede utilizar audio a 8 kHz para mejorar la precisión del reconocimiento de voz en aplicaciones de audio por teléfono.

Como respuesta, Amazon Lex devuelve el siguiente mensaje para transmitirlo al usuario. Considere los siguientes ejemplos de mensaje:

- Para la entrada de usuario “Quiero una pizza”, Amazon Lex puede devolver una respuesta con un mensaje para obtener datos de ranura (por ejemplo, `PizzaSize`): “¿De qué tamaño quiere la pizza?”.
- Una vez que el usuario haya proporcionado toda la información necesaria para pedir la pizza, Amazon Lex puede devolver una respuesta con un mensaje para obtener la confirmación del usuario: “¿Desea pedir la pizza?”.
- Si el usuario responde “sí” a la pregunta de confirmación, es posible que Amazon Lex devuelva una afirmación de cierre: “Muchas gracias. Se ha realizado el pedido de su pizza de quesos”.

No todos los mensajes de Amazon Lex requieren una respuesta del usuario. Por ejemplo, las afirmaciones de cierre no requieren respuesta. Algunos mensajes solo requieren una respuesta afirmativa o negativa. Además de `message`, Amazon Lex proporciona contexto adicional sobre el mensaje de la respuesta para que pueda mejorar el comportamiento del cliente, como mostrar la interfaz de usuario adecuada al cliente. Considere los siguientes ejemplos:

- Si el mensaje tiene como objetivo obtener datos de ranuras, Amazon Lex devuelve la siguiente información de contexto:
  - El encabezado `x-amz-lex-dialog-state` establecido en `ElicitSlot`
  - El encabezado `x-amz-lex-intent-name` establecido en el nombre de la intención en el contexto actual
  - El encabezado `x-amz-lex-slot-to-elicit` establecido en el nombre de la ranura para el que `message` obtiene información

  - El encabezado `x-amz-lex-slots` establecido en una asignación de ranuras configurada para la intención con sus valores actuales
- Si el mensaje es una pregunta de confirmación, el encabezado `x-amz-lex-dialog-state` se establece en `Confirmation` y el encabezado `x-amz-lex-slot-to-elicit` se omite.
- Si el mensaje es una pregunta aclaratoria configurada para la intención, lo que indica que no se ha comprendido la intención del usuario, el encabezado `x-amz-dialog-state` se establece en `ElicitIntent` y el encabezado `x-amz-slot-to-elicit` se omite.

Además, Amazon Lex también devuelve los valores `sessionAttributes` específicos de la aplicación. Para obtener más información, consulte Administración del contexto de la conversación.

Sintaxis de la solicitud

```
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
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

accept

Este valor se transfiere como el encabezado HTTP `Accept`.

El mensaje que Amazon Lex devuelve en la respuesta puede ser texto o voz basado en el valor del encabezado HTTP `Accept` de la solicitud.

- Si el valor es `text/plain; charset=utf-8`, Amazon Lex devuelve texto en la respuesta.
- Si el valor comienza por `audio/`, Amazon Lex devuelve voz en la respuesta. Amazon Lex utiliza Amazon Polly para generar la voz (con la configuración que ha especificado en el encabezado `Accept`). Por ejemplo, si especifica `audio/mpeg` como valor, Amazon Lex devuelve voz en formato MPEG.

- Si el valor es `audio/pcm`, la voz que se devuelve es `audio/pcm` en formato little endian de 16 bits.
- A continuación se muestran los valores aceptados:
  - audio/mpeg
  - audio/ogg
  - audio/pcm
  - text/plain; charset=utf-8
  - audio/* (el valor predeterminado es mpeg)

activeContexts

Una lista de los contextos activos para la solicitud. Un contexto se puede activar cuando se cumple una intención anterior o al incluir el contexto en la solicitud,

Si no especifica una lista de contextos, Amazon Lex utilizará la lista de contextos actual en la sesión. Si especifica una lista vacía, se borran todos los contextos de la sesión.

botAlias

El alias del bot de Amazon Lex.

Obligatorio: sí

botName

El nombre del bot de Amazon Lex.

Obligatorio: sí

contentType

Este valor se transfiere como el encabezado HTTP `Content-Type`.

Indica el formato de audio o de texto. El valor del encabezado debe comenzar por uno de los siguientes prefijos:

- En el formato PCM, los datos de audio deben estar en el orden de bytes little endian.
  - audio/l16; rate=16000; channels=1
  - audio/x-l16; sample-rate=16000; channel-count=1
  - audio/lpcm; frecuencia de muestreo = 8000; =16; recuento de canales=1; =false sample-size-bits is-big-endian

- Formato Opus
  - audio/ -preámbulox-cbr-opus-with; tamaño del preámbulo = 0; tasa de bits=256000; =4 frame-size-milliseconds
- Formato de texto
  - text/plain; charset=utf-8

Obligatorio: sí

requestAttributes

Este valor se transfiere como el encabezado HTTP `x-amz-lex-request-attributes`.

La información específica de la solicitud que se pasa entre Amazon Lex y una aplicación cliente. El valor debe ser una asignación codificada en base64 y serializada en formato JSON con claves de cadena y valores. El tamaño total de los encabezados `requestAttributes` y `sessionAttributes` no puede superar 12 kB.

El espacio de nombres `x-amz-lex:` está reservado para atributos especiales. No cree atributos de solicitud con el prefijo `x-amz-lex:`.

Para obtener más información, consulte Configuración de atributos de solicitud.

sessionAttributes

Este valor se transfiere como el encabezado HTTP `x-amz-lex-session-attributes`.

La información específica de la aplicación que se pasa entre Amazon Lex y una aplicación cliente. El valor debe ser una asignación codificada en base64 y serializada en formato JSON con claves de cadena y valores. El tamaño total de los encabezados `sessionAttributes` y `requestAttributes` no puede superar 12 kB.

Para obtener más información, consulte Configuración de atributos de sesión.

userId

El ID del usuario de la aplicación cliente. Amazon Lex lo utiliza para identificar una conversación del usuario con el bot. En tiempo de ejecución, cada solicitud debe contener el campo `userID`.

Para decidir qué ID de usuario utilizará en la aplicación, tenga en cuenta lo siguiente.

- El campo `userID` no debe contener información de identificación personal del usuario como, por ejemplo, nombre, número de identificación personal u otro tipo de datos personales del usuario final.

- Si desea que un usuario inicie una conversación en un dispositivo y esta continúe en otro dispositivo, utilice un identificador específico del usuario.
- Si desea que el mismo usuario pueda mantener dos conversaciones independientes en dos dispositivos distintos, elija un identificador específico del dispositivo.
- Un usuario no puede mantener dos conversaciones independientes con dos versiones distintas del mismo bot. Por ejemplo, un usuario no puede mantener una conversación con las versiones PROD y BETA del mismo bot. Si piensa que un usuario podría necesitar dos versiones distintas para mantener conversaciones (por ejemplo, para realizar pruebas), incluya el alias del bot en el ID del usuario para separar las dos conversaciones.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Patrón: `[0-9a-zA-Z._:-]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos binarios.

inputStream

La entrada del usuario en formato de audio PCM u Opus o en formato de texto, tal como se describe en el encabezado HTTP `Content-Type`.

Puede transmitir datos de audio a Amazon Lex o crear un búfer local que recopile todos los datos de audio antes de enviarlos. En general, transmitir datos de audio ofrece un mejor rendimiento que almacenar los datos en un búfer local.

Obligatorio: sí

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

La respuesta devuelve los siguientes encabezados HTTP.

activeContexts

Una lista de los contextos activos para la sesión. Se puede establecer un contexto cuando se cumple una intención o mediante una llamada a la operación `PostContent`, `PostText` o `PutSession`.

Puede utilizar un contexto para controlar las intenciones que pueden acompañar una intención o para modificar la operación de la aplicación.

alternativeIntents

Entre una y cuatro intenciones alternativas que pueden ser aplicables a la intención del usuario.

Cada alternativa incluye una puntuación que indica la confianza de Amazon Lex en que la intención coincide con la intención del usuario. Las intenciones se ordenan por puntuación de confianza.

botVersion

La versión del bot que ha respondido a la conversación. Puede utilizar esta información para determinar si una versión de un bot rinde mejor que otra versión.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+|\$LATEST`

contentType

El tipo de contenido, tal como se especifica en el encabezado HTTP `Accept` de la solicitud.

dialogState

Identifica el estado actual de la interacción del usuario. Amazon Lex devuelve uno de los siguientes valores como `dialogState`. Si lo desea, el cliente puede utilizar esta información para personalizar la interfaz de usuario.

- `ElicitIntent`: Amazon Lex quiere obtener la intención del usuario. Considere los siguientes ejemplos:

  Por ejemplo, un usuario puede expresar una intención (“Quiero pedir una pizza”). Si Amazon Lex no puede deducir la intención del usuario a partir de esta expresión, devolverá este estado de diálogo.
- `ConfirmIntent`: Amazon Lex espera “sí” o “no” como respuesta.

  Por ejemplo, Amazon Lex solicita la confirmación del usuario antes de cumplir con una intención. El usuario, en lugar de responder “sí” o “no”, puede responder con información adicional. Por ejemplo, “sí, pero quiero una pizza con masa gruesa” o “no, quiero pedir bebida”. Amazon Lex puede procesar dicha información adicional (en estos ejemplos, actualizar la ranura tipo corteza o cambiar la intención de OrderPizza a OrderDrink).
- `ElicitSlot`: Amazon Lex espera el valor de una ranura para la intención actual.

  Por ejemplo, supongamos que, en la respuesta, Amazon Lex envía el mensaje “¿De qué tamaño quiere la pizza?”. Un usuario puede responder con el valor de ranura (p. ej., “mediana”). El usuario también puede proporcionar información adicional en la respuesta (p. ej., “una pizza mediana con masa gruesa”). Amazon Lex puede procesar esta información adicional de forma adecuada.
- `Fulfilled`: indica que la función de Lambda ha cumplido con la intención correctamente.
- `ReadyForFulfillment`: indica que el cliente tiene que cumplir la solicitud.
- `Failed`: indica que la conversación con el usuario ha fallado.

  Esto puede ocurrir porque el usuario no ha proporcionado una respuesta adecuada a las preguntas del servicio (puede configurar el número de veces que Amazon Lex puede solicitar cierta información al usuario), porque la función de Lambda no ha podido cumplir con la intención o por otros motivos.

Valores válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedInputTranscript

El texto que se utiliza para procesar la solicitud.

Si la entrada era una transmisión de audio, el campo `encodedInputTranscript` contiene el texto que se ha extraído de la transmisión de audio. Este es el texto que se procesa realmente para reconocer los valores de las intenciones y slot. Puede utilizar esta información para determinar si Amazon Lex está procesando correctamente el audio que ha enviado.

El campo `encodedInputTranscript` está codificado en base-64. Debe descodificar el campo antes de que pueda utilizar el valor.

encodedMessage

El mensaje que se va a transmitir al usuario. El mensaje puede provenir de la configuración del bot o de una función de Lambda.

Si la intención no está configurada con una función de Lambda o si la función de Lambda ha devuelto `Delegate` como `dialogAction.type` en su respuesta, Amazon Lex decide el siguiente procedimiento y selecciona un mensaje adecuado de la configuración del bot en función del contexto de la interacción actual. Por ejemplo, si Amazon Lex no puede entender las entradas del usuario, utiliza una pregunta aclaratoria.

Al crear una intención, puede asignar mensajes a grupos. Si los mensajes están asignados a grupos, Amazon Lex devuelve un mensaje de cada grupo en la respuesta. El campo del mensaje es una cadena JSON con secuencias de escape que contiene los mensajes. Para obtener más información acerca de la estructura de la cadena JSON devuelta, consulte Formatos de mensajes admitidos.

Si la función de Lambda devuelve un mensaje, Amazon Lex lo envía al cliente en su respuesta.

El campo `encodedMessage` está codificado en base-64. Debe descodificar el campo antes de que pueda utilizar el valor.

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1366.

inputTranscript

Este encabezado ha quedado obsoleto.

El texto que se utiliza para procesar la solicitud.

Solo puede utilizar este campo en las configuraciones regionales de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. En el resto de configuraciones regionales, el campo `inputTranscript` es nulo. En su lugar, debe utilizar el campo `encodedInputTranscript`.

Si la entrada era una transmisión de audio, el campo `inputTranscript` contiene el texto que se ha extraído de la transmisión de audio. Este es el texto que se procesa realmente para reconocer los valores de las intenciones y slot. Puede utilizar esta información para determinar si Amazon Lex está procesando correctamente el audio que ha enviado.

intentName

La intención del usuario actual de la que Amazon Lex está pendiente.

message

Este encabezado ha quedado obsoleto.

Solo puede utilizar este campo en las configuraciones regionales de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. En el resto de configuraciones regionales, el campo `message` es nulo. En su lugar, debe utilizar el campo `encodedMessage`.

El mensaje que se va a transmitir al usuario. El mensaje puede provenir de la configuración del bot o de una función de Lambda.

Si la intención no está configurada con una función de Lambda o si la función de Lambda ha devuelto `Delegate` como `dialogAction.type` en su respuesta, Amazon Lex decide el siguiente procedimiento y selecciona un mensaje adecuado de la configuración del bot en función del contexto de la interacción actual. Por ejemplo, si Amazon Lex no puede entender las entradas del usuario, utiliza una pregunta aclaratoria.

Al crear una intención, puede asignar mensajes a grupos. Si los mensajes están asignados a grupos, Amazon Lex devuelve un mensaje de cada grupo en la respuesta. El campo del mensaje es una cadena JSON con secuencias de escape que contiene los mensajes. Para obtener más información acerca de la estructura de la cadena JSON devuelta, consulte Formatos de mensajes admitidos.

Si la función de Lambda devuelve un mensaje, Amazon Lex lo envía al cliente en su respuesta.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1024 caracteres.

messageFormat

El formato del mensaje de respuesta. Uno de los valores siguientes:

- `PlainText`: el mensaje contiene texto UTF-8 sin formato.
- `CustomPayload`: el mensaje está en un formato personalizado del cliente.
- `SSML`: el mensaje contiene texto con formato para salida de voz.
- `Composite`: el mensaje contiene un objeto JSON con secuencias de escape que contiene uno o más mensajes de los grupos a los que se asignaron cuando se creó la intención.

Valores válidos: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Proporciona una puntuación que indica el grado de confianza de Amazon Lex en lo que respecta a la capacidad de una intención devuelta para satisfacer las expectativas del usuario. La puntuación es un valor entre 0,0 y 1,0.

La puntuación es relativa, no absoluta. La puntuación puede cambiar en función de las mejoras de Amazon Lex.

sentimentResponse

La opinión expresada en un enunciado.

Cuando el bot está configurado para enviar enunciados a Amazon Comprehend con el fin de analizar opiniones, este campo contiene el resultado del análisis.

sessionAttributes

Un mapa de pares clave/valor que representa la información de contexto específica de la sesión.

sessionId

El identificador único de la sesión.

slots

Una asignación de cero o más ranuras de intención (pares nombre/valor) que Amazon Lex ha detectado a partir de la entrada del usuario durante la conversación. El campo está codificado en base-64.

Amazon Lex crea una lista de resoluciones que contiene posibles valores para una ranura. El valor que devuelve viene determinado por el valor `valueSelectionStrategy` seleccionado cuando se creó o actualizó el tipo de ranura. Si `valueSelectionStrategy` se establece en `ORIGINAL_VALUE`, se devuelve el valor que proporciona el usuario, en caso de que el valor del usuario sea similar a los valores de la ranura. Si `valueSelectionStrategy`

se establece en `TOP_RESOLUTION`, Amazon Lex devuelve el primer valor de la lista de resoluciones o, si no hay ninguna lista de resoluciones, un valor nulo. Si no especifica un valor `valueSelectionStrategy`, el valor predeterminado es `ORIGINAL_VALUE`.

slotToElicit

Si el valor `dialogState` es `ElicitSlot`, devuelve el nombre de la ranura para la que Amazon Lex quiere obtener un valor.

La respuesta devuelve lo siguiente como el cuerpo HTTP.

audioStream

La solicitud (o afirmación) que se transmite al usuario. Depende de la configuración del bot y el contexto. Por ejemplo, si Amazon Lex no ha comprendido la intención del usuario, envía el valor `clarificationPrompt` configurado para el bot. Si la intención requiere confirmación antes de llevar a cabo la acción de cumplimiento, envía `confirmationPrompt`. Otro ejemplo: supongamos que la función de Lambda ha cumplido la intención satisfactoriamente y ha enviado un mensaje para transmitirlo al usuario. A continuación, Amazon Lex envía el mensaje en la respuesta.

Errores

BadGatewayException

El bot de Amazon Lex aún se está compilando o uno de los servicios dependientes (Amazon Polly o AWS Lambda) ha fallado debido a un error de servicio interno.

Código de estado HTTP: 502

BadRequestException

Se ha producido un error al validar la solicitud, no hay mensajes útiles en el contexto o la compilación del bot ha fallado, está en curso o contiene cambios sin compilar.

Código de estado HTTP: 400

ConflictException

Dos clientes utilizan la misma cuenta de AWS, el mismo bot de Amazon Lex y el mismo ID de usuario.

Código de estado HTTP: 409

DependencyFailedException

Una de las dependencias, como AWS Lambda o Amazon Polly, ha generado una excepción. Por ejemplo:

- Si Amazon Lex no tiene permisos suficientes para llamar a una función de Lambda
- Si una función de Lambda tarda más de 30 segundos en ejecutarse
- Si una función de Lambda de cumplimiento devuelve una acción de diálogo `Delegate` sin eliminar ningún valor de ranura.

Código de estado HTTP: 424

InternalFailureException

Error de servicio interno. Vuelva a intentar la llamada.

Código de estado HTTP: 500

LimitExceededException

Se ha superado un límite.

Código de estado HTTP: 429

LoopDetectedException

Esta excepción no se utiliza.

Código de estado HTTP: 508

NotAcceptableException

El encabezado de aceptación de la solicitud no cuenta con un valor válido.

Código de estado HTTP: 406

NotFoundException

No se ha encontrado el recurso (como el bot o un alias de Amazon Lex) al que se hace referencia.

Código de estado HTTP: 404

RequestTimeoutException

El mensaje de voz de la entrada es demasiado largo.

Código de estado HTTP: 408

UnsupportedMediaTypeException

El encabezado Content-Type (API `PostContent`) tiene un valor no válido.

Código de estado HTTP: 415

Ejemplos

Ejemplo 1

En esta solicitud, la URI identifica un bot (Traffic), una versión del bot ($LATEST) y el nombre del usuario final (someuser). El encabezado `Content-Type` identifica el formato del audio en el cuerpo. Amazon Lex también admite otros formatos. Para convertir audio de un formato a otro, si es necesario, puede utilizar el software de código abierto SoX. Para especificar el formato en el que desea obtener la respuesta, agregue el encabezado HTTP `Accept`.

En la respuesta, el encabezado `x-amz-lex-message` muestra la respuesta que ha devuelto Amazon Lex. A continuación, el cliente puede enviar esta respuesta al usuario. El mismo mensaje se envía en formato audio/MPEG mediante codificación fragmentada (según se solicite).

Solicitud de muestra

```
"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]"
…
```

Respuesta de ejemplo

```
"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
…
```

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3

- AWS SDK para Python
- AWS SDK para Ruby V3

PostText

Servicio: Amazon Lex Runtime Service

Envía entradas de usuarios a Amazon Lex. Las aplicaciones cliente pueden utilizar esta API para enviar solicitudes a Amazon Lex en tiempo de ejecución. A continuación, Amazon Lex interpreta la entrada del usuario con el modelo de machine learning que ha compilado para el bot.

Como respuesta, Amazon Lex devuelve el siguiente valor `message` al usuario para transmitir al usuario un valor `responseCard` opcional que se muestra. Considere los siguientes ejemplos de mensaje:

- Si un usuario escribe «Me gustaría una pizza», Amazon Lex podría devolver una respuesta con un mensaje con datos de espacio (por ejemplo, PizzaSize): «¿Qué tamaño de pizza quieres?»
- Una vez que el usuario haya proporcionado toda la información necesaria para pedir la pizza, Amazon Lex puede devolver una respuesta con un mensaje para obtener la confirmación del usuario: “¿Desea continuar con el pedido?”.
- Si el usuario responde “sí” a una pregunta de confirmación, es posible que Amazon Lex devuelva una afirmación de cierre: “Muchas gracias. Se ha realizado el pedido de su pizza de quesos”.

No todos los mensajes de Amazon Lex requieren una respuesta del usuario. Por ejemplo, las afirmaciones de cierre no requieren respuesta. Algunos mensajes solo requieren una respuesta afirmativa o negativa. Además de `message`, Amazon Lex proporciona contexto adicional sobre el mensaje de la respuesta para que pueda mejorar el comportamiento del cliente, como mostrar la interfaz de usuario adecuada al cliente. Se trata de los campos `slotToElicit`, `dialogState`, `intentName` y `slots` en la respuesta. Considere los siguientes ejemplos:

- Si el mensaje tiene como objetivo obtener datos de ranuras, Amazon Lex devuelve la siguiente información de contexto:
  - `dialogState`establecido en ElicitSlot
  - `intentName` establecido en el nombre de la intención en el contexto actual
  - `slotToElicit` establecido en el nombre de la ranura para el que `message` obtiene información
  - `slots` establecido en una asignación de ranuras configurada para la intención con sus valores conocidos
- Si el mensaje es una solicitud de confirmación, `dialogState` se establece en ConfirmIntent y `SlotToElicit` se establece en nulo.

- Si el mensaje es una solicitud de aclaración (configurada para la intención) que indica que el usuario no entiende la intención, `dialogState` se establece en ElicitIntent y `slotToElicit` se establece en nulo.

Además, Amazon Lex también devuelve los valores `sessionAttributes` específicos de la aplicación. Para obtener más información, consulte Administración del contexto de la conversación.

Sintaxis de la solicitud

```
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"
   }
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

botAlias

El alias del bot de Amazon Lex.

Obligatorio: sí

botName

El nombre del bot de Amazon Lex.

Obligatorio: sí

userId

El ID del usuario de la aplicación cliente. Amazon Lex lo utiliza para identificar una conversación del usuario con el bot. En tiempo de ejecución, cada solicitud debe contener el campo `userID`.

Para decidir qué ID de usuario utilizará en la aplicación, tenga en cuenta lo siguiente.

- El campo `userID` no debe contener información de identificación personal del usuario como, por ejemplo, nombre, número de identificación personal u otro tipo de datos personales del usuario final.
- Si desea que un usuario inicie una conversación en un dispositivo y esta continúe en otro dispositivo, utilice un identificador específico del usuario.
- Si desea que el mismo usuario pueda mantener dos conversaciones independientes en dos dispositivos distintos, elija un identificador específico del dispositivo.
- Un usuario no puede mantener dos conversaciones independientes con dos versiones distintas del mismo bot. Por ejemplo, un usuario no puede mantener una conversación con las versiones PROD y BETA del mismo bot. Si piensa que un usuario podría necesitar dos versiones distintas para mantener conversaciones (por ejemplo, para realizar pruebas), incluya el alias del bot en el ID del usuario para separar las dos conversaciones.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Patrón: `[0-9a-zA-Z._:-]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

activeContexts

Una lista de los contextos activos para la solicitud. Un contexto se puede activar cuando se cumple una intención anterior o al incluir el contexto en la solicitud,

Si no especifica una lista de contextos, Amazon Lex utilizará la lista de contextos actual en la sesión. Si especifica una lista vacía, se borran todos los contextos de la sesión.

Tipo: matriz de objetos ActiveContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 20 artículos.

Obligatorio: no

inputText

El texto que ha introducido el usuario (Amazon Lex interpreta este texto).

Cuando utiliza la CLI de AWS, no puede pasar una URL en el parámetro `--input-text`. En su lugar, pase la URL con el parámetro `--cli-input-json`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Obligatorio: sí

requestAttributes

La información específica de la solicitud que se pasa entre Amazon Lex y una aplicación cliente.

El espacio de nombres `x-amz-lex:` está reservado para atributos especiales. No cree atributos de solicitud con el prefijo `x-amz-lex:`.

Para obtener más información, consulte Configuración de atributos de solicitud.

Tipo: mapa de cadena a cadena

Obligatorio: no

sessionAttributes

La información específica de la aplicación que se pasa entre Amazon Lex y una aplicación cliente.

Para obtener más información, consulte Configuración de atributos de sesión.

Tipo: mapa de cadena a cadena

Obligatorio: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

activeContexts

Una lista de los contextos activos para la sesión. Se puede establecer un contexto cuando se cumple una intención o mediante una llamada a la operación `PostContent`, `PostText` o `PutSession`.

Puede utilizar un contexto para controlar las intenciones que pueden acompañar una intención o para modificar la operación de la aplicación.

Tipo: matriz de objetos ActiveContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 20 artículos.

alternativeIntents

Entre una y cuatro intenciones alternativas que pueden ser aplicables a la intención del usuario.

Cada alternativa incluye una puntuación que indica la confianza de Amazon Lex en que la intención coincide con la intención del usuario. Las intenciones se ordenan por puntuación de confianza.

Tipo: matriz de objetos PredictedIntent

Miembros de la matriz: número máximo de 4 elementos.

botVersion

La versión del bot que ha respondido a la conversación. Puede utilizar esta información para determinar si una versión de un bot rinde mejor que otra versión.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `[0-9]+|\$LATEST`

dialogState

Identifica el estado actual de la interacción del usuario. Amazon Lex devuelve uno de los siguientes valores como `dialogState`. Si lo desea, el cliente puede utilizar esta información para personalizar la interfaz de usuario.

- `ElicitIntent`: Amazon Lex quiere obtener la intención del usuario.

  Por ejemplo, un usuario puede expresar una intención (“Quiero pedir una pizza”). Si Amazon Lex no puede deducir la intención del usuario a partir de esta expresión, devolverá este dialogState.
- `ConfirmIntent`: Amazon Lex espera “sí” o “no” como respuesta.

  Por ejemplo, Amazon Lex solicita la confirmación del usuario antes de cumplir con una intención.

  El usuario, en lugar de responder “sí” o “no”, puede responder con información adicional. Por ejemplo, “sí, pero quiero una pizza con masa gruesa” o “no, quiero pedir bebida”. Amazon Lex puede procesar dicha información adicional (en estos ejemplos, actualizar el valor de la ranura tipo masa o cambiar la intención de OrderPizza a OrderDrink).

- `ElicitSlot`: Amazon Lex espera el valor de una ranura para la intención actual.

  Por ejemplo, supongamos que, en la respuesta, Amazon Lex envía el mensaje "¿De qué tamaño quiere la pizza?". Un usuario puede responder con el valor de ranura (p. ej., "mediana"). El usuario también puede proporcionar información adicional en la respuesta (p. ej., "una pizza mediana con masa gruesa"). Amazon Lex puede procesar esta información adicional de forma adecuada.
- `Fulfilled`: indica que la función de Lambda configurada para la intención ha cumplido con la intención correctamente.
- `ReadyForFulfillment`: indica que el cliente tiene que cumplir la intención.
- `Failed`: indica que la conversación con el usuario ha fallado.

  Esto puede ocurrir porque el usuario no ha proporcionado una respuesta adecuada a las preguntas del servicio (puede configurar el número de veces que Amazon Lex puede solicitar cierta información al usuario), porque la función de Lambda no ha podido cumplir con la intención o por otros motivos.

Tipo: cadena

Valores válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

intentName

La intención del usuario actual de la que Amazon Lex está pendiente.

Tipo: cadena

message

El mensaje que se va a transmitir al usuario. El mensaje puede provenir de la configuración del bot o de una función de Lambda.

Si la intención no está configurada con una función de Lambda o si la función de Lambda ha devuelto `Delegate` como `dialogAction.type` en su respuesta, Amazon Lex decide el siguiente procedimiento y selecciona un mensaje adecuado de la configuración del bot en función del contexto de la interacción actual. Por ejemplo, si Amazon Lex no puede entender las entradas del usuario, utiliza una pregunta aclaratoria.

Al crear una intención, puede asignar mensajes a grupos. Si los mensajes están asignados a grupos, Amazon Lex devuelve un mensaje de cada grupo en la respuesta. El campo del mensaje

es una cadena JSON con secuencias de escape que contiene los mensajes. Para obtener más información acerca de la estructura de la cadena JSON devuelta, consulte Formatos de mensajes admitidos.

Si la función de Lambda devuelve un mensaje, Amazon Lex lo envía al cliente en su respuesta.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1024 caracteres.

messageFormat

El formato del mensaje de respuesta. Uno de los valores siguientes:

- `PlainText`: el mensaje contiene texto UTF-8 sin formato.
- `CustomPayload`: el mensaje tiene el formato personalizado que define la función de Lambda.
- `SSML`: el mensaje contiene texto con formato para salida de voz.
- `Composite`: el mensaje contiene un objeto JSON con secuencias de escape que contiene uno o más mensajes de los grupos a los que se asignaron cuando se creó la intención.

Tipo: cadena

Valores válidos: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Proporciona una puntuación que indica el grado de confianza de Amazon Lex en lo que respecta a la capacidad de una intención devuelta para satisfacer las expectativas del usuario. La puntuación es un valor entre 0,0 y 1,0. Para obtener más información, consulte Puntuaciones de confianza.

La puntuación es relativa, no absoluta. La puntuación puede cambiar en función de las mejoras de Amazon Lex.

Tipo: objeto IntentConfidence

responseCard

Representa las opciones con las que el usuario puede responder a la pregunta actual. La tarjeta de respuesta puede proceder de la configuración del bot (en la consola de Amazon Lex, seleccione el botón de configuración situado junto a una ranura) o de un enlace de código (función de Lambda).

Tipo: objeto ResponseCard

sentimentResponse

La opinión expresada en un enunciado.

Cuando el bot está configurado para enviar enunciados a Amazon Comprehend con el fin de analizar opiniones, este campo contiene el resultado del análisis.

Tipo: objeto SentimentResponse

sessionAttributes

Una asignación de pares clave-valor que representa la información de contexto específica de la sesión.

Tipo: mapa de cadena a cadena

sessionId

Un identificador único de la sesión.

Tipo: cadena

slots

Las ranuras de intención que Amazon Lex ha detectado a partir de la entrada del usuario en la conversación.

Amazon Lex crea una lista de resoluciones que contiene posibles valores para una ranura. El valor que devuelve viene determinado por el valor `valueSelectionStrategy` seleccionado cuando se creó o actualizó el tipo de ranura. Si `valueSelectionStrategy` se establece en `ORIGINAL_VALUE`, se devuelve el valor que proporciona el usuario, en caso de que el valor del usuario sea similar a los valores de la ranura. Si `valueSelectionStrategy` se establece en `TOP_RESOLUTION`, Amazon Lex devuelve el primer valor de la lista de resoluciones o, si no hay ninguna lista de resoluciones, un valor nulo. Si no especifica un valor `valueSelectionStrategy`, el valor predeterminado es `ORIGINAL_VALUE`.

Tipo: mapa de cadena a cadena

slotToElicit

Si el valor `dialogState` es `ElicitSlot`, devuelve el nombre de la ranura para que Amazon Lex quiere obtener un valor.

Tipo: cadena

Errores

BadGatewayException

El bot de Amazon Lex aún se está compilando o uno de los servicios dependientes (Amazon Polly o AWS Lambda) ha fallado debido a un error de servicio interno.

Código de estado HTTP: 502

BadRequestException

Se ha producido un error al validar la solicitud, no hay mensajes útiles en el contexto o la compilación del bot ha fallado, está en curso o contiene cambios sin compilar.

Código de estado HTTP: 400

ConflictException

Dos clientes utilizan la misma cuenta de AWS, el mismo bot de Amazon Lex y el mismo ID de usuario.

Código de estado HTTP: 409

DependencyFailedException

Una de las dependencias, como AWS Lambda o Amazon Polly, ha generado una excepción. Por ejemplo:

- Si Amazon Lex no tiene permisos suficientes para llamar a una función de Lambda
- Si una función de Lambda tarda más de 30 segundos en ejecutarse
- Si una función de Lambda de cumplimiento devuelve una acción de diálogo `Delegate` sin eliminar ningún valor de ranura.

Código de estado HTTP: 424

InternalFailureException

Error de servicio interno. Vuelva a intentar la llamada.

Código de estado HTTP: 500

LimitExceededException

Se ha superado un límite.

Código de estado HTTP: 429

LoopDetectedException

Esta excepción no se utiliza.

Código de estado HTTP: 508

NotFoundException

No se ha encontrado el recurso (como el bot o un alias de Amazon Lex) al que se hace referencia.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

PutSession

Servicio: Amazon Lex Runtime Service

Crea una nueva sesión o modifica una sesión existente con un bot de Amazon Lex. Utilice esta operación para permitir que la aplicación establezca el estado del bot.

Para obtener más información, consulte Administración de las sesiones.

Sintaxis de la solicitud

```
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"
      }
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

accept

El mensaje que Amazon Lex devuelve en la respuesta puede ser texto o voz basado en el valor de este campo.

- Si el valor es `text/plain; charset=utf-8`, Amazon Lex devuelve texto en la respuesta.
- Si el valor comienza por `audio/`, Amazon Lex devuelve voz en la respuesta. Amazon Lex utiliza Amazon Polly para generar la voz en la configuración que ha especificado. Por ejemplo, si especifica `audio/mpeg` como valor, Amazon Lex devuelve voz en formato MPEG.
- Si el valor es `audio/pcm`, la voz que se devuelve es `audio/pcm` en formato little endian de 16 bits.
- A continuación se muestran los valores aceptados:
  - `audio/mpeg`
  - `audio/ogg`
  - `audio/pcm`
  - `audio/*` (el valor predeterminado es mpeg)
  - `text/plain; charset=utf-8`

botAlias

El alias en uso para el bot que contiene los datos de la sesión.

Obligatorio: sí

botName

El nombre del bot que contiene los datos de la sesión.

Obligatorio: sí

userId

El ID del usuario de la aplicación cliente. Amazon Lex lo utiliza para identificar una conversación del usuario con el bot.

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 100 caracteres.

Patrón: `[0-9a-zA-Z._:-]+`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

activeContexts

Una lista de los contextos activos para la solicitud. Un contexto se puede activar cuando se cumple una intención anterior o al incluir el contexto en la solicitud,

Si no especifica una lista de contextos, Amazon Lex utilizará la lista de contextos actual en la sesión. Si especifica una lista vacía, se borran todos los contextos de la sesión.

Tipo: matriz de objetos ActiveContext

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 20 artículos.

Obligatorio: no

dialogAction

Establece la siguiente acción que el bot debería llevar a cabo para cumplir con la conversación.

Tipo: objeto DialogAction

Obligatorio: no

recentIntentSummaryView

Un resumen de los incidentes recientes en el bot. Puede utilizar la vista de resumen de intenciones para establecer una etiqueta de punto de control en una intención y modificar los atributos de intenciones. También puede utilizarla para eliminar o agregar objetos de resumen de intención a la lista.

La intención que modifique o agregue a la lista debe tener sentido para el bot. Por ejemplo, el nombre de la intención debe ser válido para el bot. Debe proporcionar valores válidos para lo siguiente:

- `intentName`
- nombres de ranuras
- `slotToElict`

Si envía el parámetro `recentIntentSummaryView` en una solicitud `PutSession`, el contenido de la nueva vista de resumen reemplaza la vista anterior. Por ejemplo, si una solicitud `GetSession` devuelve tres intenciones en la vista de resumen y puede llamar a `PutSession` con una intención en la vista de resumen, la siguiente llamada a `GetSession` devolverá únicamente una intención.

Tipo: matriz de objetos IntentSummary

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 3 elementos.

Obligatorio: no

sessionAttributes

Un mapa de pares clave/valor que representa la información de contexto específica de la sesión. Contiene información de la aplicación pasada entre Amazon Lex y una aplicación cliente.

Tipo: mapa de cadena a cadena

Obligatorio: no

Sintaxis de la respuesta

```
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 respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

La respuesta devuelve los siguientes encabezados HTTP.

activeContexts

Una lista de los contextos activos para la sesión.

contentType

El tipo de contenido, tal como se especifica en el encabezado HTTP `Accept` de la solicitud.

dialogState

- `ConfirmIntent`: Amazon Lex espera “sí” o “no” como respuesta para confirmar la intención antes de cumplirla.
- `ElicitIntent`: Amazon Lex quiere obtener la intención del usuario.
- `ElicitSlot`: Amazon Lex espera el valor de una ranura para la intención actual.
- `Failed`: indica que la conversación con el usuario ha fallado. Esto puede ocurrir porque el usuario no ha proporcionado una respuesta adecuada a las preguntas del servicio, porque la función de Lambda no ha podido cumplir con la intención o por otros motivos.
- `Fulfilled`: indica que la función de Lambda ha cumplido con la intención satisfactoriamente.
- `ReadyForFulfillment`: indica que el cliente tiene que cumplir la intención.

Valores válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedMessage

El siguiente mensaje que debería mostrarse al usuario.

El campo `encodedMessage` está codificado en base-64. Debe descodificar el campo antes de que pueda utilizar el valor.

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1366.

intentName

El nombre de la intención actual.

message

Este encabezado ha quedado obsoleto.

El siguiente mensaje que debería mostrarse al usuario.

Solo puede utilizar este campo en las configuraciones regionales de-DE, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-CA, fr-FR e it-IT. En el resto de configuraciones regionales, el campo `message` es nulo. En su lugar, debe utilizar el campo `encodedMessage`.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1024 caracteres.

messageFormat

El formato del mensaje de respuesta. Uno de los valores siguientes:

- `PlainText`: el mensaje contiene texto UTF-8 sin formato.
- `CustomPayload`: el mensaje está en un formato personalizado del cliente.
- `SSML`: el mensaje contiene texto con formato para salida de voz.
- `Composite`: el mensaje contiene un objeto JSON con secuencias de escape que contiene uno o más mensajes de los grupos a los que se asignaron cuando se creó la intención.

Valores válidos: `PlainText | CustomPayload | SSML | Composite`

sessionAttributes

Una asignación de pares clave/valor que representa la información de contexto específica de la sesión.

sessionId

Un identificador único de la sesión.

slots

Una asignación de cero o más ranuras de intención que Amazon Lex ha detectado a partir de la entrada del usuario durante la conversación.

Amazon Lex crea una lista de resoluciones que contiene posibles valores para una ranura. El valor que devuelve viene determinado por el valor `valueSelectionStrategy` seleccionado cuando se creó o actualizó el tipo de ranura. Si `valueSelectionStrategy` se establece en `ORIGINAL_VALUE`, se devuelve el valor que proporciona el usuario, en caso de que el valor del usuario sea similar a los valores de la ranura. Si `valueSelectionStrategy` se establece en `TOP_RESOLUTION`, Amazon Lex devuelve el primer valor de la lista de resoluciones o, si no hay ninguna lista de resoluciones, un valor nulo. Si no especifica un valor `valueSelectionStrategy`, el valor predeterminado es `ORIGINAL_VALUE`.

slotToElicit

Si `dialogState` es `ElicitSlot`, devuelve el nombre de la ranura para que Amazon Lex quiere obtener un valor.

La respuesta devuelve lo siguiente como el cuerpo HTTP.

audioStream

La versión del audio del mensaje que se transmite al usuario.

Errores

BadGatewayException

El bot de Amazon Lex aún se está compilando o uno de los servicios dependientes (Amazon Polly o AWS Lambda) ha fallado debido a un error de servicio interno.

Código de estado HTTP: 502

BadRequestException

Se ha producido un error al validar la solicitud, no hay mensajes útiles en el contexto o la compilación del bot ha fallado, está en curso o contiene cambios sin compilar.

Código de estado HTTP: 400

ConflictException

Dos clientes utilizan la misma cuenta de AWS, el mismo bot de Amazon Lex y el mismo ID de usuario.

Código de estado HTTP: 409

DependencyFailedException

Una de las dependencias, como AWS Lambda o Amazon Polly, ha generado una excepción. Por ejemplo:

- Si Amazon Lex no tiene permisos suficientes para llamar a una función de Lambda
- Si una función de Lambda tarda más de 30 segundos en ejecutarse
- Si una función de Lambda de cumplimiento devuelve una acción de diálogo `Delegate` sin eliminar ningún valor de ranura.

Código de estado HTTP: 424

InternalFailureException

Error de servicio interno. Vuelva a intentar la llamada.

Código de estado HTTP: 500

LimitExceededException

Se ha superado un límite.

Código de estado HTTP: 429

NotAcceptableException

El encabezado de aceptación de la solicitud no cuenta con un valor válido.

Código de estado HTTP: 406

NotFoundException

No se ha encontrado el recurso (como el bot o un alias de Amazon Lex) al que se hace referencia.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- 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 PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

Tipos de datos

Amazon Lex Model Building Service admite los siguientes tipos de datos:

- 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

Amazon Lex Runtime Service admite los siguientes tipos de datos:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

Amazon Lex Model Building Service

Amazon Lex Model Building Service admite los siguientes tipos de datos:

- 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

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de un alias de bot.

Contenido

botName

El nombre del bot al que apunta el alias.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

botVersion

La versión del bot de Amazon Lex al que apunta el alias.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

checksum

La suma de comprobación del alias del bot.

Tipo: cadena

Requerido: no

conversationLogs

La configuración que determina cómo Amazon Lex utiliza los registros de conversaciones del alias.

Tipo: objeto ConversationLogsResponse

Obligatorio: no

createdDate

La fecha de creación del alias del bot.

Tipo: marca temporal

Obligatorio: no

description

Una descripción del alias del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

lastUpdatedDate

La fecha de actualización del alias del bot. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

Obligatorio: no

name

El nombre del alias del bot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BotChannelAssociation

Servicio: Amazon Lex Model Building Service

Representa una asociación entre un bot de Amazon Lex y una plataforma de mensajería externa.

Contenido

botAlias

Un alias que apunta a la versión exacta del bot de Amazon Lex al que se vincula esta asociación.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

botConfiguration

Proporciona información necesaria para comunicarse con la plataforma de mensajería.

Tipo: mapa de cadena a cadena

Entradas de mapa: número máximo de 10 elementos.

Obligatorio: no

botName

El nombre del bot de Amazon Lex al que se vincula esta asociación.

> **Note**
>
> Actualmente, Amazon Lex admite asociaciones con Facebook y Slack, y con Twilio.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

createdDate

La fecha de creación de la asociación entre el bot de Amazon Lex y el canal.

Tipo: marca temporal

Obligatorio: no

description

Un texto que describe la asociación que está creando.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

failureReason

Si `status` es `FAILED`, Amazon Lex proporciona el motivo por el que no se ha podido crear la asociación.

Tipo: cadena

Requerido: no

name

El nombre de la asociación entre el bot y el canal.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

status

El estado del canal del bot.

- `CREATED`: el canal se ha creado y está listo para usarse.
- `IN_PROGRESS`: el canal se está creando.

- `FAILED`: se ha producido un error al crear el canal. Para obtener información acerca del motivo del error, consulte el campo `failureReason`.

Tipo: cadena

Valores válidos: `IN_PROGRESS | CREATED | FAILED`

Obligatorio: no

type

Indica el tipo de canal que se establece entre el bot de Amazon Lex y la plataforma de mensajería externa para especificar el tipo de asociación.

Tipo: cadena

Valores válidos: `Facebook | Slack | Twilio-Sms | Kik`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BotMetadata

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de un bot.

Contenido

createdDate

La fecha de creación del bot.

Tipo: marca temporal

Obligatorio: no

description

La descripción del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

lastUpdatedDate

La fecha de actualización del bot. Cuando se crea un bot, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

Obligatorio: no

name

El nombre del bot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

status

El estado del bot.

Tipo: cadena

Valores válidos: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

Obligatorio: no

version

La versión del bot. En el caso de un bot nuevo, la versión es siempre `$LATEST`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BuiltinIntentMetadata

Servicio: Amazon Lex Model Building Service

Proporciona metadatos para una intención integrada.

Contenido

signature

Un identificador único para la intención integrada. Para encontrar la firma de una intención, consulte Intenciones integradas estándar en Alexa Skills Kit.

Tipo: cadena

Requerido: no

supportedLocales

Una lista con los identificadores de las configuraciones regionales que admite la intención.

Tipo: matriz de cadenas

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`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BuiltinIntentSlot

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de una ranura que se utiliza en una intención integrada.

Contenido

name

Una lista de las ranuras definidas para la intención.

Tipo: cadena

Requerido: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

BuiltinSlotTypeMetadata

Servicio: Amazon Lex Model Building Service

Proporciona información acerca del tipo de ranura integrado.

Contenido

signature

Un identificador único para el tipo de ranura integrada. Para encontrar la firma de un tipo de ranura, consulte Referencia del tipo de ranura en Alexa Skills Kit.

Tipo: cadena

Requerido: no

supportedLocales

Una lista de las configuraciones regionales de destino para la ranura.

Tipo: matriz de cadenas

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`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

CodeHook

Servicio: Amazon Lex Model Building Service

Especifica una función de Lambda que verifica las solicitudes a un bot o cumple con la solicitud del usuario a un bot.

Contenido

messageVersion

La versión de la solicitud y respuesta que desea que Amazon Lex use para invocar una función de Lambda. Para obtener más información, consulte Uso de funciones de Lambda.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 5.

Obligatorio: sí

uri

El nombre de recurso de Amazon (ARN) de la función de Lambda.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `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-_]+)?`

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ConversationLogsRequest

Servicio: Amazon Lex Model Building Service

Proporciona la configuración necesaria para los registros de conversaciones.

Contenido

iamRoleArn

El nombre de recurso de Amazon (ARN) de un rol de IAM con permiso para escribir en sus registros para CloudWatch los registros de texto y en su bucket de S3 para los registros de audio. Si el cifrado de audio está habilitado, este rol también concede permisos de acceso a la clave de AWS KMS que se utiliza para cifrar registros de audio. Para obtener más información, consulte Creación de una política y un rol de IAM para registros de conversaciones.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Obligatorio: sí

logSettings

La configuración de los registros de conversaciones. Puede registrar la conversación en formato de texto, formato de audio o ambos.

Tipo: matriz de objetos LogSettingsRequest

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ConversationLogsResponse

Servicio: Amazon Lex Model Building Service

Contiene información acerca de la configuración de registros de conversaciones.

Contenido

iamRoleArn

El nombre de recurso de Amazon (ARN) de la función de IAM que se utiliza para escribir los registros en Logs o en un CloudWatch bucket de S3.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Obligatorio: no

logSettings

La configuración de los registros de conversaciones. Puede registrar texto, audio o ambos.

Tipo: matriz de objetos LogSettingsResponse

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

EnumerationValue

Servicio: Amazon Lex Model Building Service

Cada tipo de ranura puede tener un conjunto de valores. Cada valor de enumeración representa uno de los valores que puede tener un tipo de ranura.

Por ejemplo, un bot para pedidos de pizza puede tener un tipo de ranura que especifique el tipo de masa de la pizza. En ese caso, el tipo de ranura podría incluir los siguientes valores:

- gruesa
- fina
- rellena

Contenido

value

El valor del tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 140 caracteres.

Obligatorio: sí

synonyms

Valores adicionales relacionados con el valor del tipo de ranura.

Tipo: matriz de cadenas

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 140 caracteres.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

FollowUpPrompt

Servicio: Amazon Lex Model Building Service

Una solicitud de actividad adicional que se envía una vez que se ha cumplido una intención. Por ejemplo, si se cumple la intención `OrderPizza`, puede enviar una solicitud al usuario para saber si quiere pedir bebida.

Contenido

prompt

Solicita información del usuario.

Tipo: objeto Prompt

Obligatorio: sí

rejectionStatement

Si el usuario responde “no” a la pregunta definida en el campo `prompt`, Amazon Lex responde con esta afirmación para confirmar que la intención se ha cancelado.

Tipo: objeto Statement

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

FulfillmentActivity

Servicio: Amazon Lex Model Building Service

Describe la forma en que se cumple con la intención después de que el usuario proporcione toda la información necesaria para la intención. Puede proporcionar una función de Lambda para que procese la intención o devuelva la información de la intención a la aplicación cliente. Le recomendamos que utilice una función de Lambda para que la lógica relevante se almacene en la nube y limite el código en el lado de cliente, principalmente a la presentación. Si necesita actualizar la lógica, tan solo tiene que actualizar la función de Lambda. No hace falta actualizar la aplicación cliente.

Considere los siguientes ejemplos:

- En una aplicación de pedidos de pizza, una vez que el usuario proporciona toda la información para realizar un pedido, se utiliza una función de Lambda para realizar un pedido en una pizzería.
- En una aplicación de juegos, cuando un usuario dice “coge una piedra”, esta información debe volver a la aplicación cliente para que pueda llevar a cabo la operación y actualizar los gráficos. En este caso, quiere que Amazon Lex devuelva los datos de la intención al cliente.

Contenido

type

Indica cómo debe cumplirse la intención, ya sea al ejecutar una función de Lambda o al devolver los datos de ranura a la aplicación cliente.

Tipo: cadena

Valores válidos: `ReturnIntent | CodeHook`

Obligatorio: sí

codeHook

Una descripción de la función de Lambda que se ejecuta para cumplir con la intención.

Tipo: objeto CodeHook

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

InputContext

Servicio: Amazon Lex Model Building Service

El nombre de un contexto que debe estar activo para que Amazon Lex seleccione una intención.

Contenido

name

El nombre del contexto.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Intent

Servicio: Amazon Lex Model Building Service

Identifica la versión específica de una intención.

Contenido

intentName

El nombre de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

intentVersion

La versión de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IntentMetadata

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de una intención.

Contenido

createdDate

La fecha en que se creó la intención.

Tipo: marca temporal

Obligatorio: no

description

Una descripción de la intención.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

lastUpdatedDate

La fecha de actualización de la intención. Cuando se crea una intención, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

Obligatorio: no

name

El nombre de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

version

La versión de la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

KendraConfiguration

Servicio: Amazon Lex Model Building Service

Proporciona información de configuración para AMAZON. KendraSearchIntentintención. Cuando usa esta intención, Amazon Lex busca en el índice de Amazon Kendra especificado y devuelve documentos del índice que coinciden con la expresión del usuario. Para obtener más información, consulte AMAZON. KendraSearchIntent.

Contenido

kendraIndex

El nombre del recurso de Amazon (ARN) del índice Amazon Kendra que desea incluir en AMAZON. KendraSearchIntent intención de buscar. El índice debe estar en la misma cuenta y región que el bot de Amazon Lex. Si el índice de Amazon Kendra no existe, obtiene una excepción al llamar a la operación `PutIntent`.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9][a-zA-Z0-9_-]*`

Obligatorio: sí

role

El nombre de recurso de Amazon (ARN) de un rol de IAM que tiene permiso para buscar en el índice de Amazon Kendra. El rol debe estar en la misma cuenta y región que el bot de Amazon Lex. Si el rol no existe, obtiene una excepción al llamar a la operación `PutIntent`.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `arn:aws:iam::[0-9]{12}:role/.*`

Obligatorio: sí

queryFilterString

Un filtro de consulta que Amazon Lex envía a Amazon Kendra para filtrar la respuesta de una consulta. El filtro está en el formato definido por Amazon Kendra. Para obtener más información, consulte Filtrado de consultas.

Puede anular esta cadena de filtros con una nueva cadena de filtros en tiempo de ejecución.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LogSettingsRequest

Servicio: Amazon Lex Model Building Service

Los parámetros que se utilizan para configurar el modo de entrega y el destino de los registros de conversaciones.

Contenido

destination

La ubicación en la que se entregan los registros. Los registros de texto se envían a un grupo de CloudWatch registros. Los registros de audio se envían a un bucket de S3.

Tipo: cadena

Valores válidos: `CLOUDWATCH_LOGS | S3`

Obligatorio: sí

logType

El tipo de registros que desea habilitar. Los registros de texto se envían a un grupo de CloudWatch registros. Los registros de audio se envían a un bucket de S3.

Tipo: cadena

Valores válidos: `AUDIO | TEXT`

Obligatorio: sí

resourceArn

El nombre de recurso de Amazon (ARN) del grupo de registros de CloudWatch registros o del depósito de S3 donde se deben entregar los registros.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: `^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])$`

Obligatorio: sí

kmsKeyArn

El nombre de recurso de Amazon (ARN) de la clave administrada por el cliente de AWS KMS para el cifrado de los registros de audio enviados a un bucket de S3. La clave no se aplica a CloudWatch los registros y es opcional para los buckets de S3.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LogSettingsResponse

Servicio: Amazon Lex Model Building Service

La configuración de los registros de conversaciones.

Contenido

destination

El destino en el que se entregan los registros.

Tipo: cadena

Valores válidos: `CLOUDWATCH_LOGS | S3`

Obligatorio: no

kmsKeyArn

El nombre de recurso de Amazon (ARN) de una clave que se utiliza para cifrar registros de audio en un bucket de S3.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Obligatorio: no

logType

El tipo de registros que están habilitados.

Tipo: cadena

Valores válidos: `AUDIO | TEXT`

Obligatorio: no

resourceArn

El nombre de recurso de Amazon (ARN) del grupo de registros de CloudWatch registros o del depósito de S3 donde se entregan los registros.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: `^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])$`

Obligatorio: no

resourcePrefix

El prefijo del recurso es la primera parte de la clave del objeto de S3 incluido en el bucket de S3 que ha especificado para almacenar registros de audio. En el caso de CloudWatch los registros, es el prefijo del nombre del flujo de registros del grupo de registros que especificó.

Tipo: cadena

Limitaciones de longitud: longitud máxima de 1024 caracteres.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Message

Servicio: Amazon Lex Model Building Service

El objeto del mensaje que proporciona el texto del mensaje y su tipo.

Contenido

content

El texto del mensaje.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1000 caracteres.

Obligatorio: sí

contentType

El tipo de contenido de la cadena de mensaje.

Tipo: cadena

Valores válidos: `PlainText | SSML | CustomPayload`

Obligatorio: sí

groupNumber

Identifica el grupo de mensajes al que pertenece el mensaje. Si un grupo está asignado a un mensaje, Amazon Lex devuelve un mensaje de cada grupo en la respuesta.

Tipo: entero

Rango válido: valor mínimo de 1. Valor máximo de 5.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

MigrationAlert

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de las alertas y advertencias que Amazon Lex envía durante una migración. Las alertas incluyen información sobre cómo solucionar el problema.

Contenido

details

Detalles adicionales sobre la alerta.

Tipo: matriz de cadenas

Obligatorio: no

message

Un mensaje que describe la forma en la que se emite la alerta.

Tipo: cadena

Requerido: no

referenceURLs

Un enlace a la documentación de Amazon Lex que describe cómo se resuelve la alerta.

Tipo: matriz de cadenas

Obligatorio: no

type

El tipo de alerta. Existen dos tipos de alertas:

- `ERROR`: se ha producido un problema con la migración que no se puede resolver. La migración se detiene.
- `WARN`: se ha producido un problema con la migración que requiere cambios manuales en el nuevo bot de Amazon Lex V2. La migración continúa.

Tipo: cadena

Valores válidos: `ERROR | WARN`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

MigrationSummary

Servicio: Amazon Lex Model Building Service

Proporciona información sobre la migración de un bot de Amazon Lex V1 a Amazon Lex V2.

Contenido

migrationId

El identificador único que Amazon Lex ha asignado a la migración.

Tipo: cadena

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

Obligatorio: no

migrationStatus

El estado de la operación. Cuando el estado es `COMPLETE`, el bot está disponible en Amazon Lex V2. Es posible que haya alertas y advertencias que deban resolverse antes de completar la migración.

Tipo: cadena

Valores válidos: `IN_PROGRESS | COMPLETED | FAILED`

Obligatorio: no

migrationStrategy

La estrategia empleada para llevar a cabo la migración.

Tipo: cadena

Valores válidos: `CREATE_NEW | UPDATE_EXISTING`

Obligatorio: no

migrationTimestamp

La fecha y hora en que comenzó la migración.

Tipo: marca temporal

Obligatorio: no

v1BotLocale

La configuración regional del bot de Amazon Lex V1 que es el origen de la migración.

Tipo: cadena

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`

Obligatorio: no

v1BotName

El nombre del bot de Amazon Lex V1 que es el origen de la migración.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 2. La longitud máxima es de 50 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

v1BotVersion

La versión del bot de Amazon Lex V1 que es el origen de la migración.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

v2BotId

El identificador único del bot de Amazon Lex V2 que es el destino de la migración.

Tipo: cadena

Limitaciones de longitud: longitud fija de 10.

Patrón: `^[0-9a-zA-Z]+$`

Obligatorio: no

v2BotRole

El rol de IAM que Amazon Lex utiliza para ejecutar el bot de Amazon Lex V2.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

OutputContext

Servicio: Amazon Lex Model Building Service

La especificación de un contexto de salida que se establece cuando se cumple una intención.

Contenido

name

El nombre del contexto.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

timeToLiveInSeconds

El número de segundos que debería estar activo el contexto después de enviarse en una respuesta `PostContent` o `PostText`. Puede establecer el valor entre 5 y 86 400 segundos (24 horas).

Tipo: entero

Rango válido: valor mínimo de 5. Valor máximo de 86 400.

Obligatorio: sí

turnsToLive

El número de turnos de conversación en los que el contexto debe estar activo. Un turno de conversación es una solicitud `PostContent` o `PostText` y la respuesta correspondiente de Amazon Lex.

Tipo: entero

Rango válido: valor mínimo de 1. Valor máximo de 20.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Prompt

Servicio: Amazon Lex Model Building Service

Obtiene información del usuario. Para definir una solicitud, proporcione uno o más mensajes y especifique el número de intentos requerido para obtener información del usuario. Si proporciona más de un mensaje, Amazon Lex elige uno de los mensajes para enviar una solicitud al usuario. Para obtener más información, consulte Funcionamiento de Amazon Lex.

Contenido

maxAttempts

El número de veces que se envían solicitudes al usuario para obtener información.

Tipo: entero

Rango válido: valor mínimo de 1. Valor máximo de 5.

Obligatorio: sí

messages

Una matriz de objetos en la que cada uno proporciona una cadena de mensaje y su tipo. Puede especificar la cadena de mensaje como texto sin formato o con el formato Speech Synthesis Markup Language (SSML).

Tipo: Matriz de Message objetos

Miembros de la matriz: número mínimo de 1 artículo. La cantidad máxima es de 15 elementos.

Obligatorio: sí

responseCard

Una tarjeta de respuesta. Amazon Lex utiliza esta solicitud durante el tiempo de ejecución, en la respuesta de la API `PostText`. Sustituye a los atributos de la sesión y a los valores de ranura de los marcadores de posición en la tarjeta de respuesta. Para obtener más información, consulte Uso de una tarjeta de respuesta.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 50 000.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ResourceReference

Servicio: Amazon Lex Model Building Service

Describe el recurso que hace referencia al recurso que intenta eliminar. Este objeto se devuelve como parte de la excepción `ResourceInUseException`.

Contenido

name

El nombre del recurso que utiliza el recurso que intenta eliminar.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `[a-zA-Z_]+`

Obligatorio: no

version

La versión del recurso que utiliza el recurso que intenta eliminar.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Slot

Servicio: Amazon Lex Model Building Service

Identifica la versión de una ranura específica.

Contenido

name

El nombre del slot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z](-|_|.)?)+$`

Obligatorio: sí

slotConstraint

Especifica si el slot es obligatorio u opcional.

Tipo: cadena

Valores válidos: `Required | Optional`

Obligatorio: sí

defaultValueSpec

Una lista de valores predeterminados para la ranura. Los valores predeterminados se utilizan cuando Amazon Lex no ha determinado el valor de una ranura. Puede especificar valores predeterminados a partir de variables de contexto, atributos de sesión y valores definidos.

Tipo: objeto SlotDefaultValueSpec

Obligatorio: no

description

Una descripción de la ranura.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

obfuscationSetting

Determina si una ranura se ofusca en los registros de conversaciones y los enunciados almacenados. Cuando una ranura se ofusca, el valor se sustituye por el nombre de la ranura entre llaves ({}). Por ejemplo, si el nombre de la ranura es “full_name”, los valores ofuscados se sustituyen por “{full_name}”. Para obtener más información, consulte Ofuscación de ranuras.

Tipo: cadena

Valores válidos: `NONE | DEFAULT_OBFUSCATION`

Obligatorio: no

priority

Indica a Amazon Lex el orden en el que debe obtener este valor de ranura del usuario. Por ejemplo, si la intención tiene dos ranuras con prioridades 1 y 2, AWS Amazon Lex obtiene en primer lugar un valor para la ranura con prioridad 1.

Si varias ranuras comparten la misma prioridad, Amazon Lex obtiene los valores en un orden arbitrario.

Tipo: entero

Rango válido: valor mínimo de 0. Valor máximo de 100.

Obligatorio: no

responseCard

Un conjunto de respuestas posibles para el tipo de ranura que se utiliza en clientes basados en texto. El usuario elige una opción de la tarjeta de respuesta, en vez de responder con texto.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 50 000.

Obligatorio: no

sampleUtterances

Si conoce un patrón específico por el que los usuarios pueden responder a una solicitud de Amazon Lex para un valor de ranura, puede proporcionar esos enunciados para mejorar la

precisión. Es opcional. En la mayoría de los casos, Amazon Lex es capaz de entender los enunciados de los usuarios.

Tipo: matriz de cadenas

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 200 caracteres.

Obligatorio: no

slotType

El tipo de la ranura, que puede ser un tipo de ranura personalizada que haya definido o uno de los tipos de ranura integrados.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Obligatorio: no

slotTypeVersion

La versión del tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

valueElicitationPrompt

El mensaje que usa Amazon Lex para obtener el valor de ranura del usuario.

Tipo: objeto Prompt

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotDefaultValue

Servicio: Amazon Lex Model Building Service

Un valor predeterminado para una ranura.

Contenido

defaultValue

El valor predeterminado de la ranura. Puede especificar uno de los siguientes valores:

- `#context-name.slot-name`: el valor de la ranura “slot-name” en el contexto “context-name”
- `{attribute}`: el valor de la ranura del atributo de la sesión “attribute”
- `'value'`: el valor independiente “value”

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 202.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotDefaultValueSpec

Servicio: Amazon Lex Model Building Service

Contiene los valores predeterminados de una ranura. Los valores predeterminados se utilizan cuando Amazon Lex no ha determinado el valor de una ranura.

Contenido

defaultValueList

Los valores predeterminados de una ranura. Puede especificar más de un valor predeterminado. Por ejemplo, puede especificar un valor predeterminado para que se utilice a partir de una variable de contexto coincidente, un atributo de la sesión o un valor fijo.

El valor predeterminado elegido se selecciona en función del orden en el que los haya especificado en la lista. Por ejemplo, si especifica una variable de contexto y un valor fijo, en ese orden, Amazon Lex utiliza la variable de contexto si está disponible y, si no, utiliza el valor fijo.

Tipo: matriz de objetos SlotDefaultValue

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 10 elementos.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotTypeConfiguration

Servicio: Amazon Lex Model Building Service

Proporciona información de configuración para un tipo de ranura.

Contenido

regexConfiguration

Una expresión regular utilizada para validar el valor de un slot.

Tipo: objeto SlotTypeRegexConfiguration

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotTypeMetadata

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de un tipo de ranura.

Contenido

createdDate

La fecha de creación del tipo de ranura.

Tipo: marca temporal

Obligatorio: no

description

Una descripción del tipo de slot.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 200 caracteres.

Obligatorio: no

lastUpdatedDate

La fecha de actualización del tipo de ranura. Cuando se crea un recurso, la fecha de creación y la fecha de la última actualización son la misma.

Tipo: marca temporal

Obligatorio: no

name

El nombre del tipo de slot.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: no

version

La versión del tipo de ranura.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

SlotTypeRegexConfiguration

Servicio: Amazon Lex Model Building Service

Proporciona una expresión regular utilizada para validar el valor de un slot.

Contenido

pattern

Una expresión regular utilizada para validar el valor de un slot.

Use una expresión regular estándar. Amazon Lex admite los siguientes caracteres en la expresión regular:

- A-Z, a-z
- 0-9
- Caracteres Unicode (“\u<Unicode>”)

Representa caracteres Unicode con cuatro dígitos, por ejemplo, “\u0041” o “\u005A”.

No se admiten los siguientes operadores de expresiones regulares:

- Repetidores infinitos: *, + o {x,} sin límite superior.
- Comodín (.)

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Statement

Servicio: Amazon Lex Model Building Service

Un conjunto de mensajes que ofrece información al usuario. En tiempo de ejecución, Amazon Lex selecciona el mensaje que se transmite.

Contenido

messages

Un conjunto de objetos de mensaje.

Tipo: Matriz de Message objetos

Miembros de la matriz: número mínimo de 1 artículo. La cantidad máxima es de 15 elementos.

Obligatorio: sí

responseCard

En tiempo de ejecución, si el cliente utiliza la PostTextAPI, Amazon Lex incluye la tarjeta de respuesta en la respuesta. Sustituye a todos los atributos de la sesión y a los valores de ranura de los marcadores de posición en la tarjeta de respuesta.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 50 000.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Tag

Servicio: Amazon Lex Model Building Service

Una lista de pares de clave/valor que identifica a un bot, un alias de bot o un canal de bot. Los caracteres permitidos para las claves y los valores de etiqueta son letras y dígitos Unicode, espacios en blanco y cualquiera de estos símbolos: _ . : / = + - @.

Contenido

key

La clave de la etiqueta. Las claves no distinguen entre mayúsculas y minúsculas y deben ser únicas.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 128 caracteres.

Obligatorio: sí

value

El valor asociado a una clave. Puede ser una cadena vacía, pero no puede ser nulo.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 256 caracteres.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

UtteranceData

Servicio: Amazon Lex Model Building Service

Proporciona información acerca de un enunciado hecho al bot.

Contenido

count

El número de veces que se ha procesado el enunciado.

Tipo: entero

Obligatorio: no

distinctUsers

El número total de personas que han usado el enunciado.

Tipo: entero

Obligatorio: no

firstUtteredDate

La fecha en la que se registró el enunciado por primera vez.

Tipo: marca temporal

Obligatorio: no

lastUtteredDate

La fecha en la que se registró el enunciado por última vez.

Tipo: marca temporal

Obligatorio: no

utteranceString

El texto introducido por el usuario o la representación en texto de un clip de audio.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 2000.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

UtteranceList

Servicio: Amazon Lex Model Building Service

Proporciona una lista de los enunciados que se han hecho a una versión específica del bot. La lista contiene un máximo de 100 enunciados.

Contenido

botVersion

La versión del bot que ha procesado la lista.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 64.

Patrón: `\$LATEST|[0-9]+`

Obligatorio: no

utterances

Uno o más objetos UtteranceData que contienen información acerca de los enunciados hechos a un bot. El número máximo de objetos es 100.

Tipo: matriz de objetos UtteranceData

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Amazon Lex Runtime Service

Amazon Lex Runtime Service admite los siguientes tipos de datos:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

ActiveContext

Servicio: Amazon Lex Runtime Service

Un contexto es una variable que contiene información acerca del estado actual de la conversación entre un usuario y Amazon Lex. Amazon Lex puede establecer un contexto automáticamente cuando se cumple una intención. También puede establecerse en el tiempo de ejecución con la operación `PutContent`, `PutText` o `PutSession`.

Contenido

name

El nombre del contexto.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Patrón: `^([A-Za-z]_?)+$`

Obligatorio: sí

parameters

Indica las variables del contexto actual. Puede utilizar estos valores como valores predeterminados de ranuras en eventos posteriores.

Tipo: mapa de cadena a cadena

Entradas de mapa: número mínimo de 0 elementos. Número máximo de 10 artículos.

Limitaciones de longitud de la clave: longitud mínima de 1. La longitud máxima es de 100 caracteres.

Limitaciones de longitud de los valores: longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Obligatorio: sí

timeToLive

El tiempo o el número de turnos que permanece activo un contexto.

Tipo: objeto ActiveContextTimeToLive

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ActiveContextTimeToLive

Servicio: Amazon Lex Runtime Service

El tiempo o el número de turnos que permanece activo un contexto.

Contenido

timeToLiveInSeconds

El número de segundos que debería estar activo el contexto después de enviarse en una respuesta `PostContent` o `PostText`. Puede establecer el valor entre 5 y 86 400 segundos (24 horas).

Tipo: entero

Rango válido: valor mínimo de 5. Valor máximo de 86 400.

Obligatorio: no

turnsToLive

El número de turnos de conversación en los que el contexto debe estar activo. Un turno de conversación es una solicitud `PostContent` o `PostText` y la respuesta correspondiente de Amazon Lex.

Tipo: entero

Rango válido: valor mínimo de 1. Valor máximo de 20.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Button

Servicio: Amazon Lex Runtime Service

Representa una opción para mostrar en la plataforma cliente (Facebook, Slack, etc.).

Contenido

text

El texto que ve el usuario en el botón.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 15.

Obligatorio: sí

value

El valor que se envía a Amazon Lex cuando un usuario selecciona el botón. Por ejemplo, si el texto del botón es “BCN”, cuando el usuario selecciona el botón, el valor que se envía puede ser “Barcelona”.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1000 caracteres.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

DialogAction

Servicio: Amazon Lex Runtime Service

Describe la siguiente acción que debería realizar el bot durante la interacción con el usuario y proporciona información acerca del contexto en el que se da la acción. Utilice el tipo de dato `DialogAction` para establecer la interacción en un estado específico o para devolverla a un estado anterior.

Contenido

type

La siguiente acción que el bot debería realizar durante la interacción con el usuario. Los valores posibles son:

- `ConfirmIntent`: la siguiente acción pregunta al usuario si la intención es completa y está lista para cumplirse. Se trata de una pregunta cuya respuesta es “Sí” o “No”, como “¿Desea realizar el pedido?”.
- `Close`: indica que no habrá respuesta del usuario. Por ejemplo, la afirmación “El pedido se ha realizado” no requiere una respuesta.
- `Delegate`: Amazon Lex determina la siguiente acción.
- `ElicitIntent`: la siguiente acción determina la intención que desea cumplir el usuario.
- `ElicitSlot`: la siguiente acción es obtener un valor de ranura por parte del usuario.

Tipo: cadena

Valores válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Obligatorio: sí

fulfillmentState

El estado de cumplimiento de la intención. Los valores posibles son:

- `Failed`: la función de Lambda asociada a la intención no ha podido cumplirla.
- `Fulfilled`: la función de Lambda asociada a la intención ha podido cumplirla.
- `ReadyForFulfillment`: toda la información necesaria para la intención está presente y la aplicación del cliente puede cumplir la intención.

Tipo: cadena

Valores válidos: `Fulfilled | Failed | ReadyForFulfillment`

Obligatorio: no

intentName

El nombre de la intención.

Tipo: cadena

Requerido: no

message

El mensaje que debería mostrarse al usuario. Si no especifica un mensaje, Amazon Lex utilizará el mensaje configurado para la intención.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Obligatorio: no

messageFormat

- `PlainText`: el mensaje contiene texto UTF-8 sin formato.
- `CustomPayload`: el mensaje está en un formato personalizado del cliente.
- `SSML`: el mensaje contiene texto con formato para salida de voz.
- `Composite`: el mensaje contiene un objeto JSON de escape que, a su vez, contiene uno o más mensajes. Para obtener más información, consulte Grupos de mensajes.

Tipo: cadena

Valores válidos: `PlainText | CustomPayload | SSML | Composite`

Obligatorio: no

slots

Un mapa de las ranuras recopiladas y sus valores.

Tipo: mapa de cadena a cadena

Obligatorio: no

slotToElicit

El nombre de la ranura que debería obtener el usuario.

Tipo: cadena

Requerido: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

GenericAttachment

Servicio: Amazon Lex Runtime Service

Representa la opción que ve el usuario cuando se muestra una solicitud. Puede ser una imagen, un botón, un enlace o texto.

Contenido

attachmentLinkUrl

La dirección URL de una vinculación a la tarjeta de respuesta.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Obligatorio: no

buttons

La lista de opciones que se muestra al usuario.

Tipo: matriz de objetos Button

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 5 artículos.

Obligatorio: no

imageUrl

La dirección URL de una imagen mostrada al usuario.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Obligatorio: no

subTitle

El subtítulo que aparece debajo del título.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 80.

Obligatorio: no

title

El título de la opción.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 80.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IntentConfidence

Servicio: Amazon Lex Runtime Service

Proporciona una puntuación que indica el grado de confianza de Amazon Lex en lo que respecta a la capacidad de una intención para satisfacer las expectativas del usuario.

Contenido

score

Una puntuación que indica el grado de confianza de Amazon Lex en lo que respecta a la capacidad de una intención para satisfacer las expectativas del usuario. Varía entre 0,00 y 1,00. Una puntuación más alta indica una mayor confianza.

Tipo: Doble

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IntentSummary

Servicio: Amazon Lex Runtime Service

Proporciona información acerca del estado de una intención. Puede utilizar esta información para obtener el estado actual de una intención para procesarla o para devolverla a su estado anterior.

Contenido

dialogActionType

La siguiente acción que el bot debería realizar durante la interacción con el usuario. Los valores posibles son:

- `ConfirmIntent`: la siguiente acción pregunta al usuario si la intención es completa y está lista para cumplirse. Se trata de una pregunta cuya respuesta es “Sí” o “No”, como “¿Desea realizar el pedido?”.
- `Close`: indica que no habrá respuesta del usuario. Por ejemplo, la afirmación “El pedido se ha realizado” no requiere una respuesta.
- `ElicitIntent`: la siguiente acción determina la intención que desea cumplir el usuario.
- `ElicitSlot`: la siguiente acción es obtener un valor de ranura por parte del usuario.

Tipo: cadena

Valores válidos: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Obligatorio: sí

checkpointLabel

Una etiqueta definida por el usuario que identifica una intención concreta. Puede utilizar esta etiqueta para volver a una intención anterior.

Utilice el parámetro `checkpointLabelFilter` de la operación `GetSessionRequest` para filtrar las intenciones que la operación ha devuelto y que incluyen dicha etiqueta.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `[a-zA-Z0-9-]+`

Obligatorio: no

confirmationStatus

El estado de la intención una vez que el usuario ha respondido a la pregunta de confirmación. Si el usuario confirma la intención, Amazon Lex establece el valor del campo en `Confirmed`. Si el usuario deniega la intención, Amazon Lex establece el valor del campo en `Denied`. Los valores posibles son:

- `Confirmed`: el usuario ha respondido “sí” a la pregunta de confirmación, por lo que se confirma que la intención es completa y que está lista para cumplirse.
- `Denied`: el usuario ha respondido “no” a la pregunta de confirmación.
- `None`: indica que en ningún momento se ha solicitado confirmación al usuario o que se le ha enviado la solicitud, pero no la ha confirmado ni denegado.

Tipo: cadena

Valores válidos: `None | Confirmed | Denied`

Obligatorio: no

fulfillmentState

El estado de cumplimiento de la intención. Los valores posibles son:

- `Failed`: la función de Lambda asociada a la intención no ha podido cumplirla.
- `Fulfilled`: la función de Lambda asociada a la intención ha podido cumplirla.
- `ReadyForFulfillment`: toda la información necesaria para la intención está presente y la aplicación del cliente puede cumplir la intención.

Tipo: cadena

Valores válidos: `Fulfilled | Failed | ReadyForFulfillment`

Obligatorio: no

intentName

El nombre de la intención.

Tipo: cadena

Requerido: no

slots

Un mapa de las ranuras recopiladas y sus valores.

Tipo: mapa de cadena a cadena

Obligatorio: no

slotToElicit

La siguiente ranura que se obtiene por parte del usuario. Si no hay ninguna ranura que se pueda obtener, el campo está vacío.

Tipo: cadena

Requerido: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

PredictedIntent

Servicio: Amazon Lex Runtime Service

Una intención sugerida por Amazon Lex satisface la intención del usuario. Incluye el nombre de la intención, la confianza que Amazon Lex tiene en que se ha satisfecho la intención del usuario y las ranuras definidas para la intención.

Contenido

intentName

El nombre de la intención sugerida por Amazon Lex que satisface la intención del usuario.

Tipo: cadena

Requerido: no

nluIntentConfidence

Indica el grado de confianza de Amazon Lex en lo que respecta a la capacidad de una intención para satisfacer las expectativas del usuario.

Tipo: objeto IntentConfidence

Obligatorio: no

slots

La ranura y los valores de ranura asociados a la intención predicha.

Tipo: mapa de cadena a cadena

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ResponseCard

Servicio: Amazon Lex Runtime Service

Si configura una tarjeta de respuesta al crear bots, Amazon Lex sustituye los atributos de la sesión y los valores de ranura que están disponibles y, a continuación, los devuelve. La tarjeta de respuesta también puede proceder de una función de Lambda (`dialogCodeHook` y `fulfillmentActivity` en una intención).

Contenido

contentType

El tipo de contenido de la respuesta.

Tipo: cadena

Valores válidos: `application/vnd.amazonaws.card.generic`

Obligatorio: no

genericAttachments

Una matriz de objetos adjuntos que representan opciones.

Tipo: matriz de objetos GenericAttachment

Miembros de la matriz: número mínimo de 0 artículos. Número máximo de 10 artículos.

Obligatorio: no

version

La versión del formato de la tarjeta de respuesta.

Tipo: cadena

Requerido: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

SentimentResponse

Servicio: Amazon Lex Runtime Service

La opinión expresada en un enunciado.

Cuando el bot está configurado para enviar enunciados a Amazon Comprehend con el fin de analizar opiniones, esta estructura de campo contiene el resultado del análisis.

Contenido

sentimentLabel

La opinión deducida en la que Amazon Comprehend tiene mayor confianza.

Tipo: cadena

Requerido: no

sentimentScore

La probabilidad de que la opinión se haya deducido correctamente.

Tipo: cadena

Requerido: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Historial de documentos de Amazon Lex

- Última actualización de la documentación: 9 de septiembre de 2021

En la siguiente tabla se describen los cambios importantes en cada versión de Amazon Lex. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS.

Cambio	Descripción	Fecha
Nueva característica	Amazon Lex ahora es compatible con la configuración regional coreana (ko-KR). Para obtener más información, consulte Idiomas compatibles con Amazon Lex.	9 de septiembre de 2021
Nueva característica	Amazon Lex ahora es compatible con la configuración regional en inglés (India). Para obtener más información, consulte Idiomas compatibles con Amazon Lex.	15 de julio de 2021
Nueva característica	Amazon Lex ahora proporciona una herramienta para migrar un bot a la API de Amazon Lex V2. Para obtener más información, consulte Migración de un bot.	13 de julio de 2021
Nueva característica	Amazon Lex ahora es compatible con la configuración regional japonesa (Japón). Para obtener más	1 de abril de 2021

	información, consulte Idiomas compatibles con Amazon Lex.	
Nueva característica	Amazon Lex ahora es compatible con las configuraciones regionales en alemán (Alemania) (de-DE) y español (Latinoamérica) (es-419). Para obtener más información, consulte Idiomas compatibles con Amazon Lex.	23 de noviembre de 2020
Nueva característica	Amazon Lex ahora admite el uso de contextos para administrar intenciones de activación. Para obtener más información, consulte Configuración del contexto de intención.	19 de noviembre de 2020
Nueva característica	Amazon Lex ahora es compatible con las configuraciones regionales en francés (fr-FR), francés de Canadá (fr-CA) y español (es-ES). Para ver una lista con todas las configuraciones regionales compatibles, consulte Idiomas compatibles con Amazon Lex.	11 de noviembre de 2020
Nueva característica	Amazon Lex ahora es compatible con la configuración regional en español (EE. UU.) (es-US). Para obtener más información, consulte Idiomas compatibles con Amazon Lex.	22 de septiembre de 2020

Nueva característica	Amazon Lex ahora es compatible con la configuración regional en inglés (Reino Unido) (en-GB). Para obtener más información, consulte Idiomas compatibles con Amazon Lex.	15 de septiembre de 2020
Nueva característica	Amazon Lex ahora es compatible con la configuración regional en inglés (Australia) (en-AU). Para obtener más información, consulte Idiomas compatibles con Amazon Lex.	8 de septiembre de 2020
Nueva característica	Amazon Lex ahora cuenta con 7 nuevas intenciones integradas y 9 nuevos tipos de ranuras integrados. Para obtener más información, consulte Intenciones y tipos de ranuras integrados.	8 de septiembre de 2020
Nuevo ejemplo	Aprenda a crear un bot de Amazon Lex que permita a los agentes de atención al cliente buscar respuestas a las preguntas de los clientes con Amazon Kendra. Para obtener más información, consulte Ejemplo: asistente para agentes de centros de llamadas.	10 de agosto de 2020

Nueva característica	Amazon Lex ahora puede devolver hasta cuatro intenciones alternativas basadas en puntuaciones de confianza. Para obtener más información, consulte Uso de puntuaciones de confianza.	6 de agosto de 2020
Expansión regional	Amazon Lex ya está disponible en Asia-Pacífico (Tokio) (ap-northeast-1).	30 de junio de 2020
Nueva característica	Ahora, Amazon Lex permite buscar respuestas a preguntas frecuentes en índices de Amazon Kendra. Para obtener más información, consulte AMAZON.KendraSearchIntent.	11 de junio de 2020
Nueva característica	Amazon Lex ahora devuelve más información en los registros de conversaciones. Para obtener más información, consulte Visualización de registros de texto en Registros de Amazon CloudWatch.	9 de junio de 2020
Expansión regional	Amazon Lex ya está disponible en Asia-Pacífico (Singapur) (ap-southeast-1), Europa (Fráncfort) (eu-central-1) y Europa (Londres) (eu-west-2).	23 de abril de 2020

Nueva característica	Amazon Lex ahora admite etiquetas. Puede utilizar el etiquetado para identificar recursos, asignar costos y controlar el acceso. Para obtener más información, consulte Etiquetado de los recursos de Amazon Lex.	12 de marzo de 2020
Nueva característica	Amazon Lex ahora admite expresiones regulares para el tipo de ranura integrado AMAZON.AlphaNumeric. Para obtener más informaci ón, consulte AMAZON.Al phaNumeric.	6 de febrero de 2020
Nueva característica	Ahora, Amazon Lex puede registrar información de conversación y ocultar valores de ranura en esos registros. Para obtener más información, consulte Creación de registros de conversaciones y Ocultació n de slots.	19 de diciembre de 2019
Expansión regional	Amazon Lex ya está disponibl e en Asia-Pacífico (Sídney) (ap-southeast-2).	17 de diciembre de 2019
Nueva característica	Amazon Lex ahora cumple con la HIPAA. Para obtener más información, consulte Validación de la conformidad para Amazon Lex.	10 de diciembre de 2019

Nueva característica	Amazon Lex ahora puede enviar expresiones del usuario a Amazon Comprehend para analizar su opinión. Para obtener más información, consulte este artículo sobre el análisis de emociones.	21 de noviembre de 2019
Nueva característica	Amazon Lex ahora cumple con los requisitos de SOC. Para obtener más informaci ón, consulte Validación de la conformidad para Amazon Lex.	19 de noviembre de 2019
Nueva característica	Amazon Lex ahora cumple con los requisitos de PCI. Para obtener más informaci ón, consulte Validación de la conformidad para Amazon Lex.	17 de octubre de 2019
Nueva característica	Se ha añadido compatibi lidad para añadir un punto de comprobación en una intención para que pueda volver fácilmente a la intención durante una conversación. Para obtener más información, consulte Administración de las sesiones.	10 de octubre de 2019

Nueva característica	Se ha añadido compatibilidad con `AMAZON.FallbackIntent` , de modo que su bot pueda gestionar situaciones en las que la entrada del usuario no es la esperada. Para obtener más información, consulte AMAZON.FallbackIntent.	3 de octubre de 2019
Nueva característica	Amazon Lex le permite administrar información sobre los bots de la sesión. Para obtener más información, consulte Administración de sesiones con la API de Amazon Lex.	8 de agosto de 2019
Expansión regional	Amazon Lex ya está disponible en Oeste de EE. UU. (Oregón) (us-west-2).	8 de mayo de 2018
Nueva característica	Se ha agregado compatibilidad para la exportación e importación en formato de Amazon Lex. Para obtener más información, consulte Importación y exportación de bots, intenciones y tipos de ranuras de Amazon Lex.	13 de febrero de 2018
Nueva característica	Amazon Lex ahora admite mensajes de respuesta adicionales de bots. Para obtener más información, consulte Respuestas.	8 de febrero de 2018

Expansión regional	Amazon Lex ya está disponible en Europa (Irlanda) (eu-west-1).	21 de noviembre de 2017
Nueva característica	Se ha agregado compatibilidad con la implementación de bots de Amazon Lex en Kik Para obtener más información, consulte Integración de un bot de Amazon Lex con Kik.	20 de noviembre de 2017
Nueva característica	Se ha agregado compatibilidad con nuevos atributos de solicitud y tipos de slot integrados. Para obtener más información, consulte Tipos de intenciones integradas y Definición de los atributos de solicitud.	3 de noviembre de 2017
Nueva característica	Se ha añadido la exportación a la función Alexa Skills Kit. Para obtener más información, consulte Exportación de una habilidad de Alexa.	7 de septiembre de 2017
Nueva característica	Se ha añadido compatibilidad con sinónimos para los valores de tipo de slot. Para obtener más información, consulte Tipos de slots personalizados.	31 de agosto de 2017

Nueva característica	Agregada integración con AWS CloudTrail. Para obtener más información, consulte Supervisión de las llamadas a la API de Amazon Lex con Registros de AWS CloudTrail.	15 de agosto de 2017
Documentación ampliada	Se han añadido ejemplos de introducción para la AWS CLI. Para obtener más informaci ón, consultar https://docs.aws.amazon.com/lex/latest/dg/gs-cli.html	22 de mayo de 2017
Nueva guía	Esta es la primera versión de la Guía del usuario de Amazon Lex.	19 de abril de 2017

Glosario de AWS

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia de Glosario de AWS.