aws

Guía para desarrolladores de V2

Amazon Lex



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Lex: Guía para desarrolladores de V2

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon Lex V2? 1
Casos de uso habituales de Amazon Lex V2 3
Pagar por Amazon Lex V2 4
¿Es usted First-time usuario de Amazon Lex V2? 4
Trabajar con los SDK de AWS 4
Uso de Amazon Lex V2 con un AWS SDK 6
Funcionalidades más recientes 8
AMAZON.BedrockAgentIntent para conectarse con Bedrock Agents y Knowledge Bases 8
Compatible con la NLU para mejorar la clasificación de intenciones y la resolución de los espacios 8
Compatibilidad con vocabulario personalizado en 17 idiomas adicionales 8
QinConnect intención integrada para la integración de Amazon Connect 9
AMAZON.Currency y las ranuras AMAZON.Confirmation integradas ahora son compatibles con todas las configuraciones regionales 9
Intención QnA mejorada con la base de conocimiento y las barreras de protección de Bedrock 9
Support para los modelos Anthropic Claude 3 9
Resiliencia global para la replicación de bots en todas las regiones 10
Soporte regional para AWS GovCloud (US-West) 10
Características de la IA generativa de Amazon Lex V2 10
AMAZON.Confirmation ranura integrada para Yes/No/Maybe/Don no conocer la desambiguación. 11
Medir el rendimiento empresarial con Analytics 11
Evaluación del rendimiento de los bots con Test Workbench 11
Plantillas de bots verticales específicas 11
Red de bots 12
Generador visual de conversaciones 12
Tipos de slot compuestos 12
Ramificación condicional 12
Diseñador de chatbots automatizados 13
Sugerencias en tiempo de ejecución 13
Vocabulario personalizado 13
Tipo de slot gramatical 13
Conceptos clave de Amazon Lex V2 15

Ruta de aprendizaje de inicio rápido 15
Proceso de desarrollo detallado 15
Conceptos básicos y terminología 16
Características avanzadas de Amazon Lex V2 18
Disponibilidad regional 19
Idiomas compatibles 19
Idiomas y configuraciones regionales compatibles 19
Lenguajes y configuraciones regionales compatibles con las funcionalidades de Amazon Lex V2 23
Regions 25
Introducción 26
Descripción general de 26
Lista de verificación de requisitos previos 26
Introducción a Amazon Lex V2 26
Paso 1: Configurar una cuenta de 27
Inscripción en AWS 27
Creación de un usuario de IAM 28
Concesión de acceso programático 29
Siguiente paso 29
Paso 2: Introducción (consola) 30
Descripción general de la consola 30
Elija su ruta de aprendizaje 30
Ejercicios fundamentales 31
Inicio rápido: crea un chatbot en 5 minutos 31
Ejercicio 1: Crear un chatbot a partir de una plantilla 33
Ejercicio 2: Revisar el flujo de la conversación 37
Ejercicio 3: Cree un chatbot avanzado de servicio al cliente 48
Mejores prácticas para empezar 70
Ejemplos de bots 75
Asistente de agente de call center 75
Paso 1: creación de un índice de Amazon Kendra 76
Paso 2: Crear un bot Amazon Lex V2 77
Paso 3: Añada una personalización y una Built-in intención 78
Paso 4: configuración de Amazon Cognito 79
Paso 5: implementación del bot como aplicación web 81
Paso 6: uso del bot 81

Uso de bots Amazon Lex V2 85
Cambios en los flujos de conversación en Amazon Lex V2 86
Distintas formas de crear un bot con Amazon Lex V2 87
Usar la consola 88
Usar plantillas de bot 89
Creación de bots Amazon Lex V2 con el Diseñador de chatbots automatizados 92
Cómo agregar un nuevo idioma a un bot Amazon Lex V2 101
Adición de intenciones 102
Configurar las indicaciones en un orden específico 104
Enunciados de muestra 105
Estructura de intenciones 107
Crear rutas de conversación 130
Uso del generador visual de conversaciones 148
Intenciones integradas 159
Adición de tipos de ranura 192
Built-in tipos de ranuras 193
Tipos de ranura personalizados 209
Tipo de slot gramatical 212
Tipos de slot compuestos 361
Probar un bot 32
Optimización de bots Lex V2 con IA generativa 373
Generador de bots descriptivo a partir de su descripción 375
Ejemplos de generador de bots descriptivo 379
Permisos necesarios para la NLD 382
Uso de la generación de enunciados para generar ejemplos de enunciados para el reconocimiento de intenciones 382
Uso de la resolución asistida de ranuras para los valores de ranuras 384
Ejemplos de resolución asistida de slots 385
Habilitación en configuraciones de IA generativa 389
Habilitación de la resolución asistida de ranuras para su ranura 390
Permisos para la resolución asistida de slots 392
Uso BedrockAgentIntent de un agente Bedrock en Amazon Lex V2 392
Habilitación en configuraciones de IA generativa 394
Habilitación de la intención del agente de Bedrock para su bot 394
Permisos para la intención del agente de Bedrock 395
Solicitud de ejemplo 395

Mejore la clasificación de intenciones y la resolución de los espacios 397
Desactivación de la NLU asistida 399
Resuelva las entradas ambiguas de los usuarios con la desambiguación de intenciones 399
Desactivar la intención (desambiguación) 402
Optimice el bot con un analizador de bots impulsado por IA 402
Amazon.qn AIntent 408
Permisos 410
Creación de una red de bots para sus bots Lex V2 412
Creación de una red de bots para sus bots Lex V2 413
Administración de su red de bots para sus bots Lex V2 414
Versiones de su red de bots para Lex V2 415
Alias para su red de bots para Lex V2 415
Integraciones de canales para su red de bots Lex V2 415
Implementación de bots Lex V2 para su entorno de producción 417
Control de versiones y alias con su bot Lex V2 417
Versiones 417
Alias de su bot Lex V2 418
Integración con una aplicación Java 420
Uso de la resiliencia global para implementar bots en otras regiones 424
Permisos 426
Implementación de la resiliencia global con su bot Lex V2 427
Integración con plataformas de mensajería 430
Integración con Facebook 431
Integración con Slack 434
Integración con Twilio SMS 439
Integración con los centros de contacto 441
Amazon Chime SDK 442
Cliente de Connect 443
Genesys Cloud 444
Descripción de las conversaciones de los bots de Amazon Lex V2 446
Contexto de conversación con sus bots Lex V2 447
Cómo establecer un contexto de intención para su bot Lex V2 448
Uso de los valores de ranura predeterminados en las intenciones de su bot Lex V2 450
Cómo establecer atributos de sesión para su bot Lex V2 451
Cómo establecer atributos de solicitud para su bot Lex V2 453
Establecer el tiempo de espera de la sesión 456

Cómo compartir información entre intenciones con su bot Lex V2 456
Cómo establecer atributos complejos en su bot Lex V2 457
Introducción a las sesiones de bots 459
Iniciar una nueva sesión 460
Cambiar intenciones 461
Reanudar una intención anterior 461
Validar valores de slot 462
Integrar AWS Lambda funciones 463
AWS Lambdaformato de evento de entrada para Lex V2 464
AWS Lambdaformato de respuesta para Lex V2 471
Campos obligatorios en la respuesta 474
Estructuras comunes 477
Intención 477
Ranuras 479
Estado de la sesión 482
Crear y adjuntar una función de Lambda a un alias de bot 486
Adjunte una AWS Lambda función a un bot de Amazon Lex V2 mediante la consola 489
Adjunte una AWS Lambda función a un bot de Amazon Lex V2 mediante operaciones de API 491
Depuración de una función de Lambda 498
Personalización de las interacciones de los bots con usuarios en Lex V2 499
Análisis del sentimiento de los usuarios en la conversación 499
Uso de puntuaciones de confianza para mejorar la precisión de la conversación 500
Uso de puntuaciones de confianza en la intención para mejorar la selección de intenciones con Lex V2 501
Uso de las puntuaciones de confianza en la transcripción de voz para mejorar las conversaciones con su bot Lex V2 504
Personalización de las transcripciones de voz para usarlas con su bot Lex V2 515
Configuración de las preferencias del modelo de reconocimiento de voz 515
Mejora del reconocimiento de voz con un vocabulario personalizado 521
Configuración de la sensibilidad de detección de actividad de voz 531
Uso de la diarización del altavoz para centrarse en el altavoz principal 533
Mejora del reconocimiento de los valores de las ranuras con sugerencias de tiempo de ejecución en la conversación 539
Captura de valores de ranura con estilos de ortografía durante la conversación 543
Uso del relleno de audio para mejorar la capacidad de respuesta de los bots 550

Tipos de relleno de audio disponibles 551
Parámetros de temporización 551
Configuración del relleno de audio 552
Relleno de audio con mensajes provisionales de un agente de IA 553
Relleno de audio con ganchos de diálogo y código de cumplimiento 554
Prácticas recomendadas para el relleno de audio 554
Supervisión del rendimiento de los bots en Lex V2 555
Medición del rendimiento empresarial con Analytics 555
Definiciones clave 556
Filtrado de resultados 558
Descripción general de 559
Panel de conversación 563
Panel de rendimiento 569
Uso de API para análisis 573
Administración de permisos de acceso para análisis 580
Cómo habilitar registros de conversaciones para sus bots Lex V2 580
Registro de conversaciones con registros de conversaciones en Lex V2 581
Cómo ocultar valores de ranura en registros de conversación de Lex V2 599
Captura selectiva del registro de conversaciones en Lex V2 600
Registro de errores con registros de errores en Lex V2 608
Políticas de IAM para registros de errores 608
Habilitación de registros de errores en Lex V2 609
Desactivación de registros de error en Lex V2 609
Monitoreo de métricas operativas en Lex V2 609
Medición de métricas operativas con CloudWatch Lex V2 610
Visualización de eventos con CloudTrail 619
Evaluación del rendimiento de los bots Lex V2 con Test Workbench 622
Generación de un conjunto de prueba para Test Workbench 623
Administrar conjuntos de prueba 636
Ejecutar una prueba 665
Cobertura del conjunto de prueba en Test Workbench 667
Ver resultados de la prueba 668
Detalles de los resultados de la prueba en Test Workbench 669
Transmisión de conversaciones a su bot Lex V2 676
Inicio de una transmisión de conversación a un bot de Amazon Lex V2 677

Secuencia temporal de los eventos de una conversación de audio cuando se utiliza un bot de Amazon Lex V2 681
Iniciar una conversación en streaming 683
Codificar secuencias de eventos 699
Permitir que se interrumpa al bot 701
Esperar a que el usuario proporcione información adicional 702
Configuración de las actualizaciones del progreso de cumplimiento para su bot Lex V2 704
Actualizaciones de cumplimiento 704
Respuesta posterior al cumplimiento 706
Tiempos de espera para entradas del usuario 708
Cómo funciona el comportamiento de interrupción en un bot Lex V2 709
Cómo establecer tiempos de espera para la entrada de voz 710
Tiempos de espera para la entrada de texto 711
Cómo establecer la configuración para la entrada de DTMF 712
Importación y exportación de bots en Lex V2 714
Exportación de bots desde Lex V2 714
Permisos de IAM requeridos para exportar bots en Lex V2 715
Exportación de un bot Lex V2 (consola) 716
Importación de bots en Lex V2 718
Permisos de IAM requeridos para importar 719
Importación de un bot Lex V2 (consola) 720
Usar una contraseña al importar o exportar 722
Formato JSON para importar y exportar bots en Lex V2 722
Estructura del archivo de manifiesto 723
Estructura del archivo del bot 724
Estructura del archivo de configuración regional del bot 724
Estructura del archivo de intenciones 724
Estructura del archivo del slot 727
Estructura de tipo de slot 730
Estructura del archivo de vocabulario personalizado 733
Recursos de etiquetado en Lex V2 734
Cómo utilizar la consola o la API para etiquetar recursos 734
Restricciones de etiquetas al usar Lex V2 735
Etiquetado de recursos (consola) 735
Seguridad 737
Protección de datos 738

Cifrado en reposo ... 738
Cifrado en tránsito ... 739
Identity and Access Management ... 739
Público ... 740
Autenticación con identidades ... 740
Administración del acceso con políticas ... 742
Cómo funciona Amazon Lex V2 con IAM ... 744
Identity-based ejemplos de políticas ... 753
Resource-based ejemplos de políticas ... 764
Políticas administradas de AWS ... 772
Cómo utilizar roles vinculados a servicios ... 791
Uso de roles de servicio ... 796
Migración a roles de servicio ... 797
Resolución de problemas ... 799
Registrar y monitorear ... 802
Validación de conformidad ... 802
Resiliencia ... 803
Seguridad de la infraestructura ... 804
Puntos de conexión de VPC (AWS PrivateLink) ... 804
Consideraciones sobre los puntos de conexión de Amazon Lex V2 ... 804
Crear un punto de conexión de VPC de interfaz para Amazon Lex V2 ... 805
Crear una política de puntos de conexión de VPC para Amazon Lex V2 ... 805
Ejemplos de código ... 807
Escenarios ... 807
Creación de un chatbot Amazon Lex V2 ... 807
Directrices y prácticas recomendadas ... 809
Cuotas ... 812
Build-time cuotas ... 812
Cuotas de tiempo de ejecución ... 815
Guía de migración ... 820
Descripción de Amazon Lex V2 ... 820
Varios idiomas en un bot ... 820
Arquitectura de la información simplificada ... 820
Mejorar la productividad de los creadores ... 821
AWS CloudFormation recursos ... 823
Amazon Lex V2 y CloudFormation plantillas ... 823

Obtenga más información sobre CloudFormation 823
Historial de revisión 825
Referencia de la API 845
AWSGlosario de 846
..... dcccxlvii

¿Qué es Amazon Lex V2?

Amazon Lex V2 es un servicio de AWS para crear interfaces de conversación para las aplicaciones que usan voz y texto. Amazon Lex V2 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 V2 permite a cualquier desarrollador crear chatbots conversacionales rápidamente. Con Amazon Lex V2, no se necesitan conocimientos de aprendizaje profundo: para crear un chatbot, debe especificar el flujo de conversación básico en la consola Amazon Lex V2. Amazon Lex V2 gestiona 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 chatbots de dispositivos móviles, aplicaciones web y plataformas de chat (por ejemplo, Facebook Messenger).

Amazon Lex V2 ahora incluye AI-powered funciones avanzadas que hacen que la creación de bots sea aún más potente y accesible. Assisted NLU utiliza modelos de lenguaje extensos (LLM) para mejorar la clasificación de intenciones y la resolución de los espacios, sin perder de vista los espacios e intenciones configurados por el bot. Esto se traduce en una mejor comprensión de las solicitudes de los usuarios y se requieren menos datos de formación. Amazon Lex V2 también admite vocabularios personalizados en 17 idiomas adicionales, lo que permite su implementación global con una precisión de reconocimiento de voz mejorada en diversos mercados.

Con la Multi-Region replicación (MRR), ahora puede implementar sus bots en varias regiones de AWS para mejorar la disponibilidad y la recuperación ante desastres, lo que garantiza que sus interfaces de conversación sigan siendo accesibles para los usuarios de todo el mundo.

Amazon Lex V2 ofrece integración con AWS Lambda muchos otros servicios de la plataforma AWS y usted puede integrarlos, como Connect Customer, Amazon Comprehend y Amazon Kendra. La integración con Lambda proporciona a los bots acceso a conectores empresariales sin servidor integrados previamente para vincular datos de aplicaciones de SaaS, por ejemplo con Salesforce.

En el caso de los bots creados después del 17 de agosto de 2022, puede utilizar la ramificación condicional para controlar el flujo de la conversación con su bot. Con la ramificación condicional, puede crear conversaciones complejas sin necesidad de escribir código Lambda.

Amazon Lex V2 ofrece las siguientes ventajas:

- Simplicidad: Amazon Lex V2 le guía a través de la utilización de la consola para crear su propio chatbot en cuestión de minutos. Solo tiene que proporcionar algunas frases de ejemplo y Amazon Lex V2 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 V2 proporciona tecnologías de ASR y NLU para crear un sistema de comprensión del lenguaje hablado (SLU). Por medio de SLU, Amazon Lex V2 recibe la entrada de texto y voz en lenguaje natural, entiende la intención de la entrada y lleva a cabo la intención del usuario invocando la función de negocio apropiada.

  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 V2 pone las tecnologías de aprendizaje profundo al alcance de todos los desarrolladores. Los chatbots de Amazon Lex V2 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 sus bots y crear un valor añadido diferente para sus clientes, para definir categorías de productos totalmente nuevas gracias a las interfaces de conversación.

- Implementación y escalado sin problemas: con Amazon Lex V2 puede crear, probar e implementar sus chatbots directamente desde la consola de Amazon Lex V2. Amazon Lex V2 le permite publicar fácilmente sus chatbots de voz o texto para su uso en dispositivos móviles, aplicaciones web y servicios de chat (por ejemplo, Facebook Messenger). Amazon Lex V2 se escala automáticamente. No tiene que preocuparse por el aprovisionamiento del hardware y la administración de la infraestructura para impulsar la experiencia de su bot.

- Built-in integración con la plataforma AWS: Amazon Lex V2 funciona de forma nativa con otros servicios de AWS, como AWS Lambda Amazon CloudWatch. Puede aprovechar la eficacia de la plataforma de AWS para seguridad, monitorización, autenticación de usuarios, lógica de negocio, almacenamiento y desarrollo de aplicaciones móviles.

- Cost-effectiveness— Con Amazon Lex V2, no hay costes iniciales ni tarifas 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. Con el nivel gratuito de Amazon Lex V2 puede probar fácilmente Amazon Lex V2 sin ninguna inversión inicial.

Casos de uso habituales de Amazon Lex V2

Amazon Lex V2 le permite crear interfaces de conversación sofisticadas para una amplia variedad de escenarios empresariales. Estos son algunos de los casos de uso más populares:

- Customer Support: cree chatbots inteligentes que puedan gestionar las consultas habituales de los clientes, solucionar problemas y trasladar problemas complejos a agentes humanos. Intégrelo con sus sistemas de CRM y base de conocimientos existentes.
- E-commerce y venta minorista: cree chatbots de asistencia de compras que ayuden a los clientes a encontrar productos, comprobar el estado de los pedidos, procesar las devoluciones y ofrecer recomendaciones personalizadas basadas en el historial de compras.
- Reserva de citas: desarrolla chatbots de programación para el sector de la salud, los servicios profesionales o la hostelería que puedan comprobar la disponibilidad, reservar citas, enviar recordatorios y gestionar las cancelaciones.
- Servicio de asistencia de TI: cree chatbots de soporte internos que puedan restablecer las contraseñas, proporcionar orientación sobre la instalación del software, realizar un seguimiento de las solicitudes de TI y conectar a los empleados con los recursos técnicos adecuados.
- Servicios financieros: cree chatbots bancarios que puedan comprobar los saldos de las cuentas, transferir fondos, proporcionar el historial de transacciones y ofrecer asesoramiento financiero, manteniendo estrictos estándares de seguridad.
- Recursos humanos y servicios para empleados: desarrolle chatbots auxiliares de recursos humanos que puedan responder a preguntas sobre políticas, ayudar con la inscripción a las prestaciones, procesar las solicitudes de tiempo libre y brindar apoyo para la incorporación de nuevos empleados.

Amazon Lex V2 se integra perfectamente con las plataformas de comunicación más populares, como Slack WhatsApp, Microsoft Teams, Facebook Messenger y aplicaciones web personalizadas, lo que facilita la implementación de los chatbots donde los usuarios ya se comunican.

Pagar por Amazon Lex V2

Amazon Lex V2 solo le cobra por las solicitudes de texto o voz que realice. Esto ofrece a los desarrolladores un servicio de costo variable que puede crecer junto con sus empresas mientras disfrutan de las ventajas de costos que ofrece la infraestructura de AWS. Para obtener más información, consulte los precios de Amazon Lex V2.

Cuando se registra en AWS, su cuenta de AWS se registra automáticamente en todos los servicios de AWS, incluido Amazon Lex V2. No obstante, solo se le cobrará por los servicios que utilice. Si es un cliente nuevo de Amazon Lex V2, puede empezar a utilizar Amazon Lex V2 de forma gratuita. Para obtener más información, consulte Nivel gratuito de AWS.

Para ver su factura, vaya al panel de facturación y gestión de costes en la consola de Administración de facturación y costos de AWS. Para obtener más información sobre la Cuenta de AWS facturación, consulte la Guía AWS Billing del usuario. Si tiene preguntas sobre la facturación de AWS Cuentas de AWS, póngase en contacto con AWS Support.

¿Es usted First-time usuario de Amazon Lex V2?

Si es la primera vez que utiliza Amazon Lex V2, le recomendamos que lea las siguientes secciones en orden:

1. Conceptos clave de Amazon Lex V2: en esta sección se presenta Amazon Lex V2 y las funciones que se utilizan para crear un chatbot.
2. Introducción a Amazon Lex V2: en esta sección va a configurar su cuenta y probar Amazon Lex V2
3. Referencia de la API: esta sección contiene detalles sobre las operaciones de la API.

Uso de Amazon Lex V2 con un AWS SDK

Los kits de desarrollo de software (SDK) de AWS están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ ejemplos de código

Documentación de SDK	Ejemplos de código
CLI de AWS	Ejemplos de código de AWS CLI
AWS SDK para Go	AWS SDK para Go ejemplos de código
AWS SDK para Java	AWS SDK para Java ejemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK para .NET	AWS SDK para .NET ejemplos de código
AWS SDK para PHP	AWS SDK para PHP ejemplos de código
Herramientas de AWS para PowerShell	Herramientas de AWS para PowerShell ejemplos de código
SDK de AWS para Python v4	Ejemplos de código de AWS SDK para Python v4
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) ejemplos de código
AWS SDK para Ruby	AWS SDK para Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

Uso de Amazon Lex V2 con un AWS SDK

Más allá de la interfaz de la consola, Amazon Lex V2 proporciona acceso programático integral a través de los AWS SDK. Esto le permite integrar las capacidades de IA conversacional directamente en sus aplicaciones, automatizar las tareas de administración de bots y crear soluciones escalables.

Al usar un AWS SDK con Amazon Lex V2, puede:

- Automatizar la creación y la administración de bots: cree, actualice e implemente bots mediante programación sin la interacción manual de la consola
- Integrarlo con las aplicaciones existentes: añada interfaces de conversación a aplicaciones web, aplicaciones móviles y sistemas empresariales
- Escalar las operaciones de los bots: administre varios bots, intenciones y tipos de ranuras de forma eficiente mediante el código
- Implemente flujos de trabajo personalizados: cree flujos de conversación sofisticados que se integren con su lógica empresarial

En la siguiente tabla se muestran los AWS SDK que admiten las operaciones de Amazon Lex V2. Elija el SDK que mejor se adapte a su entorno de desarrollo y siga los enlaces proporcionados para comenzar con la instalación y la implementación.

Lenguaje de programación	SDK de AWS	Recursos para empezar
Java	AWS SDK para Java 2.x	Guía para desarrolladores Referencia de la API
Python	AWS SDK para Python (Boto3)	Introducción Referencia de la API
JavaScript/ Node.js	AWS SDK para la JavaScript versión 3	Guía para desarrolladores Referencia de la API
.NET/C#	AWS SDK para .NET	Guía para desarrolladores

Lenguaje de programación	SDK de AWS	Recursos para empezar
		Referencia de la API
Go	AWS SDK para Go v2	Guía para desarrolladores Referencia de la API
Ruby	AWS SDK para Ruby	Guía para desarrolladores Referencia de la API
PHP	AWS SDK para PHP	Guía para desarrolladores Referencia de la API

Para empezar a utilizar cualquier SDK:

1. Instala el SDK para tu lenguaje de programación preferido siguiendo las instrucciones de instalación de la Guía para desarrolladores
2. Configure sus credenciales de AWS y los ajustes de la región
3. Configurar los permisos de IAM necesarios para las operaciones de Amazon Lex V2.
4. Consulte la documentación de referencia de la API para ver las operaciones específicas que necesita
5. Comience con operaciones básicas, como crear un bot o enumerar los recursos existentes

Para ver ejemplos detallados y una guía paso a paso sobre la creación de bots mediante programación, consulta los enlaces a la documentación del SDK que aparecen en la tabla anterior.

Funcionalidades más recientes

Este tema proporciona información sobre las características más recientes que ofrece Amazon Lex V2:

AMAZON.BedrockAgentIntent para conectarse con Bedrock Agents y Knowledge Bases

Amazon Lex V2 ahora ofrece la intención AMAZON.BedrockAgentIntent integrada de conectar sin problemas su bot con los agentes y bases de conocimiento de Amazon Bedrock para mejorar las capacidades de IA conversacional.

- Documentación

Compatible con la NLU para mejorar la clasificación de intenciones y la resolución de los espacios

Amazon Lex V2 ahora ofrece una NLU asistida para mejorar la clasificación de intenciones y la resolución de ranuras mediante modelos de lenguaje de gran tamaño (LLM) y, al mismo tiempo, permanecer dentro de las intenciones y los espacios configurados por el bot.

- Documentación

Compatibilidad con vocabulario personalizado en 17 idiomas adicionales

Amazon Lex V2 ahora admite vocabularios personalizados en 17 idiomas adicionales para mejorar la precisión del reconocimiento de voz.

- Documentación

QinConnect intención integrada para la integración de Amazon Connect

Amazon Lex V2 ahora ofrece la intención QinConnect integrada de conectar sin problemas su bot con Amazon Connect para mejorar las experiencias de los centros de contacto.

- Documentación

AMAZON.Currency y las ranuras AMAZON.Confirmation integradas ahora son compatibles con todas las configuraciones regionales

Amazon Lex V2 ahora admite AMAZON.Currency tipos de ranuras AMAZON.Confirmation integrados en todas las configuraciones regionales, lo que va más allá de las configuraciones regionales en inglés y español admitidas anteriormente.

- Documentación

Intención QnA mejorada con la base de conocimiento y las barreras de protección de Bedrock

La intención integrada de QnA para IA generativa ahora es compatible con la base de conocimiento y las barreras de protección de Bedrock para brindar respuestas más seguras y precisas.

- Documentación

Support para los modelos Anthropic Claude 3

La intención integrada de QnA para IA generativa ahora es compatible con los modelos Anthropic Claude 3 Haiku y Anthropic Claude 3 Sonnet para mejorar las capacidades de conversación.

- Documentación

Resiliencia global para la replicación de bots en todas las regiones

Amazon Lex V2 ahora ofrece resiliencia global para replicar su bot en una segunda región de AWS para mejorar la disponibilidad y la recuperación ante desastres.

- Documentación

Soporte regional para AWS GovCloud (US-West)

Amazon Lex V2 ya está disponible en AWS GovCloud (US-West).

- Puntos de conexión y cuotas de Amazon Lex

Características de la IA generativa de Amazon Lex V2

Amazon Lex V2 permite ahora aprovechar las capacidades de la IA generativa de Amazon Bedrock para su bot.

- Generador de bots descriptivo
  - Publicación sobre novedades
  - Documentación
- Resolución de slots asistida
  - Publicación sobre novedades
  - Documentación
- Generación de enunciados
  - Publicación sobre novedades
  - Documentación
- `AMAZON.QnAIntent` (Preguntas frecuentes sobre conversación)
  - Publicación sobre novedades
  - Documentación
- Entrada de blog sobre machine learning de AWS

AMAZON.Confirmation ranura integrada para Yes/No/Maybe/Don no conocer la desambiguación.

Amazon Lex V2 ahora incluye una ranura `AMAZON.Confimation` integrada para mejorar la precisión de la confirmación de la ranura y de las respuestas Yes/No/Maybe/Don no conocidas.

- Documentación

Medir el rendimiento empresarial con Analytics

Amazon Lex V2 ahora ofrece a los usuarios la posibilidad de ver el rendimiento de las intenciones y los slots en el panel de Analytics.

- Publicación sobre novedades
- Documentación

Evaluación del rendimiento de los bots con Test Workbench

Amazon Lex V2 ahora ofrece a los usuarios la posibilidad de crear y ejecutar conjuntos de pruebas para medir el rendimiento de los bots y mejorar sus métricas.

- Publicación sobre novedades
- Documentación
- Entrada de blog sobre machine learning de AWS

Plantillas de bots verticales específicas

Amazon Lex V2 ahora ofrece a los usuarios plantillas de bots previamente compilados con flujos de conversación listos para su uso junto con datos de entrenamiento y mensajes de diálogo, tanto para las modalidades de voz como de chat.

- Publicación sobre novedades
- Documentación

Red de bots

Amazon Lex V2 ahora ofrece a los usuarios la posibilidad de combinar varios bots en una sola red y la posibilidad de enrutar las solicitudes al bot correspondiente en función de las entradas del usuario.

- Publicación sobre novedades
- Documentación

Generador visual de conversaciones

Amazon Lex V2 ahora ofrece un generador visual de conversaciones de arrastrar y soltar para diseñar y visualizar fácilmente las rutas de conversación mediante el uso de intenciones en un entorno visual enriquecido.

- Publicación sobre novedades
- Documentación
- Entrada de blog sobre machine learning de AWS

Tipos de slot compuestos

Amazon Lex V2 ahora ofrece a los usuarios la posibilidad de combinar varios slots en un slot compuesto mediante expresiones lógicas.

- Publicación sobre novedades
- Documentación

Ramificación condicional

Amazon Lex V2 ahora ofrece a los usuarios la posibilidad de escribir condiciones para controlar mejor el camino que siguen los clientes a lo largo de una conversación con su bot.

- Publicación sobre novedades
- Documentación

Diseñador de chatbots automatizados

Amazon Lex V2 ahora ofrece a los usuarios la opción de diseñar automáticamente un chatbot a partir de las transcripciones de las conversaciones. Lea los ejemplos de uso.

- Publicación sobre novedades
- Documentación
- Entrada de blog sobre machine learning de AWS
- Página del Diseñador de chatbots automatizados de Amazon Lex

Sugerencias en tiempo de ejecución

Amazon Lex V2 ahora ofrece a los usuarios la opción de configurar sugerencias en tiempo de ejecución para mejorar el reconocimiento de frases y mejorar la obtención de los valores de los slots.

- Publicación sobre novedades
- Documentación

Vocabulario personalizado

Amazon Lex V2 ahora ofrece a los usuarios la opción de crear un vocabulario personalizado, una lista de frases que pueden incluir nombres propios o palabras de dominios específicos, para que Amazon Lex V2 las reconozca en la entrada de audio.

- Publicación sobre novedades
- Documentación
- Entrada de blog sobre machine learning de AWS

Tipo de slot gramatical

Amazon Lex V2 ahora ofrece a los usuarios la posibilidad de crear gramáticas en formato XML siguiendo la especificación gramatical de reconocimiento de voz (SRGS) para recopilar información en una conversación.

- Publicación sobre novedades
- Documentación
- Entrada de blog sobre machine learning de AWS

Conceptos clave de Amazon Lex V2

Amazon Lex V2 le permite crear aplicaciones de chat (bots) para obtener información de los usuarios para realizar una tarea. Por ejemplo, puedes crear un chatbot para brindar atención al cliente, responder a las preguntas frecuentes o reservar citas. A continuación, presentamos los pasos habituales para trabajar con Amazon Lex V2:

Ruta de aprendizaje de inicio rápido

¿Eres nuevo en Amazon Lex V2? Siga esta ruta de aprendizaje progresivo para empezar rápidamente:

1. Comience con una plantilla (5 minutos): elija entre las plantillas de chatbot prediseñadas, como Customer Support FAQ, Appointment Booking o Order Status. Las plantillas incluyen intenciones, ranuras y ejemplos de expresiones preconfiguradas.
2. Personaliza tu Chatbot (15 minutos): modifica la plantilla para que se adapte a tu caso de uso específico. Añade tus propias intenciones, actualiza ejemplos de expresiones y configura los tipos de ranuras para tu dominio.
3. Pruebe y perfeccione (10 minutos): utilice la consola de pruebas integrada para mantener conversaciones con su chatbot. Activa la NLU asistida para mejorar la comprensión con un mínimo de datos de entrenamiento.
4. Implemente e integre (20 minutos): publique su chatbot e intégrelo en su plataforma preferida (Slack, aplicación web o aplicación móvil).

Tiempo total de funcionamiento del chatbot: unos 50 minutos

Para una comprensión más completa, continúe con el proceso de desarrollo detallado que se indica a continuación.

Proceso de desarrollo detallado

Para bots más complejos o para crear desde cero, sigue este proceso de desarrollo integral:

1. Cree un bot y añada uno o más idiomas. Configure el bot para que entienda el objetivo del usuario, 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 proporcionada por la consola de Amazon Lex V2.
3. Publique una versión y cree un alias.
4. Implemente el bot. Puede implementar el bot en sus propias aplicaciones o plataformas de mensajería, como Facebook Messenger o Slack.

Conceptos básicos y terminología

Antes de empezar, familiarícese con los siguientes conceptos y terminología esenciales de Amazon Lex V2:

- Bot: un bot realiza tareas automatizadas como, por ejemplo, pedir una pizza, reservar un hotel, pedir flores, etc. Un bot de Amazon Lex V2 emplea capacidades de reconocimiento automático de voz (ASR) y comprensión del lenguaje natural (NLU).

  Los bots de Amazon Lex V2 pueden comprender entradas del usuario en forma de voz o texto y conversar en lenguaje natural.
- Idioma: un bot de Amazon Lex V2 puede conversar en uno o más idiomas. Cada idioma es independiente de los demás. Puede configurar Amazon Lex V2 para que converse con un usuario mediante palabras y frases en su idioma nativo. Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.
- 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: nombre descriptivo de la intención. Por ejemplo, **`OrderPizza`**.
  - 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 llevar a cabo la intención: la manera en que desea llevar a cabo la intención después de que el usuario proporcione la información necesaria. Es recomendable crear una función de Lambda para el cumplimiento de la intención.

    De forma opcional, puede configurar la intención para que Amazon Lex V2 simplemente devuelva la información a la aplicación cliente y que esta se ocupe del cumplimiento.

Además de las intenciones personalizadas, Amazon Lex V2 proporciona intenciones integradas para configurar rápidamente un bot. Para obtener más información, consulte Intenciones integradas.

Amazon Lex siempre incluye una intención alternativa para cada bot. La intención alternativa se utiliza siempre que Amazon Lex no pueda deducir la intención del usuario. Para obtener más información, consulte AMAZON.FallbackIntent.

- Slot: una intención puede requerir ninguno o varios slots o parámetros. Puede añadir parámetros como parte de la configuración. En tiempo de ejecución, Amazon Lex V2 solicita al usuario valores de slot específicos. El usuario deben proporcionar valores para todos los slots obligatorios para que Amazon Lex V2e pueda atender a la intención.

  Por ejemplo, la intención `OrderPizza` requiere slots como el tamaño, el tipo de masa y el número de pizzas. Para cada slot, hay que proporcionar un tipo de slot y una o más indicaciones para que Amazon Lex V2 los envíe al cliente y obtenga valores del usuario. Un usuario puede responder con un valor de slot que contenga palabras adicionales, como «una pizza grande, por favor» o «prefiero el tamaño pequeño». Amazon Lex V2 sigue entendiendo el valor del slot.

- Tipo de slot: cada slot tiene un tipo. Puede crear sus propios tipos de slot o utilizar tipos de slot integrados. 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 V2 también ofrece tipos de slot 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 integradas.

- Versión: 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. Una vez que haya creado una versión, podrá usar un bot tal y como existía cuando se creó la versión. Después de crear una versión, esta no cambia mientras continúa trabajando en su aplicación.

- Alias: un alias es un puntero hacia una versión específica de un bot. Con un alias, puede actualizar la versión que usan las aplicaciones de su cliente. Por ejemplo, puede apuntar un alias hacia la versión 1 de su bot. Cuando esté listo para actualizar el bot, publique la versión 2 y cambie 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.

Características avanzadas de Amazon Lex V2

Además de los conceptos básicos anteriores, Amazon Lex V2 incluye funciones avanzadas que mejoran las capacidades de los bots:

- NLU asistida: utiliza modelos de lenguaje extensos (LLMs) para mejorar la clasificación de intenciones y la resolución de los espacios. Esta función ayuda a tu bot a entender las solicitudes de los usuarios con mayor precisión, incluso cuando utilizan una redacción diferente a la de tus ejemplos de entrenamiento. La NLU asistida funciona dentro de las intenciones y los espacios configurados, lo que proporciona una mejor comprensión sin requerir una gran cantidad de datos de entrenamiento.
- Conversaciones en varios turnos: Amazon Lex V2 puede mantener el contexto en varios turnos de conversación, lo que permite back-and-forth interacciones naturales. Los usuarios pueden proporcionar información de forma gradual, cambiar de opinión o hacer preguntas aclaratorias sin perder el contexto de la conversación.
- Cambio de contexto: los bots avanzados pueden gestionar los cambios de tema dentro de una conversación. Por ejemplo, un usuario puede empezar a preguntar sobre la información de la cuenta, pasar a realizar un pedido y volver al tema original. Amazon Lex V2 puede gestionar estos cambios de contexto sin problemas.
- Estrategias alternativas: cuando Amazon Lex V2 no entiende la solicitud de un usuario, puede configurar comportamientos alternativos sofisticados, como aclarar preguntas, solicitar sugerencias o recurrir a agentes humanos. Esto garantiza que los usuarios siempre tengan un camino a seguir en la conversación.
- Gestión del flujo de conversación: utilice la ramificación condicional y los controles de flujo de conversación para crear patrones de diálogo complejos sin necesidad de escribir código. Puede enrutar las conversaciones en función de las respuestas de los usuarios, los valores de los intervalos o los datos externos.

Disponibilidad regional

Para obtener una lista de las regiones de AWS en las que está disponible Amazon Lex V2, consulte los puntos de enlace y las cuotas de Amazon Lex V2 en la Referencia general de Amazon Web Services.

Idiomas y configuraciones regionales compatibles con Amazon Lex V2

Amazon Lex V2 admite una variedad de idiomas y configuraciones regionales. En este tema se muestran los idiomas compatibles y las características que admiten estos idiomas.

Idiomas y configuraciones regionales compatibles

Amazon Lex V2 admite los siguientes idiomas y configuraciones regionales.

Código	Lenguaje y configuración
af_za*	Afrikaans (Sudáfrica)
ar_AE	Árabe del Golfo (Emiratos Árabes Unidos)
Ar_sa*	Árabe (Arabia Saudí)
BG_BG*	Búlgaro (Bulgaria)
ca_ES	Catalán (España)
CS_CZ*	Checo (República Checa)
CY_GB*	Galés (Reino Unido)
da_DK*	Danés (Dinamarca)
de_AT	Alemán (austriaco)
de_CH*	Alemán (Suiza)
de_DE	Alemán (Alemania)

Código	Lenguaje y configuración
en_AB*	Inglés (Escocia)
en_AU	Inglés (Australia)
en_GB	Inglés (Reino Unido)
en_ES*	Inglés (Irlanda)
en_IN	Inglés (India)
en_NZ*	Inglés (Nueva Zelanda)
en_US	English (EE. UU.)
es_WL*	Inglés (Gales)
en_ZA	Inglés (Sudáfrica)
es_419	Español (Latinoamérica)
es_ES	Español (España)
es_MX*	Español (México)
es_US	Español (EE. UU.)
et_ET*	Estonio (Estonia)
Fa_ir*	Farsi (Irán)
fi_FI	Finés (Finlandia)
fr_be*	Francés (Bélgica)
fr_CA	Francés (Canadá)
fr_FR	Francés (Francia)
He_il*	Hebreo (Israel)

Código	Lenguaje y configuración
hi_IN	Hindi (India)
HR_hr*	Croata (Croacia)
Hu_hu*	Húngaro (Hungría)
ID_ID*	Indonesio (Indonesia)
Es_es*	Islandés (Islandia)
It_it	Italiano (Italia)
ja_JP	Japonés (Japón)
KM_kh*	Khmer (Camboya)
ko_KR	Coreano (Corea)
lt_LT*	Lituano (Lituania)
lv_LV*	Letón (Letonia)
Señora_mi*	Malayo (Malasia)
NL_be*	Flamenco (Bélgica)
nl_NL	Neerlandés (Países Bajos)
no_NO	Noruego (Noruega)
pl_PL	Polaco (Polonia)
pt_BR	Portugués (Brasil)
pt_PT	Portugués (Portugal)
Ro_ro*	Rumano (Rumanía)
ru_RU*	Ruso (Rusia)

Código	Lenguaje y configuración
SK_SK*	Eslovaco (Eslovaquia)
SL_si*	Esloveno (Eslovenia)
So_so*	Somalí (Somalia)
SR_RS*	Serbio (Serbia)
SU_ID*	Sundanese (Indonesia)
sv_SE	Sueco (Suecia)
Th_th*	Tailandés (Tailandia)
TL_ph*	Tagalog/Filipino (Filipinas)
tr_TR*	Turco (Turquía)
UK_UA*	Ucraniano (Ucrania)
VI_VN*	Vietnamita (Vietnam)
zh_CN	Mandarín (PRC)
zh_HK	Cantonés (Hong Kong)
Zu_za*	Zulú (Sudáfrica)

Locales con soporte de funciones limitado

Las configuraciones regionales marcadas con un asterisco (*) admiten funciones limitadas mediante IA generativa y el reconocimiento automático de voz (ASR) o Text-To-Speech (TTS) de terceros. Consulta la siguiente tabla para ver una lista completa de las funciones compatibles.

Compatibilidad con funciones de IA generativa

La compatibilidad con las funciones de IA generativa incluye la NLU asistida en modo principal, la resolución asistida de ranuras y la desambiguación intencional.

Lenguajes y configuraciones regionales compatibles con las funcionalidades de Amazon Lex V2

En la siguiente tabla se enumeran las funcionalidades de Amazon Lex V2 que están limitadas a determinados idiomas y configuraciones regionales. Todas las demás funcionalidades de Amazon Lex V2 se admiten en todos los idiomas y configuraciones regionales.

Funcionalidad	Idiomas y configuraciones regionales compatibles
AMAZON.AlphaNumeric	Todos los idiomas y configuraciones regionales excepto el coreano (ko_KR) y algunas funciones admiten configuraciones regionales
AMAZON.KendraSearchIntent	Inglés (EE. UU.) (en-US)
Mejora del reconocimiento de voz con un vocabulario personalizado	Inglés (Reino Unido) (en_GB) Inglés (EE. UU.) (en-US)
Diseñador de chatbots automatizados	Inglés (EE. UU.) (en-US)
Disponibilidad por región	Los siguientes idiomas y configuraciones regionales no están disponibles en las regiones de Asia-Pacífico (Singapur) (ap-southeast-1) y África (Ciudad del Cabo) (ap-south-1): • Árabe del Golfo (Emiratos Árabes Unidos) (ar_AE) • Catalán (España) (ca_ES) • Finés (Finlandia) (fi_FI) • Hindi de la India (hi-IN) • Neerlandés (Países Bajos) • Noruego (Noruega) (no_NO) • Polaco (pl-PL) • Portugués (Brasil) (pt-BR) • pt-PT: portugués (Portugal)

Funcionalidad	Idiomas y configuraciones regionales compatibles
	• Sueco (sv-SE) • Mandarín (PRC) (zh_CN) • Cantonés (Hong Kong) (zh_HK)
Cómo establecer un contexto de intención para su bot Lex V2	Inglés (EE. UU.) (en-US)
Tipo de slot gramatical	Inglés (Australia) (en-AU) Inglés de Reino Unido (es-GB) Inglés (EE. UU.) (en-US)
Uso de valores múltiples en un slot	Inglés (EE. UU.) (en-US)
Mejora del reconocimiento de los valores de las ranuras con sugerencias de tiempo de ejecución en la conversación	Inglés (Reino Unido) (en_GB) Inglés (EE. UU.) (en-US)
Captura de valores de ranura con estilos de ortografía durante la conversación	Inglés (Australia) (en-AU) Inglés de Reino Unido (es-GB) Inglés (EE. UU.) (en-US)
Uso de puntuaciones de confianza para mejorar la precisión de la conversación	Inglés (Reino Unido) (en_GB) Inglés (EE. UU.) (en-US)
Solo con Third-party ASR (Deepgram)	Neerlandés (Bélgica) (nl_BE) Francés (Bélgica) (fr_BE)

Funcionalidad	Idiomas y configuraciones regionales compatibles
Solo con Third-party TTS () ElevenLabs	afrikáans (Sudáfrica) (Af_za) Búlgaro (Bulgaria) (bg_BG) Croata (Croacia) (hr_HR) Inglés (Escocia) (en_AB) Estonio (Estonia) (et_ET) Farsi (Irán) (fa_IR) Hebreo (Israel) (He_il) Húngaro (Hungría) (hu_HU) Indonesio (Indonesia) (id_ID) Letón (Letonia) (lv_LV) Lituano (Lituania) (lt_LT) Malayo (Malasia) (MS_my) Serbio (Serbia) (SR_RS) Eslovaco (Eslovaquia) (SK_sk) Esloveno (Eslovenia) (SL_si)

Regions

Para obtener una lista de las regiones de AWS en las que Amazon Lex V2 está disponible, consulte las regiones y puntos de enlace de AWS en. Referencia general de AWS

Introducción a Amazon Lex V2

Descripción general de

Si es la primera vez que utiliza Amazon Lex V2, le recomendamos que lea Conceptos clave de Amazon Lex V2 antes de continuar.

Lista de verificación de requisitos previos

Antes de empezar a crear su primer bot de Amazon Lex V2, asegúrese de tener lo siguiente:

- Cuenta de AWS: necesitará una cuenta de AWS activa. Si no tiene uno, puede crearlo de forma gratuita y aprovechar la capa gratuita de AWS, que incluye 10 000 solicitudes de texto y 5 000 solicitudes de voz al mes para Amazon Lex V2. Para obtener instrucciones de configuración detalladas, consulteInscripción en AWS.
- Usuario de IAM con permisos: cree un usuario de IAM con los permisos de Amazon Lex V2 adecuados. Para empezar, le guiaremos en la creación de un usuario administrador. Para obtener información completa sobre los permisos y las políticas de IAM, consulteAdministración de identidades y accesos para Amazon Lex V2.
- Navegador compatible: utilice un navegador web moderno (Chrome, Firefox, Safari o Edge) para acceder a la consola Amazon Lex V2.
- Compromiso de tiempo: espere de 30 a 60 minutos para completar los ejercicios de introducción, según su nivel de experiencia.

Estimación de costos: los ejercicios de esta guía utilizan los recursos de la capa gratuita de AWS. Si se encuentra dentro de los límites de la capa gratuita, no debería haber ningún cargo. De lo contrario, completar todos los ejercicios suele costar menos de 1 dólar.

Introducción a Amazon Lex V2

Antes de poder usar Amazon Lex V2, debe crear una cuenta de AWS y obtener un ID de cuenta de AWS. Usted crea una cuenta de usuario y un rol de IAM que le otorgan los permisos necesarios para usar Amazon Lex V2

Tras crear sus cuentas, puede empezar a utilizar Amazon Lex V2 con la consola de AWS o con la API.

Amazon Lex V2 ofrece operaciones API que se pueden integrar con las aplicaciones existentes. Para ver una lista de operaciones compatibles, consulte la Referencia de la API. Puede utilizar cualquiera de las siguientes opciones:

- AWS SDK: al usar el, SDKs sus solicitudes a Amazon Lex V2 se firman y autentican automáticamente con las credenciales que usted proporciona. Le recomendamos que utilice un SDK para crear la aplicación.
- AWS CLI: puede utilizar la CLI de AWS para acceder a cualquier función de Amazon Lex V2 sin tener que escribir ningún código.

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

Antes de usar Amazon Lex V2 por primera vez, complete las siguientes tareas:

1. Inscripción en AWS
2. Creación de un usuario de IAM

Inscripción en AWS

Si ya tiene una cuenta de AWS, omita esta tarea.

Al suscribirse a Amazon Web Services (AWS), su cuenta de AWS se registra automáticamente en todos los servicios de AWS, incluido Amazon Lex V2. Solo se le cobrará por los servicios que utilice.

Con Amazon Lex V2, paga solo por los recursos que usa. Si es un cliente nuevo de AWS, puede empezar a utilizar Amazon Lex V2 de forma gratuita. Para obtener más información, consulte Capa gratuita de AWS.

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

Crear una cuenta de AWS

- Para crear una cuenta de AWS, consulte ¿Cómo creo y activo una nueva cuenta de AWS? en el Centro de conocimiento de AWS.

Anote el ID de su cuenta de AWS porque lo necesitará para la siguiente tarea.

Creación de un usuario de IAM

Los servicios de AWS, como Amazon Lex V2, 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.

Cree una cuenta de usuario de IAM para acceder a su cuenta de Amazon Lex V2. Para obtener una guía completa sobre la creación de usuarios de IAM, consulte Introducción a IAM en la Guía del usuario de IAM.

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

Ahora, puede obtener acceso a AWS mediante una dirección URL especial y las credenciales de usuario de IAM.

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 Crear el primer grupo de usuarios y administradores de IAM en la Guía del usuario de IAM.
2. Como usuario, puede iniciar sesión en la consola de administración de AWS 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 acerca de IAM, consulte la guía del usuario de IAM.

Problemas comunes de configuración

Si tienes problemas durante la configuración de la cuenta, estas son algunas soluciones habituales:

- Errores de permiso denegado: asegúrese de que su usuario de IAM tenga los permisos de Amazon Lex V2 necesarios. Para empezar, se recomiendan permisos de administrador. Para

obtener información detallada sobre los permisos de Amazon Lex V2, consulte la sección Seguridad de esta guía.

- Problemas de acceso a la consola: asegúrese de iniciar sesión con la URL de usuario de IAM correcta para su cuenta, no con la consola raíz de AWS. Para obtener más información sobre las prácticas recomendadas de IAM, consulte la sección de seguridad de esta guía.
- Disponibilidad regional: Amazon Lex V2 no está disponible en todas las regiones de AWS. Compruebe los puntos de destino y las cuotas de Amazon Lex para confirmar la disponibilidad en la región que prefiera. Para obtener información sobre las configuraciones regionales compatibles, consulteIdiomas y configuraciones regionales compatibles con Amazon Lex V2.
- Límites de nivel gratuito: supervisa tu uso para mantenerte dentro de los límites del nivel gratuito: 10 000 solicitudes de texto y 5000 solicitudes de voz al mes. Para obtener información detallada sobre los precios, consulte Amazon Lex Pricing.
- Límites de cuota: si encuentra límites de servicio, revise las cuotas actuales y solicite aumentos si es necesario. Para obtener más información, consulte Directrices y prácticas recomendadas.

Para obtener ayuda adicional sobre la solución de problemas, consulte la sección Supervisión de esta guía para obtener información sobre los registros de errores y la depuración.

Concesión de acceso programático

Para obtener información sobre cómo obtener el identificador de clave de acceso y la clave de acceso secreta, consulte Cómo entender y obtener sus credenciales de AWS en la Referencia general de AWS.

Configuración del entorno de desarrolladores

Para desarrolladores: si planea usar la AWS CLI o SDKs, también, querrá configurar su entorno de desarrollo. Instale la AWS CLI y configúrela con sus claves de acceso. Para obtener instrucciones detalladas, consulte Instalación de la AWS CLI. Para ver ejemplos de SDK, consulteEjemplos de bots.

Siguiente paso

Paso 2: Introducción (consola)

Paso 2: Introducción (consola)

Descripción general de la consola

La forma más sencilla de aprender a utilizar Amazon Lex V2 es mediante la consola. Elija la ruta de aprendizaje que mejor se adapte a su función y nivel de experiencia:

Elija su ruta de aprendizaje

Elija la ruta de aprendizaje que mejor se adapte a su función y nivel de experiencia:

- Ruta sin código (usuarios empresariales): perfecta para analistas de negocios, gerentes de producto y usuarios sin conocimientos técnicos que desean crear bots utilizando únicamente la interfaz de la consola.
  - Comience con: Inicio rápido: crea un chatbot en 5 minutos
  - A continuación, intente lo siguiente: Ejercicio 1: Crear un chatbot a partir de una plantilla
  - Aprenda: Mejores prácticas para empezar
- Developer Path: ideal para los desarrolladores de software que desean integrar Amazon Lex V2 con aplicaciones y utilizar APIs e implementar una lógica empresarial personalizada.
  - Comience con: Ejercicio 1: Crear un chatbot a partir de una plantilla
  - Avanzado: Ejercicio 3: Cree un chatbot avanzado de servicio al cliente
  - A continuación, explore: Integración de una AWS Lambda función en su bot Amazon Lex V2
  - Reseña: Ejemplos de bots
  - Avanzado: integración de API y uso del SDK
- Enterprise Path: diseñado para arquitectos e ingenieros que planifican despliegues a gran escala con requisitos de seguridad, cumplimiento y escalabilidad.
  - Comience con: Ejercicio 1: Crear un chatbot a partir de una plantilla
  - Ejemplo empresarial: Ejercicio 3: Cree un chatbot avanzado de servicio al cliente
  - Reseña: mejores prácticas de seguridad e IAM
  - Explore: Uso de la resiliencia global para implementar bots en otras regiones
  - Planifique: monitoreo, registro y optimización de costos

Ejercicios fundamentales

Independientemente del camino que elija, le recomendamos comenzar con estos ejercicios fundamentales:

- Ejercicio 1: Cree un bot de Amazon Lex V2 utilizando una plantilla, un bot predefinido que proporcione toda la configuración de bot necesaria. Solo debe hacer un mínimo de trabajo para probar la end-to-end configuración.
- Ejercicio 2: Revisar las estructuras JSON enviadas entre la aplicación cliente y un bot de Amazon Lex V2.
- Ejercicio 3: Cree un bot de chat avanzado de servicio al cliente que demuestre las capacidades de nivel empresarial, como la gestión de pedidos, el aumento inteligente de ventas, la generación de clientes potenciales y las funciones impulsadas por la IA para ofrecer experiencias personalizadas a los clientes.

Inicio rápido: crea un chatbot en 5 minutos

Esta guía de inicio rápido le ayuda a crear un chatbot Amazon Lex V2 que funcione en solo 5 minutos utilizando plantillas prediseñadas. Dispondrá de un chatbot funcional que podrá probar y personalizar inmediatamente según sus necesidades.

Paso 1: Elige una plantilla

Amazon Lex V2 proporciona varias plantillas prediseñadas para casos de uso comunes:

- Preguntas frecuentes sobre Customer Support: gestiona las consultas habituales del servicio de atención al cliente
- Reserva de citas: programe y administre citas
- Estado del pedido: compruebe la información del pedido y el estado de la entrega
- Servicio de asistencia de TI: proporciona asistencia técnica y solución de problemas

Para seleccionar una plantilla

1. Abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/.
2. Elija Crear bot.
3. Seleccione Comenzar con una plantilla.

4. Elija la plantilla de preguntas frecuentes de Customer Support para este inicio rápido.
5. Introduzca un nombre de bot, como**`MyFirstChatbot`**.
6. Seleccione Crear.

Paso 2: Personalización rápida (opcional)

La plantilla viene preconfigurada, pero puedes personalizarla rápidamente para adaptarla a tus necesidades específicas:

Para personalizar tu chatbot

1. En la descripción general del bot, revisa las intenciones preconfiguradas.
2. Elige una intención para modificarla, por ejemplo. GetAccountInfo
3. Agrega tus propios ejemplos de expresiones que coincidan con la forma en que tus clientes podrían formular las solicitudes.
4. Actualiza los mensajes de respuesta para que coincidan con la voz de tu marca.
5. Seleccione Guardar intención.

Paso 3: Pon a prueba tu bot

Pon a prueba tu chatbot inmediatamente con la consola de pruebas integrada:

Para probar tu chatbot

1. Elige Construir para compilar tu bot.
2. Espera a que se complete la compilación (normalmente de 1 a 2 minutos).
3. En la consola de pruebas de la derecha, escribe: **`I need help with my account`**
4. Presiona Entrar y observa la respuesta del bot.
5. Prueba con otras frases para poner a prueba la comprensión del bot.

Paso 4: Despliega tu bot

Una vez que estés satisfecho con las respuestas de tu chatbot, despliégalo para su uso:

Para implementar tu chatbot

1. Selecciona Publicar en el menú de acciones del bot.
2. Crea una nueva versión seleccionando Crear versión.
3. Crea un alias (como «Producción») que apunte a tu versión.
4. Elige tu método de integración.

Siguientes pasos

¡Enhorabuena! Ahora tiene un chatbot Amazon Lex V2 en funcionamiento. Esto es lo que puede hacer a continuación:

- Habilite la NLU asistida: mejore la comprensión con el procesamiento del lenguaje natural impulsado por IA.
- Agregue más intenciones: amplíe las capacidades de su chatbot con casos de uso adicionales.
- Integre con Lambda: añada lógica empresarial e integración de sistemas externos. Consulte Integración de una AWS Lambda función en su bot Amazon Lex V2.
- Configure la supervisión: realice un seguimiento del uso y el rendimiento. Consulte Medición de las métricas operativas con Amazon CloudWatch.
- Conozca las funciones avanzadas: explore la gestión del flujo de conversaciones, los diálogos de varios turnos y el cambio de contexto.

Para obtener un recorrido más detallado, continúa con. Ejercicio 1: Crear un chatbot a partir de una plantilla

Ejercicio 1: Crear un chatbot a partir de una plantilla

En este ejercicio, creará su primer chatbot Amazon Lex V2 y lo probará en la consola Amazon Lex V2. Para este ejercicio, utilizará la OrderFlowersplantilla, que muestra un caso de uso práctico y real del comercio electrónico.

OrderFlowers Ejemplo de bot

Utiliza la OrderFlowersplantilla para crear un chatbot Amazon Lex V2 que pueda gestionar las solicitudes de pedidos de flores. Este ejemplo demuestra cómo las empresas pueden automatizar la

toma de pedidos con chatbots inteligentes. Para obtener más información sobre la estructura de un bot, consulte Conceptos clave de Amazon Lex V2.

- Intentos: el bot incluye un objetivo principal:
  - `OrderFlowers`- Gestiona las solicitudes de pedidos de flores recopilando el tipo de flor, la fecha y la hora de recogida
- Tipos de ranuras: el bot usa tipos de ranuras integrados que reconocen y manejan automáticamente los formatos de datos comunes:
  - AMAZON.Date- Reconoce fechas como «mañana», «el próximo viernes» o «15 de marzo»
  - AMAZON.Time- Reconoce horas como «2 p.m.», «mediodía» o «tres y cuarto»
  - `FlowerTypes`(personalizado) - Variedades de flores específicas como «rosas», «tulipanes», «lirios»
- Tragaperras: la OrderFlowers intención requiere la siguiente información antes de que el bot pueda procesar el pedido de flores:
  - `FlowerType`(tipo FlowerTypes personalizado): el tipo de flores que se van a pedir
  - `PickupDate`(AMAZON.Datetipo) - ¿Cuándo recoger las flores
  - `PickupTime`(AMAZON.Timetipo) - ¿A qué hora recoger las flores
- Ejemplos de expresiones: los siguientes ejemplos de expresiones muestran las formas naturales en que los usuarios pueden solicitar pedidos de flores:
  - "Me gustaría recoger unas flores"
  - «Quiero pedir unas flores»
  - «¿Me pueden dar flores para recogerlas?»
  - «Necesito comprar 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 por el `PickupDate` espacio: «¿Qué día quieres que recojan el {FlowerType}?»
  - Pregunta sobre el `PickupTime` espacio: «¿A qué hora quieres que recojan el {FlowerType}?»
  - Declaración de confirmación: «De acuerdo, tu {FlowerType} estará listo para ser recogido antes del {PickupTime} el {PickupDate}. ¿Te parece bien?»

Crea tu bot

Para crear un bot de Amazon Lex V2 (consola)

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione Crear bot.
3. Para el Método de creación, seleccione Empezar con un ejemplo.
4. En la sección Ejemplos de bots, elija una opción OrderFlowersde la lista.
5. En la sección Configuración del bot, asígnele un nombre y una descripción opcional. El nombre debe ser exclusivo en su cuenta.
6. En la sección Permisos, seleccione Crear un nuevo rol con permisos básicos de Amazon Lex. Esto creará un rol de AWS Identity and Access Management (IAM) con los permisos que Amazon Lex V2 necesita para ejecutar el bot.
7. En la sección Ley de Protección de la Privacidad en Línea para Niños (COPPA), seleccione la respuesta adecuada.
8. En las secciones Tiempo de espera de la sesión agotado y Configuración avanzada, deje la configuración predeterminada.
9. Seleccione Siguiente. Amazon Lex V2 crea su bot.

Añada un idioma a su bot

Después de crear el bot, debe añadir uno o más idiomas compatibles con el bot. Un idioma contiene las intenciones, los tipos de slots y los slots que el bot usa para conversar con los usuarios.

Añadir un idioma a su bot

1. En la sección Idioma, seleccione un idioma compatible y añada una descripción.
2. Deje los campos Interacción por voz y Umbral de puntuación de confianza de la clasificación de intenciones con sus valores predeterminados.
3. Seleccione Listo para añadir el idioma al bot.

Pon a prueba tu bot

Cuando seleccione Listo, la consola abre el editor de intenciones. Puede usar el editor de intenciones para examinar las intenciones utilizadas por el bot. Cuando termine de examinar el bot, puede probarlo.

Para probar el OrderFlowers bot

1. En la parte superior de la página, seleccione Compilar. Espere a que el bot se compile.
2. Cuando se complete la compilación, seleccione Probar para abrir la ventana de prueba.
3. Pruebe el bot. Comience la conversación con uno de los ejemplos de enunciados, como «Me gustaría recoger unas flores».

Permita que la NLU mejore la comprensión

Ahora que tienes un chatbot en funcionamiento, vamos a mejorarlo con la NLU asistida para mejorar el reconocimiento de intenciones y la resolución de los espacios. Assisted NLU utiliza modelos lingüísticos extensos (LLMs) para comprender mejor las solicitudes de los usuarios, incluso cuando utilizan una redacción diferente a la de los ejemplos de formación.

Para habilitar la NLU asistida

1. En la consola Amazon Lex V2, navegue hasta la configuración de su bot.
2. En el panel de navegación izquierdo, selecciona Configuración del bot.
3. En NLU asistida, selecciona Activar.
4. Seleccione Guardar para aplicar los cambios.
5. Vuelva a crear su bot para aplicar la mejora de la NLU asistida.

Prueba la mejora: prueba estas variantes en tu consola de pruebas para ver cómo la NLU asistida gestiona las diferentes frases:

- «Quiero comprar unas rosas» (debería activar la OrderFlowers intención y la captura) FlowerType
- «¿Me pueden entregar flores mañana?» (debería activar OrderFlowers la intención y la captura PickupDate)
- «Necesito tulipanes para recogerlos a las 3 p.m.» (debería activar la OrderFlowers intención y la captura FlowerType y PickupTime)

Observa cómo el chatbot puede entender estas variaciones naturales sin tener que añadirlas como ejemplos explícitos de expresiones. Esto funciona con Assisted NLU, que utiliza la IA para mejorar la comprensión del lenguaje natural.

Siguientes pasos

Ahora que ha creado su primer bot con una plantilla, puede usar la consola para crear su propio bot. Para obtener instrucciones sobre cómo crear un bot personalizado y obtener más información sobre cómo crear bots, consulte Uso de bots Amazon Lex V2.

Ejercicio 2: Revisar el flujo de la conversación

En este ejercicio, revisará las estructuras JSON que se envían entre su aplicación cliente y el bot de Amazon Lex V2 que creó en Ejercicio 1: Crear un chatbot a partir de una plantilla. La conversación utiliza la RecognizeTextoperación para generar las estructuras JSON. RecognizeUtteranceDevuelve la misma información que los encabezados HTTP de la respuesta.

Las estructuras JSON se dividen por cada turno de la conversación. Un turno es una solicitud de la aplicación cliente y una respuesta del bot.

Turno 1

Durante el primer turno de la conversación, la aplicación cliente inicia la conversación con su bot. Tanto el URI como el cuerpo de la solicitud proporcionan información sobre la solicitud.

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

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

- El URI identifica el bot con el que se está comunicando la aplicación cliente. También incluye un identificador de sesión generado por la aplicación cliente que identifica una conversación específica entre un usuario y el bot.
- El cuerpo de la solicitud contiene el texto que el usuario escribió en la aplicación cliente. En este caso, solo se envía el texto; sin embargo, la aplicación puede enviar información adicional, como los atributos de la solicitud o el estado de la sesión. Para obtener más información, consulte la operación RecognizeText.

Desde `text`, Amazon Lex V2 detecta la intención del usuario de pedir flores. Amazon Lex V2 elige una de los slots de la intención (`FlowerType`) y una de las indicaciones del slot y, a continuación, envía la siguiente respuesta a la aplicación cliente. El cliente muestra el mensaje en la respuesta.


```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": null,
                    "PickupDate": null,
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 0.95
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "What type of flowers would you like to order?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "FlowerType",
            "type": "ElicitSlot"
        },
        "intent": {
            "confirmationState": "None",
```

```
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": null,
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Turno 2

En el turno 2, el usuario responde a la solicitud del bot Amazon Lex V2 del turno 1 con un valor que llena el slot `FlowerType`.

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

La respuesta del turno 2 muestra el slot `FlowerType` ocupado y proporciona un mensaje para obtener el valor del siguiente slot.

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

```
            },
            "state": "InProgress"
        },
        "nluConfidence": {
            "score": 0.98
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "What day do you want the dozen roses to be picked up?",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
"sessionState": {
    "dialogAction": {
        "slotToElicit": "PickupDate",
        "type": "ElicitSlot"
    },
    "intent": {
        "confirmationState": "None",
        "name": "OrderFlowers",
        "slots": {
            "FlowerType": {
                "value": {
                    "interpretedValue": "dozen roses",
                    "originalValue": "dozen roses",
                    "resolvedValues": []
                }
            },
            "PickupDate": null,
            "PickupTime": null
        },
        "state": "InProgress"
    },
    "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
}
```

```
}
```

Turno 3

En el turno 3, el usuario responde a la solicitud del bot Amazon Lex V2 del turno 2 con un valor que llena el slot `PickupDate`.

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

La respuesta del turno 3 muestra los slots ocupados `FlowerType` y `PickupDate` y proporciona un mensaje para obtener el valor del último slot.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2022-12-28",
                            "originalValue": "next monday",
                            "resolvedValues": [
                                "2021-01-04"
                            ]
                        }
                    },
                    "PickupTime": null
                },
                "state": "InProgress"
```

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

```
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f",
        "sessionAttributes": {}
    }
}
```

Turno 4

En el turno 4, el usuario proporciona el valor final del slot para la intención, es decir, el momento en que se recogen las flores.

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

En respuesta, Amazon Lex V2 envía una solicitud de confirmación al usuario para confirmar que la solicitud es correcta. El `dialogAction` está configurado en `ConfirmIntent` y el `confirmationState` es `None`.

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2021-01-04",
                            "originalValue": "next monday",
```

```
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "InProgress"
        },
        "nluConfidence": {
            "score": 1.0
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "Okay, your dozen roses will be ready for pickup by 17:00 on
2021-01-04.  Does this sound okay?",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
"sessionState": {
    "dialogAction": {
        "type": "ConfirmIntent"
    },
    "intent": {
        "confirmationState": "None",
        "name": "OrderFlowers",
        "slots": {
```

```
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```


Turno 5

En el último turno, el usuario responde con la indicación de confirmación.


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

En la respuesta que envía Amazon Lex V2, se indica que se ha cumplido la intención al establecer `confirmationState` en `Confirmed` y `dialogAction` para el cierre. Todos los valores de los slots están disponibles en la aplicación cliente.


```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "Confirmed",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2021-01-04",
                            "originalValue": "next monday",
                            "resolvedValues": [
                                "2021-01-04"
                            ]
                        }
                    },
                    "PickupTime": {
                        "value": {
                            "interpretedValue": "17:00",
                            "originalValue": "5 evening",
                            "resolvedValues": [
                                "17:00"
                            ]
                        }
                    }
                },
                "state": "Fulfilled"
            },
            "nluConfidence": {
                "score": 1.0
            }
```

```
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "Thanks. ",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "type": "Close"
        },
        "intent": {
            "confirmationState": "Confirmed",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
```

```
                            "17:00"
                        ]
                    }
                }
            },
            "state": "Fulfilled"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

Ejercicio 3: Cree un chatbot avanzado de servicio al cliente

En este ejercicio avanzado, crearás un sofisticado chatbot de servicio al cliente para una empresa de comercio electrónico. Este bot demuestra las capacidades de nivel empresarial, como la gestión de pedidos, las ventas adicionales inteligentes, la generación de clientes potenciales y la optimización de los ingresos. El chatbot utiliza funciones impulsadas por la IA para ofrecer experiencias personalizadas a los clientes e impulsar el crecimiento empresarial.

SmartCommerce Descripción general del bot de servicio al cliente

El bot de servicio SmartCommerce al cliente está diseñado para gestionar interacciones complejas con los clientes y, al mismo tiempo, maximizar las oportunidades de ingresos. Este ejemplo demuestra cómo las empresas pueden automatizar el servicio al cliente y, al mismo tiempo, impulsar el crecimiento de las ventas mediante una gestión inteligente de las conversaciones.

- Intenciones personalizadas: el bot incluye varias intenciones personalizadas para ofrecer un servicio de atención al cliente integral:
  - `CheckOrderStatus`- Verifica y proporciona información sobre el estado del pedido
  - `ProcessReturn`- Gestiona las solicitudes de devolución y los cambios
  - `UpsellProducts`- Recomienda productos y servicios adicionales
  - `CaptureLeadInfo`- Recopila información de los clientes para la generación de leads
  - `ScheduleCallback`- Reserva las llamadas del servicio de atención al cliente
- Intenciones integradas: aprovecha las intenciones integradas de Amazon Lex V2 para interacciones comunes:
  - AMAZON.HelpIntent- Proporciona ayuda y orientación
  - AMAZON.CancelIntent- Gestiona las solicitudes de cancelación

  - AMAZON.StopIntent- Termina las conversaciones con elegancia
- Tipos de ranuras personalizadas: tipos de ranuras especializadas para datos específicos de la empresa:
  - `ProductCategories`- Electrónica, ropa, hogar, libros, deportes
  - `ReturnReasons`- Defectuoso, tamaño incorrecto, cambio de opinión, no es como se describe
  - `ContactPreferences`- Notificación integrada en la aplicación por correo electrónico, SMS, llamada telefónica
  - `CustomerTiers`- Bronce, plata, oro, platino
- Tipos de ranuras integradas: utiliza las ranuras integradas de Amazon Lex V2 para los formatos de datos más comunes:
  - AMAZON.Date- Para fechas de entrega y programación de llamadas
  - AMAZON.Time- Para los horarios de devolución de llamadas y los plazos de entrega
  - AMAZON.EmailAddress- Para la recopilación de correos electrónicos de los clientes
  - AMAZON.PhoneNumber- Para los números de teléfono de los clientes
  - AMAZON.Number- Para números de pedido y cantidades

Configuración de intención detallada

CheckOrderStatus Intención

Esta intención gestiona la verificación de los pedidos y las consultas sobre el estado de los mismos, proporcionando a los clientes información sobre los pedidos en tiempo real y, al mismo tiempo, identificando oportunidades de ventas adicionales.

- Ranuras requeridas:
  - `OrderNumber`(AMAZON.Number) - Número de pedido del cliente
  - `CustomerEmail`(AMAZON.EmailAddress) - Correo electrónico de verificación
- Ejemplos de expresiones:
  - «¿Cuál es el estado de mi pedido {OrderNumber}»
  - «Necesito comprobar el número de pedido {OrderNumber}»
  - «¿Dónde está mi paquete»
  - «Rastrear mi pedido {OrderNumber}»
  - «¿Ya se ha enviado mi pedido?»

- Funciones impulsadas por IA:
  - Utiliza la NLU asistida para entender variantes como «mi paquete», «mis cosas» o «mi entrega»
  - Extrae automáticamente los números de pedido a partir de patrones de voz naturales

UpsellProducts Intención

Esta intención identifica de forma proactiva las oportunidades de ingresos mediante la recomendación de productos complementarios y servicios premium en función del historial del cliente y del contexto de interacción actual.

- Ranuras requeridas:
  - `ProductCategory`(ProductCategories) - Categoría de interés
  - `CustomerTier`(CustomerTiers) - Nivel de fidelidad de los clientes
  - `Budget`(AMAZON.Number) - Rango de presupuesto del cliente
- Ejemplos de expresiones:
  - «Muéstrame productos relacionados»
  - «¿Qué más podría venir con esto?»
  - «¿Tienen alguna oferta?»
  - «Estoy interesado en {ProductCategory} artículos»
  - «¿Qué hay en oferta hoy?»
- Características de optimización de ingresos:
  - Precios dinámicos basados en el nivel de cliente
  - Recomendaciones personalizadas mediante el historial de compras
  - Creación de ofertas por tiempo limitado para impulsar las compras inmediatas

CaptureLeadInfo Intención

Esta intención recopila sistemáticamente la información de los clientes para generar clientes potenciales, lo que crea datos valiosos que pueden venderse a los agregadores o utilizarse en campañas de marketing específicas.

- Ranuras requeridas:
  - `CustomerFirstName`(AMAZON.FirstName): nombre del cliente para personalizarlo

  - `CustomerLastName`(AMAZON.LastName) - Apellido del cliente para la búsqueda del historial de ventas
  - `CustomerEmail`(AMAZON.EmailAddress) - Dirección de correo electrónico
  - `CustomerPhone`(AMAZON.PhoneNumber) - Número de teléfono
  - `InterestCategory`(ProductCategories) - Intereses por los productos
  - `ContactPreference`(ContactPreferences) - Método de contacto preferido
- Ejemplos de expresiones:
  - «Quiero recibir información sobre nuevos productos»
  - «Inscríbeme para recibir actualizaciones»
  - «Estoy interesado en {InterestCategory}»
  - «Manténgame informado sobre las ventas»
  - «Quiero unirme a tu lista de correo»
  - «Mi nombre es {CustomerFirstName} {CustomerLastName}»
  - «Soy {CustomerFirstName} {CustomerLastName} y quiero recomendaciones personalizadas»
- Características de generación de leads:
  - Recopilación de datos estructurados para la integración de CRM
  - Puntuación de clientes potenciales basada en la calidad de la interacción
  - Calificación y enrutamiento automatizados de clientes potenciales
  - Formateo de datos para ventas de agregadores
- Integración del historial de ventas:
  - Búsqueda de nombres de clientes en tablas de historial de compras almacenadas en S3
  - Análisis del patrón de compra para obtener recomendaciones personalizadas
  - Identificación del comportamiento de compra entre categorías
  - Análisis de tendencias de compra estacionales para ofertas específicas
  - Cálculo del valor de vida útil del cliente para fijar precios por niveles

Flujo de conversación en varios turnos

El SmartCommerce bot está diseñado para gestionar conversaciones complejas y de varios turnos que abarcan múltiples intenciones y mantienen el contexto durante toda la interacción. Esto crea una experiencia natural similar a la humana y, al mismo tiempo, maximiza las oportunidades de ingresos.

Ejemplo de flujo de conversación

Turnos 1-3: Verificación del estado del pedido

```
Customer: "Hi, I need to check on my order"
Bot: "I'd be happy to help you check your order status. Can you provide your order
 number?"
Customer: "It's order 12345"
Bot: "Thank you. To verify your identity, can you provide the email address associated
 with this order?"
Customer: "john.doe@email.com"
Bot: "Perfect! Order #12345 is currently being prepared for shipment. You should
 receive it by tomorrow.
     Since you're a Gold tier customer, would you like to hear about some exclusive
 deals on related items?"
```

Turnos 4 a 6: oportunidad de aumentar las ventas

```
Customer: "Sure, what do you have?"
Bot: "Based on your Electronics purchase, I can offer you a 20% discount on premium
 accessories.
     What's your budget range for additional items?"
Customer: "Maybe around $50"
Bot: "Excellent! I have a wireless charging pad normally $60, now $48 with your Gold
 discount.
     Plus, if you add it to your current order, shipping is free. Should I add this to
 your cart?"
```

Turno 7-11: generación de leads con búsqueda del historial de ventas

```
Customer: "Not right now, but I'd like to hear about future deals"
Bot: "Absolutely! I can set you up to receive personalized offers. What's your first
 name?"
Customer: "John"
Bot: "Thanks John! And your last name so I can look up your purchase history for better
 recommendations?"
Customer: "Smith"
```

```
Bot: "Perfect! Let me check your purchase history... I see you've bought electronics
 and home items before.
     Based on your past purchases, I can offer you exclusive deals on smart home
 accessories.
     How would you prefer to be contacted?"
Customer: "Email is fine"
Bot: "Great! I'll make sure john.doe@email.com gets personalized deals based on your
 purchase history.
     You're all set for our premium recommendations program!"
```

Funciones de gestión del contexto

- Persistencia de la sesión: mantiene la información del cliente con múltiples intenciones
- Encadenamiento de intenciones: pasa sin problemas entre la verificación de pedidos, el aumento de ventas y la captación de clientes potenciales
- Personalización dinámica: ajusta las ofertas en función del nivel de cliente y del historial de compras
- Seguimiento de los ingresos: supervisa las tasas de éxito de las ventas adicionales y los puntajes de calidad de los clientes potenciales

Funciones impulsadas por IA

Este chatbot aprovecha múltiples capacidades de inteligencia artificial para brindar un servicio al cliente inteligente y personalizado, al tiempo que impulsa el crecimiento empresarial.

Implementación asistida de la NLU

Assisted NLU utiliza modelos lingüísticos extensos para comprender la intención de los clientes, incluso cuando utilizan una redacción no estándar o combinan varias solicitudes en un solo enunciado.

- Comprensión del lenguaje natural:
  - «Mis cosas aún no han llegado» → CheckOrderStatus intención
  - «Quiero devolver esta basura» → ProcessReturn intención
  - «Muéstrame qué más tienes» → UpsellProducts intención
- Reconocimiento de múltiples intenciones:

- «Comprueba mi pedido 12345 e inscríbeme para recibir ofertas» → + CheckOrderStatus CaptureLeadInfo

Resolución generativa de ranuras

Utiliza la IA para extraer los valores de las ranuras a partir de entradas complejas en lenguaje natural sin necesidad de que coincidan exactamente con los datos de entrenamiento.

- Extracción inteligente:
  - «Compré algunos aparatos electrónicos la semana pasada, el número de pedido era algo así como 12345" → OrderNumber: 12345, ProductCategory: Electrónica
  - «Llámeme a mi teléfono móvil al 555-123-4567 o envíeme un correo electrónico a john@company.com" → Teléfono: 555-123-4567, correo electrónico: john@company.com

Integración del análisis de sentimientos

Supervisa la opinión de los clientes a lo largo de la conversación para ajustar el enfoque y ampliarlo cuando sea necesario.

- Enrutamiento basado en las opiniones:
  - Sentimiento positivo → Enfoque agresivo de ventas adicionales
  - Sentimiento neutral → Servicio estándar con ventas adicionales moderadas
  - Sentimiento negativo → Céntrese en la resolución de problemas, con un aumento mínimo de las ventas

Sistema de verificación y confirmación de pedidos

El bot incluye un completo sistema de confirmación que verifica la identidad del cliente, confirma los detalles del pedido y envía las confirmaciones a través de varios canales.

Proceso de verificación multifactorial

- Verificación primaria:
  - Validación del número de pedido
  - Confirmación de dirección de correo
  - Los últimos 4 dígitos del método de pago (para operaciones delicadas)

- Verificación secundaria:
  - Confirmación de dirección de envío
  - Validación de la fecha de compra
  - Verificación de los detalles del producto

Sistema de confirmación multicanal

El bot envía automáticamente las confirmaciones a través de varios canales según las preferencias del cliente y el tipo de acción.

- Confirmaciones por correo electrónico:
  - Actualizaciones del estado del pedido con información de seguimiento
  - Autorización de devolución con etiquetas de envío prepagadas
  - Amplíe las confirmaciones de compra con los detalles de entrega
  - Confirmación de captura de leads con ofertas de bienvenida
- Confirmaciones por SMS:
  - Actualizaciones inmediatas del estado de los pedidos
  - Notificaciones de entrega con intervalos de tiempo
  - Alertas de venta relámpago para los clientes interesados
- Notificaciones integradas en la aplicación:
  - Actualizaciones de pedidos en tiempo real
  - Recomendaciones de productos personalizadas
  - Actualizaciones del programa de fidelización

Integración de puntos de venta (POS)

El chatbot se integra directamente con los sistemas POS para procesar transacciones, aplicar descuentos y actualizar el inventario en tiempo real.

Procesamiento de transacciones en tiempo real

- Flujo de transacciones de ventas adicionales:

```
1. Customer accepts upsell offer
```

```
2. Bot validates inventory availability
3. Bot applies customer-tier discount
4. Bot processes payment using stored payment method
5. Bot updates order with additional items
6. Bot sends confirmation via preferred channel
7. Bot updates customer profile with purchase data
```

- Seguimiento de ingresos:
  - Aumente las tasas de conversión de ventas por nivel de cliente
  - Incremento medio del valor del pedido por interacción
  - Atribución de ingresos a las interacciones con los chatbots
  - Impacto en el valor de vida del cliente

Motor de precios dinámico

- Precios basados en niveles:
  - Bronce: precios estándar
  - Plata: 5% de descuento en ventas adicionales
  - Oro: 10-20% de descuento en ventas adicionales
  - Platinum: 25% de descuento+envío gratis
- Precios contextuales:
  - Escenarios de resolución de problemas: discount adicional para mantener la satisfacción
  - Clientes de alto valor: niveles de precios exclusivos
  - Liquidación de inventario: descuentos agresivos para artículos que se mueven lentamente

Sistema avanzado de generación de leads

El chatbot incluye un sofisticado sistema de generación de leads que captura, califica y monetiza los datos de los clientes a través de múltiples fuentes de ingresos.

Estrategia de captura de leads multitáctil

- Captura oportunista:
  - Durante las comprobaciones del estado del pedido: «¿Quieres recibir actualizaciones sobre productos similares?»

    - Tras la resolución del problema: «¿Podemos informarte sobre las mejoras del producto?»
    - Durante las interacciones de ventas adicionales: «¿Deberíamos avisarte de futuras ofertas en esta categoría?»
- Captura de valor añadido:
    - Acceso exclusivo a los precios para miembros
    - Acceso anticipado a los lanzamientos de nuevos productos
    - Recomendaciones de productos personalizadas
    - Ofertas especiales de cumpleaños y aniversario

Calificación automatizada de clientes potenciales

- Criterios de puntuación:
    - Valor del historial de compras: 0-25 puntos
    - Nivel de compromiso: 0-20 puntos
    - Integridad de la información de contacto: 0-15 puntos
    - Amplitud de intereses por categoría de producto: 0-15 puntos
    - Respuesta a los intentos de venta adicional: de 0 a 25 puntos
- Categorías de clientes potenciales:
    - Clientes potenciales (80-100 puntos): contacto inmediato con el equipo de ventas
    - Warm Leads (60-79 puntos): secuencia de crianza automatizada
    - Cold Leads (40-59 puntos): boletín mensual
    - Prospectos (0-39 puntos): correos electrónicos promocionales trimestrales

Monetización de datos de leads para agregadores

El sistema formatea y empaqueta los datos de clientes potenciales para venderlos a agregadores de terceros, lo que crea una fuente de ingresos adicional.

- Paquetes de datos:
    - Paquete Premium: información de contacto completa, historial de compras, preferencias, puntuaciones de participación
    - Paquete estándar: información de contacto, preferencias básicas, intereses de categoría
    - Paquete básico: dirección de correo electrónico, categoría de interés principal

- Características de cumplimiento:
  - Cumplimiento del GDPR con el seguimiento del consentimiento explícito
  - Cumplimiento de los mecanismos de exclusión de la CCPA
  - Cumplimiento de las normas CAN-SPAM para el marketing por correo electrónico
  - Políticas de retención y eliminación de datos
- Optimización de ingresos:
  - Precios dinámicos basados en los puntajes de calidad de los clientes potenciales
  - Alianzas de datos exclusivas con agregadores premium
  - Licitaciones en tiempo real para clientes potenciales de alto valor

Estrategias de optimización de ingresos

Cada interacción está diseñada para maximizar los ingresos mediante la optimización inteligente de las ventas adicionales, las ventas cruzadas y el valor a lo largo de la vida útil del cliente.

Estrategias inteligentes de venta adicional

- Ventas adicionales contextuales:
  - Verificación del estado del pedido → «¡Su pedido está listo! ¿Añadir el envío rápido por solo 5$?»
  - Solicitud de devolución → «En lugar de devolver, ¿quieres cambiarlo por nuestro modelo actualizado?»
  - Consulta de producto → «Este artículo combina perfectamente con [producto complementario]: combínalo y ahorra un 15%»
- Venta por urgencia:
  - «Esta oferta vence en 24 horas»
  - «Solo quedan 3 en stock a este precio»
  - «La venta flash termina a medianoche»

Oportunidades de venta cruzada

- Venta del ecosistema de productos:
  - Electrónica → Accesorios, garantías, servicios de instalación
  - Ropa → Artículos a juego, productos de cuidado, servicios de estilismo

    - Artículos para el hogar → Artículos complementarios, productos de mantenimiento, consultas de diseño
- Venta adicional de servicios:
    - Garantías ampliadas y planes de protección
    - Niveles de atención al cliente premium
    - Servicios de suscripción y entrega automática
    - Instalación y configuración profesionales

Optimización del valor de por vida del cliente

- Integración del programa de fidelización:
    - Actualizaciones automáticas de niveles basadas en las interacciones del chatbot
    - Puntos extra por participar en ofertas de ventas adicionales
    - Recompensas y descuentos exclusivos solo para chatbots
- Estrategias de retención:
    - Resolución proactiva de problemas para evitar la pérdida de clientes
    - Ofertas personalizadas de recuperación para clientes inactivos
    - Ofertas para celebrar aniversarios e hitos

Procedimientos de implementación

Siga estos step-by-step procedimientos para crear su chatbot avanzado de servicio al cliente con todas las funciones de optimización de ingresos.

Requisitos previos y requisitos de configuración

Antes de crear el SmartCommerce bot, asegúrese de tener la configuración de la cuenta de AWS y los permisos necesarios y de comprender las consideraciones del servicio.

Requisitos de acceso y cuenta de AWS

- Cuenta de AWS: necesita una cuenta de AWS activa con la facturación habilitada. Si no tiene una, regístrese en aws.amazon.com.
- Permisos de IAM: su usuario o rol de AWS debe tener los siguientes permisos:
    - `lex:*`- Permisos completos de Amazon Lex V2 para la creación y administración de bots

  - `iam:CreateRole`- Crear funciones de servicio para el bot
  - `iam:AttachRolePolicy`- Adjuntar políticas a las funciones de servicio
  - `lambda:CreateFunction`- Para integración con Lambda (opcional)
  - `logs:CreateLogGroup`- Para el registro de Amazon CloudWatch Logs
- Selección de región: elija una región de AWS que sea compatible con Amazon Lex V2 y con su público objetivo. Las regiones recomendadas incluyen:
  - Este de EE. UU. (Norte de Virginia): us-east-1
  - Oeste de EE. UU. (Oregón): us-west-2
  - Europa (Irlanda): eu-west-1
  - Asia Pacífico (Sídney): ap-southeast-2

Límites y cuotas de servicio

Amazon Lex V2 tiene varios límites de servicio que pueden afectar al desarrollo y la implementación de sus bots:

- Límites de bots:
  - Máximo de 100 bots por cuenta y región
  - Máximo 100 intentos por bot
  - Máximo 100 tipos de ranuras por bot
  - Máximo 200 espacios por intento
- Límites de tiempo de ejecución:
  - Tiempo máximo de espera de sesión de 15 minutos
  - Máximo 1000 solicitudes por segundo (se puede aumentar)
  - Máximo 1500 caracteres por entrada de texto
- Límites de datos de entrenamiento:
  - Máximo 1500 ejemplos de expresiones por intención
  - Máximo de 10 000 valores de tipo de ranura
  - Máximo 140 caracteres por ejemplo de expresión

> **Note**
>
> Si necesita límites más altos para el uso en producción, puede solicitar aumentos de cuota a través del AWS Support Center.

Consideraciones sobre costos

Entender los precios de Amazon Lex V2 le ayuda a planificar y presupuestar la implementación de su chatbot:

- Precios basados en solicitudes:
  - Solicitudes de texto: 0,00075 USD por solicitud después de las primeras 10 000 solicitudes del mes
  - Solicitudes de voz: 0,004 USD por solicitud después de las primeras 1000 solicitudes del mes
- Nivel gratuito:
  - 10 000 solicitudes de texto al mes durante el primer año
  - 1000 solicitudes de voz al mes durante el primer año
- Costos adicionales:
  - Funciones Lambda (si se utilizan): 0,20$ por 1 millón de solicitudes más tiempo de procesamiento
  - CloudWatch Registros de Amazon Logs: 0,50 USD por GB ingerido
  - Costes de transferencia de datos para integraciones externas
- Consejos de optimización de costos:
  - Utilice los atributos de sesión para reducir las llamadas redundantes a la API
  - Implemente flujos de conversación eficientes para minimizar los turnos
  - Supervise el uso a través CloudWatch de los registros para identificar las oportunidades de optimización

> **Important**
>
> Para obtener información sobre los precios actuales, visita la página de precios de Amazon Lex, ya que las tarifas pueden cambiar.

Descripción general de la integración de Lambda

Si bien este ejercicio se centra en la configuración del bot Lex, las funciones avanzadas descritas (integración de puntos de venta, generación de clientes potenciales, optimización de ingresos) normalmente requerirían funciones Lambda para el procesamiento de back-end.

- Cuando se necesita Lambda:
  - Verificación del estado del pedido con bases de datos externas
  - Controles de inventario en tiempo real para aumentar las ventas
  - Almacenamiento y recuperación de datos de clientes
  - Búsqueda del historial de ventas en tablas de S3 utilizando el nombre y apellidos del cliente
  - Análisis del patrón de compra para obtener recomendaciones personalizadas
  - Integración del procesamiento de pagos
  - Envío de confirmaciones por correo electrónico y SMS
- Integración del historial de ventas de S3:
  - Consulte tablas de S3 utilizando CustomerLastName ranuras CustomerFirstName y ranuras
  - Analice el historial de compras para identificar los patrones y preferencias de compra
  - Genera recomendaciones de productos personalizadas basadas en compras anteriores
  - Calcule el valor de vida útil del cliente para aplicar estrategias de precios dinámicas
  - Identifique las oportunidades de venta cruzada a partir del análisis de correlación de compras
- Requisitos básicos de configuración de Lambda:
  - Función de ejecución de Lambda con los permisos adecuados
  - Configuración de VPC si se accede a recursos privados
  - Variables de entorno para la configuración
  - Gestión de errores e implementación del registro
- Puntos de integración:
  - Cumplimiento de intenciones: procesa las intenciones completadas
  - Validación de ranuras: valida las entradas del usuario en tiempo real
  - Gestión del diálogo: controle el flujo de la conversación

> **Note**
>
> Para este ejercicio, nos centraremos en la configuración del bot Lex. La integración Lambda se puede añadir más adelante a medida que evolucionen sus requisitos. El bot funcionará con respuestas estáticas con fines de prueba y demostración.

Lista de verificación previa a la implementación

Antes de continuar con la creación del bot, compruebe que tiene:

- ✓ Cuenta AWS activa con facturación habilitada
- ✓ Permisos de IAM adecuados para Lex y servicios relacionados
- ✓ Región de AWS de destino seleccionada
- ✓ Se revisaron los límites y las cuotas del servicio
- ✓ Comprendió las implicaciones de los costos
- ✓ Enfoque de integración Lambda planificado (si es necesario)
- ✓ Acceso a la consola de administración de AWS (AWS Management Console)

Cree el SmartCommerce bot

Para crear el bot de servicio SmartCommerce al cliente

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione Crear bot.
3. Para el Método de creación, seleccione Crear un bot en blanco.
4. En la sección de configuración del bot:
   - Nombre del bot: **`SmartCommerceBot`**
   - Descripción: **`Advanced customer service chatbot with upselling, lead generation, and revenue optimization`**
5. En la sección Permisos, seleccione Crear un nuevo rol con permisos básicos de Amazon Lex.
6. En la sección Ley de Protección de la Privacidad en Línea para Niños (COPPA), seleccione No.

7. En la sección Tiempo de espera de la sesión, **`15 minutes`** configúrelo para permitir conversaciones complejas de varios turnos.
8. Elija Siguiente.
9. Agrega el inglés (EE. UU.) como idioma y selecciona Listo.

Cree tipos de ranuras personalizados

Para crear los tipos de ranuras personalizados

1. En el panel de navegación izquierdo, seleccione Tipos de ranuras.
2. Seleccione Añadir tipo de ranura y cree los siguientes tipos de ranura:
    a. ProductCategories:
        - Electrónica
        - Ropa
        - Inicio
        - Libros
        - Sports
        - Belleza
        - Automotriz
    b. ReturnReasons:
        - Defectuoso
        - Tamaño incorrecto
        - ¿Cambió de opinión
        - No como se describe
        - Dañado durante el envío
        - Se encontró un mejor precio
    c. ContactPreferences:
        - Correo electrónico
        - SMS
        - Llamada telefónica

- Notificación en la aplicación
- Correo electrónico

d. CustomerTiers:

- Bronce
- Plata
- Oro
- Platino

Crea intenciones personalizadas

Para crear la intención CheckOrderStatus

1. En el panel de navegación izquierdo, elija Intenciones.
2. Elija Añadir intención y asígnele un nombre. **`CheckOrderStatus`**
3. Añada los siguientes ejemplos de expresiones:
   - «¿Cuál es el estado de mi pedido {OrderNumber}»
   - «Necesito comprobar el número de pedido {OrderNumber}»
   - «¿Dónde está mi paquete»
   - «Rastrear mi pedido {OrderNumber}»
   - «¿Ya se ha enviado mi pedido?»
   - «¿Cuándo llegará mi pedido?»
   - «Quiero información sobre mi envío»
4. Añada las siguientes ranuras:
   - `OrderNumber`(AMAZON.Number) - Necesario

     Mensaje: «¿Puede proporcionarnos su número de pedido?»
   - `CustomerEmail`(AMAZON.EmailAddress) - Necesario

     Mensaje: «Para verificar tu identidad, proporciona la dirección de correo electrónico asociada a este pedido».
5. Establece el mensaje de confirmación: «Permíteme comprobar el número de pedido {OrderNumber} de {CustomerEmail}. ¿Es correcto?».

6. Configura el mensaje de envío: «¡He encontrado tu pedido! El número de pedido {OrderNumber} es {OrderStatus}. {UpsellMessage}»

Para crear la UpsellProducts intención

1. Cree una nueva intención con el nombre**`UpsellProducts`**.
2. Agregue ejemplos de expresiones:
   - «Muéstrame productos relacionados»
   - «¿Qué más podría venir con esto?»
   - «¿Tienen alguna oferta?»
   - «Estoy interesado en {ProductCategory} artículos»
   - «¿Qué hay en oferta hoy?»
   - «Muéstrame más opciones»
3. Añadir ranuras:
   - `ProductCategory`(ProductCategories) - Necesario

     Mensaje: «¿Qué tipo de productos le interesan?»
   - `Budget`(AMAZON.Number) - Opcional

     Mensaje: «¿Cuál es su rango de presupuesto para artículos adicionales?»

Para crear la CaptureLeadInfo intención

1. Cree una nueva intención con el nombre**`CaptureLeadInfo`**.
2. Agregue ejemplos de expresiones:
   - «Quiero recibir información sobre nuevos productos»
   - «Inscríbeme para recibir actualizaciones»
   - «Estoy interesado en {InterestCategory}»
   - «Manténgame informado sobre las ventas»
   - «Quiero unirme a tu lista de correo»
   - «Envíame ofertas y promociones»
   - «Mi nombre es {CustomerFirstName} {CustomerLastName}»

- «Soy {CustomerFirstName} {CustomerLastName} y quiero recomendaciones personalizadas»

3. Añadir ranuras:

   - `CustomerFirstName`(AMAZON.FirstName) - Necesario

     Mensaje: «¿Cuál es su nombre de pila para un servicio personalizado?»
   - `CustomerLastName`(AMAZON.LastName) - Necesario

     Mensaje: «¿Y tu apellido para poder consultar tu historial de compras?»
   - `CustomerEmail`(AMAZON.EmailAddress) - Necesario
   - `CustomerPhone`(AMAZON.PhoneNumber) - Opcional
   - `InterestCategory`(ProductCategories) - Necesario
   - `ContactPreference`(ContactPreferences) - Necesario

Habilite las funciones impulsadas por IA

Para habilitar la NLU asistida y otras funciones de IA

1. En el panel de navegación izquierdo, selecciona Configuración del bot.
2. En NLU asistida, selecciona Activar.
3. En IA generativa, habilita:

   - Resolución de slots asistida
   - Creación descriptiva de bots
4. En Análisis de sentimientos, selecciona Activar.
5. Seleccione Save.

Pruebe e implemente el bot

Para probar el SmartCommerce bot

1. Elige Construir para compilar tu bot.
2. Una vez completada la compilación, elige Probar.
3. Pruebe los siguientes flujos de conversación:

   a. Flujo de estado del pedido:

- Usuario: «Comprueba mi pedido 12345"
- Bot: Solicita la verificación por correo electrónico
- Usuario: proporciona el correo electrónico
- Bot: proporciona el estado y ofrece ventas adicionales

b. Flujo de ventas adicionales:

- Usuario: «Muéstrame ofertas»
- Bot: Solicita la categoría del producto
- Usuario: «Electrónica»
- Bot: presenta ofertas personalizadas

c. Flujo de generación de leads:

- Usuario: «Inscríbeme para recibir actualizaciones»
- Bot: recopila información de contacto
- Usuario: proporciona detalles
- Bot: confirma la suscripción y ofrece un descuento inmediato

4. Comprueba que las funciones de IA funcionan:

- Prueba variantes del lenguaje natural, como «mis cosas no han llegado»
- Verifica que el análisis de opiniones ajuste las respuestas de los bots
- Confirme que la resolución de la ranura funcione con entradas complejas

5. Una vez finalizadas las pruebas, elige Publicar para implementar tu bot.

Análisis y optimización del rendimiento

Supervisa y optimiza el rendimiento de tu SmartCommerce bot mediante un análisis exhaustivo y un seguimiento de los ingresos.

Métricas clave de rendimiento

- Métricas de ingresos:
  - Tasa de conversión de ventas adicionales por nivel de cliente
  - Incremento medio del valor del pedido por interacción
  - Ingresos por conversación

    - Impacto en el valor de vida del cliente
- Métricas de generación de leads:
    - Tasa de captación de leads por tipo de interacción
    - Puntuaciones de calidad de los leads y tasas de conversión
    - Ingresos por monetización de datos procedentes de las ventas de agregadores
    - Tasa de crecimiento de las suscripciones de correo electrónico
- Métricas operativas:
    - Precisión del reconocimiento de intenciones
    - Tasa de éxito en el llenado de ranuras
    - Tasa de finalización de conversaciones
    - Puntuaciones de satisfacción del cliente

Estrategias de optimización continua

- Pruebas A/B:
    - Pruebe diferentes enfoques de mensajería de ventas adicionales
    - Compare las técnicas de captura de clientes potenciales agresivas con las suaves
    - Optimice los porcentajes de descuento para obtener la máxima conversión
- Optimización de Machine Learning:
    - Analice los patrones de conversación exitosos
    - Identifique el momento óptimo para las ofertas de ventas adicionales
    - Perfeccione los algoritmos de clasificación por niveles de clientes

Conclusión del ejercicio

¡Enhorabuena! Ha creado con éxito un chatbot avanzado de servicio al cliente que demuestra las capacidades de nivel empresarial para la optimización de los ingresos y la fidelización de los clientes. Este SmartCommerce bot muestra cómo las empresas pueden aprovechar las funciones impulsadas por IA de Amazon Lex V2 para:

- Maximice los ingresos: mediante estrategias inteligentes de venta adicional, venta cruzada y precios dinámicos

- Genere clientes potenciales: recopilando y calificando sistemáticamente la información de los clientes para obtener múltiples fuentes de ingresos
- Mejore la experiencia del cliente: utilice la IA para comprender el lenguaje natural y ofrecer interacciones personalizadas
- Optimice las operaciones: mediante un servicio de atención al cliente automatizado que escale de manera eficiente y mantenga la calidad

Las técnicas demostradas en este ejercicio se pueden adaptar y ampliar para diversas industrias y casos de uso, lo que proporciona una base para crear soluciones sofisticadas de inteligencia artificial conversacional que impulsen el crecimiento empresarial.

Siguientes pasos

Para mejorar aún más tu SmartCommerce bot, considera la posibilidad de implementar:

- Integraciones avanzadas:
  - Integración del sistema CRM para perfiles completos de clientes
  - Conexiones al sistema de gestión de inventario para una disponibilidad en tiempo real
  - Integración del procesamiento de pagos para transacciones fluidas
- Despliegue multicanal:
  - Integración del widget de chat del sitio web
  - Conexiones a plataformas de redes sociales
  - Implementación de la interfaz de voz
- Análisis avanzado:
  - Desarrollo de paneles personalizados para métricas empresariales
  - Análisis predictivo del comportamiento de los clientes
  - Modelado de atribución y seguimiento del ROI

Mejores prácticas para empezar

Principios del diseño de conversaciones

Seguir estos principios de diseño de conversaciones desde el principio le ayudará a crear chatbots Amazon Lex V2 más eficaces, fáciles de mantener y fáciles de usar que ofrezcan interacciones naturales.

Principios básicos de diseño

- Comience con los objetivos de los usuarios: diseñe su bot en función de lo que los usuarios quieren lograr, no de lo que su sistema puede hacer. Céntrese en el recorrido del usuario y en los resultados deseados.
- Utilice un lenguaje natural: escriba las indicaciones y las respuestas de forma conversacional. Evita la jerga técnica y habla como lo haría un humano servicial.
- Ofrezca opciones claras: cuando los usuarios se queden atascados, ofrezca ejemplos específicos de lo que pueden decir en lugar de un texto de ayuda genérico.
- Hágalo simple: comience con la funcionalidad básica y añada complejidad gradualmente. Los usuarios deberían poder completar las tareas comunes rápidamente.
- Maneje los errores con elegancia: cuando el bot no entienda, proporcione una guía útil en lugar de simplemente decir «no entiendo».
- Confirma las acciones importantes: confirma siempre antes de realizar acciones que no se puedan deshacer fácilmente, como realizar pedidos o eliminar información.
- Proporcione rutas de escape: ofrezca siempre a los usuarios una forma de volver a empezar, obtener ayuda o ponerse en contacto con una persona cuando la necesiten.

Mejores prácticas para el flujo de conversaciones

- Establezca expectativas claras: deje que los usuarios sepan lo que el bot puede y no puede hacer al principio de la conversación.
- Utilice la divulgación progresiva: pida información pieza por pieza en lugar de abrumar a los usuarios con múltiples preguntas.
- Proporcione contexto: recuerde a los usuarios qué información ya ha recopilado y qué es lo que aún necesita.
- Facilite las correcciones: permita a los usuarios corregir la información sin tener que empezar de cero por completo.

Casos de uso y ejemplos del mundo real

Estos ejemplos prácticos muestran cómo aplicar los principios del diseño de conversaciones a los escenarios habituales a los que se enfrentan los nuevos usuarios de Amazon Lex V2.

Caso de uso 1: reserva de citas

Escenario: Un consultorio médico quiere automatizar la programación de citas.

Desafío: los usuarios deben proporcionar varios datos (tipo de servicio, fecha, hora, información de contacto) y es posible que deseen cambiar los detalles.

Enfoque de solución:

- Comience de manera amplia: «¿Qué tipo de cita le gustaría programar?» (odontológico, examen ocular, consulta)
- Reduzca: «¿Cuándo preferiría su cita con el dentista?» (Acepte entradas flexibles como «la semana que viene» o «viernes por la tarde»)
- Cambios de confirmación y oferta: «Tengo programada una limpieza dental para el viernes 15 de marzo a las 14:00 horas. ¿Esto funciona para ti?»
- Gestionar los cambios: si el usuario dice: «¿Podemos hacerlo a las 3 de la tarde en su lugar?» , actualiza la hora sin reiniciar todo el proceso.

Técnicas clave:

- Utilice AMAZON.Date y AMAZON.Time para una date/time entrada flexible
- Cree tipos de espacios personalizados para los tipos de citas
- Utilice las instrucciones de confirmación antes de finalizar las reservas

Caso de uso 2: Consulta del estado del pedido

Escenario: una empresa de comercio electrónico quiere que los clientes comprueben el estado del pedido sin llamar al servicio de asistencia.

Desafío: es posible que los usuarios no tengan a mano su número de pedido o que lo pregunten de distintas formas.

Enfoque de solución:

- Múltiples puntos de entrada: acepte la pregunta «¿Dónde está mi pedido?» , «Rastrear mi paquete» o «Estado del pedido»
- Identificación flexible: «Puedo ayudarlo a rastrear su pedido. ¿Tiene su número de pedido o prefiere usar su dirección de correo electrónico?»

- Guía útil: «El número de pedido suele estar en el correo electrónico de confirmación y comienza por 'ORD-'»
- Resultados claros: «Tu pedido #ORD -12345 se envió ayer y llegará mañana a las 20:00 horas. ¿Desea obtener detalles de seguimiento?»

Técnicas clave:

- Utiliza tipos de ranuras integradas como AMAZON. AlphaNumeric para números de pedido
- Proporcione varias formas de identificar los pedidos (correo electrónico, teléfono, número de pedido)
- Proporcione información clara y procesable en las respuestas

Caso de uso 3: Preguntas frecuentes y soporte

Escenario: una empresa de software quiere gestionar automáticamente las preguntas de soporte más habituales.

Desafío: los usuarios hacen las mismas preguntas de muchas maneras diferentes, y algunos problemas requieren la ayuda humana.

Enfoque de solución:

- Reconocimiento amplio de intenciones: capacite las intenciones para que reconozcan variaciones como «No puedo iniciar sesión», «Problemas con el inicio de sesión» o «La contraseña no funciona»
- Solución de problemas guiada: «Probemos algunos pasos rápidos. En primer lugar, ¿has intentado restablecer tu contraseña?»
- Ruta de escalación: «Si estos pasos no funcionan, puedo ponerte en contacto con nuestro equipo de soporte. ¿Quieres que lo haga?»
- Seguimiento: «¿Solucionó esto tu problema? Siempre puedes decir «ayuda» si necesitas ayuda con alguna otra cosa».

Técnicas clave:

- Utilice muchos ejemplos de expresiones para captar las diferentes formas en que los usuarios expresan sus problemas

- Ofrezca step-by-step orientación en lugar de largas explicaciones
- Ofrezca siempre un camino hacia el apoyo humano cuando la ayuda automatizada no sea suficiente

Manejo de escenarios complejos

Incluso los bots más simples se enfrentan a situaciones complejas. A continuación, te explicamos cómo manejarlas con elegancia:

- Cuando los usuarios cambian de opinión:
  - Usuario: «En realidad, quiero rosas en vez de tulipanes»
  - Bot: «¡No hay problema! He actualizado tu pedido a rosas. Tus rosas estarán listas para recogerlas el viernes a las 2 p.m. ¿Es correcto?»
- Cuando la información no está clara:
  - Usuario: «Necesito flores para la semana que viene»
  - Bot: «¡Estaré encantado de ayudarte! ¿Qué día de la semana que viene es mejor para ti? Puedes decir algo como «lunes» o «el próximo martes».
- Cuando el bot no entiende:
  - Usuario: «Quiero los morados con tallos largos»
  - Bot: «Quiero asegurarme de que lo he hecho bien. ¿Buscas flores moradas? Tenemos rosas moradas, tulipanes morados e lirios morados disponibles».
- Cuando los usuarios necesitan empezar de nuevo:
  - Usuario: «Cancela esto, quiero empezar de nuevo»
  - Bot: «¡Por supuesto! He aprobado tu pedido actual. ¿Qué tipo de flores quieres pedir hoy?»

Ejemplos de bots

En esta sección se proporcionan ejemplos de bots de Amazon Lex que puede utilizar como punto de partida para sus propias aplicaciones.

Temas

- Asistente de agente de call center

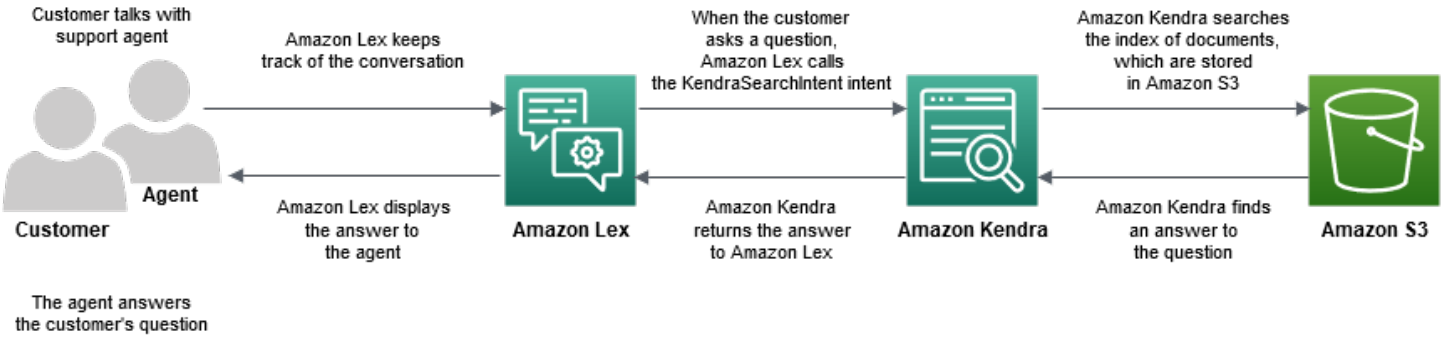
Asistente de agente de call center

En este tutorial, utilizará Amazon Lex V2 con Amazon Kendra para crear un bot de asistencia a los agentes que ayude a los agentes de atención al cliente y lo publique como una 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.

Los bots Amazon Lex V2 se utilizan ampliamente 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, creamos un bot de Amazon Lex V2 que los agentes utilizan 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$ por hora para un índice de Amazon Kendra y 0,75$ para 1000 solicitudes de Amazon Lex V2. El índice de Amazon Kendra se seguirá ejecutando una vez haya finalizado con este ejercicio. Asegúrese de eliminarlo para evitar costos 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
- Paso 2: Crear un bot Amazon Lex V2
- Paso 3: Añada una personalización y una Built-in intención
- Paso 4: configuración de Amazon Cognito
- Paso 5: implementación del bot como aplicación web
- Paso 6: uso del bot

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. Un índice proporciona una API de búsqueda para las consultas de los clientes. El índice se crea a partir de los 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 FAQ.csv archivo ML_ en GitHub.

Siguiente paso

Paso 2: Crear un bot Amazon Lex V2

Paso 2: Crear un bot Amazon Lex V2

Amazon Lex V2 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 V2 que muestra en el bot. Solo el agente puede ver esta respuesta.

Creación de un bot de asistencia para agentes

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el panel de navegación, seleccione Bots.
3. Seleccione 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 Crear. Tras crear el bot, Amazon Lex V2 muestra la pestaña del editor de bots.

Siguiente paso

Paso 3: Añada una personalización y una Built-in intención

Paso 3: Añada una personalización y una Built-in intención

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 V2 tiene dos tipos de intenciones: intenciones personalizadas e intenciones integradas. `AMAZON.KendraSearchIntent`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. Seleccione Guardar intención.

Para añadir el mensaje de **`AMAZON.KendraSearchIntent`**intención y 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. Seleccione Guardar intención.
9. Para crear el bot, elija Crear.

Siguiente paso

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 credenciales de AWS que permiten a los usuarios acceder a otros servicios de AWS. Para este tutorial, proporciona acceso a Amazon Lex V2.

Al crear un grupo de identidades, Amazon Cognito le proporciona las funciones de AWS Identity and Access Management (IAM) para los usuarios autenticados y no autenticados. Para modificar las funciones de IAM, añada políticas que concedan acceso a Amazon Lex V2.

Configuración de Amazon Cognito

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Cognito en. https://console.aws.amazon.com/cognito/
2. Seleccione Administrar grupos de identidades.
3. Elija Crear nuevo grupo de identidades.
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 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 Permitir.
9. En la página Introducción a Amazon Cognito, en Plataforma, elija. JavaScript
10. En la sección Obtenga credenciales de AWS, busque y registre el ID del grupo de identidades.
11. Para permitir el acceso a Amazon Lex V2, modifique las funciones de IAM autenticadas y no autenticadas.
    a. Inicie sesión en la consola de administración de AWS 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 Asociar políticas.
        ii. En el cuadro de búsqueda, escriba **`AmazonLexRunBotsOnly`** y marque la casilla situada al lado del rol.
        iii. Elija 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 Asociar políticas.

ii. En el cuadro de búsqueda, escriba **`AmazonLexRunBotsOnly`** y marque la casilla situada al lado del rol.

iii. Elija Asociar política.

Siguiente paso

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 en https://github.com/awsdocs/amazon-lex-developer-guide/blob/master/example_apps/agent_assistance_bot/su ordenador.
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.

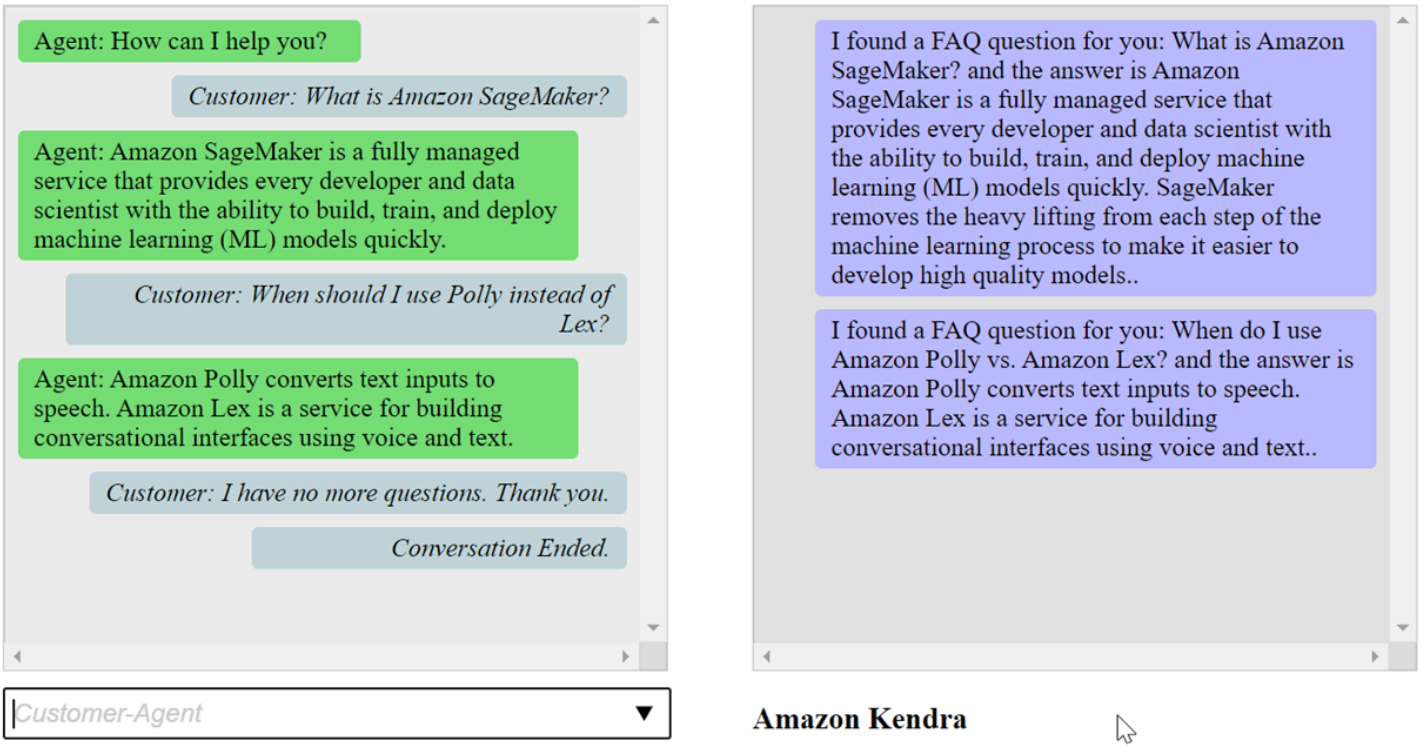
   b. En la sección `Amazon Lex V2 runtime parameters`, introduzca el nombre del bot.

   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.

Ejecute su aplicación web abriéndola `index.html` para entablar una conversación con su bot como esta:.



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 información como cliente, la API de tiempo de ejecución de Amazon Lex V2 la envía a Amazon Lex V2.

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.

Uso de bots Amazon Lex V2

Cree un bot de Amazon Lex V2 para interactuar con sus usuarios y obtener información para realizar una tarea. Por ejemplo, puede crear un bot que recopile la información necesaria para pedir un ramo de flores o reservar una habitación de hotel.

Para compilar un bot, necesitará la siguiente información:

1. El idioma que utiliza el bot para interactuar con el cliente. Puede elegir uno o más idiomas; cada idioma contiene intenciones, slots y tipos de slots independientes.
2. Las intenciones o los objetivos que el bot ayuda a cumplir al usuario. Un bot puede contener una o más intenciones, como pedir flores o reservar un hotel o un coche de alquiler. Debe decidir qué afirmaciones o enunciados hace el usuario para iniciar la intención.
3. La información o los slots que necesita recopilar del usuario para cumplir una intención. Por ejemplo, es posible que necesite obtener el tipo de flores del usuario o la fecha de inicio de una reserva de hotel. Debe definir una o más de las solicitudes que usa Amazon Lex V2 para obtener el valor de slot del usuario.
4. El tipo de slots que necesita del usuario. Puede que tenga que crear un tipo de slot personalizado, como una lista de flores que un usuario pueda pedir, o puede usar un tipo de slot integrado, como el tipo de slot `AMAZON.Date` para la fecha de inicio de una reserva.
5. La interacción del usuario fluye dentro de las intenciones y entre ellas. Puede configurar el flujo de conversación para definir la interacción entre el usuario y el bot una vez que se invoque la intención. Puede crear una función de Lambda para validar y cumplir la intención.

Temas

- Cambios en los flujos de conversación en Amazon Lex V2
- Distintas formas de crear un bot con Amazon Lex V2
- Cómo agregar un nuevo idioma a un bot Amazon Lex V2
- Adición de intenciones
- Adición de tipos de ranura
- Probar un bot con la consola

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Cambios en los flujos de conversación en Amazon Lex V2

El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación.

Antes del cambio, Amazon Lex V2 gestionaba la conversación seleccionando máquinas tragamonedas en función de sus prioridades e intenciones. Puede modificar este comportamiento de forma dinámica y cambiar la ruta de conversación en función de las entradas del usuario mediante DialogAction en la función de Lambda. Esto se puede hacer realizando un seguimiento del estado actual de la conversación y decidiendo mediante programación qué hacer a continuación en función del estado de la sesión.

Con este cambio, puede crear rutas de conversación y ramas condicionales mediante la consola o las API de Amazon Lex V2 sin utilizar una función de Lambda. Amazon Lex V2 rastrea el estado de la conversación y controla qué hacer a continuación en función de las condiciones definidas al crear el bot. Esto le permite crear conversaciones complejas fácilmente mientras diseña su bot.

Estos cambios le dan un control total sobre la conversación con su cliente. Sin embargo, no es necesario que defina una ruta. Si no especifica una ruta de conversación, Amazon Lex V2 crea una ruta predeterminada en función de la prioridad de los slots que desee. Puede seguir utilizando las funciones de Lambda para definir las rutas de conversación de forma dinámica. En tal escenario, la conversación se reanuda en función del estado de la sesión configurado en la función de Lambda.

Esta actualización proporciona lo siguiente:

- Una nueva experiencia de consola para la creación de bots con flujos de conversación complejos.
- Actualizaciones de las API existentes para crear bots que admitan los nuevos flujos de conversación.

- Una respuesta inicial para enviar un mensaje cuando se invoca la intención.
- Nuevas respuestas para la obtención de slots, la invocación de Lambda como enlace de código de diálogo y la confirmación.
- Posibilidad de especificar los próximos pasos en cada turno de la conversación.
- Evaluación de las condiciones para diseñar múltiples vías de conversación.
- Configuración de los valores de los slots y los atributos de sesión en cualquier momento de la conversación.

Tenga en cuenta lo siguiente para los bots más antiguos:

- Los bots creados antes del 17 de agosto de 2022 siguen utilizando el mecanismo anterior para gestionar los flujos de conversación. Los bots creados después de esa fecha utilizan la nueva forma de gestionar el flujo de conversación.
- Los nuevos bots creados mediante importaciones después del 17 de agosto de 2022 utilizan la nueva gestión del flujo de conversaciones. Las importaciones de los bots existentes siguen utilizando la antigua forma de gestionar las conversaciones.
- Para habilitar la nueva gestión del flujo de conversación para un bot creado antes del 17 de agosto de 2022, exporte el bot y, a continuación, impórtelo con un nombre de bot nuevo. El bot recién creado a partir de la importación utiliza la nueva gestión del flujo de conversaciones.

Tenga en cuenta lo siguiente para los bots nuevos creados después del 17 de agosto de 2022:

- Amazon Lex V2 sigue el flujo de conversación definido exactamente como se diseñó para ofrecer la experiencia deseada. Debe configurar todas las ramas del flujo para evitar las rutas de conversación predeterminadas durante el tiempo de ejecución.
- Los pasos de conversación que siguen a un enlace de código deben estar completamente configurados, ya que los pasos incompletos pueden provocar el fallo del bot. Le recomendamos que valide los bots creados antes del 17 de agosto de 2022, ya que estos bots no validan automáticamente los pasos de la conversación tras un enlace de código.

Distintas formas de crear un bot con Amazon Lex V2

Puede crear un bot con Amazon Lex V2 de una de las siguientes formas:

1. Utilice la consola de Amazon Lex V2 para crear un bot mediante la interfaz de un sitio web. Para obtener más información, consulte Creación de un bot mediante la consola de Amazon Lex V2.
2. Utilice el generador de bots descriptivo para crear un bot con las capacidades de IA generativa de Amazon Bedrock. Para obtener más información, consulte Uso de una descripción para crear un bot en Lex V2 con el generador de bots descriptivo.
3. Utilice plantillas de bots para crear un bot preconfigurado que se adapte a los casos de uso empresariales habituales. Para obtener más información, consulte Creación de bots Amazon Lex V2 con plantillas.
4. Utilice un SDK de AWS para crear un bot mediante operaciones de API.
5. Utilice el Diseñador de chatbots automatizados para crear un bot a partir de las transcripciones de chat existentes entre agentes y clientes. Para obtener más información, consulte Creación de bots Amazon Lex V2 con el Diseñador de chatbots automatizados.
6. Importe una definición de bot existente. Para obtener más información, consulte Importación de bots en Lex V2.
7. Se utiliza CloudFormation para crear un bot. Para obtener más información, consulte Creación de recursos de Amazon Lex V2 con AWS CloudFormation.

Temas

- Creación de un bot mediante la consola de Amazon Lex V2
- Creación de bots Amazon Lex V2 con plantillas
- Creación de bots Amazon Lex V2 con el Diseñador de chatbots automatizados

Creación de un bot mediante la consola de Amazon Lex V2

Cree un bot de Amazon Lex V2 para interactuar con sus usuarios y obtener información para realizar una tarea. Por ejemplo, puede crear un bot que recopile la información necesaria para encargar flores o reservar una habitación de hotel. Para crear un bot con la consola de AWS, comience por definir el nombre, la descripción y algunos datos básicos.

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija Crear bot.
3. En la sección Método de creación, seleccione Tradicional y, a continuación, Crear un bot en blanco.

4. En la sección Configuración del bot, asígnele un nombre y una descripción opcional.
5. En la sección de permisos de IAM, elija un rol de AWS Identity and Access Management (IAM) que otorgue permiso a Amazon Lex V2 para acceder a otros servicios de AWS, como Amazon. CloudWatch Puede hacer que Amazon Lex V2 cree el rol o puede elegir un rol existente con CloudWatch permisos.
6. En la sección Ley de Protección de la Privacidad en Línea para Niños (COPPA), seleccione la respuesta adecuada.
7. En la sección Tiempo de espera de la sesión inactivo, seleccione el tiempo que Amazon Lex V2 mantiene abierta una sesión con un usuario. Amazon Lex V2 mantiene las variables de sesión durante la sesión para que el bot pueda reanudar una conversación con las mismas variables.
8. En la sección Configuración avanzada, si quiere, agregue etiquetas que ayuden a identificar el bot, controlar el acceso y monitorear los recursos.
9. Seleccione Siguiente para crear el bot y pasar a añadir un idioma.

Creación de bots Amazon Lex V2 con plantillas

Amazon Lex V2 ofrece soluciones prediseñadas para crear experiencias a escala e impulsar el compromiso digital. Las plantillas de bots prediseñadas automatizan y estandarizan las experiencias de los clientes. Las plantillas de bots proporcionan flujos de conversación listos para usar junto con datos de entrenamiento y mensajes de diálogo, tanto para las modalidades de voz como de chat. Puede agilizar la entrega de soluciones de bots y, al mismo tiempo, optimizar los recursos para centrarse en las relaciones con los clientes.

Puede crear bots prediseñados en función de su caso de uso empresarial. Puede utilizar la CloudFormation consola para seleccionar las opciones prediseñadas para los servicios relacionados, como Amazon S3, Connect Customer y DynamoDB.

Actualmente, Amazon Lex V2 es compatible con los siguientes verticales de negocio:

- Servicios financieros
- Pedidos minoristas
- Seguro de automóvil
- Telecomunicaciones
- Servicios de aerolíneas
- Habrá plantillas adicionales disponibles en el futuro.

Puede compilar un bot con la plantilla de solución empresarial proporcionada y personalizarlo para que se adapte a las necesidades de su empresa.

> **Note**
>
> Las plantillas crean recursos fuera de Amazon Lex V2 mediante pilas de CloudFormation . Es posible que sea necesario modificar la pila en otras consolas, como Lambda y DynamoDB.

Requisitos previos para compilar e implementar la plantilla de bot:

- Una cuenta de AWS
- Acceso a los siguientes servicios de AWS:
  - Amazon Lex V2 para crear bots
  - Lambda para las funciones de inicio de sesión empresarial
  - DynamoDB para crear las tablas
  - Acceso a IAM para crear políticas y funciones
  - AWS CloudFormation para ejecutar la pila
- Credenciales de acceso a IAM y clave secreta
- Instancia Connect Customer (opcional)

> **Note**
>
> El uso de diferentes servicios de AWS conlleva los costes de uso respectivos para cada servicio.

Para compilar un bot a partir de plantillas de Amazon Lex V2:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el panel de navegación izquierdo, seleccione Plantillas de bots.
3. Seleccione qué vertical empresarial desea utilizar para su plantilla de bot. NOTA: Actualmente hay 5 plantillas de bots disponibles. Próximamente habrá más.

4. Seleccione Crear para la plantilla que desea usar. Se abre una pestaña CloudFormation en la que puede editar los parámetros de la CloudFormation pila. Todas las opciones de la plantilla que ha elegido ya están completas. También puede obtener más información sobre cómo funciona la plantilla del bot seleccionando Más información.
5. En la CloudFormation consola, CloudFormation crea una configuración predeterminada para cada uno de los valores de la plantilla que ha elegido. También puede seleccionar su propio nombre de pila, CloudFormation parámetros, tabla de Amazon DynamoDB y (opcional) parámetros de Connect Customer.
6. En la parte inferior de la ventana, seleccione Crear pila.
7. CloudFormation procesa la solicitud en segundo plano durante varios minutos para configurar el nuevo bot. NOTA: El proceso crea automáticamente recursos para una tabla de DynamoDB, un flujo de contactos de Connect Customer y una instancia de Connect Customer. Puede realizar un seguimiento del progreso en la CloudFormation consola y, a continuación, volver a la consola Amazon Lex V2 una vez que se haya completado la creación de la CloudFormation pila.
8. Si se creó correctamente, aparecerá un mensaje y podrá seleccionar Ir a la lista de bots para ir a la página de bots, donde encontrará el nuevo bot que está listo para probarlo y usarlo.

Configurar su plantilla de bot

Funciones de Lambda: la plantilla de bot crea automáticamente las funciones de Lambda necesarias para su implementación. Si varios bots forman parte de la solución de plantilla, en los parámetros de CloudFormation se muestran varias funciones de Lambda. Si tiene funciones de Lambda existentes para implementar con su bot, puede introducir el nombre de la función de Lambda personalizada.

Amazon DynamoDB: la plantilla de bot crea automáticamente la tabla de DynamoDB necesaria para cargar los datos de la política de muestra. También puede ingresar el nombre de su tabla personalizada de DynamoDB. La tabla personalizada de DynamoDB debe tener el mismo formato que la tabla predeterminada creada por la implementación de la plantilla de bot.

Connect Customer: puede configurar su instancia de Connect Customer para que funcione con su nueva plantilla de bot introduciendo el ConnectInstance ARN y un único. ContactFlowName Con el uso de Connect Customer, puedes probar tu bot mediante un sistema IVR de principio a fin.

Solución de problemas con su plantilla de bot

- Compruebe que tiene los permisos adecuados para crear la plantilla que está eligiendo. Los usuarios necesitan un CloudFormation:CreateStack permiso junto con los permisos para los

recursos de AWS que aparecen en la plantilla. En la parte inferior de la página de Crear plantilla hay una lista de los recursos que necesitan permisos de usuario.

- Si no se puede crear la plantilla de bot, el banner rojo de la consola Amazon Lex V2 proporciona un enlace a la CloudFormation pila responsable de crear la plantilla. En la CloudFormation consola, puedes ver la pestaña de eventos para ver el error específico que provocó el error de la plantilla. Una vez que haya revisado el CloudFormation error, consulte Solución de problemas CloudFormation para obtener más información.
- Las plantillas de bots solo funcionan con los datos de muestra. Debe rellenar la tabla de DynamoDB con sus datos para que las plantillas funcionen con los datos personalizados.

Creación de bots Amazon Lex V2 con el Diseñador de chatbots automatizados

El diseñador de chatbots automatizados le ayuda a diseñar bots a partir de transcripciones de conversaciones existentes. Analiza las transcripciones y sugiere un diseño inicial con intenciones y tipos de slot. Puede modificar el diseño del bot, añadir instrucciones, compilarlo, probarlo e implementarlo.

Tras crear un bot nuevo o añadir un idioma al bot mediante la API o la consola de Amazon Lex V2, puede cargar las transcripciones de las conversaciones entre dos partes. El diseñador de chatbots automatizados analiza las transcripciones y determina las intenciones y los tipos de slot del bot. También etiqueta las conversaciones que influyeron en la creación de una intención o un tipo de slot en particular para que las revise.

Utilice la consola de Amazon Lex V2 o la API para analizar las transcripciones de las conversaciones y sugerir intenciones y tipos de slot para un bot.

> **Note**
>
> Solo puede usar transcripciones en inglés (EE. UU.).

Puede revisar las intenciones y los tipos de slot sugeridos una vez que el diseñador del chatbot finalice el análisis. Una vez que haya añadido una intención o un tipo de slot sugerido, puede modificarlo o eliminarlo del diseño del bot mediante la consola o la API.

El diseñador de chatbots automatizado admite archivos de transcripciones de conversaciones mediante el esquema Contact Lens for Connect Customer. Si utiliza una aplicación de centro de

contacto diferente, debe transformar las transcripciones de las conversaciones al formato utilizado por el diseñador del chatbot. Para obtener información, consulte Formato de la transcripción de entrada.

Para utilizar el diseñador de chatbots automatizados, debe permitir el acceso al rol de IAM que ejecuta el diseñador. Para conocer la política de IAM específica, consulte Permitir a los usuarios utilizar el Diseñador de chatbots automatizados. Para permitir que Amazon Lex V2 cifre los datos de salida con una AWS KMS clave opcional, debe actualizar la clave con la política que se muestra enPermita a los usuarios utilizar un AWS KMS clave para cifrar y descifrar archivos.

> **Note**
>
> Si utiliza una KMS key, debe proporcionar una KMS key política, independientemente de la IAM función utilizada.

Temas

- Importar transcripciones de conversaciones
- Crear intenciones y tipos de slot
- Formato de la transcripción de entrada
- Formato de la transcripción de entrada

Importar transcripciones de conversaciones

Importar transcripciones de conversaciones es un proceso de tres pasos:

1. Prepare las transcripciones para su importación convirtiéndolas al formato correcto. Si utiliza Contact Lens for Connect Customer, las transcripciones ya están en el formato correcto.
2. Cargue las transcripciones en un bucket de Amazon S3. Si utiliza Contact Lens, sus transcripciones ya están en un bucket de S3.
3. Analice las transcripciones con la consola de Amazon Lex V2 o las operaciones de la API. El tiempo que se tarda en completar la formación depende del volumen de las transcripciones y de la complejidad de la conversación. Por lo general, se analizan 500 líneas de transcripciones por minuto.

En las siguientes secciones se describen cada uno de estos pasos.

Importación de transcripciones desde Contact Lens for Connect Customer

El diseñador de chatbots automatizados Amazon Lex V2 es compatible con los archivos de transcripciones de Contact Lens. Para usar los archivos de transcripciones de Contact Lens, debe activar Contact Lens y anotar la ubicación de sus archivos de salida.

Exportar las transcripciones de Contact Lens

1. Activa Contact Lens en tu instancia de Connect Customer. Para obtener instrucciones, consulte Habilitar lentes de contacto para Connect Customer en la guía del administrador de Connect Customer.
2. Anote la ubicación del depósito de S3 que Connect Customer utiliza para su instancia. Para ver la ubicación, abra la página de almacenamiento de datos en la consola Connect Customer. Para obtener instrucciones, consulte Actualizar la configuración de la instancia en la guía del administrador de Connect Customer.

Una vez que haya activado Contact Lens y haya anotado la ubicación de los archivos de sus transcripciones, consulte Analice sus transcripciones con la consola de Amazon Lex V2 para ver las instrucciones para importar y analizar las transcripciones.

Preparar las transcripciones

Prepare sus transcripciones creando archivos de transcripciones.

- Cree un archivo de transcripción por conversación en el que figure la interacción entre las partes. Cada interacción de la conversación puede abarcar varias líneas. Puede proporcionar versiones redactadas y no redactadas de la conversación.
- El archivo debe estar en el formato JSON especificado en Formato de la transcripción de entrada.
- Debe proporcionar al menos 1000 turnos de conversación. Para mejorar la detección de sus intenciones y los tipos de slot debe proporcionar alrededor de 10 000 turnos o más de conversación. El diseñador de chatbots automatizados solo procesará los primeros 700 000 turnos.
- No hay límite en la cantidad de archivos de transcripciones que puede cargar, ni existe una restricción de tamaño del archivo.

Si planea filtrar las transcripciones importadas por fecha, los archivos deben tener la siguiente estructura de directorio:

```
<path or bucket root>
```

```
--> yyyy
   --> mm
      --> dd
         --> transcript files
```

El archivo de transcripción debe contener la fecha en el formato «aaaa-mm-dd» en algún lugar del nombre del archivo.

Exportar las transcripciones de otras aplicaciones de centros de contacto

1. Utilice las herramientas de su aplicación del centro de contacto para exportar las conversaciones. La conversación debe contener al menos la información especificada en Formato de la transcripción de entrada.
2. Transforme las transcripciones producidas por su aplicación del centro de contacto al formato descrito en Formato de la transcripción de entrada. Usted es responsable de realizar la transformación.

Proporcionamos tres scripts para preparar las transcripciones. Son los siguientes:

- Un script para combinar las transcripciones de Contact Lens con los registros de conversaciones de Amazon Lex V2. Las transcripciones de Contact Lens no incluyen partes de las conversaciones de Connect Customer que interactúan con los bots de Amazon Lex V2. El script requiere que los registros de conversaciones estén activados para Amazon Lex V2 y los permisos adecuados para consultar los CloudWatch registros de conversaciones y los buckets de Contact Lens S3.
- Un script para transformar los análisis de llamadas con Amazon Transcribe al formato de entrada de Amazon Lex V2.
- Un script para transformar las transcripciones de los chats de Connect Customer al formato de entrada Amazon Lex V2.

Puede descargar los scripts desde este GitHub repositorio: https://github.com/aws-samples/amazon-lex-bot-recommendation-integration.

Cargar las transcripciones en un bucket de S3

Si utiliza Contact Lens, sus archivos transcripciones ya están en un bucket de S3. Para conocer la ubicación y los nombres de los archivos de sus transcripciones, consulte Ejemplos de archivos de salida de Contact Lens en la guía del administrador de Connect Customer.

Si utiliza otra aplicación de centro de contacto y no ha configurado un bucket de S3 para sus archivos de transcripciones, siga este procedimiento. De lo contrario, si ya tiene un bucket de S3, después de iniciar sesión en la consola de Amazon S3, siga este procedimiento empezando por el paso 5.

Cargar los archivos en un bucket de S3

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.
2. Seleccione Crear bucket.
3. Escriba un nombre y seleccione una región para el bucket. La región debe ser la misma que utiliza para Amazon Lex V2. Configure las demás opciones según sus necesidades.
4. Seleccione Crear bucket.
5. En la lista de buckets, seleccione un bucket existente o el bucket que acaba de crear.
6. Seleccione Cargar.
7. Agregue los archivos de transcripciones que desee cargar.
8. Seleccione Cargar.

Analice sus transcripciones con la consola de Amazon Lex V2

Solo puede utilizar un diseño de bots automatizado en un idioma vacío. Puede añadir un nuevo idioma a un bot existente o crear un nuevo bot.

Crear un nuevo idioma en un nuevo bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione Crear bot.
3. Seleccione Empezar con el diseñador de chatbots automatizados. Rellene la información para crear su nuevo bot.
4. Seleccione Siguiente.
5. En Añadir idioma al bot, rellene la información del idioma.
6. En la sección Ubicación del archivo de transcripciones en S3, seleccione el bucket de S3 que contiene sus archivos de transcripciones y, si es necesario, la ruta local a los archivos.
7. Puede elegir lo siguiente:

- Una AWS KMS clave para cifrar los datos de la transcripción durante el procesamiento. Si no selecciona una clave, se utilizará una AWS KMS clave de servicio.
- Filtrar las transcripciones a un intervalo de fechas específico. Si decide filtrar las transcripciones, deben estar en la estructura de carpetas correcta. Para obtener más información, consulte Preparar las transcripciones.

8. Seleccione Listo.

Espere a que Amazon Lex V2 procese la transcripción. Verá un mensaje de finalización cuando se completa el análisis.

¿Cómo dejar de analizar su transcripción?

En caso de que necesite parar el análisis de las transcripciones que ha subido, puede detener un trabajo `BotRecommendation` en ejecución que tenga un estado `BotRecommendationStatus` como procesando. Puede hacer clic en el botón Detener procesamiento que aparece en el banner después de enviar un trabajo desde la consola o mediante CLI SDK para la API de `StopBotRecommendation`. Para obtener más información, consulte StopBotRecommendation

Después de solicitar la `StopBotRecommendation`, el `BotRecommendationStatus` interno se configura como `Stopping` y no se le cobrará nada. Para asegurarse de que el trabajo se ha detenido, puede llamar a la API de `DescribeBotRecommendation` y comprobar que el `BotRecommendationStatus` es `Stopped`. Esto suele tardar entre 3 y 4 minutos.

No se le cobrará por el procesamiento una vez que se llame a la API de `StopBotRecommendation`.

Crear intenciones y tipos de slot

Una vez que el diseñador del chatbot haya creado las intenciones y los tipos de slot, usted selecciona las intenciones y los tipos de slot para añadirlos al bot. Puede revisar los detalles de cada intención y tipo de slot para decidir qué recomendaciones son las más relevantes para su caso.

Puede hacer clic en el nombre de una intención recomendada para ver los ejemplos de enunciados y slots sugeridos por el diseñador del chatbot. Si selecciona Mostrar transcripciones asociadas, también puede desplazarse por las conversaciones que proporcionó. Estas transcripciones influyen en la recomendación del diseñador del chatbot sobre esta intención. Si hace clic en un ejemplo de enunciado, puede revisar la conversación principal y el giro de diálogo correspondiente que influyó en ese enunciado específico.

Puede hacer clic en el nombre de un tipo de slot específico para ver los valores de slot recomendados. Si selecciona Mostrar transcripciones asociadas, puede revisar las conversaciones que influyeron en este tipo de slot, con el indicador del agente que solicita el tipo de slot resaltado. Si hace clic en un valor de tipo de slot específico, puede revisar la conversación principal y el giro de diálogo correspondiente que influyó en ese valor.

Revisar y añadir intenciones y tipos de slot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, seleccione el bot con el que quiere trabajar.
3. Seleccione Ver idiomas.
4. En la lista de idiomas, seleccione el idioma con el que quiere trabajar.
5. En Estructura de conversación, seleccione Revisar.
6. En la lista de intenciones y tipos de slot, seleccione los que quiera añadir al bot. Puede elegir una intención o un tipo de slot para ver los detalles y las transcripciones asociadas.

Las intenciones se ordenan según la confianza que Amazon Lex V2 deposite en que la intención está asociada a las transcripciones procesadas.

Formato de la transcripción de entrada

El siguiente es el formato de archivo de entrada para generar intenciones y tipos de slot para su bot. El archivo de entrada debe contener estos campos. Los demás campos se ignoran.

El formato de entrada es compatible con el formato de salida de Contact Lens for Connect Customer. Si está utilizando Contact Lens, no es necesario modificar sus archivos de transcripciones. Para obtener más información, consulte Ejemplos de archivos de salida de Contact Lens. Si está utilizando otra aplicación de centro de contacto, debe transformar el archivo de transcripción a este formato.

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

```
        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string"
    },
    "Transcript": [
        {
            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
        }
    ]
}
```

Los siguientes campos deben estar presentes en el archivo de entrada:

- Participantes: identifica a los participantes de la conversación y el papel que desempeñan.
- Versión: la versión del formato del archivo de entrada. Siempre «1.1.0».
- ContentMetadataIndica si ha eliminado información confidencial de la transcripción. Defina el campo `Output` como «Sin procesar» si la transcripción contiene información confidencial.
- CustomerMetadataUn identificador único para la conversación.
- Transcripción: el texto de la conversación entre las partes de la conversación. Cada turno de la conversación se identifica con un identificador único.

Formato de la transcripción de entrada

El formato de la transcripción de salida es prácticamente el mismo que el formato de la transcripción de entrada. Sin embargo, también incluye algunos metadatos de los clientes y un campo en el que se enumeran los segmentos que influyeron en la sugerencia de intenciones y tipos de slot. Puede descargar la transcripción de salida desde la página de Revisión de la consola o mediante la API de Amazon Lex V2. Para obtener más información, consulte Formato de la transcripción de entrada.

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
```

```
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {

        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string",
        "FileName": "string",
        "InputFormat": "Lex"
    },
    "InfluencingSegments": [
        {
            "Id": "string",
            "StartTurnIndex": number,
            "EndTurnIndex": number,
            "Intents": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SampleUtteranceIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ],
            "SlotTypes": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SlotValueIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ]
```

```
        }
    ],
    "Transcript": [
        {

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

- CustomerMetadata— Se añaden dos campos al `CustomerMetadata` campo, el nombre del archivo de entrada que contiene la conversación y el formato de entrada, que siempre es «Lex».
- InfluencingSegments— Identifica los segmentos de la conversación que influyeron en la sugerencia de una intención o un tipo de espacio. El identificador de la intención o el tipo de slot identifica a la persona específica influenciada por la conversación.

Cómo agregar un nuevo idioma a un bot Amazon Lex V2

Añada uno o más idiomas y configuraciones regionales a su bot para que pueda comunicarse con los usuarios en sus respectivos idiomas. Defina las intenciones, los slots y los tipos de slots por separado para cada idioma, de modo que los enunciados, las instrucciones y los valores de los slots sean específicos del idioma.

El bot debe contener al menos un idioma.

Añadir un idioma a su bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la sección Bots, seleccione el bot al que quiere agregar un idioma.
3. En la sección Agregar idiomas, haga clic en Ver idiomas.
4. En la sección Todos los idiomas, haga clic en Agregar idioma.
5. En la sección Agregar un nuevo idioma, seleccione Agregar un idioma desde cero.
6. En la sección Detalles del idioma, seleccione el idioma que desee utilizar.

7. Si su bot admite la interacción por voz, en la sección Voz, seleccione la voz de Amazon Polly que Amazon Lex V2 utiliza para comunicarse con el usuario. Si el bot no admite la voz, seleccione Ninguno.
8. En la sección Umbral de puntuación de confianza de la clasificación, defina el valor que Amazon Lex V2 utiliza para determinar si una intención es la correcta. Puede ajustar este valor después de probar el bot.
9. Seleccione Añadir.

Adición de intenciones

Las intenciones son los objetivos que los usuarios quieren conseguir, como pedir flores o reservar un hotel. Su bot debe tener, como mínimo, una intención.

De forma predeterminada, todos los bots contienen una única intención integrada, la intención alternativa. Esta intención se utiliza cuando Amazon Lex V2 no reconoce ninguna otra intención. Por ejemplo, si un usuario dice «Quiero pedir flores» a una intención de reserva de hotel, se activa la intención alternativa.

Agregar una intención

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, seleccione el bot al que quiere añadir la intención y, en Añadir idiomas, seleccione Ver idiomas.
3. Seleccione el idioma al que quiere añadir la intención y, a continuación, seleccione Intenciones.
4. Seleccione Agregar intención, dele un nombre a la intención y, a continuación, seleccione Agregar.
5. En el editor de intenciones, añada los detalles de su intención.

   - Flujo de conversación: use el diagrama de flujo de conversación para ver cómo se vería un diálogo con su bot. Puede elegir diferentes secciones de la conversación para ir a esa sección del editor de intenciones.
   - Detalles de la intención: dele un nombre y una descripción a la intención para ayudar a identificar el propósito de la intención. También puede ver el identificador único que Amazon Lex V2 asignó a la intención.

- Contextos: defina los contextos de entrada y salida de la intención. Un contexto es una variable de estado asociada a una intención. Un contexto de salida se establece cuando se cumple una intención. Una intención con un contexto de entrada solo se puede reconocer si el contexto está activo. Siempre se puede reconocer una intención sin contextos de entrada.
- Ejemplos de enunciados: debe proporcionar 10 o más frases que espere que sus usuarios utilicen para iniciar una intención. Amazon Lex V2 generaliza a partir de estas frases para reconocer que el usuario quiere iniciar la intención.
- Respuesta inicial: el mensaje inicial que se envía al usuario después de invocar la intención. Puede proporcionar respuestas, inicializar valores y definir el siguiente paso que debe dar Amazon Lex V2 para responder al usuario al principio de la intención.
- Slots: defina los slots, o los parámetros, necesarios para cumplir la intención. Cada slot tiene un tipo que define los valores que se pueden introducir en el slot. Puede elegir entre sus tipos de slot personalizados o puede elegir un tipo de slot integrado.
- Confirmación: estas indicaciones y respuestas se utilizan para confirmar o rechazar el cumplimiento de la intención. El mensaje de confirmación pide al usuario que revise los valores del slot. Por ejemplo, «He reservado una habitación de hotel para el viernes. ¿Es correcto?». La respuesta negativa se envía al usuario cuando rechaza la confirmación. Puede proporcionar respuestas, establecer valores y definir el siguiente paso que realizará Amazon Lex V2 correspondiente a una respuesta de confirmación o rechazo del usuario.
- Cumplimiento: respuesta que se envía al usuario durante el transcurso del cumplimiento. Puede configurar las actualizaciones del progreso de cumplimiento al inicio de esta y periódicamente mientras esté en curso. Por ejemplo, «Estoy cambiando la contraseña, esto puede tardar unos minutos» y «Todavía estoy trabajando en tu solicitud». Las actualizaciones de cumplimiento solo se pueden usar para conversaciones en streaming. También puede configurar un mensaje de confirmación, de error y de tiempo de espera agotado posterior al cumplimiento. Puede enviar mensajes posteriores al cumplimiento tanto para conversaciones en streaming como conversaciones habituales. Por ejemplo, si el cumplimiento se realiza correctamente, puede enviar el mensaje «He cambiado tu contraseña». Si el cumplimiento no se realiza correctamente, puede enviar una respuesta con más información, como «No he podido cambiar tu contraseña, ponte en contacto con el servicio de asistencia para obtener ayuda». Si el proceso de cumplimiento supera el tiempo de espera configurado, puede enviar un mensaje informando al usuario, por ejemplo: «Nuestros servidores están muy ocupados en este momento. Intenta volver a realizar la solicitud más tarde». Puede proporcionar respuestas, establecer valores y definir el siguiente paso que debe dar Amazon Lex V2 para responder al usuario.

- Respuestas finales: respuesta que se envía al usuario una vez que se ha cumplido la intención y se han reproducido todos los demás mensajes. Por ejemplo, un agradecimiento por reservar una habitación de hotel. O bien, puede hacer que el usuario inicie una intención diferente, como: «Gracias por reservar una habitación. ¿Desea reservar un coche de alquiler?». Puede proporcionar respuestas y configurar las próximas acciones de seguimiento después de que se cumpla la intención y se responda con la respuesta final.
- Enlaces de códigos: indican si está utilizando una función de AWS Lambda para inicializar la intención y validar las entradas del usuario. La función de Lambda se especifica en el alias que se utiliza para ejecutar el bot.

6. Seleccione Guardar intención para guardar la intención.

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Configurar las indicaciones en un orden específico

Puede configurar el bot para que reproduzca los mensajes en un orden predefinido marcando la casilla Reproducir los mensajes en orden. De lo contrario, el bot reproduce el mensaje y las variaciones en orden aleatorio.

Las indicaciones ordenadas permiten que el mensaje y las variantes de un grupo de mensajes se reproduzcan en orden entre reintentos. Puede utilizar una reformulación alternativa de un mensaje cuando el usuario dé una respuesta no válida a la solicitud o para confirmar la intención. Se pueden configurar hasta dos variantes del mensaje original en cada slot. Puede elegir si quiere reproducir los mensajes en orden o de forma aleatoria.

El mensaje ordenado admite los cuatro tipos de mensajes: texto, respuesta de carga personalizada, SSML y grupo de tarjetas. Las respuestas se ordenan dentro del mismo grupo de mensajes. Los distintos grupos de mensajes son independientes.

Temas

- Enunciados de muestra
- Estructura de intenciones
- Crear rutas de conversación
- Uso del generador visual de conversaciones
- Intenciones integradas

Enunciados de muestra

Cree ejemplos de enunciados que son variaciones de frases que espera que los usuarios utilicen para iniciar una intención. Por ejemplo, para la intención **`BookFlight`**, puede incluir enunciados como los siguientes:

1. Quiero reservar un vuelo
2. ayúdame a conseguir un vuelo.
3. billetes de avión, ¡por favor!
4. vuelo desde {*`DepartureCity`*} a {*`DestinationCity`*}

Debería proporcionar 10 o más enunciados de muestra. Proporcione ejemplos que representen una amplia gama de estructuras de oraciones y palabras que los usuarios puedan pronunciar. Considere también las oraciones incompletas, como las de los ejemplos 3 y 4 anteriores. También puede usar slots que haya definido para la intención en un ejemplo de enunciado poniendo el nombre del slot entre corchetes, como en {*`DepartureCity`*} en el ejemplo 4. Si incluye nombres de slots en un enunciado de muestra, Amazon Lex V2 rellena los slots de la intención con los valores que el usuario proporciona en el enunciado.

Una variedad de ejemplos de enunciados ayuda a Amazon Lex V2 a generalizar para reconocer de manera efectiva que el usuario quiere iniciar la intención.

Puede añadir ejemplos de enunciados en el editor de intenciones, en el generador visual de conversaciones o mediante las operaciones de la API CreateIntent o UpdateIntent. También puede generar enunciados de ejemplo automáticamente aprovechando las capacidades de la IA generativa de Amazon Bedrock. Para obtener más información, consulte Uso de la generación de enunciados para generar ejemplos de enunciados para el reconocimiento de intenciones.

Use el editor de intenciones o el generador visual de conversaciones

1. En el editor de intenciones, navegue hasta la sección Ejemplos de enunciados. En el generador visual de conversaciones, busque la sección Ejemplos de enunciados en el bloque Inicio.
2. En el cuadro con el texto transparente **`I want to book a flight`**, escriba un ejemplo de enunciado. Seleccione Añadir enunciado para añadir el enunciado.
3. Vea los ejemplos de enunciados que ha agregado en el modo Vista previa o Texto sin formato. En Texto sin formato, cada línea es un enunciado independiente. En el modo Vista previa, coloque el cursor sobre un enunciado para ver las siguientes opciones:
   - Seleccione el cuadro de texto para editar el enunciado.
   - Seleccione el botón x situado a la derecha del cuadro de texto para eliminar el enunciado.
   - Arrastre el botón situado a la izquierda del cuadro de texto para cambiar el orden de los ejemplos de enunciados.
4. Use la barra de búsqueda situada en la parte superior para buscar entre los ejemplos de enunciados y el menú desplegable situado junto a ella para ordenarlos por el orden en que los añadió o por orden alfabético.

Usar una operación de la API

1. Cree una nueva intención con la operación CreateIntent o actualice una existente con la operación UpdateIntent.
2. La solicitud de API incluye un campo `sampleUtterances` que se asigna a una matriz de objetos SampleUtterance.
3. Para cada enunciado de ejemplo que desee añadir, añada un objeto `SampleUtterance` a la matriz. Añada el enunciado de muestra como valor del campo `utterance`.
4. Para editar y eliminar ejemplos de enunciados, envíe una solicitud de `UpdateIntent`. La lista de enunciados que proporciona en el campo `sampleUtterances` reemplaza a los enunciados existentes.

> **Important**
>
> Cualquier campo que deje en blanco en la solicitud de `UpdateIntent` provocará que se eliminen las configuraciones existentes en su intención. Utilice la operación

DescribeIntent para devolver la configuración del bot y copie en la solicitud de `UpdateIntent` las configuraciones que no desee que se eliminen.

Estructura de intenciones

Las intenciones son los objetivos que los usuarios quieren conseguir, como pedir flores o reservar un hotel. Su bot debe tener, como mínimo, una intención. Una intención consta de los siguientes componentes

- Respuesta inicial: el mensaje inicial que se envía al usuario después de invocar la intención. Puede proporcionar respuestas, inicializar valores y definir el siguiente paso que debe dar su bot para responder al usuario al principio de la intención.
- Ranuras: los parámetros necesarios para cumplir una intención. Cada slot tiene un tipo que define los valores que se pueden introducir en el slot. Puede elegir entre sus tipos de ranura personalizados o puede elegir un tipo de ranura integrado.
- Confirmación: una vez finalizada la conversación con el usuario y rellenados los valores de las ranuras correspondientes a la intención, puede configurar un mensaje de confirmación para preguntar al usuario si los valores de las ranuras son correctos.
- Cumplimiento: respuesta que se envía al usuario durante el transcurso del cumplimiento. Puede configurar las actualizaciones del progreso de cumplimiento al inicio de esta y periódicamente mientras esté en curso. También puede configurar un mensaje de confirmación, de error y de tiempo de espera agotado posterior al cumplimiento.
- Respuesta de cierre: la respuesta de cierre se envía al usuario una vez que se ha cumplido su intención. Puede usar la respuesta de cierre para finalizar la conversación o para indicarle al usuario que puede continuar con otra intención.

Temas

- Respuesta inicial
- Ranuras
- Confirmación
- Cumplimiento
- Respuesta de cierre

Respuesta inicial

La respuesta inicial se envía al usuario después de que Amazon Lex V2 determine la intención y antes de que comience a obtener valores de slots. Puede usar esta respuesta para informar al usuario de la intención que ha reconocido y prepararlo para que la información que recopile pueda cumplir con esa intención.

Por ejemplo, si la intención es programar una cita de servicio para un automóvil, la respuesta inicial podría ser:

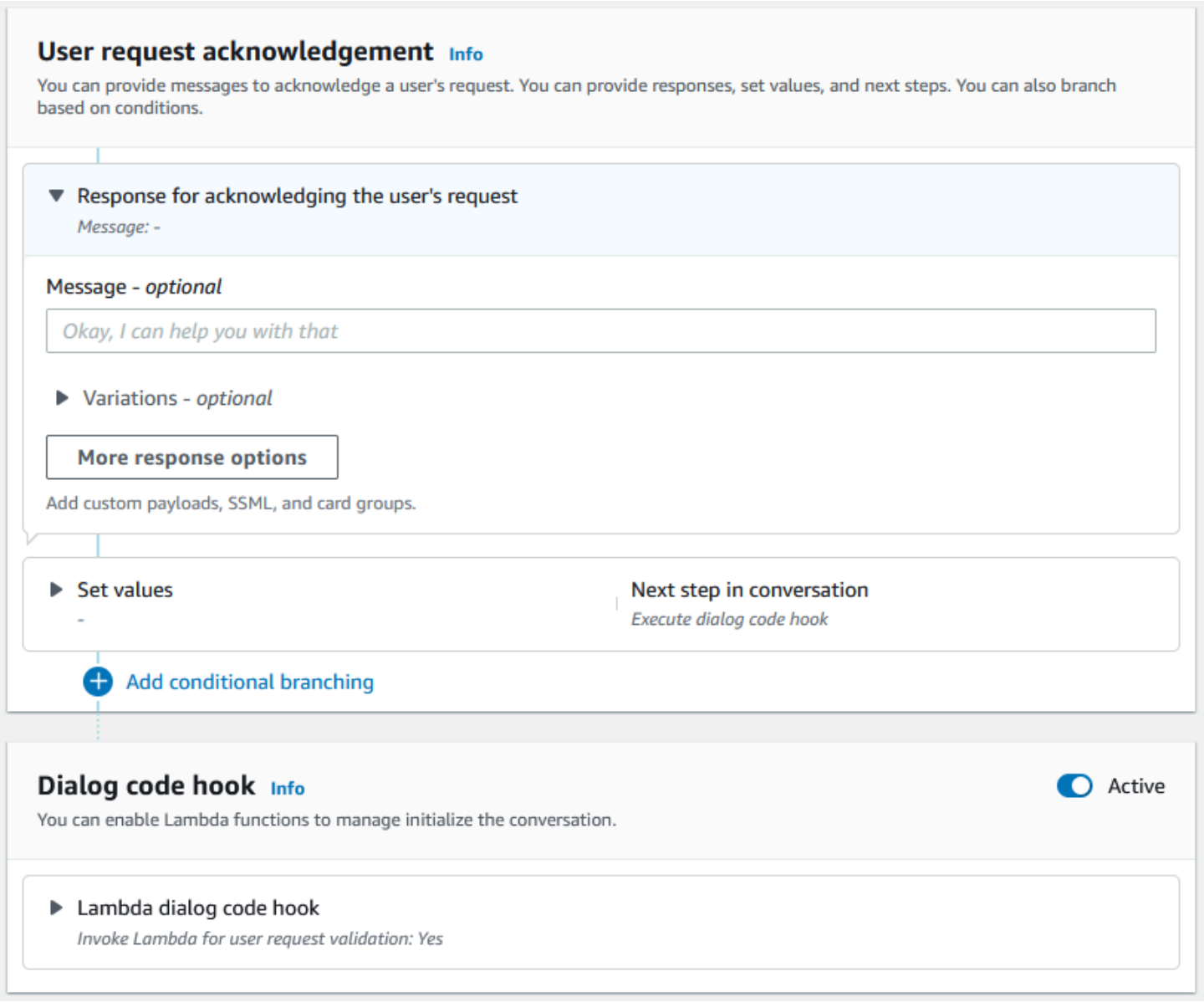
> Puedo ayudarte a programar una cita. Deberá proporcionar la marca, el modelo y el año del automóvil.

No es obligatorio tener un mensaje de respuesta inicial. Si no lo proporciona, Amazon Lex V2 seguirá con el siguiente paso de la respuesta inicial.

Puede configurar las siguientes opciones en la respuesta inicial:

- Configurar el siguiente paso: puede indicar el siguiente paso de la conversación, como pasar a una acción de diálogo específica, seleccionar un slot concreto o pasar a una intención diferente. Para obtener más información, consulte Configurar los siguientes pasos de la conversación.
- Establecer valores: puede establecer valores para los slots y los atributos de sesión. Para obtener más información, consulte Establecer valores durante la conversación
- Añadir ramificación condicional: puede aplicar condiciones después de reproducir la respuesta inicial. Cuando una condición se evalúa como «verdadero», se llevan a cabo las acciones que usted defina. Para obtener más información, consulte Añadir condiciones a las conversaciones ramificadas.
- Ejecutar enlace de código de diálogo: puede definir un enlace de código Lambda para inicializar los datos y ejecutar la lógica empresarial. Para obtener más información, consulte Invocar el enlace de código de diálogo. Si la opción de ejecutar la función de Lambda está habilitada para la intención, el enlace de código del cuadro de diálogo se ejecuta de forma predeterminada. Puede desactivar el enlace de código del cuadro de diálogo pulsando el botón Activo.

En ausencia de una condición o de un siguiente paso explícito, Amazon Lex V2 pasa a la siguiente ranura en orden de prioridad.



> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Ranuras

Los slots son valores que proporciona el usuario para cumplir con la intención. Existen dos tipos de slots:

- Tipo de slot integrado: puede utilizar los tipos de slot integrados para capturar valores estándar como el número, el nombre y la ciudad. Para obtener una lista de los tipos compatibles, consulte Built-in tipos de ranuras.
- Tipo de slot personalizado: puede utilizar tipos de slot personalizados para obtener valores personalizados específicos de la intención. Por ejemplo, puede utilizar un tipo de slot personalizado para obtener el tipo de cuenta como «Cuenta corriente» o «Ahorros». Para obtener más información, consulte Tipos de ranura personalizados.

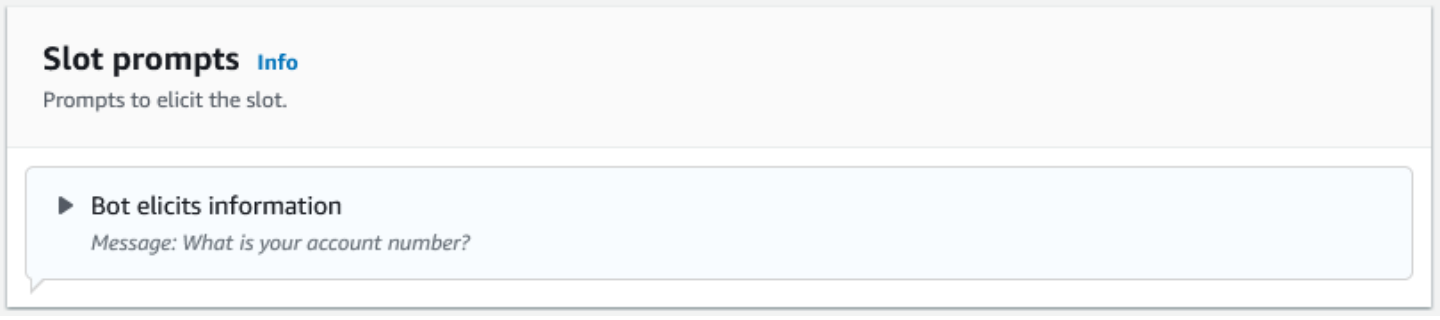
Para definir un slot en una intención, debe configurar lo siguiente:

- Información sobre el slot: este campo contiene un nombre y una descripción opcional para el slot. Por ejemplo, puede proporcionar el nombre de la ranura como «AccountNumber» para capturar los números de cuenta. Si el slot es obligatorio como parte del flujo de la conversación para cumplir con la intención, debe marcarse como obligatorio.
- Tipo de slot: un tipo de slot define la lista de valores que puede aceptar una slot. Puede crear un tipo de slot personalizado o utilizar un tipo de slot predefinido.
- Solicitud de slot: una solicitud de slot es una pregunta que se hace al usuario para recopilar información. Puede configurar el número de reintentos que se utilizan para recopilar información y la variación de la solicitud utilizada para cada reintento. También puede habilitar la invocación de una función de Lambda después de cada reintento para procesar la entrada recopilada e intentar convertirla en una entrada válida.
- Esperar y continuar (opcional): al habilitar este comportamiento, los usuarios pueden decir frases como «espera un segundo» para que el bot espere a que encuentren la información y la proporcionen. Esto solo está habilitado para la transmisión de conversaciones. Para obtener más información, consulte Cómo permitir que el bot Amazon Lex V2 espere a que el usuario proporcione más información durante una pausa.
- Respuestas de captura de slots: puede configurar una respuesta correcta y una respuesta de error en función del resultado de capturar el valor del slot a partir de las entradas del usuario.
- Ramificación condicional: puede aplicar condiciones después de reproducir la respuesta inicial. Cuando una condición se evalúa como «verdadero», se llevan a cabo las acciones que usted

defina. Para obtener más información, consulte Añadir condiciones a las conversaciones ramificadas.

- Enlace de código de diálogo: también puede utilizar un enlace de código Lambda para validar los valores de los slots y ejecutar la lógica empresarial. Para obtener más información, consulte Invocar el enlace de código de diálogo.
- Tipo de entrada de usuario: puede configurar el tipo de entrada para que el bot pueda aceptar una modalidad específica. De forma predeterminada, se aceptan las modalidades de audio y DTMF. Puede configurarlo de forma selectiva como solo audio o solo DTMF.
- Tiempos de espera y duraciones de entrada de audio: puede configurar los tiempos de espera de audio, incluidos los tiempos de espera de voz y tiempo de espera de silencio. Además, puede establecer la duración máxima del audio.
- Tiempo de espera de entrada DTMF, caracteres y longitudes: puede configurar el tiempo de espera de DTMF junto con el carácter de eliminación y el carácter final. Además, puede establecer la duración máxima del DTMF.
- Longitud del texto: puede establecer la longitud máxima para la modalidad del texto.

Una vez reproducido el mensaje del slot, el usuario proporciona el valor del slot como entrada. Si Amazon Lex V2 no entiende el valor de un slot proporcionado por el usuario, volverá a intentar obtener el slot hasta que comprenda un valor o hasta que supere el número máximo de reintentos que configuró para el slot. Con la configuración avanzada de reintentos, puede configurar los tiempos de espera, restringir el tipo de entrada y activar o desactivar la interrupción durante el mensaje inicial y los reintentos. Tras cada intento de captura de la entrada, Amazon Lex V2 puede llamar a la función de Lambda configurada para el bot con una etiqueta de invocación incluida para los reintentos. Puede usar la función de Lambda, por ejemplo, para aplicar su lógica empresarial e intentar resolverla en un valor válido. Esta función de Lambda se puede activar en las Opciones avanzadas para las indicaciones de slots.



Puede definir las respuestas que el bot debe enviar al usuario una vez introducido el valor del slot o si se supera el número máximo de reintentos. Por ejemplo, si se trata de un bot para programar

el servicio de un coche, puede enviar un mensaje al usuario cuando introduzca el número de identificación del vehículo (VIN):

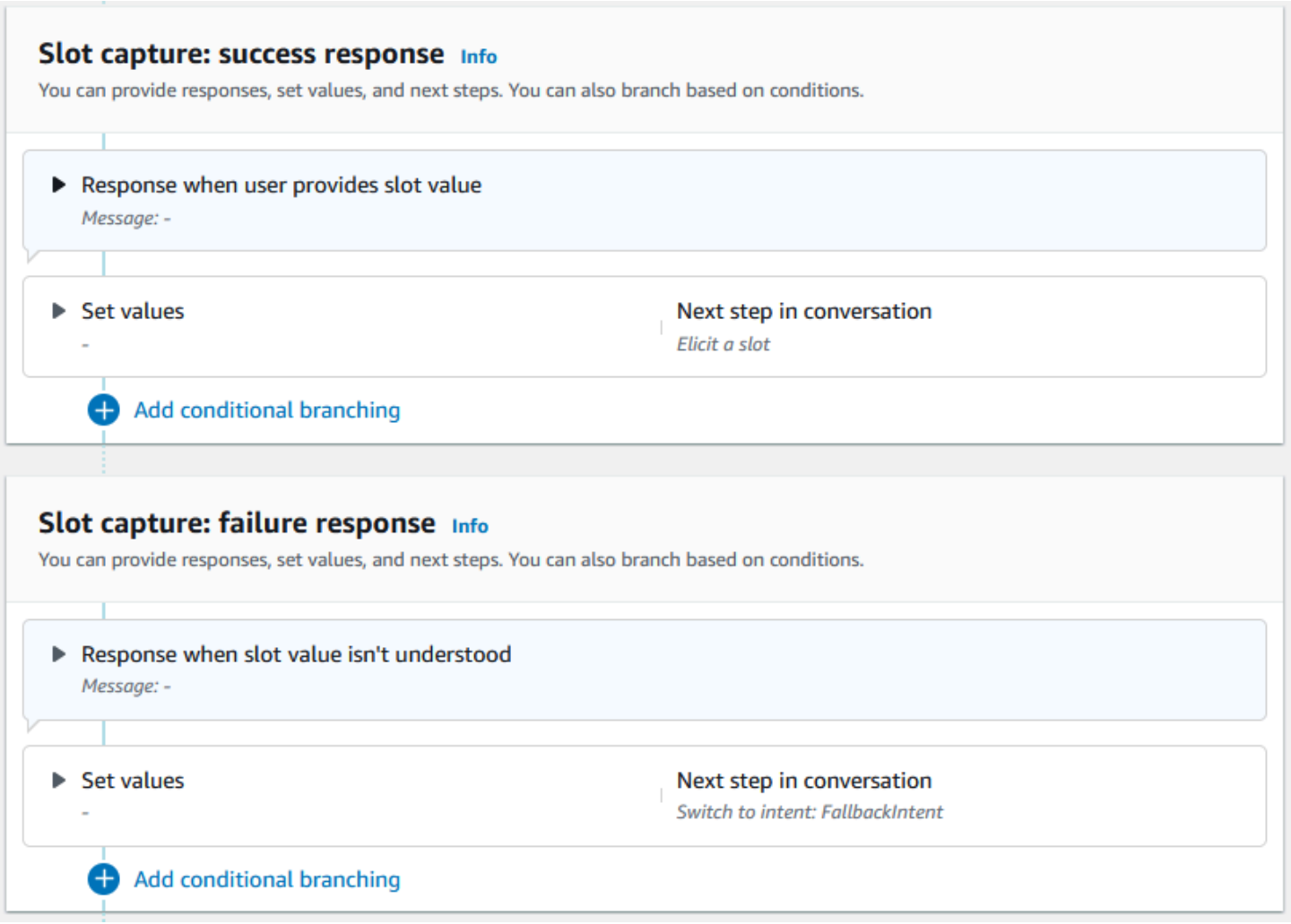
> Gracias por proporcionarnos el número VIN de tu coche. Ahora procederé a programar una cita.

Puede crear dos respuestas:

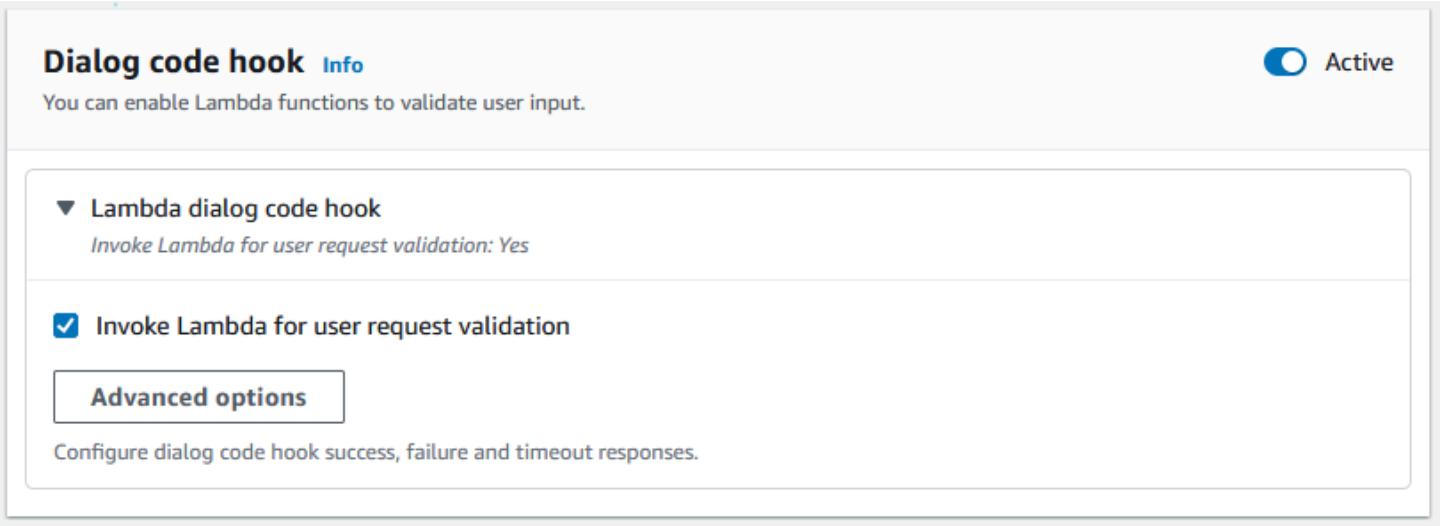
- Respuesta correcta: se envía cuando Amazon Lex V2 entiende el valor de un slot.
- Respuesta de error: se envía cuando Amazon Lex V2 no entiende el valor de un slot del usuario tras el número máximo de reintentos.

Puede establecer valores, configurar los siguientes pasos y aplicar las condiciones correspondientes a cada respuesta para diseñar el flujo de la conversación.

En ausencia de una condición o de un siguiente paso explícito, Amazon Lex V2 pasa al siguiente slot en orden de prioridad.



Puede utilizar una función de Lambda para validar un valor de slot que haya introducido un usuario y determinar cuál debe ser la siguiente acción. Por ejemplo, puede usar la función de validación para asegurarse de que el valor introducido se encuentra en el rango correcto o que tiene el formato correcto. Para activar la función de Lambda, seleccione la casilla de verificación Invocar la función de Lambda y el botón Activo en la sección Enlace del código de diálogo. Puede especificar una etiqueta de invocación para el enlace de código del diálogo. Esta etiqueta de invocación se puede utilizar en la función de Lambda para escribir la lógica empresarial correspondiente a la obtención de slots.



Los slots que no son necesarios para la intención no forman parte del flujo principal de la conversación. Sin embargo, si el enunciado de un usuario contiene un valor que su bot identifica como correspondiente a un slot opcional, puede rellenar el slot con ese valor. Por ejemplo, si configura un bot de inteligencia empresarial para que tenga un slot `City` opcional y el enunciado del usuario **`What is the sales for April in San Diego?`**, el bot rellenará el slot opcional con **`San Diego`**. Puede configurar la lógica empresarial para que utilice el valor de slot opcional, si está presente.

Los slots que no sean necesarios para la intención no se pueden obtener mediante los siguientes pasos. Estos pasos solo se pueden completar durante la obtención de intenciones (como en el ejemplo anterior) o se pueden obtener configurando el estado del diálogo en la función de Lambda. Si el slot se obtiene mediante la función de Lambda, debe usar la función de Lambda para decidir el siguiente paso de la conversación una vez que se complete la obtención del slot. Para permitir el siguiente paso a la hora de compilar el bot, debe marcar el slot como obligatorio para esa intención.

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

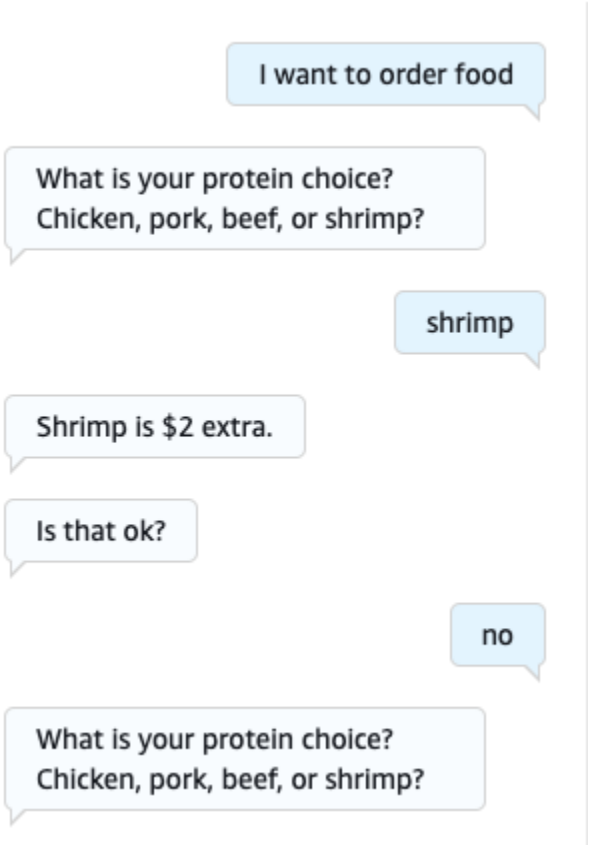
En los siguientes temas, se describe cómo configurar un bot para que vuelva a obtener un valor de slot que ya se ha ocupado y cómo crear un slot compuesto por varios valores:

Temas

- Obtención de slots de nuevo
- Uso de valores múltiples en un slot

Obtención de slots de nuevo

Puede configurar su bot para que vuelva a obtener un slot que ya se ha ocupado fijando el valor de ese slot en **`null`** y configurando el siguiente paso de la conversación para volver a obtener este slot. Por ejemplo, tal vez quiera volver a obtener un slot después de que su cliente rechace la confirmación del mismo basándose en información adicional, como en la siguiente conversación:



Puede configurar un bucle a partir de la respuesta de confirmación para volver a obtener el slot con el editor de intenciones o con el Uso del generador visual de conversaciones.

> **Note**
>
> Puede retroceder para volver a obtener un slot en cualquier momento de la conversación, siempre que haya establecido ese valor de antemano en **`null`**.

Reproducir el ejemplo anterior con el editor de intenciones

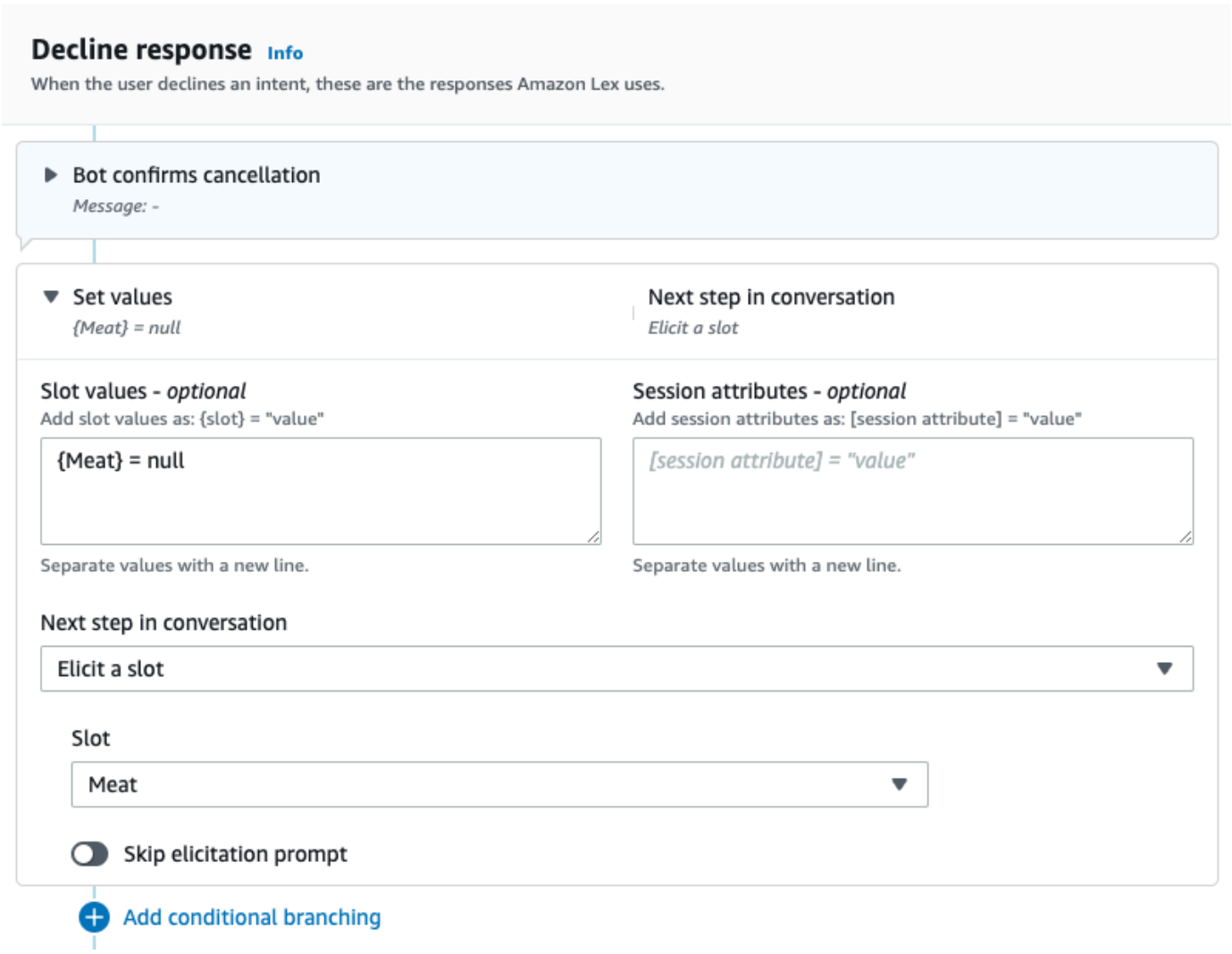
1. En la sección Confirmación del editor de intenciones, seleccione la flecha derecha situada junto a Instrucciones para confirmar la intención para expandir la sección.
2. Seleccione Opciones avanzadas en la parte inferior.
3. En la sección Rechazar respuesta, seleccione la flecha derecha situada junto a Establecer valores para expandir la sección. Complete esta sección con los siguientes pasos, como se muestra en la siguiente imagen:

    a. Establezca el valor del slot que desea volver a obtener en **`null`**. En este ejemplo, queremos volver a obtener el slot `Meat`, por lo que introducimos **`{Meat} = null`** en la sección de Valores del slot.

    b. En el menú desplegable de Siguiente paso de la conversación, seleccione Obtener un slot.

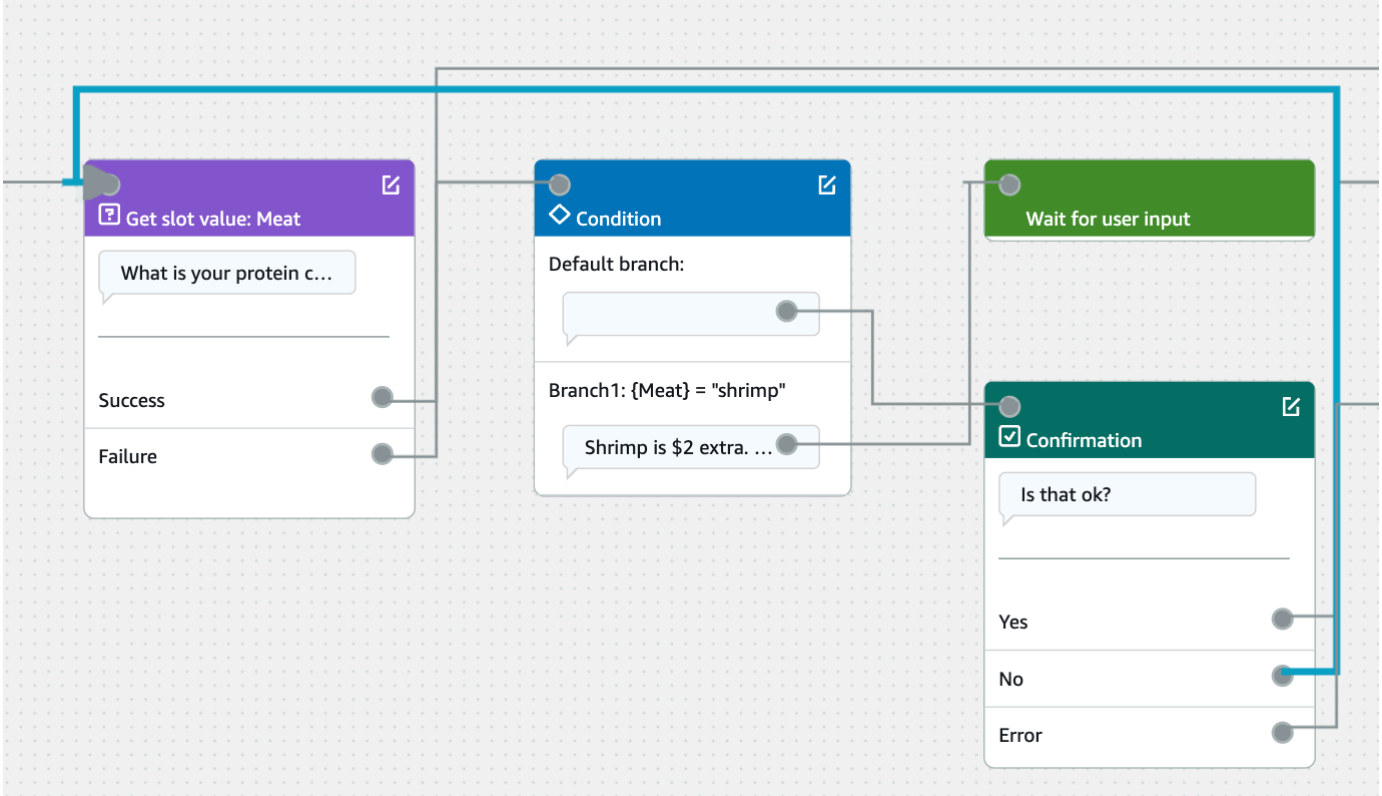
    c. Aparecerá una sección de Slot. En el menú desplegable que hay debajo, seleccione el slot que quiere volver a obtener.

    d. Seleccione Opciones de actualización para confirmar los cambios.

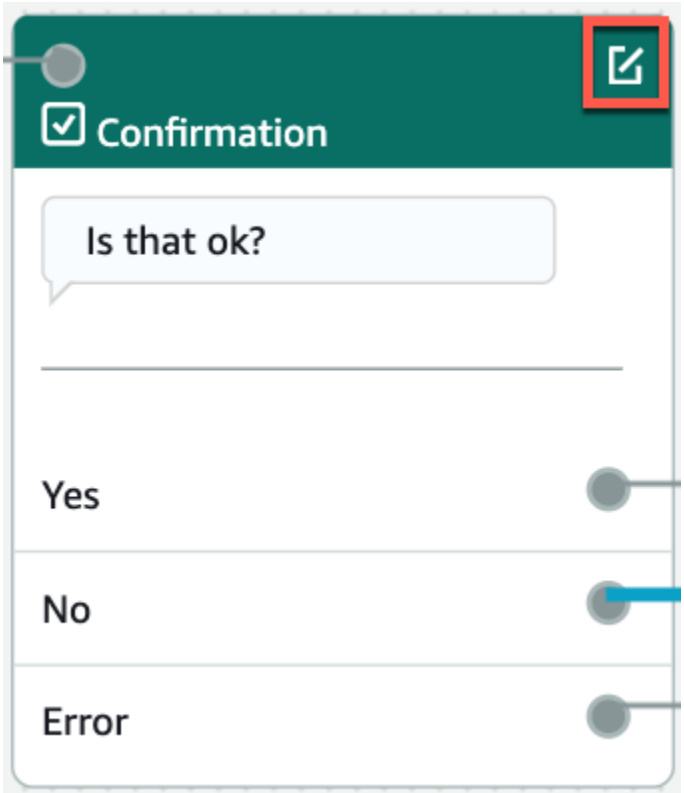


Reproducir el ejemplo anterior con el generador visual de conversaciones

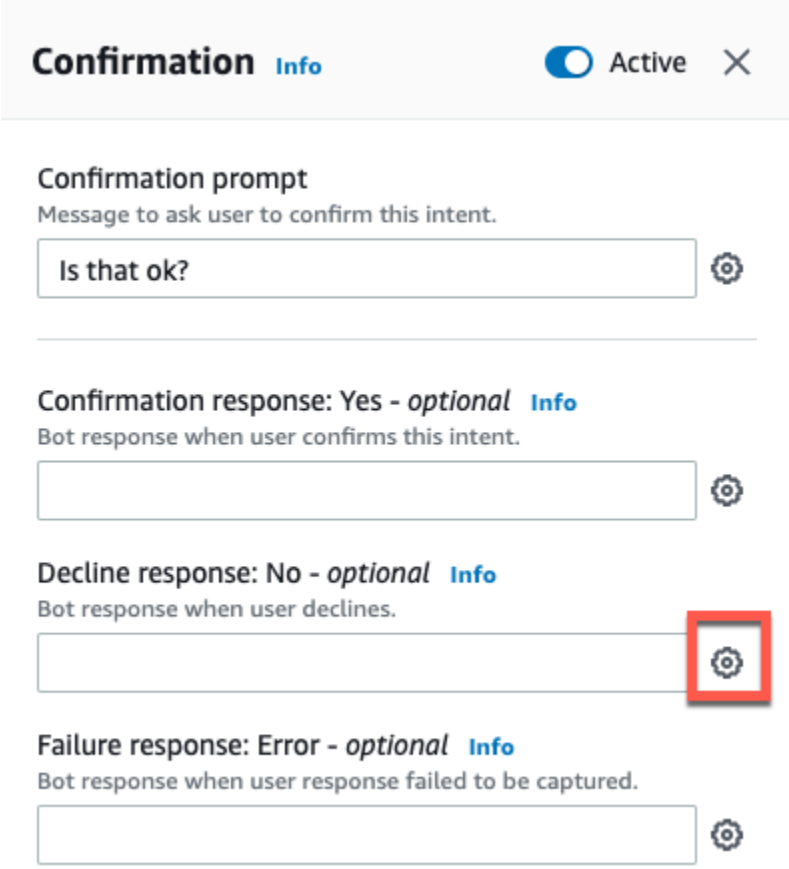
1. Cree una conexión desde el puerto No del bloque de Confirmación hasta el puerto entrante del bloque Obtener valor de slot: carne.



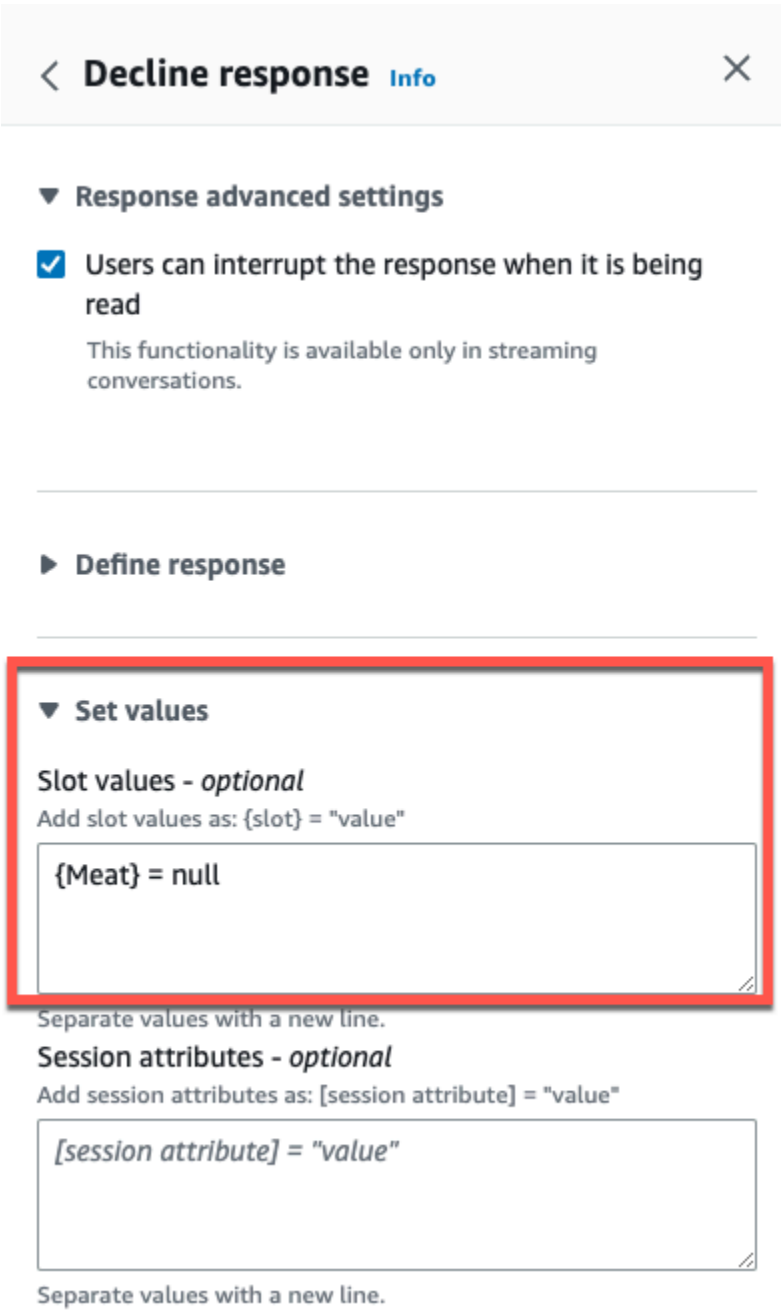
2. Seleccione el icono de Editar en la esquina superior derecha del bloque de Confirmación.



3. Seleccione el icono con forma de engranaje situado junto a la respuesta del bot en la sección Rechazar respuesta.



4. En la sección Establecer valores, añada «{Meat} = null» en el cuadro de Valores del slot.



5. Seleccione Guardar intención.

Uso de valores múltiples en un slot

> **Note**
>
> Los slots de valores múltiples solo se admiten en inglés (EE. UU.).

Para algunas intenciones, es posible que desee capturar varios valores para un solo slot. Por ejemplo, un bot de pedidos de pizza podría tener una intención con el siguiente enunciado:

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

La intención espera que el slot `{toppings}` contenga una lista de los ingredientes que el cliente quiere incluir en su pizza, por ejemplo, «pepperoni y piña».

Para configurar un slot de modo que capture varios valores, debe establecer el campo `allowMultipleValues` del slot en «verdadero». Puede configurar el campo mediante la consola o con la UpdateSlotoperación CreateSloto.

Solo puede marcar los slots con tipos de slots personalizados como slots con valores múltiples.

Para una ranura con varios valores, Amazon Lex V2 devuelve una lista de valores de ranura en respuesta a la RecognizeUtteranceoperación RecognizeTexto. La siguiente es la información sobre las ranuras que devuelve el bot para la expresión «Quiero una pizza con pepperoni y piña». OrderPizza

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

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

Los slots con varios valores siempre devuelven una lista de valores. Cuando el enunciado solo contiene un valor, la lista de valores devuelta solo contiene una respuesta.

Amazon Lex V2 reconoce varios valores separados por espacios, comas (,) y la conjunción «y». Los slots con varios valores funcionan tanto con la entrada de texto como con la de voz.

Puede utilizar slots con varios valores en las indicaciones. Por ejemplo, puede establecer que el mensaje de confirmación de una intención sea

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

Cuando Amazon Lex V2 envía el mensaje al usuario, dice: «¿Quieres que pida tu pizza de pepperoni y piña?»

Los slots con varios valores admiten valores predeterminados únicos. Si se proporcionan varios valores predeterminados, Amazon Lex V2 rellena el slot solo con el primer valor disponible. Para obtener más información, consulte Uso de los valores de ranura predeterminados en las intenciones de su bot Lex V2.

Puede utilizar la ofuscación de slots para enmascarar los valores de un slot con varios valores en los registros de conversaciones. Cuando elige ofuscar los valores de slot, el valor de cada valor de slot se reemplaza con el nombre del slot. Para obtener más información, consulte Cómo ocultar valores de ranura en registros de conversación de Lex V2.

Confirmación

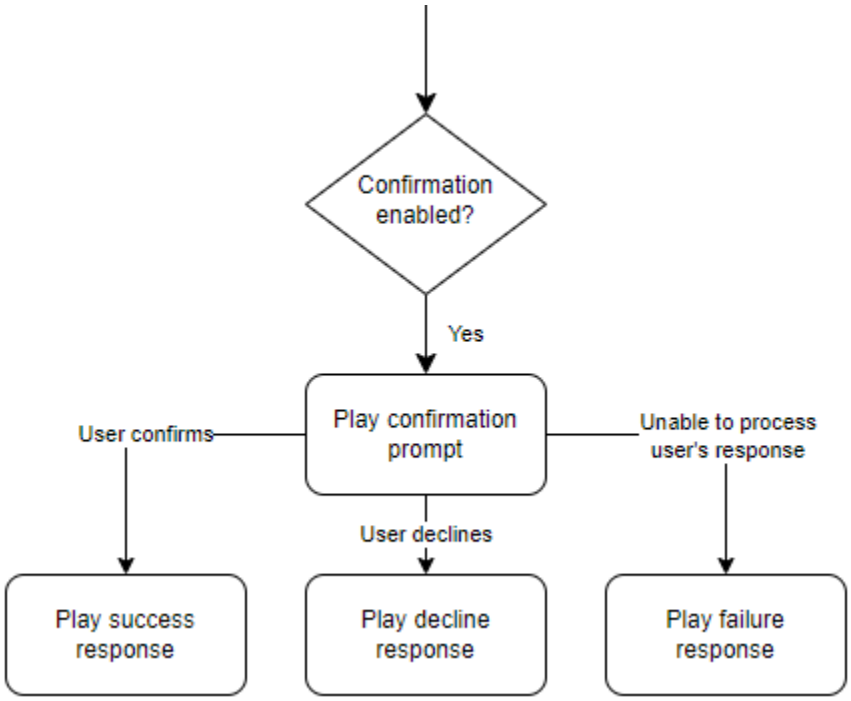
Una vez finalizada la conversación con el usuario y rellenados los valores de los slots correspondientes a la intención, puede configurar un mensaje de confirmación para preguntar al

usuario si los valores de los slots son correctos. Por ejemplo, un bot que programe citas de servicio para automóviles podría preguntar al usuario lo siguiente:

> Tengo programado el servicio para tu Honda Civic 2017 para el 25 de marzo a las 15:00 h. ¿Es correcto?

Puede definir 3 tipos de respuestas a la solicitud de confirmación:

- Respuesta de confirmación: esta respuesta se envía al usuario cuando este confirma su intención. Por ejemplo, después de que el usuario responda «sí» a la pregunta «¿deseas realizar el pedido?».
- Respuesta de negación: esta respuesta se envía al usuario cuando este rechaza la intención. Por ejemplo, después de que el usuario responda «no» a la pregunta «¿deseas realizar el pedido?».
- Respuesta de error: esta respuesta se envía al usuario cuando no se puede procesar el mensaje de confirmación. Por ejemplo, si la respuesta del usuario no se pudo entender o no se pudo resolver con un sí o un no.

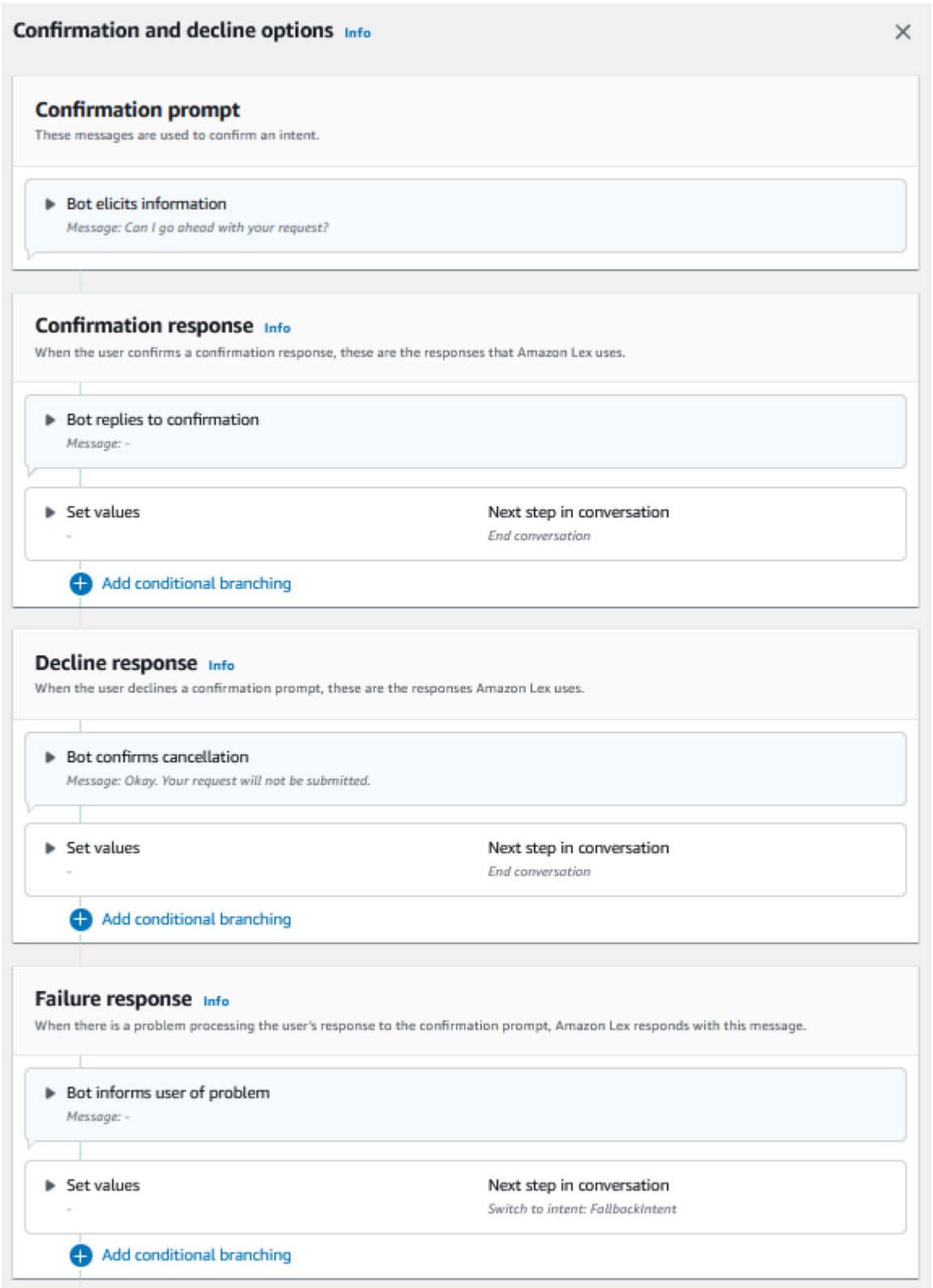


Si no especifica un mensaje de confirmación, Amazon Lex V2 pasa al paso de cumplimiento o a la respuesta de cierre.

Puede establecer valores, configurar los pasos siguientes y aplicar las condiciones correspondientes a cada respuesta para diseñar el flujo de la conversación. En ausencia de una condición o de un siguiente paso explícito, Amazon Lex V2 pasa al paso de cumplimiento.

También puede activar el enlace de códigos del cuadro de diálogo para validar la información recopilada en la intención antes de enviarla para su cumplimiento. Para usar un enlace de códigos, habilite el enlace de códigos del cuadro de diálogo en las opciones avanzadas del mensaje de confirmación. Además, configure el siguiente paso del estado anterior para ejecutar el enlace de código del diálogo. Para obtener más información, consulte Invocar el enlace de código de diálogo.

> **Note**
>
> Si utiliza un enlace de código para activar el paso de confirmación en tiempo de ejecución, debe marcar el paso de confirmación como Activo en el momento de la compilación.


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

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Usar una función de Lambda para validar una intención.

Puede definir un enlace de código Lambda para validar la intención antes de enviarla para su cumplimiento. Para usar un enlace de códigos, habilite el enlace de códigos del cuadro de diálogo en las opciones avanzadas del mensaje de confirmación.

Cuando utiliza un enlace de código, puede definir las acciones que Amazon Lex V2 lleva a cabo una vez ejecutado el enlace de código. Puede crear tres tipos de respuestas:

- Respuesta correcta: se envía al usuario cuando el enlace de código se completa correctamente.
- Respuesta de error: se envía al usuario cuando el enlace de código no se ejecuta correctamente o cuando el enlace de código devuelve `Failure` en la respuesta.
- Respuesta de tiempo de espera agotado: se envía al usuario cuando el enlace de código no se completa en el período de tiempo de espera configurado.

Cumplimiento

Una vez que el usuario haya proporcionado todos los valores de los slots para la intención, Amazon Lex V2 cumplirá con la solicitud del usuario. Puede configurar las siguientes opciones para el cumplimiento:

- Enlace de código de cumplimiento: puede usar esta opción para controlar la invocación de Lambda de cumplimiento. Si la opción está desactivada, el cumplimiento se realiza correctamente sin invocar la función de Lambda.
- Actualizaciones de cumplimiento: puede habilitar las actualizaciones de cumplimiento para las funciones de Lambda que tardan más de unos segundos en completarse, de modo que el usuario sepa que el proceso está en curso. Para obtener más información, consulte Configuración de las

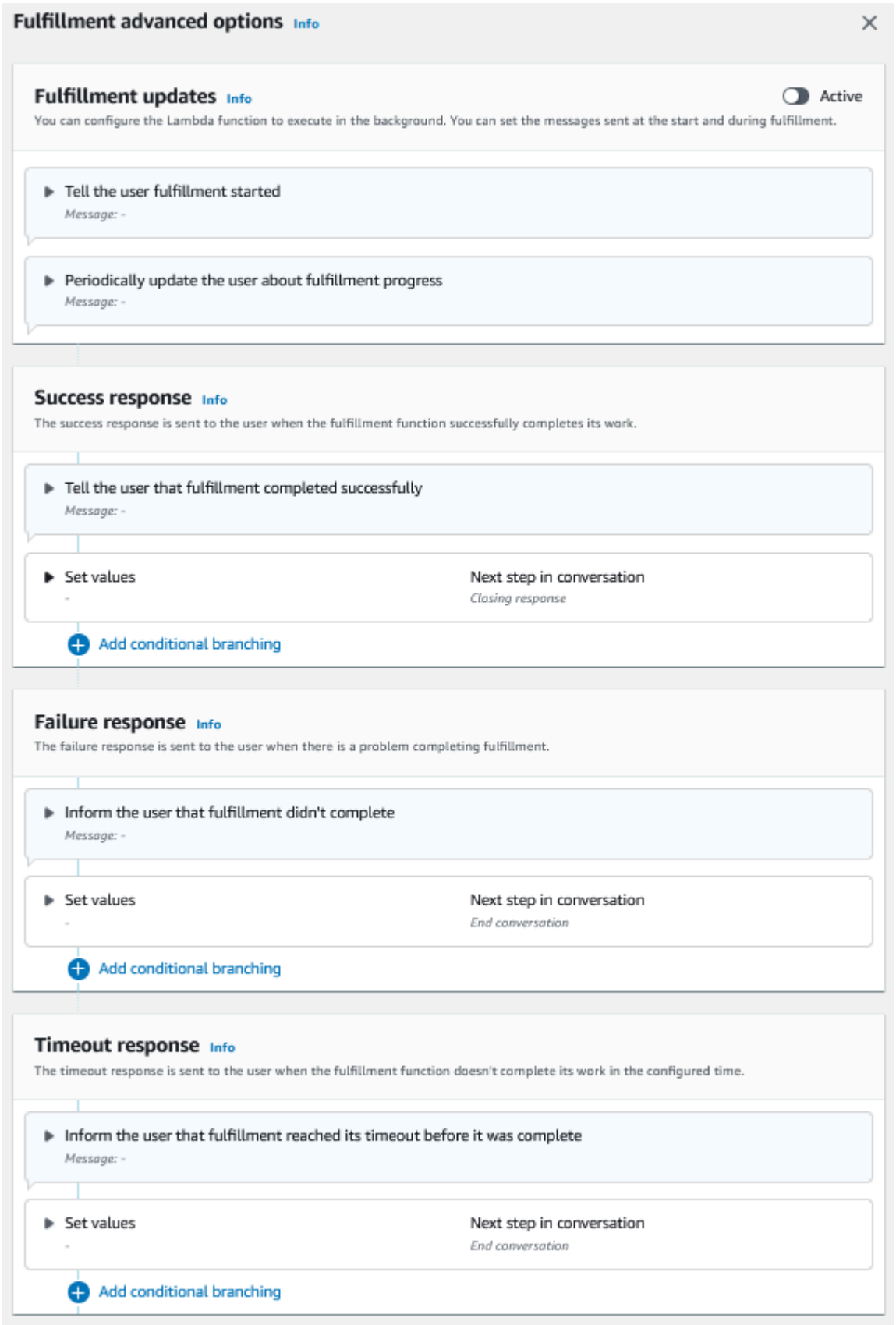
actualizaciones del progreso de cumplimiento para su bot Lex V2. Esta funcionalidad solo está disponible para conversaciones en streaming.

- Respuestas de cumplimiento: puede configurar una respuesta de éxito, una respuesta de error y una respuesta de tiempo de espera agotado. La respuesta adecuada se devuelve al usuario en función del estado de la invocación de Lambda de cumplimiento.

Hay tres posibles respuestas de cumplimiento:

- Respuesta correcta: mensaje que se envía cuando el cumplimiento Lambda se completa correctamente.
- Respuesta fallida: mensaje que se envía si se ha producido un error en el cumplimiento o si Lambda no se puede completar por algún motivo.
- Respuesta de tiempo de espera agotado: mensaje que se envía si la función de Lambda de cumplimiento no finaliza dentro del tiempo de espera configurado.

Puede establecer valores, configurar los pasos siguientes y aplicar las condiciones correspondientes a cada respuesta para diseñar el flujo de la conversación. En ausencia de una condición o de un siguiente paso explícito, Amazon Lex V2 pasa a la respuesta de cierre.


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

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

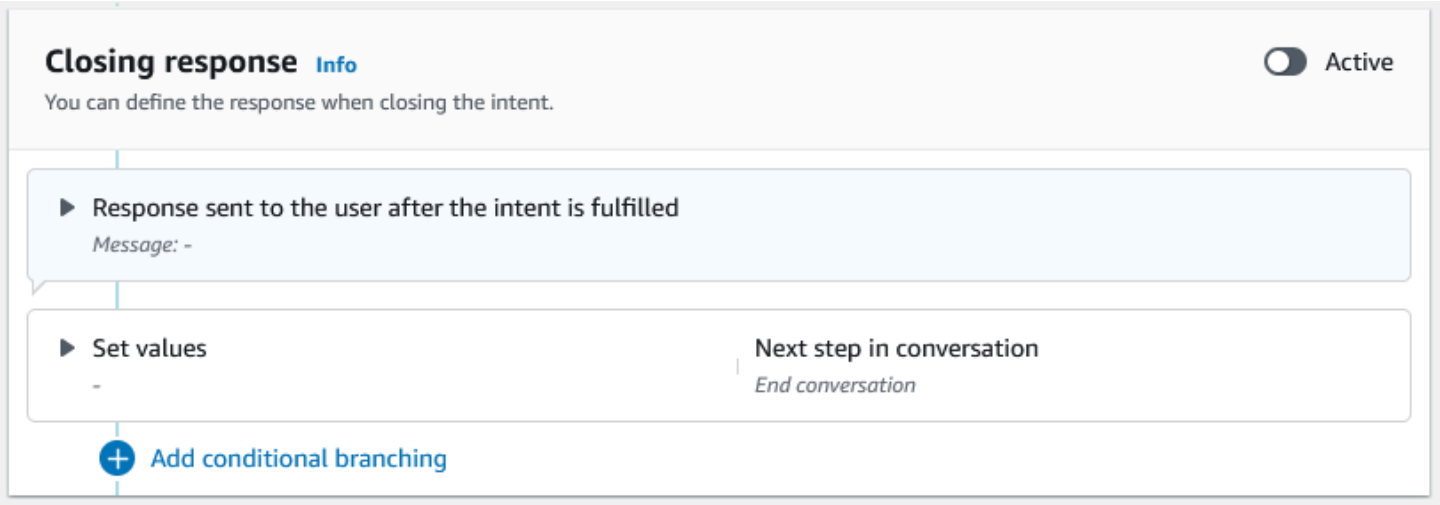
Respuesta de cierre

La respuesta de cierre se envía al usuario una vez que se ha cumplido su intención. Puede usar la respuesta de cierre para finalizar la conversación o puede usarla para hacerle saber al usuario que puede continuar con otra intención. Por ejemplo, en un bot de reservas de viajes, puede configurar la respuesta de cierre para la intención de reservar una habitación de hotel de la siguiente manera:

```
Muy bien, he reservado tu habitación de hotel. ¿Puedo ayudarte con algo más?
```

Puede establecer valores, configurar los pasos siguientes y aplicar las condiciones después de cada respuesta final al diseño de la ruta de la conversación. En ausencia de una condición o de un siguiente paso explícito, Amazon Lex V2 finaliza la conversación.

Si no proporciona una respuesta final o si ninguna de las condiciones resulta ser verdadera, Amazon Lex V2 finaliza la conversación con su bot.

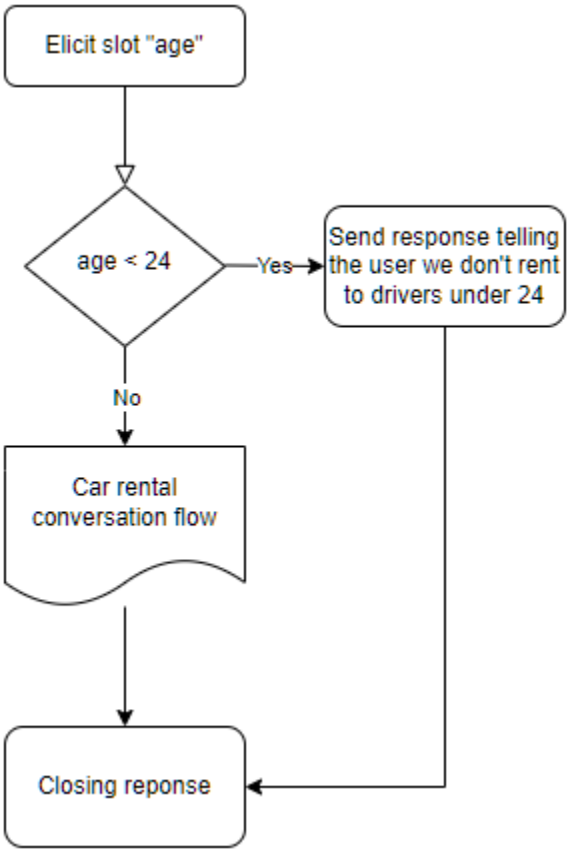


> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Crear rutas de conversación

Normalmente, Amazon Lex V2 gestiona el flujo de las conversaciones con los usuarios. En el caso de los bots simples, el flujo predeterminado puede ser suficiente para crear una buena experiencia para los usuarios. Sin embargo, en el caso de los bots más complejos, es posible que desee tomar el control de la conversación y dirigir el flujo hacia rutas más complejas.

Por ejemplo, en el caso de un bot para reservar coches de alquiler, es posible que no quiera alquilarlos a conductores más jóvenes. En este caso, puede crear una condición que compruebe si un conductor tiene menos de cierta edad y, de ser así, pasar directamente a la respuesta de cierre.



Para diseñar estas interacciones, puede configurar el siguiente paso en cada punto de la conversación, evaluar las condiciones, establecer valores e invocar enlaces de código.

La ramificación condicional le ayuda a crear rutas para sus usuarios a través de interacciones complejas. Puede usar una ramificación condicional en cualquier momento en el que pase el control de la conversación a su bot. Por ejemplo, puede crear una condición antes de que el bot obtenga el primer valor del slot, puede crear una condición entre la obtención de cada valor de slot o puede crear una condición antes de que el bot finalice la conversación. Para ver una lista de los puntos en los que puede añadir condiciones, consulte Adición de intenciones.

Al crear un bot, Amazon Lex V2 crea una ruta predeterminada a través de la conversación en función del orden de prioridad de los slots. Para personalizar la ruta de la conversación, puede modificar el siguiente paso en cualquier momento de la conversación. Para obtener más información, consulte Configurar los siguientes pasos de la conversación.

Para crear rutas alternativas en función de las condiciones, puede usar una rama condicional en cualquier punto de la conversación. Por ejemplo, puede crear una condición antes de que el bot obtenga el primer valor del slot. Puede crear una condición entre la obtención de cada valor de slot o puede crear una condición antes de que el bot finalice la conversación. Para ver una lista de los

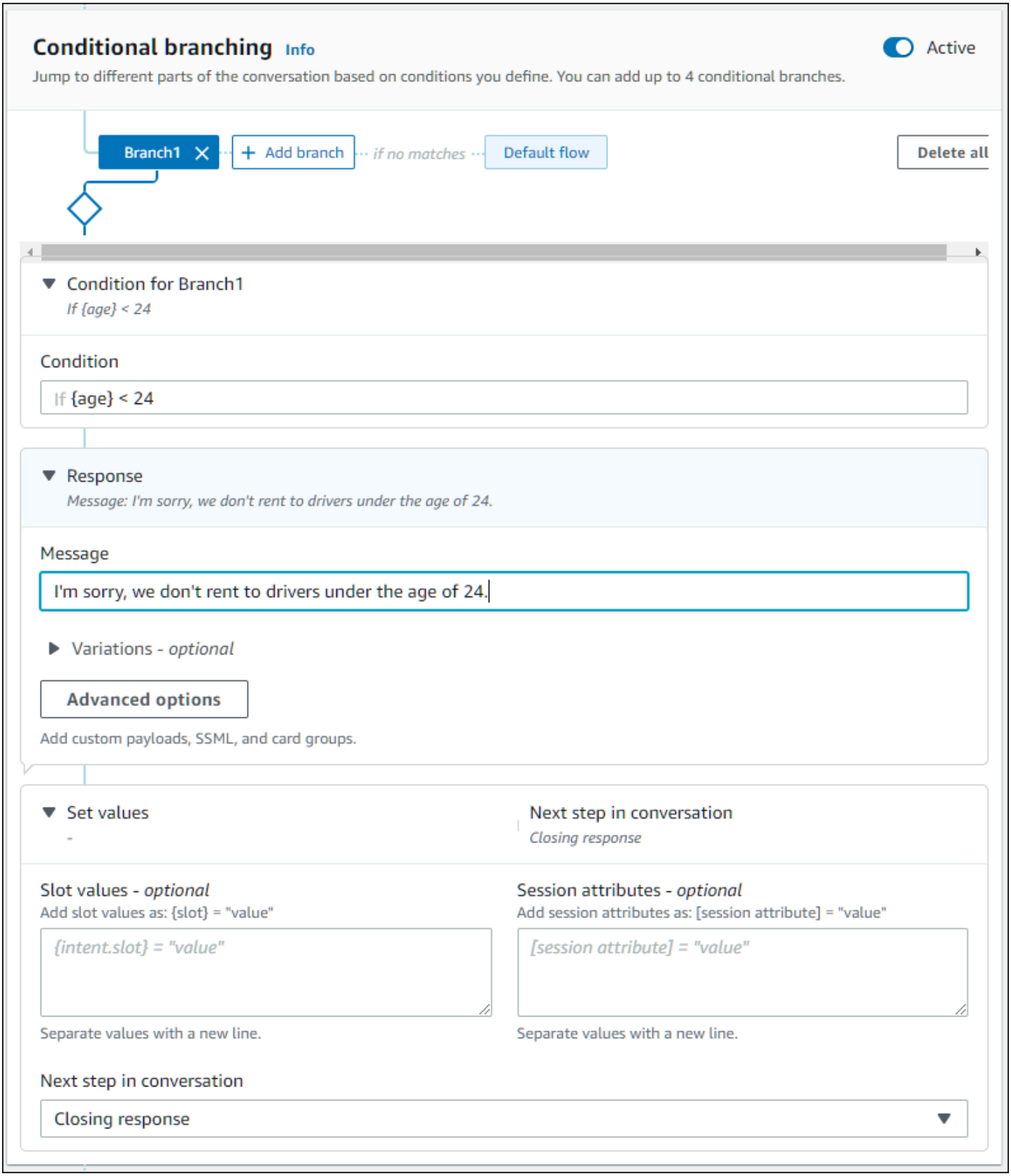
puntos en los que puede añadir condiciones, consulte Añadir condiciones a las conversaciones ramificadas.

Puede establecer condiciones en función de los valores de los slots, los atributos de sesión, el modo de entrada y la transcripción de entrada o una respuesta de Amazon Kendra.

Puede configurar los valores de los atributos y los slots de la sesión en cualquier momento de la conversación. Para obtener más información, consulte Establecer valores durante la conversación.

También puede configurar la siguiente acción en un enlace de código de diálogo para ejecutar una función de Lambda. Para obtener más información, consulte Invocar el enlace de código de diálogo.

En la imagen siguiente, se muestra la creación de una ruta para un slot en la consola. En este ejemplo, Amazon Lex V2 obtendrá el slot «edad». Si el valor de la ranura es inferior a 24, Amazon Lex V2 pasa a la respuesta de cierre; de lo contrario, Amazon Lex V2 seguirá la ruta predeterminada.


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

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Configurar los siguientes pasos de la conversación

Puede configurar el siguiente paso en cada etapa de la conversación para diseñar las conversaciones. Normalmente, Amazon Lex V2 configura automáticamente los siguientes pasos predeterminados para cada etapa de la conversación según el siguiente orden.

Respuesta inicial → Obtención de slot → Confirmación (si está activa) → Cumplimiento (si está activo) → Respuesta de cierre (si está activa) → Fin de la conversación

Puede modificar los siguientes pasos predeterminados y diseñar la conversación en función de la experiencia del usuario esperada. Los siguientes pasos se pueden configurar en cada etapa de la conversación:

Saltar a

- Respuesta inicial: la conversación se reinicia desde el principio de la intención. Puede optar por omitir la respuesta inicial al configurar el siguiente paso.
- Obtener un slot: puede obtener cualquier slot en la intención.
- Evaluar las condiciones: puede evaluar las condiciones y diversificar la conversación en cualquier etapa de la conversación.
- Invocar el enlace de código de diálogo: puede invocar la lógica empresarial en cualquier momento.
- Confirmar la intención: se le pedirá al usuario que confirme la intención.
- Cumplir la intención: el cumplimiento de la intención comenzará como el siguiente paso.
- Respuesta de cierre: la respuesta de cierre se devolverá al usuario.

Cambiar a

- Intención: puede cambiar a una intención diferente y continuar la conversación con esa intención. Si lo desea, puede omitir la respuesta inicial de la intención mientras realiza la transición.
- Intención: slot específico: puede obtener directamente un slot específico con una intención diferente si ya ha capturado algunos valores de slot en la intención actual.

Espera a la entrada del usuario: el bot espera a que el usuario introduzca información para reconocer cualquier nueva intención. Puede configurar mensajes como «¿Puedo ayudarte con algo más?» antes de configurar el siguiente paso. El bot tendrá el estado `ElicitIntent` de diálogo.

Finalizar conversación: se cierra la conversación con el bot.

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Establecer valores durante la conversación

Amazon Lex V2 ofrece la posibilidad de establecer valores de slot y valores de atributos de sesión en cada paso de la conversación. A continuación, puede utilizar estos valores durante la conversación para evaluar las condiciones o utilizarlos para cumplir la intención.

Puede establecer valores de slot para la intención actual. Si el siguiente paso de la conversación es invocar otra intención, puede establecer los valores de slot de la nueva intención.

Si el slot asignado no está ocupado o si no se puede analizar la ruta JSON, el atributo se establecerá en `null`.

Utilice la siguiente sintaxis cuando utilice valores de slot y atributos de sesión:

- Valores de slot: escriba el nombre del slot entre corchetes («{ }»). Para los valores de slot con la intención actual, solo necesita usar el nombre del slot. Por ejemplo, `{slot}`. Si va a establecer un valor en la siguiente intención, debe utilizar tanto el nombre de la intención como el nombre del slot para identificarlo. Por ejemplo, `{intent.slot}`.

Ejemplos:

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

El valor de un slot puede ser cualquiera de los siguientes:

- una cadena constante
- una ruta JSON que hace referencia al bloque de transcripciones de la respuesta de Amazon Lex (para en-US y en-GB)
- un atributo de sesión

Ejemplos:

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

> **Note**
>
> Los valores de slot también se pueden configurar como `null`. Si necesita volver a obtener el valor de un slot que ya se ha ocupado, debe establecer el valor en `null` antes de volver a solicitar al cliente el valor del slot. Si el slot asignado no está ocupado o si no se puede analizar la ruta JSON, el atributo se establecerá en `null`.

- Atributos de sesión: escriba el nombre del atributo entre corchetes («[ ]»). Por ejemplo, `[sessionAttribute]`.

  Ejemplos:

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

  El valor del atributo de sesión puede ser uno de los siguientes valores:

  - una cadena constante

- una ruta JSON que hace referencia al bloque de transcripciones de la respuesta de Amazon Lex (para en-US y en-GB)
- una referencia de valor de slot

> **Note**
>
> Si el slot asignado no está ocupado o si no se puede analizar la ruta JSON, el atributo se establecerá en `null`.

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Añadir condiciones a las conversaciones ramificadas

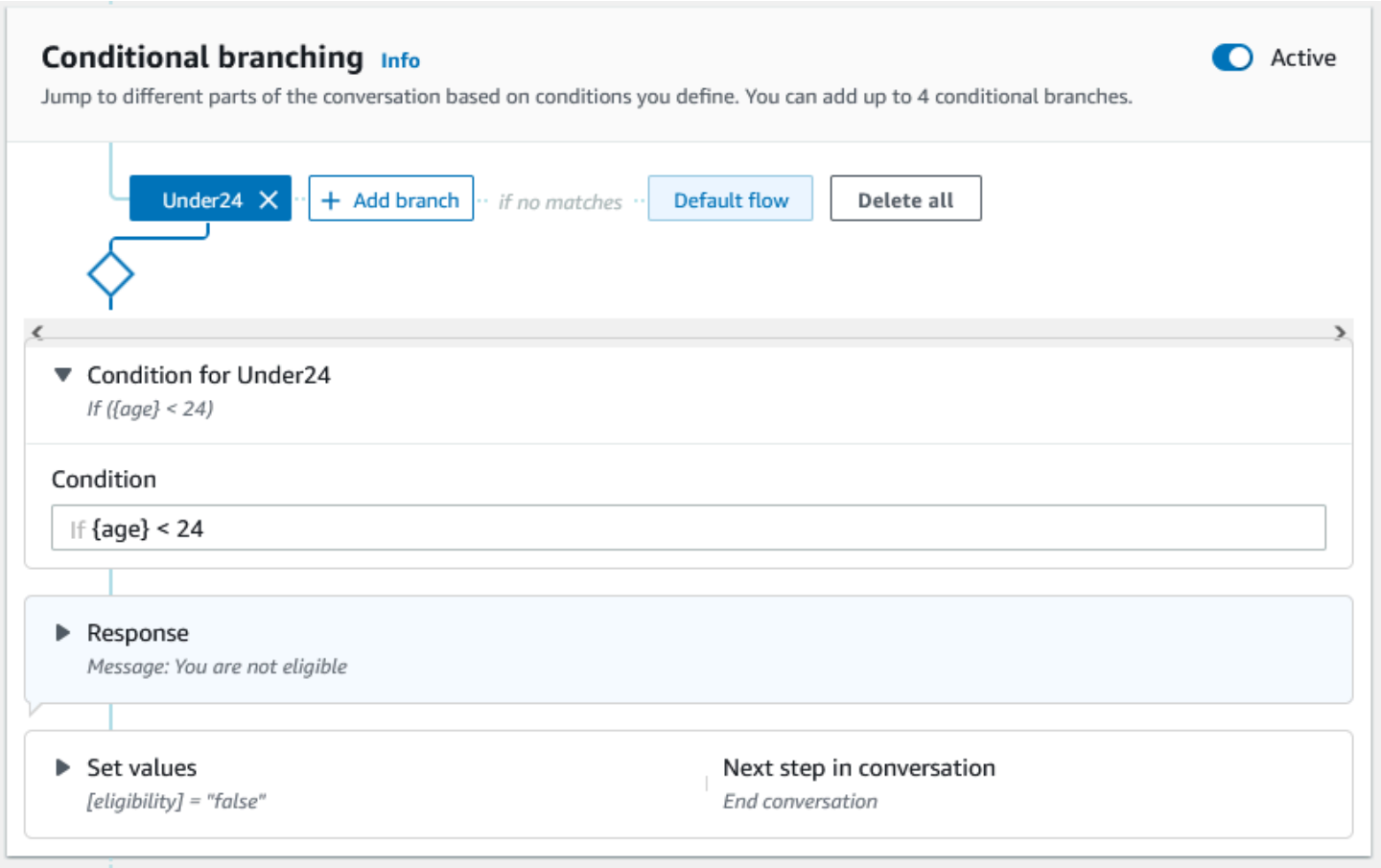
Puede usar la ramificación condicional para controlar la ruta que sigue su cliente durante la conversación con su bot. Puede ramificar la conversación en función de los valores de los slots, los atributos de sesión, los contenidos del modo de entrada y los campos de transcripción de entrada o una respuesta de Amazon Kendra.

Puede definir hasta cuatro ramas. Cada rama tiene una condición que debe cumplirse para que Amazon Lex V2 siga esa rama. Si ninguna de las ramas cumple su condición, se sigue una rama predeterminada.

Cuando define una rama, define la acción que Amazon Lex V2 debe realizar si las condiciones correspondientes a esa rama resultan verdaderas. Puede definir cualquiera de las siguientes acciones:

- Una respuesta enviada al usuario.
- Valores de slots para aplicarlos a los slots.
- Valores de los atributos de sesión para la sesión actual.

- El siguiente paso de la conversación. Para obtener más información, consulte Crear rutas de conversación.



Cada rama condicional tiene una expresión booleana que debe cumplirse para que Amazon Lex V2 siga esa rama. Existen operadores de comparación y booleanos, funciones y operadores cuantificadores que puede utilizar para sus condiciones. Por ejemplo, la siguiente condición se vuelve verdadera si el slot {edad} es inferior a 24.

```
{age} < 24
```

La siguiente condición se cumple si el slot multivalor {ingredientes} contiene la palabra «piña».

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

Puede combinar varios operadores de comparación con un operador booleano para condiciones más complejas. Por ejemplo, la siguiente condición se cumple si el valor del slot {marca} es «Honda» y el valor del slot {modelo} es «Civic». Use paréntesis para establecer el orden de evaluación.

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

En los siguientes temas se proporcionan detalles sobre los operadores y las funciones de las ramas condicionales.

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Temas

- Operadores de comparación
- Operadores booleanos
- Operadores cuantificadores
- Funciones
- Ejemplos de expresiones condicionales

Operadores de comparación

Amazon Lex V2 admite los siguientes operadores de comparación para las condiciones:

- Igual (=)
- Distinto de (!=)
- Menor que (<)
- Menor que o igual a (<=)
- Mayor que (>)
- Mayor que o igual a (>=)

Cuando se utiliza un operador de comparación, se utilizan las siguientes reglas.

- El lado izquierdo debe ser una referencia. Por ejemplo, para hacer referencia a un valor del slot, utilice `{slotName}`. Para hacer referencia a un valor de atributo de sesión, utilice `[attribute]`. Para el modo de entrada y la transcripción de entrada, se utiliza `$.inputMode` y `$.inputTranscript`.
- El lado derecho debe ser constante y del mismo tipo que el lado izquierdo.
- Cualquier expresión que haga referencia a un atributo que no se haya establecido se considera inválida y no se evalúa.
- Al comparar un slot con varios valores, el valor utilizado es una lista separada por comas de todos los valores interpretados.

Las comparaciones se basan en el tipo de slot de la referencia. Se resuelven de la siguiente manera:

- Cadenas: las cadenas se comparan en función de su representación en ASCII. Esta comparación no distingue entre mayúsculas y minúsculas.
- Números: los slots basados en números se convierten de la representación de cadena a un número y, a continuación, se comparan.
- Fecha/hora: los slots basados en el tiempo se comparan en función de la serie temporal. La fecha u hora anterior se considera más pequeña. En cuanto a la duración, los períodos más cortos se consideran más pequeños.

Operadores booleanos

Amazon Lex V2 admite operadores booleanos para combinar operadores de comparación. Permiten crear declaraciones similares a las siguientes:

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

También puede utilizar las siguientes operaciones:

- Y (&&)
- O (||)
- NO (!)

Operadores cuantificadores

Los operadores cuantificadores evalúan los elementos de una secuencia y determinan si uno o más elementos cumplen la condición.

- CONTIENE: determina si el valor especificado está contenido en un slot con varios valores y devuelve el valor verdadero si lo es. Por ejemplo, `{toppings} CONTAINS "pineapple"` devuelve el valor verdadero si el usuario pidió piña para su pizza.

Funciones

Las funciones deben ir precedidas de la cadena `fn.`. El argumento de la función es una referencia a un slot, un atributo de sesión o un atributo de solicitud. Amazon Lex V2 proporciona dos funciones para obtener información de los valores de los slots: SessionAttribute o RequestAttribute.

- fn.COUNT (): cuenta el número de valores de un slot con varios valores.

  Por ejemplo, si el slot `{toppings}` contiene el valor «pepperoni, piña»:

  ```
  fn.COUNT({toppings}) = 2
  ```
- fn.IS_SET (): el valor es verdadero si se establece un slot, un atributo de sesión o un atributo de solicitud en la sesión actual.

  Basado en el ejemplo anterior:

  ```
  fn.IS_SET({toppings})
  ```
- fn.LENGTH(): el valor es la longitud del valor del atributo de sesión, el valor de ranura o el atributo de ranura que se establece en la sesión actual. Esta función no admite ranuras de valores múltiples ni ranuras compuestas.

  Ejemplo:

  Si la ranura `{credit-card-number}` contiene el valor “123456781234”:

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

Ejemplos de expresiones condicionales

Estos son algunos ejemplos de expresiones condicionales. NOTA: `$.` representa el punto de entrada a la respuesta JSON de Amazon Lex V2. El siguiente valor se `$.` analizará en la respuesta de

Amazon Lex V2 para recuperar el valor. Las expresiones condicionales que utilicen la referencia de ruta JSON al bloque de transcripciones en la respuesta de Amazon Lex V2 solo se admitirán en las mismas configuraciones regionales que admiten las puntuaciones de transcripción de ASR.

Tipo de valor	Caso de uso	Expresión condicional
Slot personalizado	El valor del slot `pizzaSize` es igual a grande	`{pizzaSize} = "large"`
Slot personalizado	`pizzaSize` es igual a grande o mediana	`{pizzaSize} = "large"` O `{pizzaSize} = "medium"`
Slot personalizado	Expresiones con `()` y `AND/OR`	`{pizzaType} = "pepperoni"` O `{pizzaSize} = "medium"` O `{pizzaSize} = "small"`
Slot personalizado (slot con varios valores)	Compruebe si uno de los ingredientes es cebolla	`{toppings} CONTAINS "Onion"`
Slot personalizado (slot con varios valores)	El número de ingredientes es superior a 3	`fn.COUNT({topping}) > 2`
`AMAZON.AlphaNumeric`	`bookingID` es ABC123	`{bookingID} = "ABC123"`
`AMAZON.Number`	el valor del slot de edad es superior a 30	`{age} > 30`
`AMAZON.Number`	el valor del slot de edad es igual a 10	`{age} = 10`
`AMAZON.Date`	valor del slot `dateOfBirth` antes de 1990	`{dateOfBirth} < "1990-10-01"`

Tipo de valor	Caso de uso	Expresión condicional
AMAZON.State	el valor del slot destinationState es igual a Washington	{destinationState} = "washington"
AMAZON.Country	el valor del slot destinationCountry es no Estados Unidos	{destinationCountry} != "united states"
AMAZON.FirstName	el valor del slot firstName es John	{firstName} = "John"
AMAZON.PhoneNumber	el valor del slot phoneNumber es 716767891932	{phoneNumer} = 716767891932
AMAZON.Percentage	Compruebe si el valor del slot porcentual es mayor o igual a 78	{percentage} >= 78
AMAZON.EmailAddress	el valor del slot emailAddress es userA@hmail.com	{emailAddress} = "userA@hmail.com"
AMAZON.LastName	el valor del slot lastName es Doe	{lastName} = "Doe"
AMAZON.City	El valor del slot de la ciudad es igual a Seattle	{city} = "Seattle"
AMAZON.Time	La hora es después de las 8 p. m.	{time} > "20:00"
AMAZON.StreetName	el valor del slot streetName es Boren Avenue	{streetName} = "boren avenue"
AMAZON.Duration	el valor del slot travelDuration es inferior a 2 horas	{travelDuration} < P2H

Tipo de valor	Caso de uso	Expresión condicional
Modo de entrada	El modo de entrada es voz	`$.inputMode = "Speech"`
Transcripción de entrada	La transcripción de entrada equivale a «Quiero una pizza grande»	`$.inputTranscript = "I want a large pizza"`
Atributos de sesión	compruebe el atributo customer_subscription_type	`[customer_subcription_type] = "yearly"`
Atributo de solicitud	compruebe el indicador retry_enabled	`((retry_enabled)) = "TRUE"`
Respuesta de Kendra	La respuesta de Kendra contiene preguntas frecuentes	`fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1)))`
Expresión condicional con transcripciones	Expresiones condicionales que utilizan la ruta JSON de transcripciones	`$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5`
Establecer atributos de sesión	Establezca los atributos de sesión mediante la ruta JSON de las transcripciones y los valores de los slots.	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`

Tipo de valor	Caso de uso	Expresión condicional
Establecer valores de slot	Establezca los valores de slot utilizando atributos de sesión y la ruta JSON de las transcripciones.	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName`hace referencia al nombre de una ranura en el bot Amazon Lex V2. Si el slot no está resuelto (nulo) o si no existe, las asignaciones se ignoran durante el tiempo de ejecución. `sessionAttribute` hace referencia al nombre del atributo de sesión que establece el cliente en el momento de la compilación.

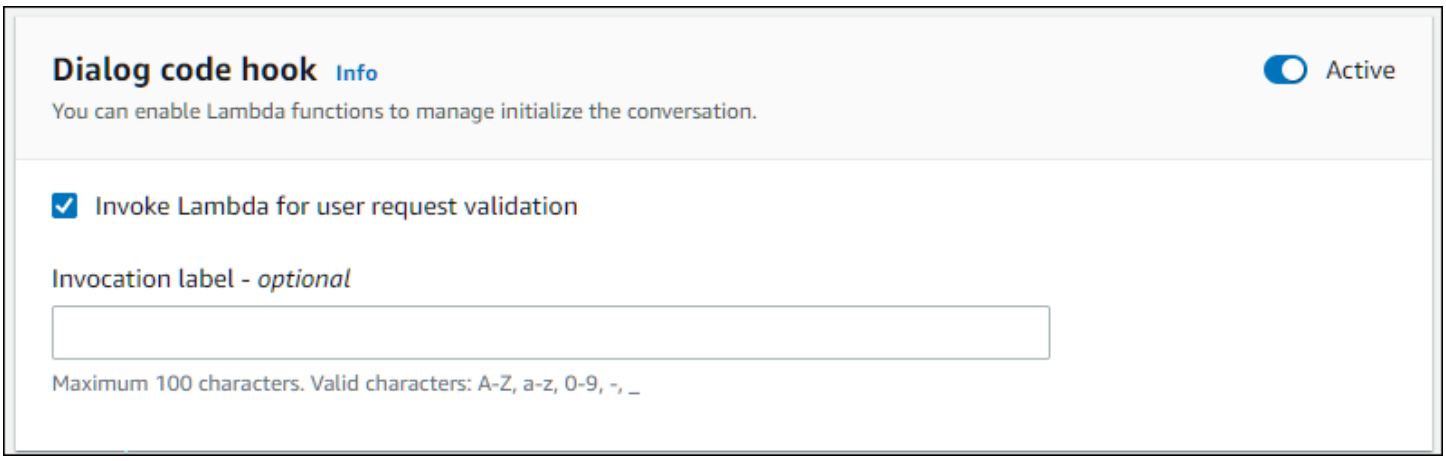
Invocar el enlace de código de diálogo

En cada paso de la conversación, cuando Amazon Lex V2 envíe un mensaje al usuario, podrá utilizar una función Lambda como siguiente paso de la conversación. Puede utilizar la función para implementar la lógica empresarial en función del estado actual de la conversación.

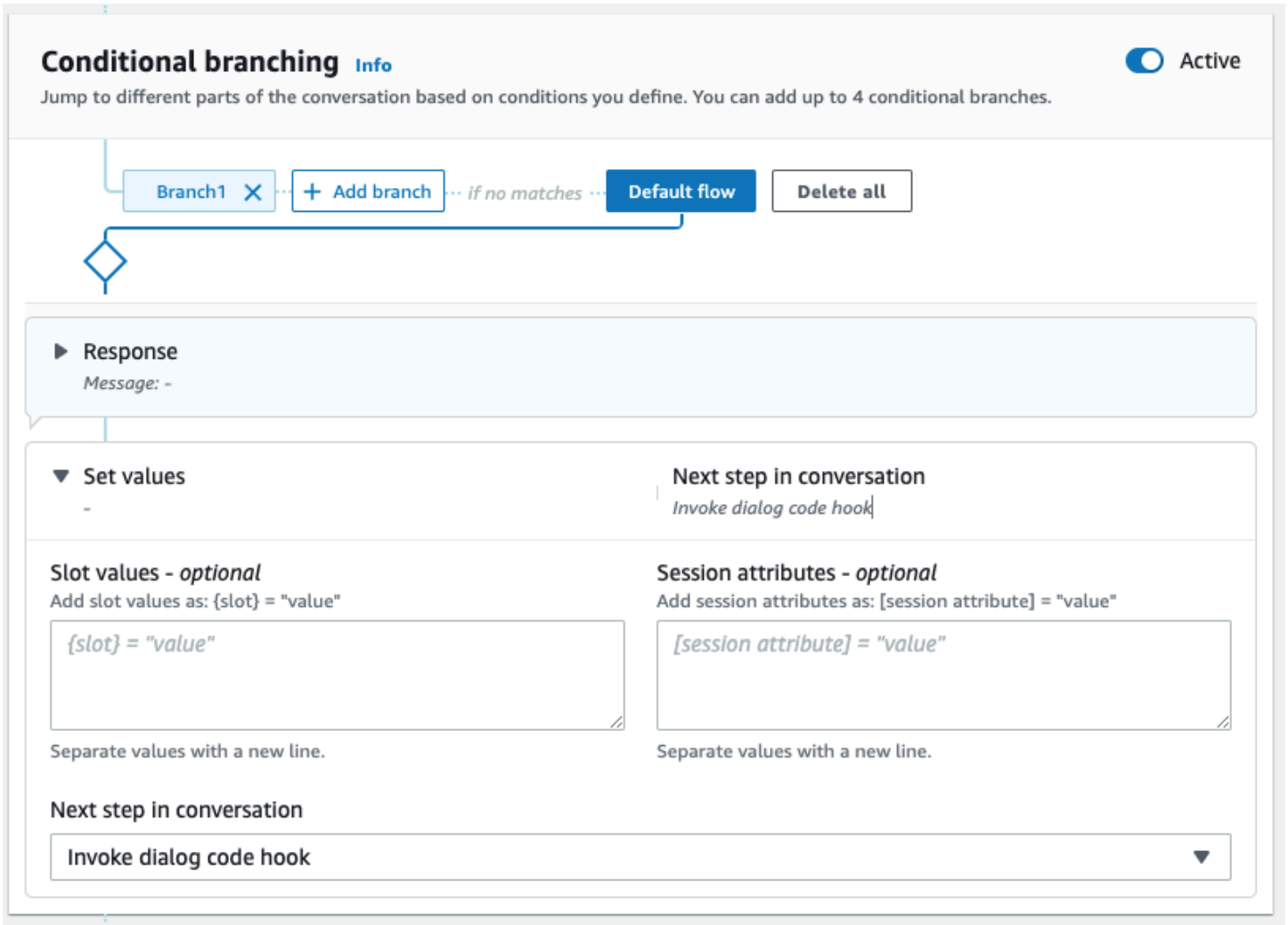
La función de Lambda que se ejecuta está asociada al alias del bot que está utilizando. Para invocar la función de Lambda en todos los enlaces de código de diálogo en su intención, debe seleccionar Utilizar una función de Lambda para inicializar y validar la intención. Para obtener más información sobre elegir una función de Lambda, consulte Creación de una AWS Lambda función para su bot Amazon Lex V2.

Para utilizar una función de Lambda, se deben realizar dos pasos. En primer lugar, debe activar el enlace de código de diálogo en cualquier momento de la conversación. En segundo lugar, debe configurar el siguiente paso de la conversación para utilizar el enlace de código de diálogo.

En la imagen siguiente, se muestra el enlace de código de diálogo activado.

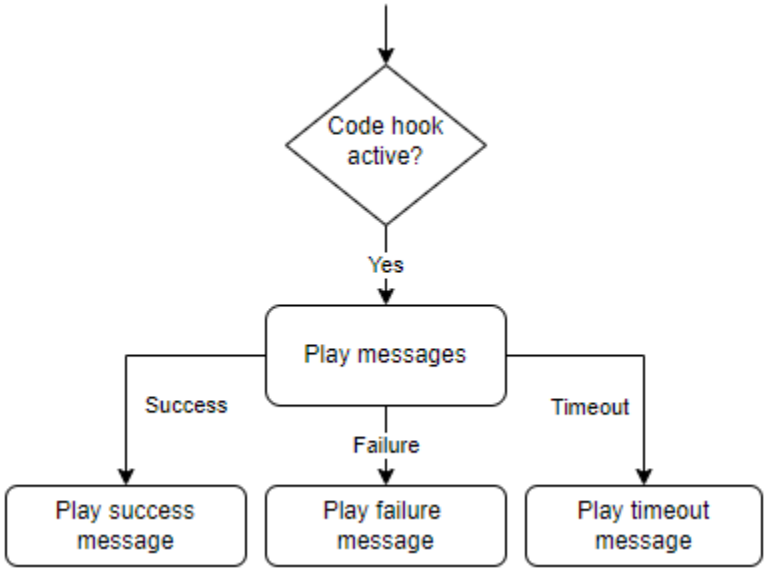


A continuación, establezca el enlace de código como la siguiente acción del paso de la conversación. Para ello, configure el siguiente paso de la conversación para invocar el enlace de código de diálogo. La siguiente imagen muestra una rama condicional en la que el siguiente paso para la ruta predeterminada de la conversación es invocar el enlace de código de diálogo.



Cuando los enlaces de código están activos, puede configurar tres respuestas para devolverlas al usuario:

- Éxito: se envía cuando la función de Lambda se completó correctamente.
- Error: se envía si ocurrió un problema al ejecutar la función de Lambda o si la función de Lambda devolvió un valor `intent.state` de `Failed`.
- Tiempo de espera agotado: se envía si la función de Lambda no se completó en el período de tiempo de espera configurado.



Seleccione el enlace de código de diálogo Lambda y, a continuación, seleccione Opciones avanzadas para ver las tres opciones de respuestas que corresponden a la invocación de la función de Lambda. Puede establecer valores, configurar los siguientes pasos y aplicar las condiciones correspondientes a cada respuesta para diseñar el flujo de la conversación. En ausencia de una condición o de un siguiente paso explícito, Amazon Lex V2 decide el siguiente paso en función del estado actual de la conversación.

En la página Opciones avanzadas, también puede optar por habilitar o deshabilitar la invocación de la función de Lambda. Cuando la función está habilitada, el enlace de código de diálogo se invoca con una invocación a Lambda, seguida del mensaje de éxito, error o tiempo de espera agotado basado en los resultados de la invocación a Lambda. Cuando la función está deshabilitada, Amazon Lex V2 no ejecuta la función de Lambda y continúa como si el enlace de código de diálogo se hubiera realizado correctamente.

También puede establecer una etiqueta de invocación que se envíe a la función de Lambda cuando se invoque mediante este mensaje. Puede usar esto para ayudar a identificar la sección de la función de Lambda que debe ejecutarse.

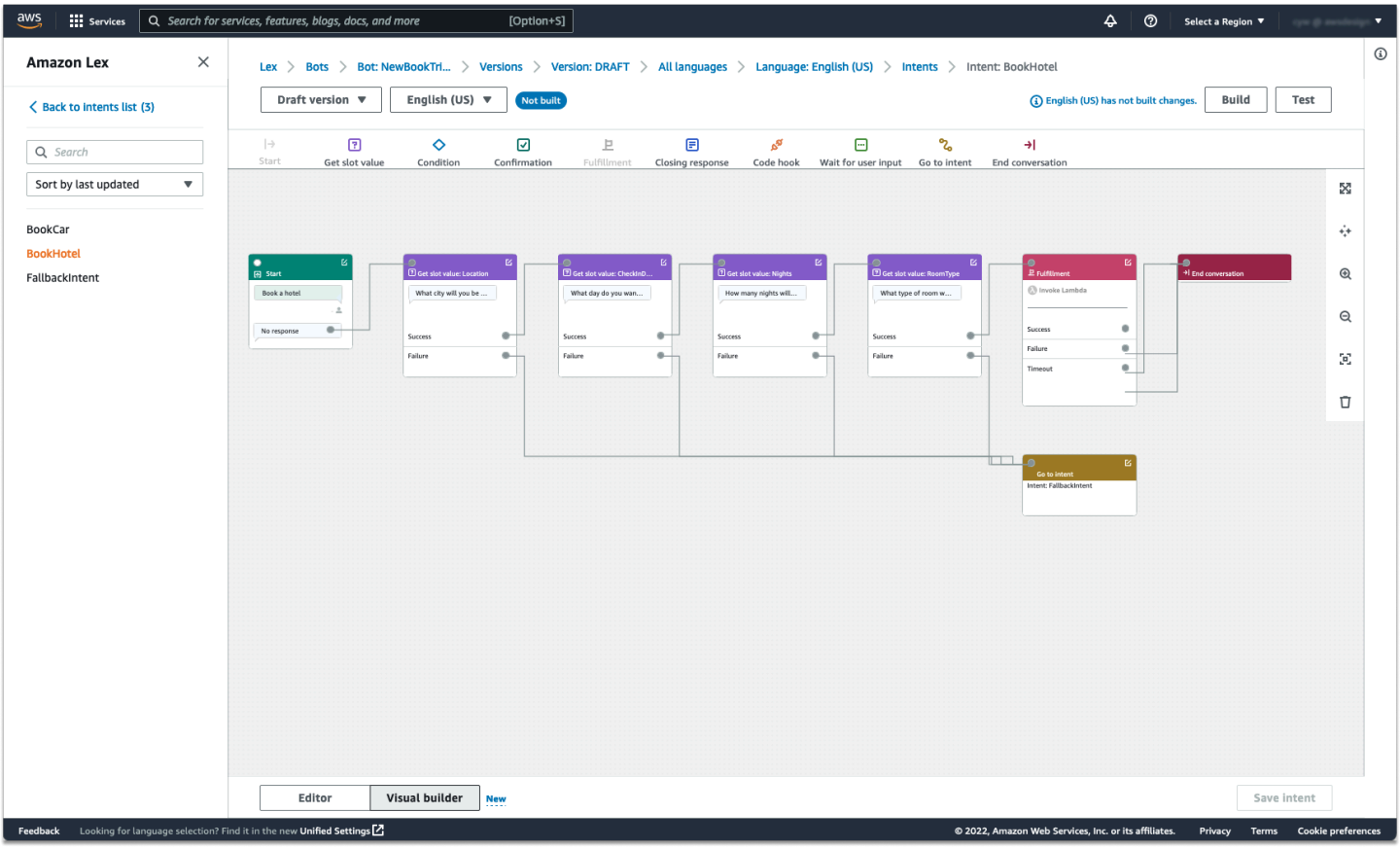
> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

Uso del generador visual de conversaciones

El generador visual de conversaciones es un generador de conversaciones de arrastrar y soltar para diseñar y visualizar fácilmente las rutas de conversación mediante el uso de intenciones en un entorno visual enriquecido.

Acceder al generador visual de conversaciones

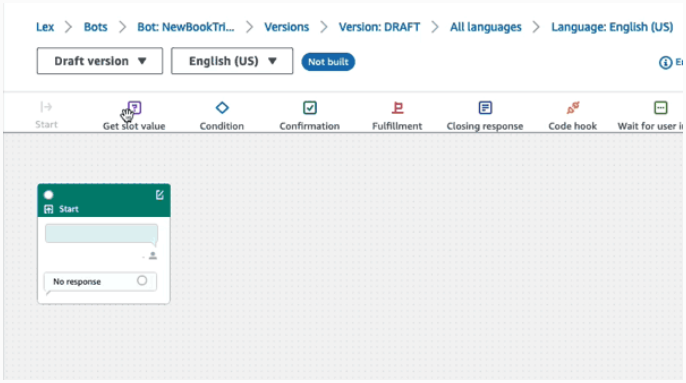
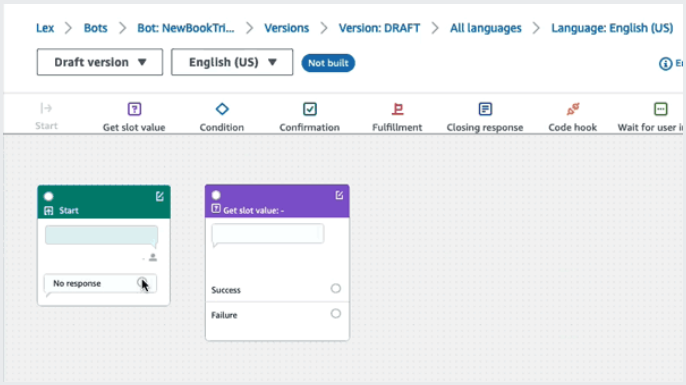
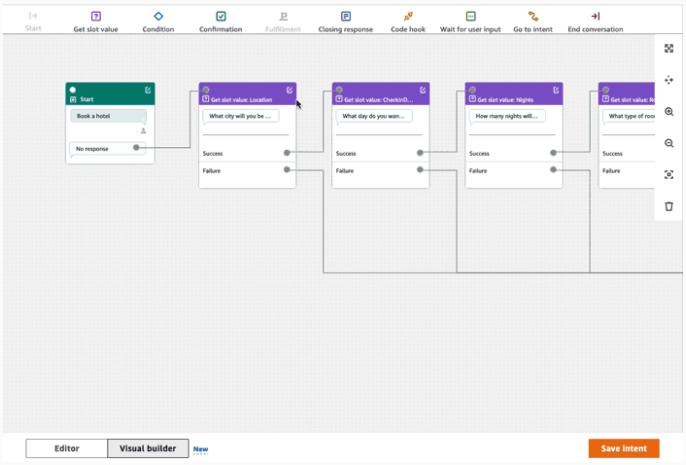
1. En la consola de Amazon Lex V2, seleccione un bot y seleccione Intenciones en el panel de navegación izquierdo.
2. Vaya al editor de intenciones de una de las siguientes formas:
   - Seleccione Añadir intención en la esquina superior derecha de la sección Intenciones y, a continuación, seleccione añadir una intención vacía o una intención integrada.
   - Seleccione el nombre de una intención en la sección Intenciones.
3. En el editor de intenciones, seleccione Generador visual en el panel de la parte inferior de la pantalla para acceder al generador de conversaciones visual.
4. Para volver a la interfaz del editor de intenciones del menú, seleccione Editor.

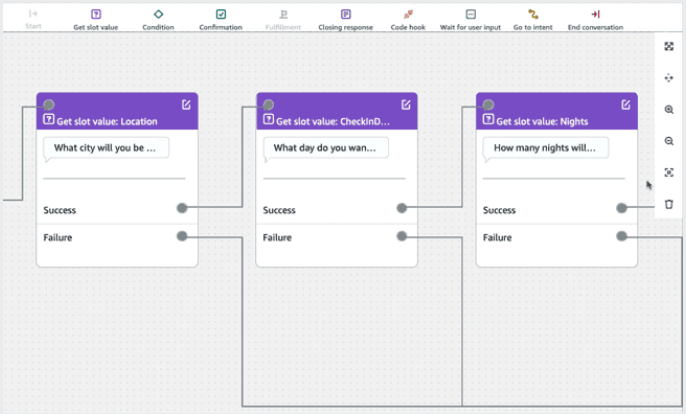
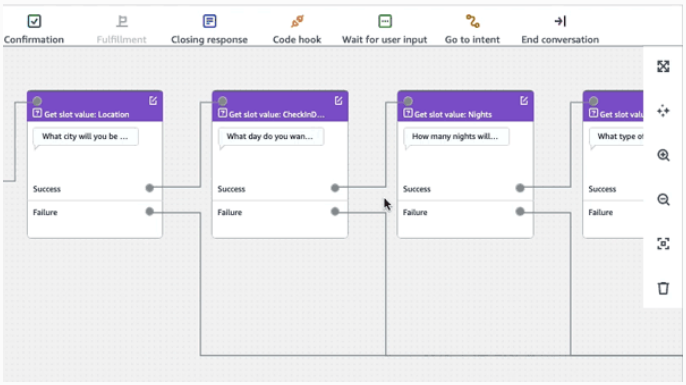
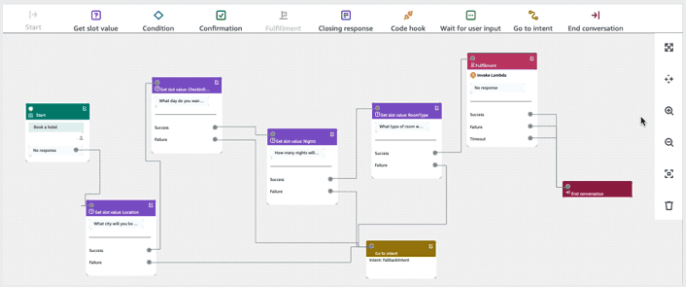


El generador visual de conversaciones ofrece una interfaz de usuario más intuitiva con la capacidad de visualizar y modificar el flujo de la conversación. Al arrastrar y soltar los bloques, puede ampliar un flujo existente o reordenar los pasos de la conversación. Puede desarrollar un flujo de conversación con ramificaciones complejas sin necesidad de escribir ningún código Lambda.

Este cambio ayuda a disociar el diseño del flujo de conversación de otras lógicas empresariales en Lambda. El generador visual de conversaciones se puede usar junto con el editor de intenciones existente y se puede usar para crear flujos de conversación. Sin embargo, se recomienda utilizar la vista del editor visual para flujos de conversación más complejos.

Cuando guarda una intención, Amazon Lex V2 puede conectar automáticamente las intenciones cuando determina que hay conexiones perdidas, Amazon Lex V2 sugiere una conexión o puede seleccionar su propia conexión para el bloqueo.

Acción	Ejemplo
Añadir un bloque al espacio de trabajo	
Hacer una conexión entre bloques	
Abrir el panel de configuración en un bloque	

Acción	Ejemplo
Zoom para ajustar	
Eliminar un bloque del flujo de conversación	
Limpiar automáticamente el espacio de trabajo	

Terminología:

Bloque: la unidad básica de creación de un flujo de conversación. Cada bloque tiene una funcionalidad específica para gestionar diferentes casos de uso de una conversación.

Puerto: cada bloque contiene puertos, que se pueden usar para conectar un bloque a otro. Los bloques pueden contener puertos de entrada y puertos de salida. Cada puerto de salida representa una variación funcional particular de un bloque (como errores, tiempos de espera agotados o éxitos).

Periferia: una periferia es una conexión entre el puerto de salida de un bloque y el puerto de entrada de otro bloque. Forma parte de una rama del flujo de una conversación.

Flujo de conversación: conjunto de bloques conectados por periferias que describe las interacciones a nivel de intención con un cliente.

Bloques

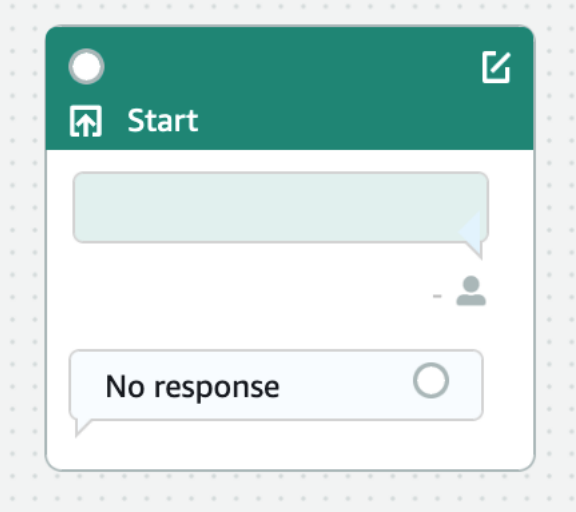
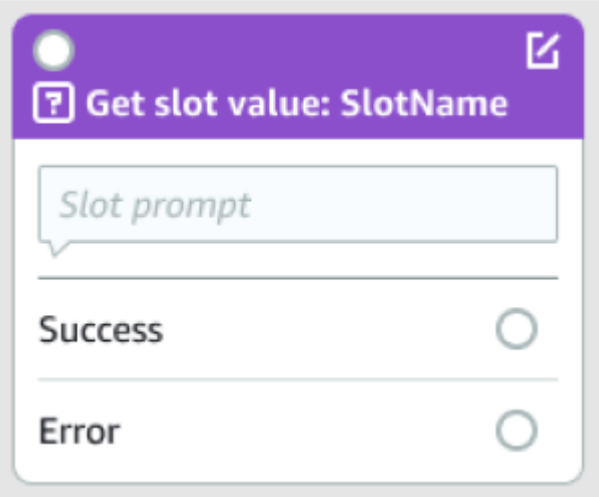
Los bloques son los bloques de creación de un diseño de flujo de conversación. Representan diferentes estados dentro de la intención, que van desde el inicio de la intención hasta la entrada del usuario y su cierre.

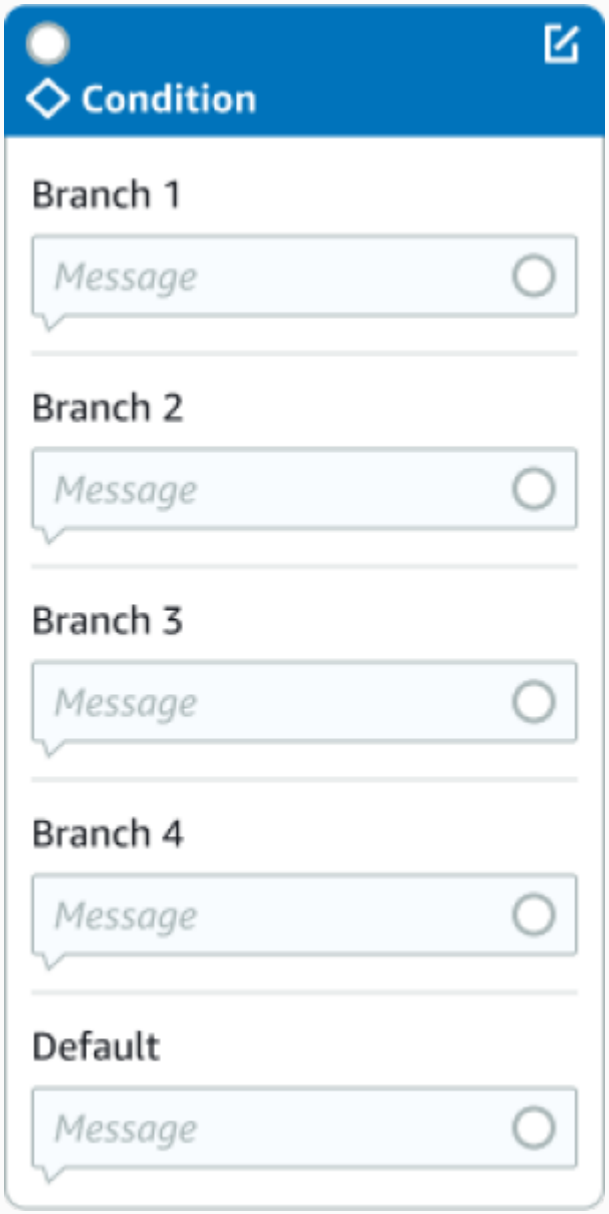
Cada bloque tiene un punto de entrada y uno o varios puntos de salida según el tipo de bloque. Cada punto de salida se puede configurar con el mensaje correspondiente a medida que la conversación avanza por los puntos de salida. En el caso de los bloques con varios puntos de salida, los puntos de salida se refieren al estado correspondiente al nodo. Para un nodo de condición, los puntos de salida representan las diferentes condiciones.

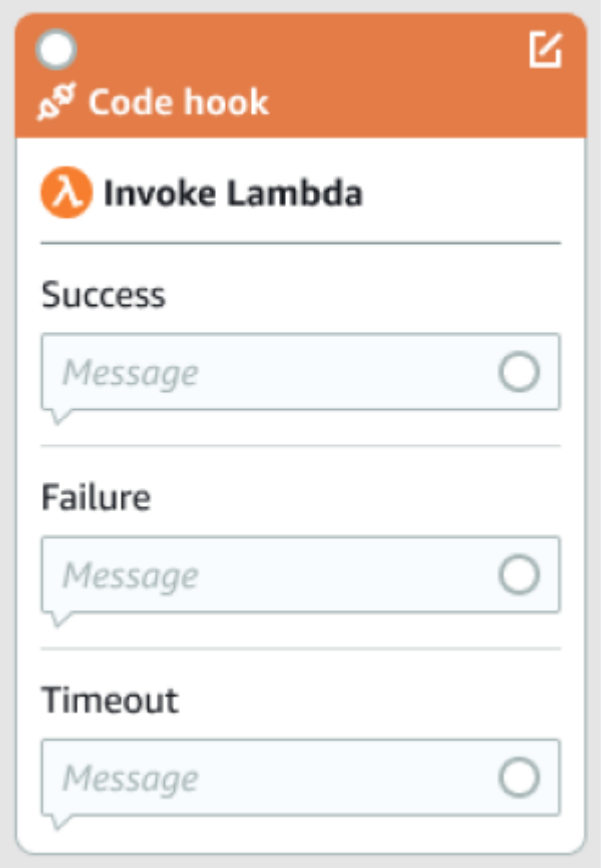
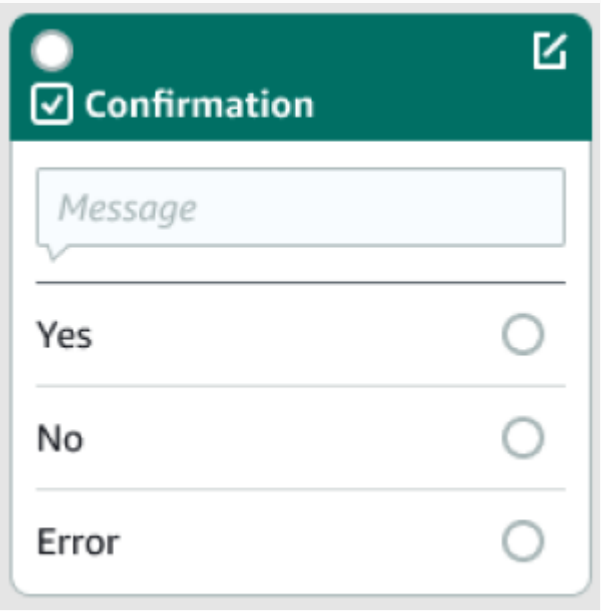
Cada bloque tiene un panel de configuración, que se abre haciendo clic en el icono de Edición situado en la esquina superior derecha del bloque. El panel de configuración contiene campos detallados que se pueden configurar para que se correspondan con cada bloque.

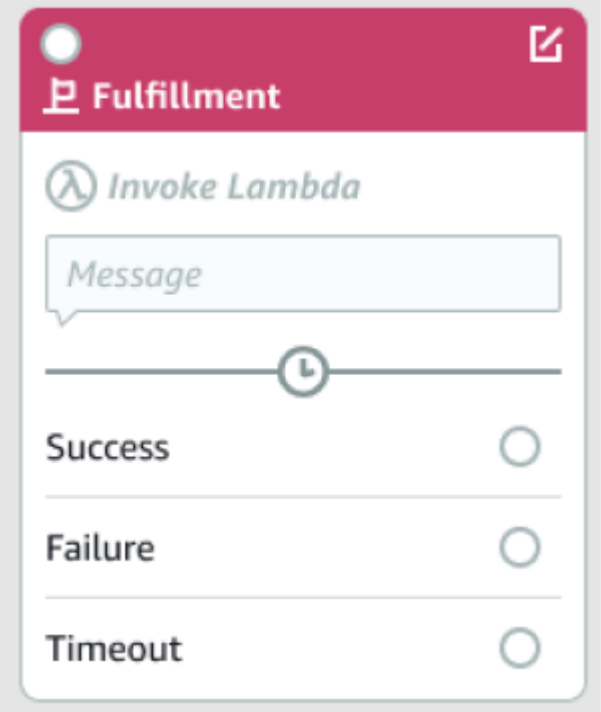
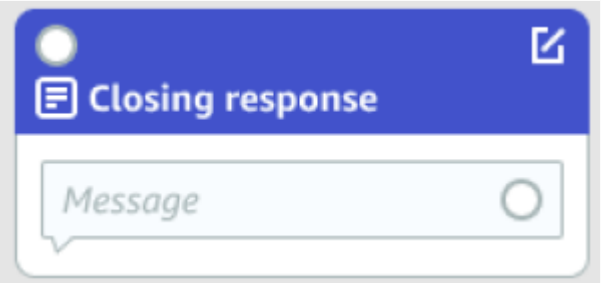
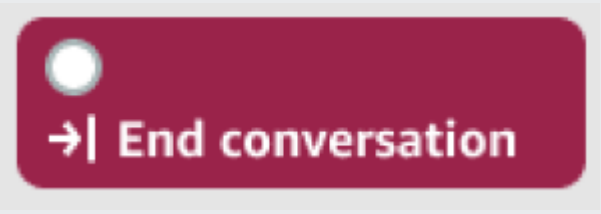
Las indicaciones y los mensajes del bot se pueden configurar directamente en el nodo arrastrando un nuevo bloque, o se pueden modificar en el panel derecho, junto con otros atributos del bloque.

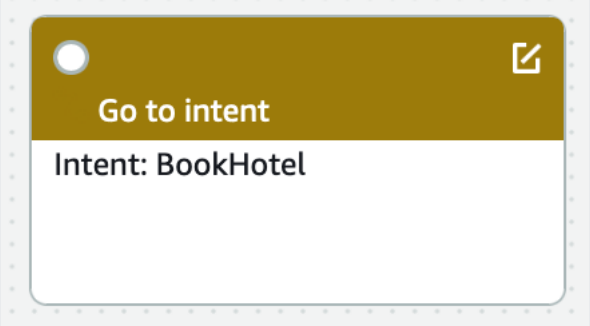
Tipos de bloques: estos son los tipos de bloques que puede usar con el generador visual de conversaciones.

Tipo de block	Bloque
Inicio: el bloque raíz o el primer bloque del flujo de la conversación. Este bloque también se puede configurar de manera que el bot pueda enviar una respuesta inicial (mensaje en el que se indique que se ha reconocido la intención). Para obtener más información, consulte Respuesta inicial.	
Obtener el valor de un slot: este bloque intenta obtener el valor de un solo slot. Este bloque está configurado para esperar a que el cliente responda a la solicitud de obtención de slot. Para obtener más información, consulte Ranuras.	

Tipo de block	Bloque
Condición: este bloque contiene condicion ales. Contiene hasta 4 ramas personalizadas (con condiciones) y una rama predeterminada. Para obtener más información, consulte Añadir condiciones a las conversaciones ramificadas.	

Tipo de block	Bloque
Enlace de código de diálogo: este bloque gestiona la invocación de la función de Lambda de diálogo. Este bloque contiene las respuesta s de los bots basadas en el éxito, el fallo o la caducidad de la función de Lambda de diálogo. Para obtener más información, consulte Invocar el enlace de código de diálogo.	
Confirmación: este bloque consulta al cliente antes de cumplir la intención. Contiene respuestas de bots basadas en las respuesta s del cliente que dicen sí o no a la solicitud de confirmación. Para obtener más información, consulte Confirmación.	

Tipo de block	Bloque
Cumplimiento: este bloque gestiona el cumplimiento de la intención, normalmente después de la obtención de slots. Se puede configurar para invocar funciones de Lambda, así como para responder con mensajes, si el cumplimiento se realiza correctamente o no. Para obtener más información, consulte Cumplimiento.	
Respuesta de cierre: este bloque permite al bot responder con un mensaje antes de finalizar la conversación. Para obtener más información, consulte Respuesta de cierre.	
Finalizar la conversación: este bloque indica el final del flujo de la conversación.	
Espera a la entrada del usuario: este bloque se puede usar para capturar las opiniones del cliente y cambiar a otra intención en función del enunciado.	

Tipo de block	Bloque
Ir a la intención: este bloque se puede usar para ir a una nueva intención o para obtener directamente un slot específico de esa intención.	

Tipos de puertos

Todos los bloques contienen un puerto de entrada, que se utiliza para conectar sus bloques principales. La conversación solo puede fluir al puerto de entrada de un bloque en particular desde el puerto de salida de su bloque principal. Sin embargo, los bloques pueden no tener, tener uno o muchos puertos de salida. Los bloques sin ningún puerto de salida significan el final del flujo de conversación en la intención actual (`GoToIntent`, `EndConversation`, `WaitForUserInput`).

Reglas de diseño de intenciones:

- Todos los flujos de una intención comienzan con el bloque de inicio.
- Los mensajes correspondientes a cada punto de salida son opcionales.
- Puede configurar los bloques para establecer los valores correspondientes a cada punto de salida en el panel de configuración.
- Solo puede existir un único bloque de inicio, confirmación, cumplimiento y cierre en un mismo flujo dentro de una intención. Es posible que existan varios bloques de condiciones, como el enlace de códigos de diálogo, la obtención de valores de los slots, la finalización de la conversación, la transferencia y la espera a que el usuario introduzca los valores.
- Un bloque de condiciones no puede tener una conexión directa con un bloque de condiciones. Lo mismo se aplica al enlace de código de diálogo.
- Se permiten flujos circulares de tres bloques, pero no se permite un conector entrante a Iniciar intención.
- Un slot opcional no tiene un conector de entrada ni una conexión de salida y se utiliza principalmente para capturar los datos presentes durante la obtención de la intención. Todas los demás slots que formen parte de la ruta de conversación deben ser obligatorios.

Bloques:

- El bloque de inicio debe tener una periferia saliente.
- Cada bloque para obtener el valor del slot debe tener una periferia de salida desde el puerto de éxito, si se requiere el slot.
- Cada bloque de condiciones debe tener una periferia de salida desde cada rama si el bloque está activo.
- Un bloque de condiciones no puede tener más de un elemento principal.
- Un bloque de condiciones activo debe tener una periferia entrante.
- Cada bloque de enlace de código activo debe tener una periferia de salida de cada puerto: éxito, error y tiempo de espera agotado.
- Un bloque de enlace de código activo debe tener una periferia entrante.
- Un bloque de confirmación activo debe tener una periferia entrante.
- Un bloque de cumplimiento activo debe tener una periferia entrante.
- Un bloque de cierre activo debe tener una periferia entrante.
- Un bloque de condiciones debe tener al menos una rama no predeterminada.
- Un bloque de intención debe tener una intención especificada.

Periferias:

- Un bloque de condiciones no puede tener una conexión directa con otro bloque de condiciones.
- Un bloque de enlace de códigos no se puede conectar a otro bloque de enlace de códigos.
- Un bloque de condiciones solo se puede conectar a cero o a un bloque de enlace de código.
- La conexión (enlace de código -> condición -> enlace de código) no es válida.
- Un bloque de cumplimiento no puede tener un bloque de enlace de códigos como elemento secundario.
- Un bloque de condiciones, que es un elemento secundario del bloque de cumplimiento, no puede tener un bloque de enlace de códigos secundario.
- Un bloque de cierre no puede tener un bloque de enlace de códigos como elemento secundario.
- Un bloque de condiciones que es un elemento secundario del bloque de cierre no puede tener un bloque de enlace de códigos secundario.
- Un bloque de inicio, confirmación u obtención de valores de slots no puede tener más de un bloque de enlace de códigos en su cadena de dependencias.

> **Note**
>
> El 17 de agosto de 2022, Amazon Lex V2 publicó un cambio en la forma en que se gestionan las conversaciones con el usuario. Este cambio le da más control sobre la ruta que sigue el usuario a lo largo de la conversación. Para obtener más información, consulte Cambios en los flujos de conversación en Amazon Lex V2. Los bots creados antes del 17 de agosto de 2022 no admiten mensajes de enlace de código de diálogo, ni permiten establecer valores, configurar los pasos siguientes ni añadir condiciones.

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 consola de administración de AWS 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 menú de la izquierda, seleccione el idioma y, a continuación, seleccione Intenciones.
4. Seleccione Añadir intención y, a continuación, seleccione Usar intención integrada.
5. En Intención integrada, seleccione la intención que quiere usar.
6. Asigne un nombre a la intención y, a continuación, seleccione Añadir.
7. Use el editor de intenciones para configurar la intención según sea necesario para su bot.

Temas

- AMAZON.BedrockAgentIntent

- AMAZON.CancelIntent
- AMAZON.FallbackIntent
- AMAZON.HelpIntent
- AMAZON.KendraSearchIntent
- AMAZON.PauseIntent
- AMAZON.QnAIntent
- AMAZON.QnAIntent (multiple use support)
- AMAZON.QinConnectIntent
- AMAZON.RepeatIntent
- AMAZON.ResumeIntent
- AMAZON.StartOverIntent
- AMAZON.StopIntent

AMAZON.BedrockAgentIntent

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.
> 2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Activa los agentes de Amazon Bedrock que se definen con la intención de responder a las solicitudes de los clientes y activa los flujos de trabajo agénticos para realizar la tarea definida. Esta característica está disponible en todas las configuraciones regionales compatibles con Amazon Lex V2 y en todas las regiones comerciales en las que estén presentes Amazon Lex V2 y agentes de Amazon Bedrock.

Si esta intención prevaleceFallbackIntent, la intención se activa cuando un enunciado no está clasificado en ninguna de las demás intenciones presentes en el bot; de lo contrario, solo se activa

cuando un enunciado se clasifica en esta intención. Es importante tener en cuenta que esta intención no se activará en el caso de expresiones omitidas cuando se obtenga un valor de espacio.

Cuando su bot de Amazon Lex V2 la reconozca, la `AMAZON.BedrockAgentIntent` activa el `BedrockAgent` o la `BedrockKnowledgeBase` definidos para responder al cliente. Si utiliza agentes para Amazon Bedrock, la conversación permanece en la `BedrockAgentIntent` y las solicitudes de los usuarios se transmiten a los agentes hasta que el agente para Amazon Bedrock determine que la conversación haya `FINISH`. Solo entonces, Amazon Lex V2 asumirá el control de la conversación y seguirá los siguientes pasos definidos en la `AMAZON.BedrockAgentIntent`.

Responde a las preguntas de los clientes usando los agentes y bases de conocimiento de Amazon Bedrock para responder a las preguntas de los clientes y proporcionar respuestas detalladas.

> **Warning**
>
> No puedes usar expresiones sin muestras, `AMAZON.BedrockAgentIntent` `AMAZON.QnAIntent` sin expresiones de muestra y `AMAZON.KendraSearchIntent` en la misma configuración regional del bot.

Si selecciona esta intención, configure los siguientes campos y, a continuación, seleccione Agregar para agregar la intención.

- ID del agente de Amazon Bedrock: es el identificador del agente de Amazon Bedrock. Elija el agente de Bedrock que desea usar.
- ID del alias del agente de Amazon Bedrock: es el identificador del alias del agente de Amazon Bedrock.

> **Important**
>
> Al crear el agente de Amazon Bedrock para usarlo con Amazon Lex V2, compruebe que la Entrada del usuario en Configuración adicional sea `ENABLED`. Esta configuración es fundamental para que los agentes puedan hacer preguntas aclaratorias o de seguimiento, y permite a Amazon Lex V2 delegar en los agentes la tarea correspondiente.

(Opcional) También puedes añadir un BedrockAgentIntent con estas opciones:

- Modelo Amazon Bedrock: elija el modelo de proveedor y base que se utilizará para este propósito. Actualmente, se admiten algunos modelos Anthropic Claude.
- Base de conocimiento de Amazon Bedrock: si elige esta opción, especifique el ID de la base de conocimiento de Amazon Bedrock. Puede buscar el ID en la página de detalles de la base de conocimiento de Amazon Bedrock en la consola o enviando una solicitud `GetKnowledgeBase`.

Las respuestas del se BedrockAgentIntent almacenarán en los atributos de sesión y solicitud, tal y como se muestra a continuación:

- `x-amz-lex:bedrock-agent-search-response`: respuesta del agente de Amazon Bedrock a la pregunta o al enunciado.
- `x-amz-lex:bedrock-knowledge-base-search-response-source`: señala al documento o la lista de documentos utilizados para generar la respuesta si se utiliza la configuración de la base de conocimiento de Amazon Bedrock.
- `x-amz-lex:bedrock-agent-action-group-invocation-input`: objeto que contiene los valores de entrada recopilados por el grupo de acciones de los agentes. Para obtener más información sobre los grupos de acción de los agentes, consulte ActionGroupInvocationInput.
- `x-amz-lex:bedrock-agent-knowledge-base-lookup-input`: objeto que contiene los detalles relacionados con la búsqueda en la base de conocimiento de Amazon Bedrock.
- `x-amz-lex:bedrock-agent-agent-collaborator-details`: objeto que contiene detalles de las entradas y salidas de los subagentes que se han invocado como parte de las invocaciones de colaboración entre múltiples agentes.

Para obtener más información, consulte Uso BedrockAgentIntent de un agente Bedrock en Amazon Lex.

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 V2 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 V2 invoca la intención alternativa para gestionar la respuesta.

El tipo de `AMAZON.FallbackIntent` intención integrado se añade al bot automáticamente cuando se crea un bot mediante la consola o cuando se añade una configuración regional a un bot mediante esta operación. CreateBotLocale

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.

Amazon Lex V2 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 V2 no puede determinar qué intención debe invocar.

La intención alternativa se invoca cuando:

- 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.

No puede añadir lo siguiente a una intención alternativa:

- Enunciados
- Ranuras
- Una solicitud de confirmación

Uso de una función de Lambda con una intención alternativa

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

- Devuelve la información de la intención a la aplicación cliente.
- Llama a la función Lambda de validación y cumplimiento del alias. 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 `fulfillmentCodeHook` de la operación CreateIntent.

Si utiliza la función de Lambda con 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.

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 V2 no puede inferir 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 su 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 Cómo establecer atributos de sesión para su bot Lex V2.

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 V2 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 machine-learning-based búsqueda 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 V2 la llamará siempre que no pueda determinar el enunciado del usuario en un slot o una intención. Si no hay respuesta de Amazon Kendra, la conversación continuará tal y como está configurada en el bot.

> **Note**
>
> Actualmente, Amazon Lex V2 no admite la `AMAZON.KendraSearchIntent` durante la obtención del slot. Si Amazon Lex V2 no puede determinar el enunciado del usuario para un slot, llama al `AMAZON.FallbackIntent`.

Cuando use la `AMAZON.KendraSearchIntent` con la `AMAZON.FallbackIntent` en el mismo bot, Amazon Lex V2 utiliza las intenciones de la siguiente manera:

1. Amazon Lex V2 llama a la `AMAZON.KendraSearchIntent`. La intención llama a la operación `Query` de Amazon Kendra.
2. Si Amazon Kendra devuelve una respuesta, Amazon Lex V2 muestra el resultado al usuario.
3. Si no hay respuesta por parte de Amazon Kendra, Amazon Lex V2 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 V2 reconoce, como llenar un valor de slot 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 V2 reconozca, Amazon Lex V2 hará otra llamada a la operación `Query`.
4. Si tras un número establecido de nuevos intentos no hay ninguna respuesta, Amazon Lex V2 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 V2 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 V2 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. 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 utilizando la función de Lambda. Puede crear una solicitud de consulta de Amazon Kendra completa para que Amazon Lex V2 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 V2 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 V2 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 utilizar la `AMAZON.KendraSearchIntent` intención, debe utilizar una función que proporcione políticas de AWS Identity and Access Management (IAM) que permitan a Amazon Lex V2 asumir una función en tiempo de ejecución que tenga permiso para denominar la intención de Amazon `Query` Kendra. La configuración de IAM que utilice dependerá de si la ha creado `AMAZON.KendraSearchIntent` con la consola Amazon Lex V2 o con un AWS SDK o 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 V2 para que llame a Amazon Kendra o si prefiere utilizar un rol específico para llamar a la operación `Query` de Amazon Kendra. Cuando usa la AWS CLI o un SDK para crear la intención, debe usar un rol específico para llamar a la `Query` operación.

Asociación de permisos

Puede utilizar la consola para asociar permisos que permitan al rol vinculado al servicio de Amazon Lex V2 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 V2 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.

JSON

```
{
```

```
"Version":"2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:AttachRolePolicy",
            "iam:PutRolePolicy",
            "iam:GetRolePolicy"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexBots*"
    },
    {
        "Effect": "Allow",
        "Action": "iam:ListRoles",
        "Resource": "*"
    }
]
}
```

Especificación de un rol

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

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 AWS STS condición en la Guía del usuario de AWS Identity and Access Management.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/role"
        }
```

```
    ]
}
```

El rol en tiempo de ejecución que Amazon Lex V2 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 usar 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.
2. En la AWS CLI, ejecute el siguiente comando para crear la política de IAM para ejecutar la operación de Amazon `Query` 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 V2 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 de tiempo de ejecución para AMAZON.KendraSearchIntent

1. Inicie sesión en la consola de administración de AWS 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, seleccione una de las opciones siguientes:
   - Para actualizar el rol vinculado al servicio de Amazon Lex V2 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 Cómo establecer atributos de solicitud para su bot Lex V2. Para obtener más información acerca de los atributos de sesión, consulte Cómo establecer atributos de sesión para su bot Lex V2.

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"
            }
        }
    }
]
}
```

Usar la respuesta de búsqueda

Amazon Kendra devuelve la respuesta a una búsqueda en una respuesta a la declaración `IntentClosingSetting` de intención. La intención debe tener una declaración

`closingResponse`, a menos que una función de Lambda genere un mensaje de respuesta de conclusión.

Amazon Kendra tiene cinco tipos de respuestas.

- Las dos respuestas siguientes requieren que se configure una sección de preguntas frecuentes para su índice de Amazon Kendra. Para obtener más información, consulte Agregar preguntas y respuestas directamente a un índice.
  - `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.
- Las tres respuestas siguientes requieren que se configure un origen de datos para su índice de Amazon Kendra. Para obtener más información, consulte Crear un origen de datos.
  - `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 `closingResponse` 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:

- «He encontrado una pregunta de preguntas frecuentes para usted: ((x-amz-lex: kendra-search-response-question _answer-question-1)), y la respuesta es ((x-amz-lex: _answer-answer-1))» kendra-search-response-question
- «He encontrado un extracto de un documento útil: ((: -1))» x-amz-lex kendra-search-response-document
- «Creo que la respuesta a tus 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 V2 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 V2 con el campo `dialogAction` de la respuesta establecido en `delegate`. Amazon Lex V2 envía la consulta a Amazon Kendra y, a continuación, devuelve la respuesta a la consulta al enlace de código de cumplimentación.

Usar el enlace de código de cumplimentación en la respuesta

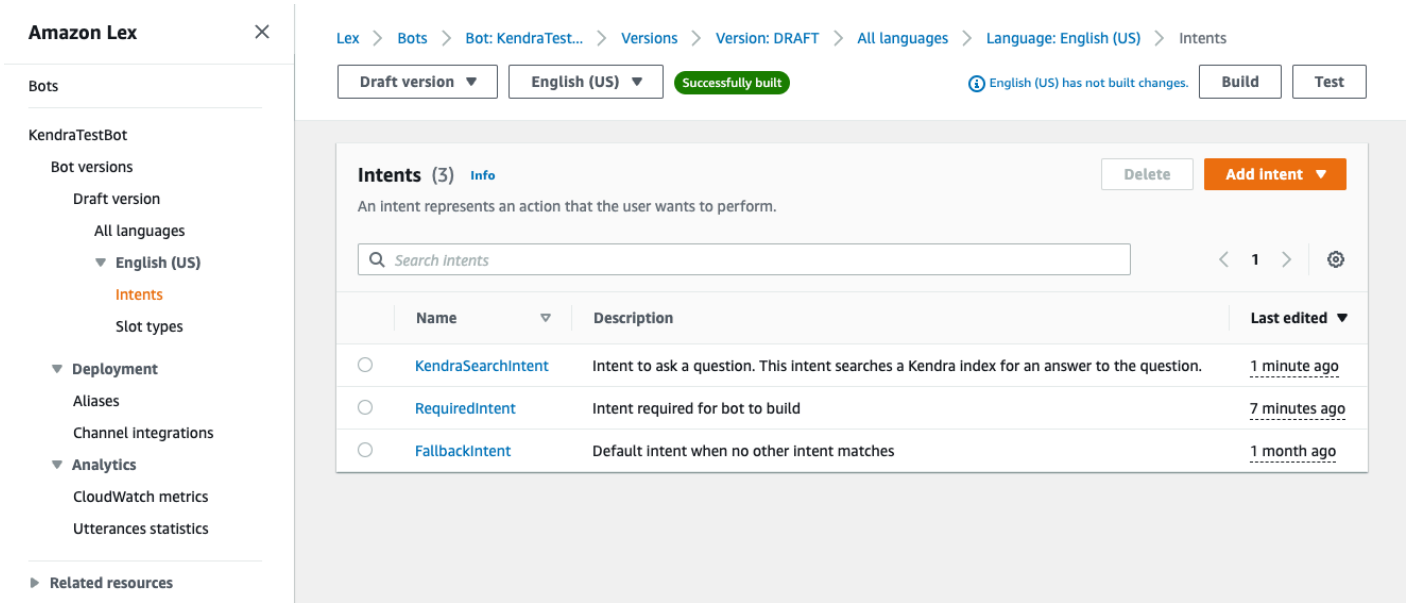
Una vez que Amazon Lex V2 envía una consulta a Amazon Kendra, la respuesta se devuelve a la función de Lambda de cumplimentación `AMAZON.KendraSearchIntent`. El evento de entrada del enlace de código contiene la respuesta completa de Amazon Kendra. Los datos de consulta tienen la misma estructura que 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 V2 utilizará la declaración `closingResponse` 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 V2 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. Este es un resumen de cómo creará su bot de preguntas frecuentes utilizando un índice de Amazon Kendra:

1. Crear un bot con el que sus clientes puedan interactuar para obtener respuestas.
2. Crear una intención personalizada. Como las `AMAZON.KendraSearchIntent` y `AMAZON.FallbackIntent` son intenciones de respaldo, su bot necesita al menos otra intención que contenga al menos un enunciado. Esta intención permitirá compilar el bot, pero no se utilizará de ninguna otra manera. Por lo tanto, su bot de preguntas frecuentes contendrá al menos tres intenciones, como se muestra en la siguiente imagen:



3. Agregar la intención `AMAZON.KendraSearchIntent` al bot y configurarlo para que funcione con el índice de Amazon Kendra.
4. Pruebe el bot realizando una consulta y verificando que los resultados de su índice de Amazon Kendra sean documentos que respondan a la consulta.

Requisitos previos

Para poder utilizar este ejemplo, primero debe crear un índice de Amazon Kendra. Para obtener más información, consulte Introducción a la consola de Amazon Kendra en la Guía para desarrolladores de Amazon Kendra. Para este ejemplo, seleccione el conjunto de datos de muestra (documentación de AWS de muestra) como origen de datos.

Para crear un bot de preguntas frecuentes

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el panel de navegación, seleccione Bots.
3. Seleccione Crear bot.
   a. Para el Método de creación, seleccione Crear un bot en blanco.
   b. En la sección Configuración del bot, asígnele un nombre que indique su finalidad, como **`KendraTestBot`**, y una descripción opcional. El nombre debe ser exclusivo en su cuenta.

c. En la sección Permisos de IAM, seleccione Crear un rol con permisos básicos de Amazon Lex. Esto creará un rol de AWS Identity and Access Management (IAM) con los permisos que Amazon Lex V2 necesita para ejecutar el bot.

d. En la sección Ley de Protección de la Privacidad en Línea para Niños (COPPA), seleccione No.

e. En las secciones Tiempo de espera de la sesión inactiva y Configuración avanzada, deje la configuración predeterminada y seleccione Siguiente.

f. Ahora se encuentra en la sección Añadir idioma al bot. En el menú de Interacción por voz, seleccione Ninguna. Esta es solo una aplicación basada en texto. No cambie la configuración predeterminada del resto de los campos.

g. Seleccione Listo. Amazon Lex V2 crea su bot y una intención predeterminada llamada NewIntent, y lo lleva a la página para configurar esta intención.

Para compilar un bot correctamente, debe crear al menos una intención independiente de la `AMAZON.FallbackIntent` y la `AMAZON.KendraSearchIntent`. Esta intención es necesaria para compilar el bot de Amazon Lex V2, pero no se usa para la respuesta de preguntas frecuentes. Esta intención debe contener al menos un ejemplo de enunciado y el enunciado no debe aplicarse a ninguna de las preguntas que haga el cliente.

Para crear la intención necesaria:

1. En la sección Detalles de la intención, dele un nombre a la intención, como **`RequiredIntent`**.
2. En la sección Ejemplos de enunciados, escriba un enunciado en el cuadro situado junto a Añadir enunciado, como **`Required utterance`**. A continuación, seleccione Añadir enunciado.
3. Seleccione Guardar intención.

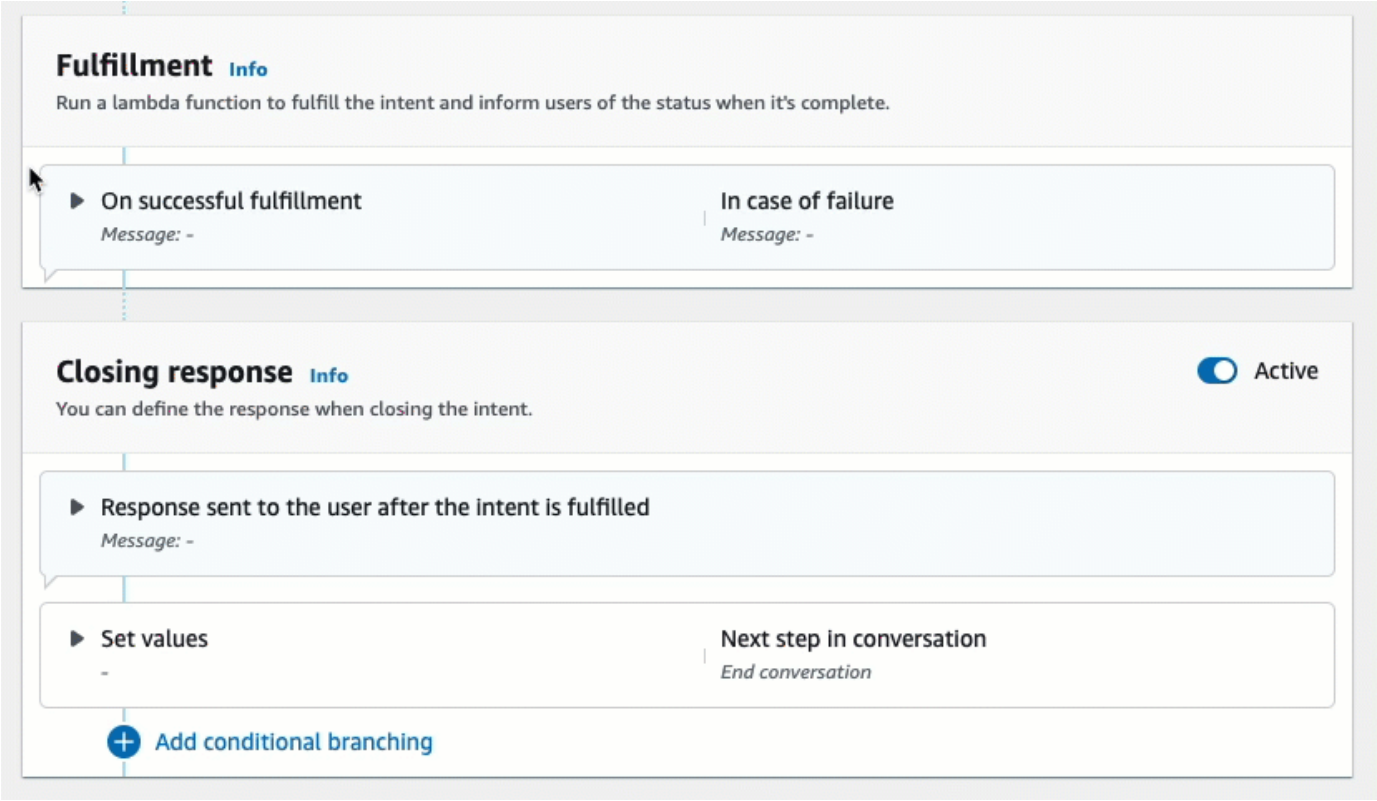
Cree la intención para buscar un índice de Amazon Kendra y los mensajes de respuesta que debe devolver.

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

1. Seleccione Volver a la lista de intenciones en el panel de navegación para volver a la página Intenciones de su bot. Seleccione Añadir intención y seleccione Usar intención integrada en el menú desplegable.
2. En el cuadro que aparece, seleccione el menú en Intención integrada. En la barra de búsqueda, escriba **`AMAZON.KendraSearchIntent`** y selecciónela de la lista.

3. Asigne a la entrada un nombre, como **`KendraSearchIntent`**.
4. En el menú desplegable Índice de Amazon Kendra, seleccione el índice donde desee buscar la intención. El índice que creó en la sección Requisitos previos debería estar disponible.
5. Seleccione Agregar.
6. En el editor de intenciones, desplácese hacia abajo hasta la sección Cumplimentación, seleccione la flecha derecha para expandir la sección y añada el siguiente mensaje en el cuadro que aparece debajo de En caso de cumplimentación exitoso:

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



Para obtener más información sobre la respuesta de búsqueda de Amazon Kendra, consulte Usar la respuesta de búsqueda.

7. Seleccione Guardar intención y, luego, Compilar para generar el bot. Cuando el bot esté listo, el banner de la parte superior de la pantalla se volverá verde y mostrará un mensaje de confirmación.

Por último, utilice la ventana de prueba de la consola para probar las respuestas del bot.

Para probar el bot de preguntas frecuentes:

1. Una vez que el bot se haya creado correctamente, seleccione Probar.
2. Introduzca **`What is Amazon Kendra?`** en la ventana de prueba de la consola. Compruebe que el bot responde con un enlace.
3. Para obtener más información sobre la configuración`AMAZON.KendraSearchIntent`, consulte AMAZON.KendraSearchIntenty KendraConfiguration.

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.QnAIntent

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.
> 2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Responde a las preguntas de los clientes mediante FM de Amazon Bedrock para buscar y resumir respuestas a preguntas frecuentes. Esta intención se activa cuando un enunciado no está clasificado en ninguna de las otras intenciones presentes en el bot. Tenga en cuenta que esta intención no se

activará en el caso de enunciados omitidos cuando se obtenga un valor de slot. Una vez reconocida, la `AMAZON.QnAIntent` utiliza el modelo de Amazon Bedrock especificado para buscar en la base de conocimiento de Amazon Bedrock configurada y responder a la pregunta del cliente.

> **Warning**
>
> No se puede utilizar la `AMAZON.QnAIntent` y la `AMAZON.KendraSearchIntent` en la misma configuración regional del bot.

Están disponibles las siguientes opciones de almacén de conocimientos. Debe haber creado ya el almacén de conocimientos e indexado los documentos que contiene.

- OpenSearch Dominio de servicio: contiene documentos indexados. Para crear un dominio, sigue los pasos que se indican en Creación y gestión de dominios OpenSearch de Amazon Service.
- Índice de Amazon Kendra: contiene documentos de preguntas frecuentes indexados. Para crear un índice de Amazon Kendra, siga los pasos que se indican en Crear un índice.
- Base de conocimiento de Amazon Bedrock: contiene orígenes de datos indexados. Para configurar una base de conocimiento de Amazon Bedrock, siga los pasos que se indican en Creación de una base de conocimiento.

Si selecciona esta intención, configure los siguientes campos y, a continuación, seleccione Agregar para agregar la intención.

- Modelo de Bedrock: elija el proveedor y el modelo fundacional que desea utilizar para esta intención. Asegúrese de comprobar los últimos modelos disponibles y el calendario de obsolescencia, y planifique las migraciones en consecuencia. Para obtener más información, consulte el ciclo de vida del modelo.
- Almacén de conocimientos: elija el origen desde el que desea que el modelo extraiga información para responder a las preguntas de los clientes. Estos son los estados disponibles:
  - OpenSearch— Configure los siguientes campos.
    - Punto de conexión del dominio: proporciona el punto de conexión del dominio que creó para el dominio o que le proporcionaron después de crear el dominio.
    - Nombre del índice: proporcione el índice para realizar búsquedas. Para obtener más información, consulta Indexación de datos en Amazon OpenSearch Service.
    - Elija cómo desea devolver la respuesta al cliente.

    - Respuesta exacta: cuando esta opción está habilitada, el valor del campo Respuesta se usa tal como está para la respuesta del bot. El modelo fundacional configurado de Amazon Bedrock se utiliza para seleccionar el contenido exacto de la respuesta tal como está, sin síntesis ni resumen del contenido. Especifique el nombre de los campos de pregunta y respuesta que se configuraron en la OpenSearch base de datos.
    - Incluir campos: devuelve una respuesta generada por el modelo mediante los campos que especifique. Especifique el nombre de un máximo de cinco campos que se configuraron en la OpenSearch base de datos. Utilice punto y coma (;) para separar campos.
- Amazon Kendra: configure los siguientes campos.
    - Índice de Amazon Kendra: seleccione el índice de Amazon Kendra que desea que busque su bot.
    - Filtro de Amazon Kendra: para crear un filtro, selecciona esta casilla. Para obtener más información sobre el formato JSON del filtro de búsqueda de Amazon Kendra, consulte Usar atributos de documentos para filtrar los resultados de búsqueda.
    - Respuesta exacta: para permitir que el bot devuelva la respuesta exacta de Amazon Kendra, seleccione esta casilla. De lo contrario, el modelo de Amazon Bedrock que seleccione generará una respuesta en función de los resultados.

      > **Note**
      >
      > Para utilizar esta función, primero debe añadir las preguntas frecuentes a su índice siguiendo los pasos que se indican en Añadir preguntas frecuentes (FAQs) a un índice.

- Base de conocimiento de Amazon Bedrock: si elige esta opción, especifique el ID de la base de conocimiento de Amazon Bedrock. Puede buscar el ID en la página de detalles de la base de conocimiento de Amazon Bedrock en la consola o enviando una solicitud GetKnowledgeBase.
    - Respuesta exacta: cuando esta opción está habilitada, el valor del campo Respuesta se usa tal como está para la respuesta del bot. El modelo fundacional configurado de Amazon Bedrock se utiliza para seleccionar el contenido exacto de la respuesta tal como está, sin síntesis ni resumen del contenido. Para utilizar la respuesta exacta en la base de conocimientos de Amazon Bedrock, debe hacer lo siguiente:
        - Cree archivos JSON individuales y que cada archivo contenga un campo de respuesta con la respuesta exacta que debe devolverse al usuario final.

- Al indexar estos documentos en la base de conocimientos de Bedrock, establezca la Estrategia de fragmentación en No fragmentar.
- Defina el campo de respuesta en Amazon Lex V2 como el campo de respuesta en Bedrock Knowledge Base.

Las respuestas del Qn se AIntent almacenarán en los atributos de la solicitud, tal y como se muestra a continuación:

- `x-amz-lex:qnA-search-response`— La respuesta del Qn AIntent a la pregunta o al enunciado.
- `x-amz-lex:qnA-search-response-source`: señala el documento o la lista de documentos utilizados para generar la respuesta.
- `x-amz-lex:qna-additional-context`— El contexto adicional utilizado por el Qn AIntent para generar la respuesta.

Configuraciones del modelo adicionales

Cuando AIntent se invoca Amazon.qn, utiliza una plantilla de mensaje predeterminada que combina las instrucciones y el contexto con la consulta del usuario para crear el mensaje que se envía al modelo para generar la respuesta. También puede proporcionar una petición personalizada o actualizar la predeterminada para que se ajuste a sus necesidades.

Puede diseñar la plantilla de petición con las siguientes herramientas:

Marcadores de posición rápidos: variables predefinidas en Amazon.qn AIntent para Amazon Bedrock que se rellenan dinámicamente en tiempo de ejecución durante la llamada a Bedrock. En la petición del sistema, puede ver estos marcadores de posición rodeados por el símbolo $. En la siguiente lista se describen los marcadores de posición que puede utilizar.

Variable	Se sustituye por	Modelo	¿Obligatorio?
$query_results$	Los resultados recuperados para la consulta del usuario del almacén de conocimientos.	Modelo de roca madre seleccionado	Sí

Variable	Se sustituye por	Modelo	¿Obligatorio?
$output_instruction$	Instrucciones subyacentes para formatear la generación de respuestas y las citas. Depende del modelo. Si define sus propias instrucciones de formato, le sugerimos que elimine este marcador de posición.	Modelo de roca madre seleccionado	No
$contexto_adicional$	El contexto adicional utilizado por el Qn para generar la respuesta AIntent	Modelo de roca madre seleccionado	No
$locale$	El idioma en el que el bot responderá a las consultas de los clientes	Modelo Bedrock seleccionado	No

La petición predeterminada que se utiliza es:

```
$query_results$

$additional_context$

Please only follow the instructions in <instruction> tags below.
<instruction>
Given the conversation history, <additional_context> and <Context>:
(1) first, identify the user query intent and classify it as one of the categories:
 FAQ_QUERY, OTHER_QUERY, GIBBERISH, GREETINGS, AFFIRMATION, CHITCHAT, or MISC;
(2) second, if the intent is FAQ_QUERY, predict the most relevant grounding passage(s)
 by providing the passage id(s) or output CANNOTANSWER;
```

```
(3) then, generate a concise, to-the-point FAQ-style response in $locale$ locale
 ONLY USING the grounding content in <Context> and <additional_context>; or output
 CANNOTANSWER if the user query/request cannot be directly answered with the grounding
 content. DO NOT mention about the grounding passages such as ids or other meta data;
 do not create new content not presented in <Context>. Do NOT respond to query that is
 ill-intented or off-topic;
(4) lastly, provide the confidence level of the above prediction as LOW, MID or HIGH.
</instruction>

$output_instruction$
```

$output_instruction$ se sustituye por:

```
Give your final response in the following form:
<answer>
<intent>FAQ_QUERY or OTHER_QUERY or GIBBERISH or GREETINGS or AFFIRMATION or CHITCHAT
 or MISC</intent>
<text>a concise FAQ-style response or CANNOTANSWER</text>
<passage_id>passage_id or CANNOTANSWER</passage_id>
<confidence>LOW or MID or HIGH</confidence>
</answer>
```

> **Note**
>
> Si decide no utilizar las instrucciones predeterminadas, cualquier resultado que proporcione el LLM se devolverá tal cual al usuario final.
> Las instrucciones de salida deben contener etiquetas <text></text> y <passageId></passageId> e instrucciones para que el LLM devuelva los passageIds a fin de proporcionar la respuesta y la atribución de la fuente.

Soporte contextual adicional a través de los atributos de la sesión

Puede transferir contexto adicional `AMAZON.QnAIntent` al tiempo de ejecución mediante el atributo session`x-amz-lex:qna-additional-context`. Esto le permite proporcionar información complementaria que el modelo puede utilizar junto con los resultados del almacén de conocimientos al generar una respuesta. El contexto adicional se inserta en la plantilla de solicitud a través del `$additional_context$` marcador de posición.

Ejemplo:

```
{"sessionAttributes": {"x-amz-lex:qna-additional-context":"Our support hours are Monday
 through Friday, 8AM-8PM EST"}}
```

Compatibilidad con el filtrado de metadatos de la base de conocimiento de Amazon Bedrock con atributos de sesión

Puede pasar los filtros de metadatos de la base de conocimiento de Amazon Bedrock como parte del atributo de sesión `x-amz-lex:bkb-retrieval-filter`.

```
            {"sessionAttributes":{"x-amz-lex:bkb-retrieval-filter":"{\"equals\":{\"key
\":\"insurancetype\",\"value\":\"farmers\"}}
```

> **Note**
>
> Debe utilizar la base de conocimientos de Amazon Bedrock como almacén de datos para AIntent que Qn utilice este filtro. Para obtener más información, consulte Metadata filtering

Configuraciones de inferencia

Puede definir las configuraciones de inferencia que se utilizarán al realizar la llamada a LLM mediante el atributo de sesión:

- temperatura: tipo Integer
- topP
- maxTokens

Ejemplo:

```
        {"sessionAttributes":{"x-amz-lex:llm-text-inference-config":"{\"temperature
\":0,\"topP\":1,\"maxTokens\":200}"}}
```

Las barreras de protección para Bedrock son compatibles con el tiempo de compilación y los atributos de sesión

- Al usar la consola en el momento de la compilación, proporcione el y el GuardrailsIdentifier . GuardrailsVersion Obtenga más información en la sección Additional Model Configurations.
- Mediante los atributos de sesión: también puede definir la configuración de las barreras de protección mediante los atributos de sesión: `x-amz-lex:bedrock-guardrails-identifier` y `x-amz-lex:bedrock-guardrails-version`.

Para obtener más información sobre el uso de barreras de protección de Bedrock, consulte Guardrails.

AMAZON.QnAIntent (multiple use support)

Puede elegir tener varios Amazon.qn AIntents dentro de una configuración regional. Amazon Lex V2 admite hasta 5 Amazon.QN AIntents dentro de una configuración regional de bots.

Amazon.qn se AIntent puede activar si se cumple uno de los siguientes casos:

- Si la configuración regional de un bot contiene solo 1 Amazon.qn AIntent y esa intención no contiene expresiones de muestra, se activa cuando una expresión no está clasificada en ninguna de las otras intenciones presentes en el bot. Esta intención se activa cuando un enunciado no está clasificado en ninguna de las otras intenciones presentes en el bot. Tenga en cuenta que esta intención no se activará en el caso de enunciados omitidos cuando se obtenga un valor de slot.

  > **Note**
  >
  > Si la respuesta del FM no es satisfactoria o falla la llamada al FM, Amazon Lex V2 invoca la `AMAZON.FallbackIntent`.

- Si Amazon.qn AIntent contiene ejemplos de expresiones, solo se activa cuando Amazon Lex V2 reconoce que el usuario quiere iniciar esa intención en función de las entradas del usuario.

  > **Note**
  >
  > Si la respuesta del FM no es satisfactoria o falla la llamada al FM, Amazon Lex V2 invoca el siguiente paso al fallo, definido en el bloque de cumplimiento.

> **Note**
>
> Si `botLocale` tiene más de 1 Amazon.QNAIntent, cada Amazon.QN AIntent debe tener al menos un ejemplo de expresión.

AMAZON.QinConnectIntent

> **Note**
>
> Para utilizar las capacidades de IA generativa con Amazon Q In Connect, debe cumplir los siguientes requisitos previos:
>
> 1. Diríjase a la consola de Amazon Connect y cree su instancia; si aún no tiene una, consulte Comenzar a utilizar Amazon Connect.
> 2. Habilita Amazon Q en Connect para tu instancia; consulta Habilitar Amazon Q en Connect para tu instancia.

AMAZON. QinConnectIntent responde a las preguntas de los clientes mediante la evolución mejorada por LLM de Amazon Connect Wisdom, que ofrece recomendaciones en tiempo real para ayudar a los clientes y agentes del centro de contacto a resolver los problemas de los clientes de forma rápida y precisa. Esta intención se activa cuando un enunciado no está clasificado en ninguna de las otras intenciones presentes en el bot. Tenga en cuenta que esta intención no se activará en el caso de enunciados omitidos cuando se obtenga un valor de slot. Una vez reconocido, el AMAZON. QinConnectIntent, utiliza el dominio Q in Connect especificado para buscar en la base de conocimientos de Amazon Bedrock configurada y responder a la pregunta del cliente.

> **Note**
>
> - No puedes usar AMAZON. QinConnectIntent junto con Amazon.qn AIntent en la misma configuración regional del bot.
> - Si seleccionas otro idioma además del inglés estadounidense, debes personalizar las instrucciones del autoservicio (`SELF_SERVICE_PRE_PROCESSINGySELF_SERVICE_ANSWER_GENERATION`) para

que respondan en el idioma especificado. Para obtener más información sobre cómo personalizar tu mensaje, consulta Cómo personalizar Amazon Q en Connect.

Si seleccionas esta intención, tendrás que configurar los siguientes campos y, a continuación, seleccionar Guardar intención para añadir la intención al bot.

- Configuración de Amazon Q In Connect: proporcione el nombre de recurso de Amazon (ARN) del asistente Amazon Q in Connect. Patrón de ARN del asistente: `.^arn:[a-z-]*?:wisdom:[a-z0-9-]*?:[0-9]{12}:[a-z-]*?/[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}(?:/[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}){0,2}$>`

Las respuestas del se QinConnectIntent almacenarán en los atributos de la solicitud, tal y como se muestra a continuación:

- `x-amz-lex:q-in-connect-response`— La respuesta QinConnectIntent a la pregunta o al enunciado.

Atributos de sesión devueltos desde QinConnectIntent

La interacción con la QinConnect intención proporciona datos adicionales sobre la conversación a través de los atributos de la sesión.

1. `x-amz-lex:q-in-connect:session-arn`— Un identificador único para la sesión creada con Amazon Q In Connect durante la conversación.
2. `x-amz-lex:q-in-connect:conversation-status`— El estado actual de la conversación con el QinConnect asistente o el dominio. Hay tres valores posibles para este estado:
   - `CLOSED`
   - `READY`
   - `PROCESSING`
3. `x-amz-lex:q-in-connect:conversation-status-reason`— Proporciona el motivo del estado actual informado con el atributo anterior. De lo contrario, las razones posibles son las siguientes:

- `SUCCESS`— Indica que al cliente no le queda nada que preguntar y que la pregunta se ha respondido correctamente.
- `FAILED`— Indica un error al responder a la pregunta del cliente. En su mayoría, se deben a la falta de comprensión de la pregunta del cliente.
- `REJECTED`: indica que el asistente se niega a responder a la pregunta del cliente y recomienda que la pregunta se aborde fuera de la interacción con el bot, por ejemplo, hablando con una persona o un agente, para obtener más información.

> **Note**
>
> Cuando QinConnectIntent se invoca un bot con durante las interacciones con los clientes impulsadas por una instancia de Amazon Connect, es necesario crear el arn de sesión y pasarlo desde la instancia de Amazon Connect. Para crear una sesión, Amazon Connect Flows se puede configurar con Amazon Q en el paso Connect.

Limitaciones

- No puede utilizar AMAZON. QinConnectIntent junto con intenciones sin expresiones específicas, como AMAZON.QN, AMAZON. AIntent BedrockAgentIntent en la misma configuración regional del bot.
- Cuando QinConnectIntent se invoca un bot con durante una interacción con un cliente impulsada por una instancia de Amazon Connect, es necesario crear el arn de sesión y pasarlo desde la instancia de Amazon Connect. Para crear una sesión, Amazon Connect Flows se puede configurar con el paso Amazon Q In Connect.
- No puede haber más de un AMAZON. QinConnectIntent por configuración regional del bot.
- El dominio Amazon Q in Connect utilizado con AMAZON. QinConnectIntent debe estar en la misma región de AWS que el bot Amazon Lex V2.

Permisos

Si el QinConnect Intent se utiliza en un bot de Amazon Lex V2 y el bot utiliza un rol vinculado a servicios (SLR), el servicio Amazon Lex V2 tiene los permisos para actualizar las políticas correspondientes del rol e integrarlo con el asistente Q in Connect. Si el bot utiliza una función de IAM personalizada, el usuario tendrá que añadir estos permisos manualmente a su función de IAM.

El rol vinculado al servicio se actualizará con los siguientes permisos si se añade la QinConnect intención. Se agregará una nueva política de QinConnect acceso:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "QInConnectAssistantPolicy",
            "Action": [
                "wisdom:CreateSession",
                "wisdom:GetAssistant"
            ],
            "Resource": [
                "arn:aws:wisdom:*:accountId:assistant/assistantId",
                "arn:aws:wisdom:*:accountId:assistant/assistantId/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "QInConnectSessionsPolicy",
            "Action": [
                "wisdom:SendMessage",
                "wisdom:GetNextMessage"
            ],
            "Resource": [
                "arn:aws:wisdom:*:accountId:session/assistantId/*"
            ]
        },
        {
            "Sid": "QInConnectKmsCMKPolicy",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:accountId:key/keyId"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "accountId",
                    "kms:ViaService": "wisdom.region.amazonaws.com",
```

```
                    "kms:EncryptionContext:aws:wisdom:assistant:arn":
 ["arn:aws:wisdom:region:accountId:assistant/assistantId"]
                }
            }
        }
    ]
}
```

> **Note**
>
> La `QInConnectKmsCMKPolicy` declaración solo es obligatoria si utiliza una clave de KMS gestionada por el cliente con el asistente Amazon Q in Connect.

Política de confianza

```
{
    "Effect": "Allow",
    "Sid": "LexV2InternalTrustPolicy",
    "Principal": {
        "Service": "lexv2.aws.internal"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "accountId"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:lex:*:accountId:bot-alias/botId/*"
        }
    }
}
```

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

Adición de tipos de ranura

Los tipos de slot definen los valores que los usuarios pueden proporcionar para las variables de intención. Defina los tipos de slot para cada idioma de modo que los valores sean específicos de ese idioma. Por ejemplo, para un tipo de slot que muestre colores de pintura, puede incluir el valor «red» en inglés, «rouge» en francés y «rojo» en español.

En este tema se describe cómo crear tipos de slot personalizados que proporcionen valores para los slots de su intención. También puede usar tipos de slots integrados para los valores estándar. Por ejemplo, puede utilizar el tipo de slot integrado `AMAZON.Country` para obtener una lista de países del mundo.

Crear tipos de slot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, seleccione el bot al que desee añadir el idioma, seleccione Estructura de conversación y, por último, Todos los idiomas.
3. Seleccione el idioma al que quiere añadir el tipo de slot y, a continuación, seleccione Tipos de slot.
4. Seleccione Añadir tipo de slot, dele un nombre a su tipo de slot y, a continuación, seleccione Agregar.
5. En el editor de tipos de slot, añada los detalles del tipo de slot.

    - Resolución de valores de slots: determina cómo se resuelven los valores de los slots. Si elige Expandir valores, Amazon Lex V2 utiliza los valores como valores representativos para el entrenamiento. Si utiliza Restringir a los valores de slots, los valores permitidos para el slot se limitan a los que usted proporcione.
    - Valores del tipo de slot: los valores para el slot. Si seleccionó Restringir a los valores de slots, puede añadir sinónimos para el valor. Por ejemplo, para el valor «fútbol» puede añadir el sinónimo «balompié». Si el usuario escribe «balompié» en una conversación con su bot, el valor real del slot es «fútbol».
    - Usar los valores de slot como vocabulario personalizado: active esta opción para ayudar a mejorar el reconocimiento de los valores de los slots y los sinónimos en las conversaciones de

audio. No habilite esta opción cuando los valores de los slots sean términos comunes, como «sí», «no», «uno», «dos», «tres», etc.

6. Seleccione Guardar tipo de slot.

Amazon Lex V2 ofrece los siguientes tipos de slot:

Temas

- Built-in tipos de ranuras
- Tipos de ranura personalizados
- Tipo de slot gramatical
- Tipos de slot compuestos

Built-in tipos de ranuras

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 los datos de ranura más utilizados, como la fecha, la hora y la ubicación. Built-in los tipos de ranuras no tienen versiones.

Tipo de slot	Descripción breve	Configuraciones regionales compatibles	
AMAZON.AlphaNumeric	Reconoce palabras compuestas de letras y números.		
AMAZON.City	Reconoce las palabras que representan una ciudad.	Todas las configuraciones locales	
AMAZON.Confirmation	Reconoce las palabras que significan «Sí», «No»,	Todas las configuraciones locales	

Tipo de slot	Descripción breve	Configuraciones regionales compatibles	
	«Quizás» y «No sé» y las convierte a un formato estándar (Yes/No/Maybe/Donno lo sé).		
AMAZON.Country	Reconoce las palabras que representan un país.	Todas las configuraciones locales	
AMAZON.Currency	Reconoce palabras que representan un valor de divisa y lo convierte en un valor y una abreviatura de divisa estándar.	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 duración y las convierte a un formato estándar.	Todas las configuraciones locales	

Tipo de slot	Descripción breve	Configuraciones regionales compatibles	
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 compatibles	
AMAZON.PhoneNumber	Reconoce palabras que representan un número de teléfono y las convierte en una cadena numérica.	Todas las configuraciones locales	
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 locales	
AMAZON.Time	Reconoce las palabras que indican horas y las convierte en un formato de hora.	Todas las configuraciones locales	
AMAZON.UKPostalCode	Reconoce las palabras que representan un código postal de Reino Unido y las convierte a un formato estándar.	Solo inglés (Reino Unido) (en-GB)	
AMAZON.FreeFormInput	Reconoce las cadenas que constan de cualquier palabra o carácter.	Todas las configuraciones locales	

AMAZON.AlphaNumeric

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

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**

El tipo de slot `AMAZON.AlphaNumeric` admite entradas mediante estilos de ortografía. Puede utilizar los estilos ortografía por letra y ortografía por palabra para ayudar a sus clientes a introducir las letras. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

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 de Canadá
- Números de permiso de conducción
- Números de identificación de vehículo

Use una expresión regular estándar. Amazon Lex V2 admite los siguientes caracteres en la expresión regular:

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

Amazon Lex V2 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 `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 CreateSlotType. La expresión regular se valida al guardar el tipo de slot. Si la expresión no es válida, Amazon Lex V2 devuelve un mensaje de error.

Cuando se utiliza una expresión regular en un tipo de slot, Amazon Lex V2 comprueba la entrada en slots 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 V2 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 V2 no pasa de ser una variante a un nombre oficial.

Ejemplos:

- Nueva York
- Reikiavik
- Tokio
- Versalles

AMAZON.Confirmation

Este tipo de slot reconoce las frases y palabras introducidas que corresponden a frases y palabras «Sí», «No», «Quizás» y «No sé» para Amazon Lex V2 y las convierte en uno de los cuatro valores.

Se puede usar para capturar la confirmación o el reconocimiento del usuario. En función del valor final resuelto, puede crear las condiciones para diseñar múltiples rutas de conversación.

Por ejemplo:

si {confirmation} = «Sí», cumple con la intención

de lo contrario, busca otro slot

Ejemplos:

- Sí: Sí, vale, claro, lo tengo, estoy de acuerdo...
- No: No, negativo, para nada, olvídalo, me niego, de ninguna manera...
- Quizás: es posible, quizás, a veces, podría ser, podría ser cierto...
- No lo sé: no sé, desconocido, ni idea, no estoy seguro de ello, quién sabe...

A partir del 17 de agosto de 2023, si existe un tipo de slot personalizado denominado «Confirmación», deberá cambiarse el nombre para evitar conflictos con el slot integrado Confirmación. En el panel de navegación izquierdo de la consola de Lex, vaya al tipo de slot (para un tipo de slot personalizado existente denominado Confirmación) y actualice el nombre del tipo de slot. El nombre del nuevo tipo de slot no debe ser «Confirmación», que es una palabra clave reservada para el tipo de slot de confirmación integrado.

AMAZON.Country

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

- Australia
- Alemania
- Japón
- Estados Unidos
- Uruguay

AMAZON.Currency

Convierte las palabras que representan una divisa en un código de divisa alfabético estándar ISO 4217 y en un número. Amazon Lex V2 reconoce las divisas, pero no convierte de una divisa a otra.

Para obtener más información, consulte Currency codes - ISO 4217 en el sitio web de la Organización Internacional de Normalización (ISO).

La divisa representada se estructura de la siguiente manera: `{Unit} {Amount}`

- {Unidad} se refiere a la unidad monetaria específica (por ejemplo, USD).
- {Cantidad} indica el valor monetario, formateado con dos decimales (por ejemplo, 300,00).

Ejemplos (todos los ejemplos siguientes utilizan la configuración regional en EE. UU.; diferentes configuraciones regionales pueden arrojar resultados diferentes):

- “3USD”: 3,00 USD
- “USD300”: 300,00 USD
- “3 dimes”: 0,30 USD
- “$1.56”: 1,56 USD
- “5c”: 0,05 USD
- “1 dólar”: 1,00 USD
- “five fifteen”: 515,00 USD
- “five dollars fifteen cents”: 5.15 USD
- «5 USD y «: 5,50 1/2 USD

AMAZON.Date

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

La fecha se proporciona según su intención en formato de ISO-8601 fecha. 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 futura, como «la semana que viene», se convierten en la fecha del último día de la semana actual. En ISO-8601 formato, 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 `2020-11-29`. Las fechas que se asignan a la semana actual o anterior se convierten

en el primer día de la semana. Por ejemplo, si hoy es 25 de noviembre de 2020, «la semana que viene» se convierte en `2020-11-16`.

- Los enunciados que se asignan a un mes futuro, 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`. Las fechas que se asignan al mes actual o anterior se convierten en el primer día del mes. Por ejemplo, si hoy es 25 de noviembre de 2020, «este mes» se convierte en `2020-11-01`.
- Los enunciados que se asignan a un año futuro, 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`. Las fechas que se asignan al año actual o anterior se convierten en el primer día del año. Por ejemplo, si hoy es 25 de noviembre de 2020, «el año pasado» se convierte en `2019-01-01`.

AMAZON.Duration

Convierte las palabras que indican duraciones en una duración numérica.

La duración se resuelve en un formato basado en el formato de ISO-8601 duración,`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 (+).

El tipo de slot `AMAZON.EmailAddress` admite entradas mediante estilos de ortografía. Puede utilizar los estilos ortografía por letra y ortografía por palabra para ayudar a sus clientes a introducir direcciones de correo electrónico. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

AMAZON.FirstName

Nombres de uso común. Este tipo de slot reconoce nombres formales, apodos informales y nombres que constan de más de una palabra. El nombre que se envía a su intención es el valor enviado por el usuario. Amazon Lex V2 no pasa del apodo al nombre formal.

Para los nombres que suenan igual pero que se escriben de forma diferente, Amazon Lex V2 envía su intención a una única forma común.

El tipo de slot `AMAZON.FirstName` admite entradas mediante estilos de ortografía. Puede utilizar los estilos ortografía por letra y ortografía por palabra para ayudar a sus clientes a introducir nombres. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

Ejemplos:

- Emily
- John
- Sofia
- Anil Kumar

AMAZON.FirstName también devuelve una lista de nombres estrechamente relacionados en función del valor original. Puede utilizar la lista de valores resueltos para recuperarse de errores tipográficos, confirmar el nombre con el usuario o buscar nombres válidos en la base de datos de su directorio de usuarios.

Por ejemplo, la entrada «John» puede provocar la devolución de nombres relacionados adicionales, como «John J» y "John-Paul».

A continuación se muestra el formato de respuesta del tipo de slot integrado `AMAZON.FirstName`:

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

AMAZON.LastName

Apellidos de uso común. Para los apellidos que suenan igual pero que se escriben de forma diferente, Amazon Lex V2 envía su intención a una única forma común.

El tipo de slot `AMAZON.LastName` admite entradas mediante estilos de ortografía. Puede utilizar los estilos ortografía por letra y ortografía por palabra para ayudar a sus clientes a introducir nombres. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

Ejemplos:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON.LastName también devuelve una lista de nombres estrechamente relacionados en función del valor original. Puede utilizar la lista de valores resueltos para recuperarse de errores tipográficos, confirmar el nombre con el usuario o buscar nombres válidos en la base de datos de su directorio de usuarios.

Por ejemplo, la entrada «Smith» puede provocar que se devuelvan otros nombres relacionados, como «Smyth» y «Smithe».

A continuación se muestra el formato de respuesta del tipo de slot integrado `AMAZON.LastName`:

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

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
-15	-15
menos 15	-15
menos quince puntos, dos, cuatro, cinco.	-15,245

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	Description (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 internacional	(03) 5115 4444 (509) 555-1212	`0351154444` `5095551212`
Número local	Número de teléfono sin código internacional ni código de área	555-1212	`5551212`

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.

Ejemplos:

- Avenida Canberra
- Front Street
- Market Road

AMAZON.Time

Convierte palabras que representan tiempos en valores temporales. `AMAZON.Time` puede resolver tiempos exactos, valores ambiguos e intervalos de tiempo. El valor del slot se puede resolver en los siguientes intervalos de tiempo:

- a. m.
- PM
- MO (mañana)
- AF (tarde)
- EV (tarde)
- NI (noche)

Cuando un usuario introduce una hora ambigua, Amazon Lex V2; utiliza el atributo `slots` de un evento 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 del campo `interpretedValue` es `null` y el campo `resolvedValues` contiene las dos posibles resoluciones de la hora. Amazon Lex V2 introduce lo siguiente en la función de Lambda:

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

Cuando el usuario responde con una hora inequívoca, Amazon Lex V2 envía la hora a la función de Lambda en el campo `interpretedValue` del atributo `slots` del evento de Lambda. Por ejemplo, si el usuario responde a la petición de una hora de entrega con «10:00 a. m.», Amazon Lex V2 introduce lo siguiente en la función de Lambda:

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

Cuando el usuario responde a la petición de una hora de entrega con «por la mañana», Amazon Lex V2 introduce lo siguiente en la función de Lambda:

```
"slots": {
    "deliveryTime": {
    "value": {
        "originalValue": "morning",
```

```
        "interpretedValue": "MO",
        "resolvedValues": [
            "MO"
        ]
    }
}
```

Para obtener más información acerca de los datos enviados de Amazon Lex V2 a una función de Lambda, consulte AWS Lambdaformato de evento de entrada para Lex V2.

AMAZON.UKPostalCode

Convierte las palabras que representan un código postal del Reino Unido a un formato estándar para los códigos postales del Reino Unido. El tipo de slot `AMAZON.UKPostalCode` valida y resuelve el código postal en un conjunto de formatos estandarizados, pero no comprueba que el código postal sea válido. La solicitud debe validar el código postal.

El tipo de slot `AMAZON.UKPostalCode` solo está disponible en la configuración regional inglesa (UK) (en-GB).

El tipo de slot `AMAZON.UKPostalCode` admite entradas mediante estilos de ortografía. Puede utilizar los estilos ortografía por letra y ortografía por palabra para ayudar a sus clientes a introducir las letras. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

El tipo de ranura reconoce todos los formatos de códigos postales válidos utilizados en el Reino Unido. Los formatos válidos son (“A” representa una letra y “9” representa un dígito):

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

Para la entrada de texto, el usuario puede introducir cualquier combinación de letras mayúsculas y minúsculas. El usuario puede usar u omitir el espacio en el código postal. El valor resuelto siempre incluirá el espacio en la ubicación adecuada para el código postal.

Para la entrada hablada, el usuario puede pronunciar los caracteres individuales o utilizar pronunciaciones de dos letras, como «doble A» o «doble 9». También pueden usar pronunciaciones de dos dígitos, como «noventa y nueve» para «99».

> **Note**
>
> No se reconocen todos los códigos postales del Reino Unido. Solo se admiten los formatos indicados anteriormente.

AMAZON.FreeFormInput

`AMAZON.FreeFormInput` se puede utilizar para capturar las entradas de forma libre del usuario final. Reconoce las cadenas que constan de cualquier palabra o carácter. El valor resuelto es todo el enunciado de entrada.

Ejemplo:

Bot: envíanos tus comentarios sobre tu experiencia de llamada.

Usuario: He obtenido las respuestas a todas mis preguntas y he podido completar la transacción.

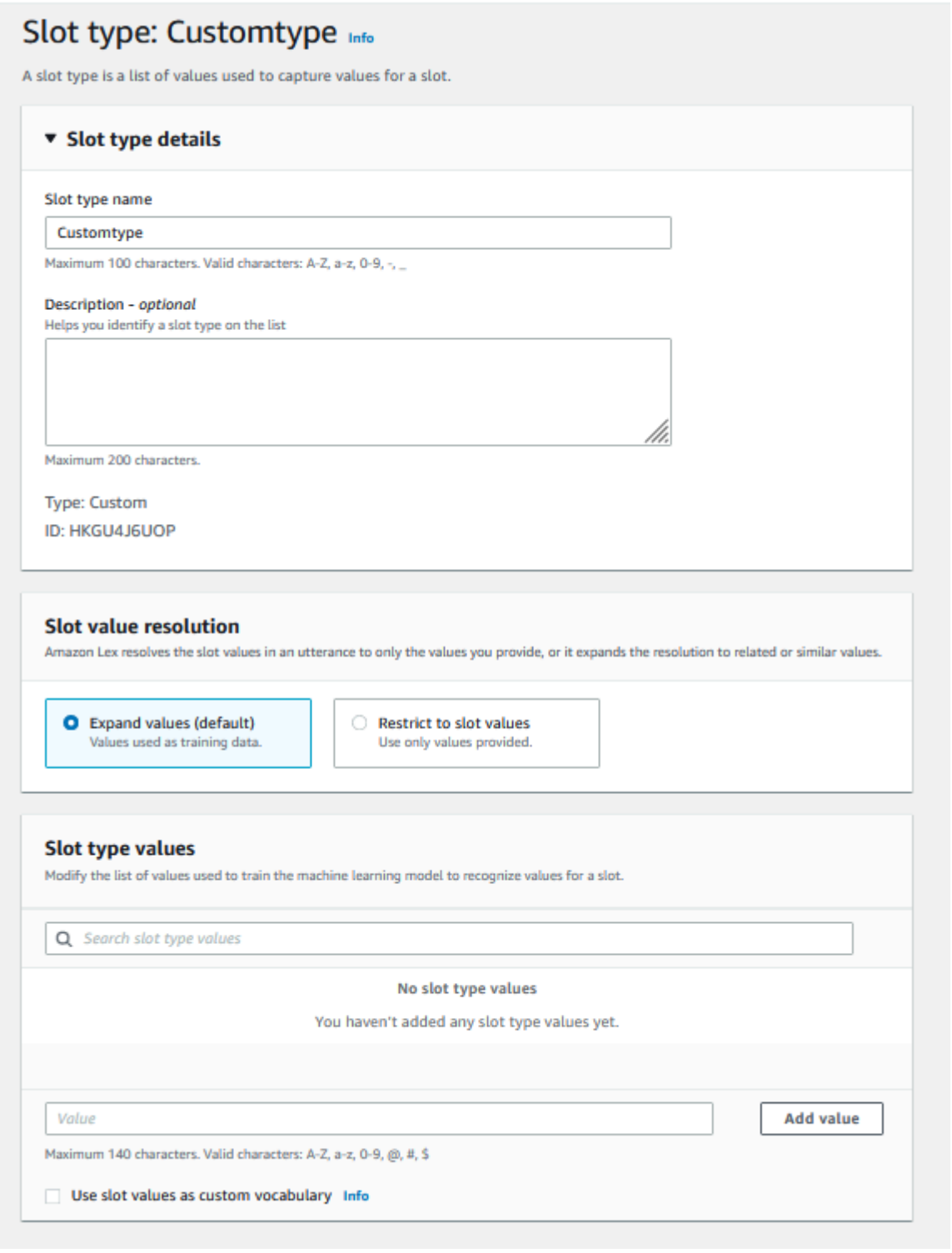
Nota:

- `AMAZON.FreeFormInput` se puede utilizar para capturar las entradas de forma libre del usuario final.
- `AMAZON.FreeFormInput` no se puede usar en ejemplos de enunciados de intención.
- `AMAZON.FreeFormInput` no puede tener ejemplos de enunciados de slot.
- `AMAZON.FreeFormInput` solo se reconoce cuando se obtiene.
- `AMAZON.FreeFormInput` no admite esperar y continuar.
- `AMAZON.FreeFormInput`actualmente no es compatible con el canal Connect Customer Chat.
- Cuando se active un slot `AMAZON.FreeFormInput`, `FallbackIntent` no se activará.
- Cuando se active un slot `AMAZON.FreeFormInput`, no habrá ningún cambio intencional.

Tipos de ranura 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 slot es una

lista de valores que Amazon Lex V2 utiliza para enseñar al modelo de machine learning a reconocer los valores de un slot. Por ejemplo, puede definir un tipo de ranura llamado `Genres` con valores como “comedia”, “aventuras”, “documental”, etc. Puede definir sinónimos 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 ampliar los valores de slot. Los valores del slot se utilizarán como datos de capacitación y el modelo resolverá el slot en el valor que proporciona el usuario si es similar a los valores del slot y los sinónimos de estos. Este es el comportamiento predeterminado. Amazon Lex V2 mantiene una lista de posibles resoluciones para un slot. Cada entrada de la lista ofrece un valor de resolución que Amazon Lex V2 reconoce como posibilidad adicional para el slot. Un valor resuelto es la mejor forma de que coincida con el valor de slot. La lista contiene hasta cinco valores.

De manera alternativa ,puede configurar el tipo de slot para restringir la resolución a los valores del slot. En este caso, el modelo convertirá un valor de slot introducido por el usuario en un valor de slot existente solo si es el mismo que ese valor de slot o si es un sinónimo. Por ejemplo, si el usuario introduce «divertido», se resolverá en el slot «comedia».

Cuando el valor especificado por el usuario es un sinónimo, el modelo devuelve ese valor de tipo de slot como la primera entrada de la lista de valores de `resolvedValues`. Por ejemplo, si el usuario escribe «gracioso», el modelo rellena el campo `originalValue` con el valor «gracioso» y la primera entrada del campo resolvedValues con «comedia». Puede configurar `valueSelectionStrategy` al crear o actualizar un tipo de slot con la operación CreateSlotType, de manera que el valor de slot se rellene con el primer valor de la lista de resolución.

Los tipos de slot personalizados admiten entradas mediante estilos de ortografía. Puede utilizar los estilos ortografía por letra y ortografía por palabra para ayudar a sus clientes a introducir las letras. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

Si utiliza una función de Lambda, el evento de entrada a la función incluye una lista de resolución llamada `resolvedValues`. El siguiente ejemplo muestra la sección del slot de la entrada a una función de Lambda:

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

Un tipo de slot personalizado no debe tener el mismo nombre que los tipos de slots integrados. Por ejemplo, un tipo de slot personalizado no debe nombrarse con las palabras clave reservadas Fecha, Número o Confirmación. Estas palabras clave están reservadas a los tipos de slots integrados. Para obtener una lista de los tipos de slots integrados, consulte Built-in tipos de ranuras.

Tipo de slot gramatical

> **Important**
>
> Amazon Lex V2 otorga acceso a los tipos de ranuras gramaticales para cada cuenta de AWS. Si su cuenta no tiene acceso, Amazon Lex V2 rechaza los intentos de añadir un tipo de ranura gramatical a un bot. Amazon Lex V2 también rechaza las compilaciones para configuraciones regionales que incluyen un tipo de ranura gramatical. Para solicitar el acceso a su cuenta, póngase en contacto con AWS Support. Si no necesita crear su propia gramática, puede utilizar en su lugar un tipo de espacio personalizado.

Con el tipo de slot gramatical, puede crear su propia gramática en formato XML según la especificación SRGS para recopilar información en una conversación. Amazon Lex V2 reconoce los enunciados que coinciden con las reglas especificadas en la gramática. También puede proporcionar reglas de interpretación semántica mediante etiquetas de ECMAScript en los archivos de gramática. A continuación, Amazon Lex devuelve las propiedades definidas en las etiquetas como valores resueltos cuando se produce una coincidencia.

Solo puede crear tipos de slot gramaticales en los idiomas inglés (Australia), inglés (Reino Unido) e inglés (EE. UU.).

Un tipo de slot gramatical consta de dos partes. La primera es la propia gramática, escrita con el formato de especificación SRGS. La gramática interpreta el enunciado del usuario. Si el enunciado es aceptado por la gramática, coincide; de lo contrario, se rechaza. Si un enunciado coincide, se pasa al guion, si lo hay.

La segunda forma parte de un tipo de slot gramatical es un guion opcional escrito en ECMAScript que transforma la entrada en los valores resueltos devueltos por el tipo de slot. Por ejemplo, puede usar un script para convertir números hablados en dígitos. Las declaraciones de ECMAScript se incluyen en el elemento **<tag>**.

El siguiente ejemplo está en formato XML según la especificación SRGS que muestra una gramática válida aceptada por Amazon Lex V2. Define un tipo de slot gramatical que acepta números de tarjetas y determina si son para cuentas normales o prémium. Para obtener más información acerca de la sintaxis aceptable, consulte Definición sobre la gramática y los temas Formato de script.

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

   <rule id="card_number" scope="public">
        <item repeat="0-1">
            card number
        </item>
        <item>
            seven
            <tag>out.value = "7";</tag>
         </item>
         <item>
            <one-of>
               <item>
                  two four one
                  <tag> out.value = out.value + "241"; out.card_type = "premium"; </
tag>
               </item>
               <item>
                  zero zero one
                  <tag> out.value = out.value + "001"; out.card_type = "regular";</tag>
               </item>
            </one-of>
         </item>
   </rule>
</grammar>
```

La gramática anterior solo acepta dos tipos de números de tarjeta: 7241 o 7001. Ambos pueden llevar opcionalmente el prefijo «número de tarjeta». También contiene etiquetas ECMAScript que se pueden utilizar para la interpretación semántica. Con la interpretación semántica, el enunciado «carta número siete dos cuatro uno» devolvería el siguiente objeto:

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

Este objeto se devuelve como una JSON-serialized cadena en el `resolvedValues` objeto devuelto por las StartConversation operaciones RecognizeText RecognizeUtterance, y.

Añadir un tipo de slot gramatical

Añadir un tipo de slot gramatical

1. Cargue la definición XML de su tipo de slot en un bucket de S3. Anote el nombre del bucket y la ruta al tipo de archivo.

   > **Note**
   >
   > El tamaño de archivo máximo es de 100 KB.

2. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
3. En el menú de la izquierda, seleccione Bots y, a continuación, seleccione el bot al que quiere añadir el tipo de slot gramatical.
4. Seleccione Ver idiomas y, a continuación, seleccione el idioma al que quiere añadir el tipo de slot gramatical.
5. Seleccione Ver tipos de slots.
6. Seleccione Añadir tipo de slot y, a continuación, seleccione Añadir tipo de slot gramatical.
7. Asigne un nombre al tipo de slot y, a continuación, seleccione Añadir.
8. Seleccione el bucket de S3 que contiene el archivo de definición e introduzca la ruta al archivo. Seleccione Guardar tipo de slot.

Definición sobre la gramática

En este tema se muestran las partes de la especificación SRGS compatibles con Amazon Lex V2. Todas las reglas se definen en la especificación SRGS. Para obtener más información, consulte la recomendación del W3C de la versión 1.0 de la Especificación gramatical del reconocimiento de voz.

Temas

- Declaraciones de encabezado
- Elementos XML compatibles
- Tokens
- Referencia a reglas
- Secuencias y encapsulación
- Repeticiones
- Idioma
- Tags
- Ponderaciones

Este documento incluye material copiado y derivado de la versión 1.0 de la especificación gramatical de reconocimiento de voz del W3C (disponible en https://www.w3.org/TR/speech-grammar/). La información sobre las citas es la siguiente:

Derechos de autor © 2004 W3C® (MIT), ERCIM, Keio, todos los derechos reservados. Se aplican las normas del W3C en materia de responsabilidad, marcas comerciales, uso de documentos y licencias de software.

El documento de especificaciones del SRGS, una recomendación del W3C, está disponible en el W3C con la siguiente licencia.

Texto de licencia

Licencia

Al and/or copiar este documento, o el documento del W3C al que se vincula esta declaración, usted (el licenciatario) acepta que ha leído, entendido y cumplirá los siguientes términos y condiciones:

Se otorga permiso para copiar y distribuir el contenido de este documento, o del documento del W3C desde el que se enlaza esta declaración, en cualquier medio, con cualquier propósito y sin pagar tasas ni regalías, siempre que incluya lo siguiente en TODAS las copias del documento, o partes del mismo, que utilice:

- Un enlace o URL al documento original del W3C.
- El aviso de derechos de autor preexistente del autor original o, si no existe, un aviso (se prefiere el hipertexto, pero se permite la representación textual) del formato: «Derechos de autor © [$date-

of-document] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang). http://www.w3.org/Consortium/Legal/2015/doc-license»

- Si existe, el ESTADO del documento del W3C.

Cuando el espacio lo permita, se debe incluir el texto completo de este AVISO. Solicitamos que se indique la autoría en cualquier software, documento u otro elemento o producto que cree de conformidad con la implementación del contenido de este documento o de cualquier parte del mismo.

No se concede ningún derecho a crear modificaciones o derivados de los documentos del W3C en virtud de esta licencia, excepto en los siguientes casos: para facilitar la implementación de las especificaciones técnicas establecidas en este documento, cualquier persona puede preparar y distribuir obras derivadas y partes de este documento en software, en los materiales de apoyo que acompañan al software y en la documentación del software, SIEMPRE que todos esos trabajos incluyan el aviso siguiente. SIN EMBARGO, queda expresamente prohibida la publicación de obras derivadas de este documento para su uso como especificación técnica.

Además, los «componentes de código» (el IDL web en las secciones claramente marcadas como IDL web; y el código de W3C-defined marcado (HTML, CSS, etc.) y el código del lenguaje de programación informática claramente marcados como ejemplos de código, se licencian con arreglo a la licencia de software del W3C.

El aviso es el siguiente:

«Derechos de autor © 2015 W3C® (MIT, ERCIM, Keio, Beihang). Este software o documento incluye material copiado o derivado de [título y URI del documento del W3C]».

Descargos de responsabilidad

ESTE DOCUMENTO SE PROPORCIONA «TAL CUAL» Y LOS TITULARES DE LOS DERECHOS DE AUTOR NO OFRECEN NINGUNA DECLARACIÓN NI GARANTÍA, EXPRESA O IMPLÍCITA, INCLUIDAS, ENTRE OTRAS, LAS GARANTÍAS DE COMERCIABILIDAD, IDONEIDAD PARA UN PROPÓSITO PARTICULAR O TÍTULO; DE QUE EL CONTENIDO DEL DOCUMENTO SEA ADECUADO PARA CUALQUIER PROPÓSITO; NI DE QUE LA IMPLEMENTACIÓN DE DICHO CONTENIDO NO INFRINJA NINGUNA PATENTE, DERECHO DE AUTOR, MARCA COMERCIAL U OTROS DERECHOS DE TERCEROS. NON-INFRINGEMENT

LOS TITULARES DE LOS DERECHOS DE AUTOR NO SERÁN RESPONSABLES DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL O CONSECUENTE QUE SURJA DEL USO DEL DOCUMENTO O DE LA EJECUCIÓN O IMPLEMENTACIÓN DE SU CONTENIDO.

El nombre y las marcas comerciales de los titulares de los derechos de autor NO se pueden utilizar en la publicidad o la publicidad relacionada con este documento o su contenido sin un permiso previo específico y por escrito. Los titulares de los derechos de autor de este documento permanecerán en todo momento en manos de los titulares de los derechos de autor.

Declaraciones de encabezado

En la siguiente tabla, se muestran las declaraciones de encabezado compatibles con el tipo de slot gramatical. Para obtener más información, consulte la recomendación del W3C sobre Declaraciones de encabezado sobre gramática en la versión 1 de la Especificación gramatical del reconocimiento de voz.

Declaración	Requisito de especificación	Formulario XML	Soporte de Amazon Lex V2	Especificación
Versión gramatical	Obligatorio	4.3: atributo `version` en el elemento `grammar`	Obligatorio	SRGS
Espacio de nombres XML	Obligatorio (solo XML)	4.3: atributo `xmlns` en el elemento `grammar`	Obligatorio	SRGS
Tipo de documento	Obligatorio (solo XML)	4.3: TIPO DE DOCUMENTO XML	Recomendado	SRGS
Codificación de caracteres	Recomendado	4.4: atributo `encoding` en la declaración XML	Recomendado	SRGS
Idioma	Necesario en el modo de voz Se ignora en el modo DTMF	4.5: atributo `xml:lang` en el elemento `grammar`	Necesario en el modo de voz Se ignora en el modo DTMF	SRGS

Declaración	Requisito de especificación	Formulario XML	Soporte de Amazon Lex V2	Especificación
Mode	Opcional	4.6: atributo `mode` en el elemento `grammar`	Opcional	SRGS
Regla raíz	Opcional	4.7: atributo `root` en el elemento `grammar`	Obligatorio	SRGS
Formato de etiqueta	Opcional	4.8: atributo `tag-format` en el elemento `grammar`	Se admiten cadenas literales y ECMAScript	SRGS, SISR
URI de base	Opcional	4.9: atributo `xml:base` en el elemento `grammar`	Opcional	SRGS
Lexicones de pronunciación	Opcional, se admiten varios	4.10: elemento `lexicon`	No compatible	SRGS, PLUS
Metadatos	Opcional, se admiten varios	4.11.1: elemento `meta`	Obligatorio	SRGS
Metadatos XML	Opcional, solo XML	4.11.2: elemento `metadata`	Opcional	SRGS
Tag	Opcional, se admiten varios	4.12: elemento `tag`	No se admiten etiquetas globales	SRGS

Ejemplo

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

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

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

Elementos XML compatibles

Amazon Lex V2 admite los siguientes elementos XML para gramáticas personalizadas:

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

Tokens

En la siguiente tabla, se muestran las especificaciones de token compatibles con el tipo de slot gramatical. Para obtener más información, consulte la recomendación del W3C sobre Tokens de la versión 1 de la Especificación gramatical del reconocimiento de voz.

Tipo de token	Ejemplo	¿Compatible?
Token único sin comillas	Hola	Sí
Token único sin comillas: no alfabético	2	Sí

Tipo de token	Ejemplo	¿Compatible?
Token único entre comillas, sin espacios en blanco	«hola»	Sí, coloque las comillas dobles cuando solo contenga un token
Dos tokens delimitados por espacios en blanco	buen viaje	Sí
Cuatro tokens delimitados por espacios en blanco	esta es una prueba	Sí
Token único entre comillas, con espacios en blanco	«San Francisco»	No
Token único XML en la etiqueta <token>	<token>San Francisco</token>	No (igual que token único entre comillas con espacios en blanco)

Notas

- Token único entre comillas con espacios en blanco: la especificación exige que las palabras entre comillas dobles se traten como un token único. Amazon Lex V2 los trata como tokens delimitados por espacios en blanco.
- Token único XML en <token>: la especificación requiere que las palabras delimitadas por <token> representen un token. Amazon Lex V2 los trata como tokens delimitados por espacios en blanco.
- Amazon Lex V2 arroja un error de validación cuando se encuentra alguno de los dos usos en la gramática.

Ejemplo

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

```
</rule>
```

Referencia a reglas

La siguiente tabla resume las diversas formas de las referencias a reglas que son posibles en los documentos gramaticales. Para obtener más información, consulte la recomendación del W3C sobre la Referencia a reglas de la versión 1 de la Especificación gramatical del reconocimiento de voz.

Tipo de referencia	Formulario XML	compatible
2.2.1 Referencia explícita a una regla local	`<ruleref uri="#rulename"/>`	Sí
2.2.2 Referencia explícita a una regla gramatical nombrada identificada por un URI	`<ruleref uri="grammarURI#rulename"/>`	No
2.2.2 Referencia implícita a una regla raíz gramatical identificada por un URI	`<ruleref uri="grammarURI"/>`	No
2.2.2 Referencia explícita a una regla gramatical nombrada identificada por un URI con un tipo de medio	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	No
2.2.2 Referencia implícita a una regla raíz gramatical identificada por un URI con un tipo de medio	`<ruleref uri="grammarURI" type="media-type"/>`	No

2.2.3 Definiciones de reglas especiales	`<ruleref special="NULL"/>`
	
	`<ruleref special="VOID"/>`
		No

Tipo de referencia	Formulario XML	compatible
	`<ruleref special=" GARBAGE"/>`	

Notas

1. El URI gramatical es un URI externo. Por ejemplo, `http://grammar.example.com/world-cities.grxml`.
2. El tipo de medio puede ser:
   - `application/srgs+xml`
   - `text/plain`

Ejemplo

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

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

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

Secuencias y encapsulación

En el siguiente ejemplo, se muestran las secuencias compatibles. Para obtener más información, consulte la recomendación del W3C sobre Encapsulación y secuencias de la versión 1 de la Especificación gramatical del reconocimiento de voz.

Ejemplo

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

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

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

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

Repeticiones

En la siguiente tabla, se muestran las expansiones repetidas compatibles con las reglas. Para obtener más información, consulte la recomendación del W3C sobre Repeticiones de la versión 1 de la Especificación gramatical del reconocimiento de voz.

Formulario XML Ejemplo	Comportamiento	¿Compatible?
repeat=«n» repeat=«6»	La expresión contenida se repite exactamente «n» veces. «n» debe ser «0» o un número entero positivo.	Sí
repeat=»m-n» repeat=«4-6»	La expansión contenida se repite entre «m» y «n» veces (inclusive). Tanto «m» como «n» deben ser «0» o un entero	Sí

Formulario XML Ejemplo	Comportamiento	¿Compatible?
	positivo, y «m» debe ser menor o igual que «n».	
repeat=»m-» repeat=«3-»	La expansión contenida se repite «m» veces o más (inclusive). «m» debe ser «0» o un entero positivo. Por ejemplo, «3-» indica que la expansión contenida puede producirse tres, cuatro, cinco o más veces.	Sí
repeat=«0-1»	La expansión contenida es opcional.	Sí
<item repeat=«2-4» repeat-prob=«0.8»>		No

Idioma

La siguiente explicación hace referencia a los identificadores de idioma aplicados a las gramáticas. Para obtener más información, consulte la recomendación del W3C sobre Idioma de la versión 1 de la Especificación gramatical del reconocimiento de voz.

De forma predeterminada, una gramática es un documento en un solo idioma con un identificador de idioma que se proporciona en la declaración de idioma del encabezado gramatical. Todos los símbolos de esa gramática, a menos que se declare lo contrario, se manejarán de acuerdo con el idioma de la gramática. Grammar-level no se admiten las declaraciones de idioma.

En el siguiente ejemplo:

1. Amazon Lex V2 admite la declaración del encabezado gramatical para el idioma «en-US».
2. Item-level no se admite el idioma adjunto (resaltado en*red*) . Amazon Lex V2 arroja un error de validación si el idioma adjunto es diferente al de la declaración del encabezado.

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

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

<!-- the default grammar language is US English -->
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0">

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

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

Tags

La siguiente explicación hace referencia a las etiquetas definidas para las gramáticas. Para obtener más información, consulte la recomendación del W3C sobre Etiquetas de la versión 1 de la Especificación gramatical del reconocimiento de voz.

Según la especificación SRGS, las etiquetas se pueden definir de las siguientes maneras:

1. Como parte de una declaración de encabezado, tal como se describe en Declaraciones de encabezado.

2. Como parte de una definición de <regla>.

Se admiten los siguientes formatos:

- `semantics/1.0` (SISR, ECMAScript)
- `semantics/1.0-literals` (literales de cadena SISR)

No se admiten los siguientes formatos:

- `swi-semantics/1.0` (propiedad de Nuance)

Ejemplo

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

Ponderaciones

Puede añadir el atributo de peso a un elemento. El peso es un valor de punto flotante positivo que representa el grado en que se potencia la frase del elemento durante el reconocimiento de voz. Para obtener más información, consulte la recomendación del W3C sobre Ponderaciones de la versión 1 de la Especificación gramatical del reconocimiento de voz.

Los pesos deben ser mayores que 0 y menores o iguales que 10 y solo pueden tener una cifra decimal. Si el peso es mayor que 0 e inferior a 1, la frase se potencia negativamente. Si el peso es mayor que 1 e inferior o igual a 10, la frase se potencia positivamente. Un peso de 1 equivale a no darle ningún peso y no hay ningún refuerzo para la frase.

Asignar los pesos adecuados a los elementos para mejorar el rendimiento del reconocimiento de voz es una tarea difícil. Estos son algunos consejos que puede seguir para asignar pesos:

- Comience con una gramática sin asignar pesos a los elementos.
- Determine qué patrones del discurso se identifican erróneamente con frecuencia.
- Aplique valores diferentes para las ponderaciones hasta que note una mejora en el rendimiento del reconocimiento de voz y no se produzcan regresiones.

Ejemplo 1

Por ejemplo, si tiene una gramática para los aeropuertos y observa que a menudo se identifica erróneamente a Nueva York como Newark, puede darle un valor positivo a Nueva York asignándole una ponderación de 5.

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

Ejemplo 2

Por ejemplo, tiene una gramática para el código de reserva de una aerolínea que comienza con un alfabeto inglés seguido de tres dígitos. Lo más probable es que el código de reserva comience por B o D, pero si observa que con frecuencia B se identifica erróneamente con P y D con T. Puede aumentar positivamente las letras B y D.

```
<rule> id="alphabet">
    <one-of>
        <item>A<tag>out.letters+='A';</tag></item>
        <item weight="3.5">B<tag>out.letters+='B';</tag></item>
        <item>C<tag>out.letters+='C';</tag></item>
        <item weight="2.9">D<tag>out.letters+='D';</tag></item>
        <item>E<tag>out.letters+='E';</tag></item>
        <item>F<tag>out.letters+='F';</tag></item>
        <item>G<tag>out.letters+='G';</tag></item>
        <item>H<tag>out.letters+='H';</tag></item>
        <item>I<tag>out.letters+='I';</tag></item>
        <item>J<tag>out.letters+='J';</tag></item>
        <item>K<tag>out.letters+='K';</tag></item>
        <item>L<tag>out.letters+='L';</tag></item>
        <item>M<tag>out.letters+='M';</tag></item>
        <item>N<tag>out.letters+='N';</tag></item>
        <item>O<tag>out.letters+='O';</tag></item>
        <item>P<tag>out.letters+='P';</tag></item>
        <item>Q<tag>out.letters+='Q';</tag></item>
        <item>R<tag>out.letters+='R';</tag></item>
        <item>S<tag>out.letters+='S';</tag></item>
        <item>T<tag>out.letters+='T';</tag></item>
        <item>U<tag>out.letters+='U';</tag></item>
        <item>V<tag>out.letters+='V';</tag></item>
        <item>W<tag>out.letters+='W';</tag></item>
        <item>X<tag>out.letters+='X';</tag></item>
        <item>Y<tag>out.letters+='Y';</tag></item>
        <item>Z<tag>out.letters+='Z';</tag></item>
    </one-of>
</rule>
```

Formato de script

Amazon Lex V2 admite las siguientes características de ECMAScript para definir gramáticas.

Amazon Lex V2 admite las siguientes características de ECMAScript al especificar etiquetas en la gramática. `tag-format` debe enviarse a `semantics/1.0` cuando se utilicen etiquetas de ECMAScript en la gramática. Para obtener más información, consulte la especificación del lenguaje ECMA-262 ECMAScript 2021.

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

Temas

- Declaración variable
- Expressions
- Declaración IF
- Declaración Switch
- Declaración de acciones
- operación de instrucción
- Declaración de bloque
- Comentarios
- Declaraciones no compatibles

Este documento contiene material del estándar ECMAScript (disponible en). https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ El documento de especificación del lenguaje ECMAScript está disponible en Ecma International con la siguiente licencia.

Texto de licencia

© 2020 Ecma International

Este documento puede copiarse, publicarse y distribuirse a terceros, y algunas obras derivadas del mismo pueden prepararse, copiarse, publicarse y distribuirse, total o parcialmente, siempre que el aviso de derechos de autor anterior y esta licencia de derechos de autor y exención de responsabilidad estén incluidos en todas esas copias y obras derivadas. Las únicas obras derivadas que están permitidas en virtud de esta licencia de derechos de autor y descargo de responsabilidad son:

(i) las obras que incorporan todo o parte de este documento con el fin de proporcionar comentarios o explicaciones (como una versión anotada del documento),

(ii) obras que incorporen todo o parte de este documento con el fin de incorporar características que proporcionen accesibilidad,

(iii) traducciones de este documento a idiomas distintos del inglés y a diferentes formatos y

(iv) funciona haciendo uso de esta especificación en productos que cumplen con los estándares mediante la implementación (por ejemplo, copiando y pegando total o parcialmente) la funcionalidad que contiene.

Sin embargo, el contenido de este documento en sí no puede modificarse de ninguna manera, ni siquiera mediante la eliminación del aviso de derechos de autor o las referencias a Ecma International, excepto cuando sea necesario traducirlo a idiomas distintos del inglés o a un formato diferente.

La versión oficial de un documento de Ecma International es la versión en inglés del sitio web de Ecma International. En caso de discrepancias entre la versión traducida y la versión oficial, prevalecerá la versión oficial.

Los permisos limitados otorgados anteriormente son perpetuos y no serán revocados por Ecma International ni sus sucesores o cesionarios. Este documento y la información aquí contenida se proporcionan «TAL CUAL» y ECMA INTERNATIONAL RENUNCIA A TODAS LAS GARANTÍAS, EXPRESAS O IMPLÍCITAS, INCLUIDA, ENTRE OTRAS, CUALQUIER GARANTÍA DE QUE EL USO DE LA INFORMACIÓN AQUÍ CONTENIDA NO INFRINGIRÁ NINGÚN DERECHO DE PROPIEDAD NI NINGUNA GARANTÍA IMPLÍCITA DE COMERCIABILIDAD O IDONEIDAD PARA UN PROPÓSITO PARTICULAR.

Declaración variable

Una declaración variable define una o más variables.

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

Expressions

Puede agregar cadenas de expresiones para realizar funciones en Amazon Lex V2. En esta tabla se muestran la sintaxis y los ejemplos que se pueden utilizar en las expresiones de SRGS.

Tipo de expresión	Sintaxis	Ejemplo	¿Compatible?
Expresión regular: literal	Cadena literal que contiene caracteres especiales de expresiones regulares válidos	`"^\d\.$"`	No
Función	`function functionName(parameters) { functionBody}`	`var x = function calc() { return 10; }`	No
Eliminar	`delete expression`	`delete obj.property;`	No
Vacío	`void expression`	`void (2 == '2');`	No
Tipo de	`typeof expression`	`typeof 42;`	No
Índice de miembros	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	Sí
Punto de miembro	`expression . identifier`	`out.value`	yes
Argumentos	`expression (arguments)`	`new Date('1994-10-11')`	Sí
Después el incremento	`expression++`	`var x=10; x++;`	Sí

Tipo de expresión	Sintaxis	Ejemplo	¿Compatible?
Después de la disminución	`expression--`	`var x=10; x--;`	Sí
Antes del incremento	`++expression`	`var x=10; ++x;`	Sí
Antes de la disminución	`--expression`	`var x=10; --x;`	Sí
Más unario / menos unario	`+expression / -expression`	`+x / -x;`	Sí
Bit no	`~ expression`	`const a = 5; consol e.log( ~a );`	Sí
Lógico no	`! expression`	`!(a > 0 || b > 0)`	Sí
Multiplicativo	`expression ('*' | '/' | '%') expression`	`(x + y) * (a / b)`	Sí
Aditivo	`expression ('+' | '-') expression`	`(a + b) - (a - (a + b))`	Sí
Cambio de bits	`expression ('<<' | '>>' | '>>>') expression`	`(a >> b) >>> c`	Sí
Relativo	`expression ('<' | '>' | '<=' | '>=') expression`	`if (a > b) { ... }`	Sí

Tipo de expresión	Sintaxis	Ejemplo	¿Compatible?
In	`expression in expression`	`fruits[0] in otherFruits;`	Sí
Igualdad	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	Sí
Bit y / xor / o	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	Sí
Lógico y/o	`expression ('&&' | '||') expression`	`if (a && (b ||c)) {...}`	Sí
Ternario	`expression ? expression : expression`	`a > b ? obj.prop : 0`	Sí
Asignación	`expression = expression`	`out.value = "string";`	Sí
Operador de asignación	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	Sí
Operador de asignación bit a bit	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	Sí

Tipo de expresión	Sintaxis	Ejemplo	¿Compatible?
Identificador	`identifierSequence` donde identifierSequence es una secuencia de caracteres válidos	`fruits=[10, 20, 30];`	Sí
Literal nulo	`null`	`x = null;`	Sí
Literal booleano	`true | false`	`x = true;`	Sí
Una cadena literal.	`'string' / "string"`	`a = 'hello', b = "world";`	Sí
Literal decimal	`integer [.] digits [exponent]`	`111.11 e+12`	Sí
Literal hexadecimal	`0 (x | X)[0-9a-fA-F]`	`0x123ABC`	Sí
Literal octal	`0 [0-7]`	`"051"`	Sí
Matriz literal	`[ expression, ... ]`	`v = [a, b, c];`	Sí
Objeto literal	`{property: value, ...}`	`out = {value: 1, flag: false};`	Sí
Entre paréntesis	`( expressions )`	`x + (x + y)`	Sí

Declaración IF

Puede agregar declaraciones if para realizar funciones en Amazon Lex V2. En este ejemplo, se muestra la sintaxis que se puede utilizar en las expresiones de SRGS.

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

Nota: En el ejemplo anterior, `expressions` y `statements` deben ser uno de los compatibles de este documento.

Declaración Switch

Puede agregar declaraciones switch para realizar funciones en Amazon Lex V2. En este ejemplo, se muestra la sintaxis que se puede utilizar en las expresiones de SRGS.

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

Nota: En el ejemplo anterior, `expressions` y `statements` deben ser uno de los compatibles de este documento.

Declaración de acciones

Puede agregar declaraciones function para realizar funciones en Amazon Lex V2. En este ejemplo, se muestra la sintaxis que se puede utilizar en las expresiones de SRGS.

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

operación de instrucción

Las declaraciones de iteración pueden ser cualquiera de las siguientes:

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


// While Loop
while (expressions) {
   statements
}

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

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

Declaración de bloque

Puede agregar bloques de declaraciones para realizar funciones en Amazon Lex V2. En este ejemplo, se muestra la sintaxis que se puede utilizar en las expresiones de SRGS.

```
{
   statements
}

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

Nota: En el ejemplo anterior, las `statements` proporcionadas en el bloque deben ser una de las compatibles de este documento.

Comentarios

Puede agregar comentarios en Amazon Lex V2. En este ejemplo, se muestra la sintaxis que se puede utilizar en las expresiones de SRGS.

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

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

Declaraciones no compatibles

Amazon Lex V2 no es compatible con las características de ECMAScript.

Temas

- Declaración vacía
- Continuar la declaración
- Declaración de interrupción
- Declaración RETURN
- Declaración THROW
- Declaración de prueba
- Declaración del depurador
- Declaración etiquetada
- Declaración de clase

Declaración vacía

La declaración vacía se utiliza para no proporcionar ninguna declaración. La siguiente es la sintaxis de una sentencia vacía:

```
;
```

Continuar la declaración

La declaración de continuación sin etiqueta es compatible con la operación de instrucción. La declaración de continuación con etiqueta no es compatible.

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

Declaración de interrupción

La declaración de interrupción sin etiqueta es compatible con la operación de instrucción. La declaración de interrupción con etiqueta no es compatible.

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

Declaración RETURN

```
return expression;
```

Declaración THROW

La declaración throw se utiliza para generar una excepción definida por el usuario.

```
throw expression;
```

Declaración de prueba

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

Declaración del depurador

La declaración del depurador se utiliza para invocar la funcionalidad de depuración proporcionada por el entorno.

```
debugger;
```

Declaración etiquetada

La declaración etiquetada se puede utilizar con declaraciones `break` o `continue`.

```
label:
   statements


// Example
let str = '';

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

console.log(str);
```

Declaración de clase

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

Gramáticas industriales

Las gramáticas industriales son un conjunto de archivos XML que se utilizan con el tipo de slot gramatical. Puede utilizarlas para ofrecer rápidamente una experiencia uniforme para el usuario final a medida que migra los flujos de trabajo de respuesta de voz interactiva a Amazon Lex V2. Puede

seleccionar entre una variedad de gramáticas prediseñadas en tres dominios: servicios financieros, seguros y telecomunicaciones. También hay un conjunto genérico de gramáticas que puede utilizar como punto de partida para sus propias gramáticas.

Las gramáticas contienen las reglas para recopilar la información y las etiquetas ECMAScript para la interpretación semántica.

Gramáticas para servicios financieros (download)

Los servicios financieros admiten las siguientes gramáticas: números de cuenta y de ruta, números de tarjetas de crédito y préstamos, calificación crediticia, fechas de apertura y cierre de cuentas y número de seguro social.

Número de cuenta

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


        <!-- Test Cases

        Grammar will support the following inputs:

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

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

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

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

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

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

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

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

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

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

```
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Número de ruta

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

         <!-- Test Cases

         Grammar will support the following inputs:

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

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

         -->

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

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

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

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

```
        </rule>
</grammar>
```

Número de tarjeta de crédito

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

         <!-- Test Cases

         Grammar will support the following inputs:

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

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

         -->

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

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

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

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

ID del préstamo

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

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


        <!-- Test Cases

        Grammar will support the following inputs:

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

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

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

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

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

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

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

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

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

Puntuación de crédito

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

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
             Scenario 1:
                 Input: The number is fifteen
                 Output: 15

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

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

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

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

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

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

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

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

</grammar>
```

Fecha de apertura de la cuenta

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

         <!-- Test Cases

         Grammar will support the following inputs:

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

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

```
                Output: 07/11

         -->

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

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

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

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

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

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

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

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

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

Fecha de pago automática

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

```
         <!-- Test Cases

         Grammar will support the following inputs:

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

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

         -->

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

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

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

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

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

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

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


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

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

Fecha de caducidad de la tarjeta

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

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

        <!-- Test Cases

        Grammar will support the following inputs:

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

            Scenario 2:
```

```
            Input: My card expiration date is may twenty six
            Output: 05 2026

    -->

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

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

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

```
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

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

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

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

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

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

```
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Fecha de la declaración

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

         <!-- Test Cases

         Grammar will support the following inputs:

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

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

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

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

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

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

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

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

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

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

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

```
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

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

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

Fecha de transacción

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

         <!-- Test Cases
```

```
         Grammar will support the following inputs:

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

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

         -->

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

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

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

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

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

```
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

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

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

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

Importe de la transferencia

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

         <!-- Test Cases

         Grammar will support the following inputs:

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

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

         -->

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

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

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

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

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

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

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

```
        </rule>


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

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

Número de la Seguridad Social

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

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

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

Gramáticas de los seguros (download)

En el ámbito de los seguros se admiten las siguientes gramáticas: números de reclamación y póliza, números de carné de conducir y matrícula, fechas de caducidad, fechas de inicio y fechas de renovación, importes de las reclamaciones y pólizas.

ID de reclamación

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My claim number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: Claim number One Five Four Four
                 Output: 1544

         -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My claim number is</item>
                <item repeat="0-1">Claim number</item>
                <item repeat="0-1">This is for claim</item>
            </one-of>
        </rule>

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

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

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

ID de política

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

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


        <!-- Test Cases

        Grammar will support the following inputs:

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

            Scenario 2:
                Input: This is the policy number 1 2 3 4 A B C
                Output: 1234ABC

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

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy number is</item>
                <item repeat="0-1">This is the policy number</item>
                <item repeat="0-1">Policy number</item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My policy Id is</item>
                <item repeat="0-1">This is the policy Id</item>
```

```
                <item repeat="0-1">Policy Id</item>
            </one-of>
        </rule>

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

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

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

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
```

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

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

Números de permiso de conducción

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My drivers license number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: driver license number One Five Four Four
                 Output: 1544

         -->

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

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

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

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

      <rule id="singleDigit">
          <item repeat="0-1"><ruleref uri="#text"/></item>
          <tag>out.digit=""</tag>
          <item repeat="1-10">
              <one-of>
                  <item>0<tag>out.digit+=0;</tag></item>
                  <item>zero<tag>out.digit+=0;</tag></item>
                  <item>1<tag>out.digit+=1;</tag></item>
                  <item>one<tag>out.digit+=1;</tag></item>
                  <item>2<tag>out.digit+=2;</tag></item>
                  <item>two<tag>out.digit+=2;</tag></item>
                  <item>3<tag>out.digit+=3;</tag></item>
                  <item>three<tag>out.digit+=3;</tag></item>
                  <item>4<tag>out.digit+=4;</tag></item>
                  <item>four<tag>out.digit+=4;</tag></item>
                  <item>5<tag>out.digit+=5;</tag></item>
                  <item>five<tag>out.digit+=5;</tag></item>
                  <item>6<tag>out.digit+=6;</tag></item>
                  <item>six<tag>out.digit+=5;</tag></item>
                  <item>7<tag>out.digit+=7;</tag></item>
                  <item>seven<tag>out.digit+=7;</tag></item>
                  <item>8<tag>out.digit+=8;</tag></item>
                  <item>eight<tag>out.digit+=8;</tag></item>
                  <item>9<tag>out.digit+=9;</tag></item>
                  <item>nine<tag>out.digit+=9;</tag></item>
              </one-of>
          </item>
      </rule>
</grammar>
```

Número de placa

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

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


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: my license plate is A B C D 1 2
                Output: ABCD12

            Scenario 2:
                Input: license plate number A B C 1 2 3 4
                Output: ABC1234

            Scenario 3:
                Input: my plates say A F G K 9 8 7 6 Thanks
                Output: AFGK9876
        -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my license plate is</item>
                <item repeat="0-1">license plate number</item>
                <item repeat="0-1">my plates say</item>
            </one-of>
        </rule>

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

```
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="3-4">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
```

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

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

Fecha de caducidad de la tarjeta

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

        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
```

```
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

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

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

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

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

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
```

```
            <item>april<tag>out="04";</tag></item>
            <item>may<tag>out="05";</tag></item>
            <item>june<tag>out="06";</tag></item>
            <item>july<tag>out="07";</tag></item>
            <item>august<tag>out="08";</tag></item>
            <item>september<tag>out="09";</tag></item>
            <item>october<tag>out="10";</tag></item>
            <item>november<tag>out="11";</tag></item>
            <item>december<tag>out="12";</tag></item>
            <item>jan<tag>out="01";</tag></item>
            <item>feb<tag>out="02";</tag></item>
            <item>aug<tag>out="08";</tag></item>
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

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

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

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

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

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

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

Fecha de caducidad de la póliza, /año day/month

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

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
            Scenario 1:
                Input: My policy expired on July Five Two Thousand and Eleven
                Output: 07/5/11

            Scenario 2:
                Input: My policy will expire on July Sixteen Two Thousand and Eleven
                Output: 07/16/11

            Scenario 3:
                Input: My policy expired on July Thirty Two Thousand and Eleven
                Output: 07/30/11
        -->

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

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My policy expired on</item>
```

```
            <item repeat="0-1">My policy will expire on</item>
         </one-of>
      </rule>

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

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

       <rule id="digits">
           <one-of>
               <item>zero<tag>out=0;</tag></item>
               <item>one<tag>out=1;</tag></item>
               <item>two<tag>out=2;</tag></item>
               <item>three<tag>out=3;</tag></item>
```

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

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

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

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

```
</grammar>
```

Fecha de renovación de la póliza, month/year

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I renewed my policy on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: My policy will renew on July Two Thousand and Eleven
                 Output: 07/11

         -->

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

```
        </item>
    </rule>

    <rule id="text">
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
        <one-of>
            <item repeat="0-1">My policy will renew on</item>
            <item repeat="0-1">My policy was renewed on</item>
            <item repeat="0-1">Renew policy on</item>
            <item repeat="0-1">I renewed my policy on</item>
        </one-of>
    </rule>

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

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

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

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

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

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

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

Fecha de inicio de la política

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I bought my policy on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My policy started on july fifteen
                 Output: 07/15

         -->

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

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">I bought my policy on</item>
                <item repeat="0-1">I bought policy on</item>
                <item repeat="0-1">My policy started on</item>
             </one-of>
          </rule>

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

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
```

```
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

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

```
        </rule>

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

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

Importe de la reclamación

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

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to make a claim of one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: Requesting claim of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to place a claim for</item>
              <item repeat="0-1">I want to make a claim of</item>
              <item repeat="0-1">I assess damage of</item>
              <item repeat="0-1">Requesting claim of</item>
           </one-of>
        </rule>
```

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

          <rule id="thanks">
              <one-of>
                 <item>Thanks</item>
                 <item>I think</item>
              </one-of>
            </rule>

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

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
```

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

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

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

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

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

Monto de la prima

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

         <!-- Test Cases
```

```
         Grammar will support the following inputs:

             Premium amounts
             Scenario 1:
                 Input: The premium for one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: RPremium amount of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">A premium of</item>
              <item repeat="0-1">Premium amount of</item>
              <item repeat="0-1">The premium for</item>
              <item repeat="0-1">Insurance premium for</item>
           </one-of>
        </rule>

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

          <rule id="thanks">
              <one-of>
```

```
            <item>Thanks</item>
            <item>I think</item>
          </one-of>
        </rule>

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

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

```
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

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

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


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

```
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

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

Cantidad de la póliza

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: I want policy for ten
                 Output: 10
```

```
         -->

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

        <rule id="text">
           <one-of>
              <item repeat="0-1">I want policy for</item>
              <item repeat="0-1">I want to order policy for</item>
              <item repeat="0-1">The number is</item>
           </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

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

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

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

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
            <item>forty<tag>out=40;</tag></item>
            <item>fifty<tag>out=50;</tag></item>
```

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

</grammar>
```

Gramáticas para telecomunicaciones (download)

Las siguientes gramáticas son compatibles con las telecomunicaciones: número de teléfono, número de serie, número de SIM, código postal estadounidense, fecha de caducidad de la tarjeta de crédito, inicio del plan, fechas de renovación y caducidad, fecha de inicio del servicio, cantidad de equipos e importe de la factura.

Número de teléfono

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

         <!-- Test Cases

         Grammar will support 10-12 digits number and here are couple of examples of
 valid inputs:
```

```
             Scenario 1:
                 Input: Mmm My phone number is two zero one two five two six seven
 eight five
                 Output: 2012526785

             Scenario 2:
                 Input: My phone number is two zero one two five two six seven eight
 five
                 Output: 2012526785

         -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My phone number is</item>
              <item repeat="0-1">Phone number is</item>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">Yes, it's</item>
              <item repeat="0-1">Yes, it is</item>
              <item repeat="0-1">Yes it is</item>
           </one-of>
        </rule>

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

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="10-12">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
```

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

Número de serie

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My serial number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456
```

```
             Scenario 2:
                 Input: Device Serial number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

          -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My serial number is</item>
                <item repeat="0-1">Device Serial number</item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">The IMEI number is</item>
            </one-of>
        </rule>

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

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="15">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
```

```
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Número de SIM

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


        <!-- Test Cases

        Grammar will support the following inputs:

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

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

            Scenario 3:
```

```
                Input: My SIM number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

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

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

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

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

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

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

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

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

Código postal de EE. UU.

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

         <!-- Test Cases

         Grammar will support 5 digits code and here are couple of examples of valid
 inputs:

             Scenario 1:
                 Input: Mmmm My zipcode is umm One Oh Nine Eight Seven
                 Output: 10987

             Scenario 2:
                 Input: My zipcode is One Oh Nine Eight Seven
                 Output: 10987

          -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
```

```
        <one-of>
           <item repeat="0-1">My zipcode is</item>
           <item repeat="0-1">Zipcode is</item>
           <item repeat="0-1">It is</item>
        </one-of>
    </rule>

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

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

```
</grammar>
```

Fecha de caducidad de la tarjeta

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

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

        <!-- Test Cases

        Grammar will support the following inputs:

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

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

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

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

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
               <item>april<tag>out="04";</tag></item>
               <item>may<tag>out="05";</tag></item>
               <item>june<tag>out="06";</tag></item>
               <item>july<tag>out="07";</tag></item>
               <item>august<tag>out="08";</tag></item>
               <item>september<tag>out="09";</tag></item>
               <item>october<tag>out="10";</tag></item>
               <item>november<tag>out="11";</tag></item>
               <item>december<tag>out="12";</tag></item>
               <item>jan<tag>out="01";</tag></item>
               <item>feb<tag>out="02";</tag></item>
               <item>aug<tag>out="08";</tag></item>
               <item>sept<tag>out="09";</tag></item>
               <item>oct<tag>out="10";</tag></item>
               <item>nov<tag>out="11";</tag></item>
               <item>dec<tag>out="12";</tag></item>
               <item>1<tag>out="01";</tag></item>
               <item>2<tag>out="02";</tag></item>
               <item>3<tag>out="03";</tag></item>
               <item>4<tag>out="04";</tag></item>
               <item>5<tag>out="05";</tag></item>
               <item>6<tag>out="06";</tag></item>
               <item>7<tag>out="07";</tag></item>
               <item>8<tag>out="08";</tag></item>
               <item>9<tag>out="09";</tag></item>
               <item>ten<tag>out="10";</tag></item>
               <item>eleven<tag>out="11";</tag></item>
               <item>twelve<tag>out="12";</tag></item>
           </one-of>
       </rule>
```

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

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

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
```

```
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

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

Fecha de caducidad del plan, /año day/month

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan expires on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: My plan will expire on July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: My plan will expire on July Thirty Two Thousand and Eleven
                 Output: 07/30/11
         -->

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

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

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
          <one-of>
               <item repeat="0-1">My plan expires on</item>
               <item repeat="0-1">My plan expired on</item>
               <item repeat="0-1">My plan will expire on</item>
           </one-of>
        </rule>

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

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

           <one-of>
             <item>january<tag>out.mon+="01";</tag></item>
             <item>february<tag>out.mon+="02";</tag></item>
             <item>march<tag>out.mon+="03";</tag></item>
             <item>april<tag>out.mon+="04";</tag></item>
             <item>may<tag>out.mon+="05";</tag></item>
             <item>june<tag>out.mon+="06";</tag></item>
             <item>july<tag>out.mon+="07";</tag></item>
             <item>august<tag>out.mon+="08";</tag></item>
             <item>september<tag>out.mon+="09";</tag></item>
             <item>october<tag>out.mon+="10";</tag></item>
```

```
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

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

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

    <rule id="thousands">
        <item>two thousand</item>
```

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

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

Fecha de renovación del plan, month/year

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will renew on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: Renew plan on July Two Thousand and Eleven
                 Output: 07/11
```

```
         -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My plan will renew on</item>
                <item repeat="0-1">My plan was renewed on</item>
                <item repeat="0-1">Renew plan on</item>
            </one-of>
        </rule>

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

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

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

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

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

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

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

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

Fecha de inicio del plan, month/day

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

```
         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will start on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My plan will start on july fifteen
                 Output: 07/15

         -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan started on</item>
                <item repeat="0-1">My plan will start on</item>
                <item repeat="0-1">I paid it on</item>
                <item repeat="0-1">I paid bill for</item>
             </one-of>
          </rule>

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

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

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

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
```

```
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

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

```
        </rule>

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

Fecha de inicio del servicio, month/day

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan starts on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: I want to activate on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
```

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan starts on</item>
                <item repeat="0-1">I want to start my plan on</item>
                <item repeat="0-1">Activation date of</item>
                <item repeat="0-1">Start activation on</item>
                <item repeat="0-1">I want to activate on</item>
                <item repeat="0-1">Activate plan starting</item>
                <item repeat="0-1">Starting</item>
                <item repeat="0-1">Start on</item>
             </one-of>
          </rule>

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

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
```

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

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

```
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

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

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

Cantidad de equipo

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

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: It is ten
                 Output: 10

         -->

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

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">It is</item>
              <item repeat="0-1">The number is</item>
```

```
            <item repeat="0-1">Order</item>
            <item repeat="0-1">I want to order</item>
            <item repeat="0-1">Total equipment</item>
         </one-of>
      </rule>

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

      <rule id="thanks">
          <one-of>
             <item>Thanks</item>
             <item>I think</item>
          </one-of>
        </rule>

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

```
        </rule>

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

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

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

</grammar>
```

Importe de la factura

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

         <!-- Test Cases

         Grammar will support the following inputs:

                 Input: I want to make a payment of one hundred ten dollars
                 Output: $110

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
```

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

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

  <rule id="thanks">
      <one-of>
         <item>Thanks</item>
         <item>I think</item>
      </one-of>
    </rule>

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

```
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

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

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

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


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

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

Gramáticas genéricas (download)

Ofrecemos las siguientes gramáticas genéricas: alfanumérica, moneda, fecha (mm/dd/yy), números, saludo, vacilación y agente.

Alfanumérico

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


        <!-- Test Cases

            Scenario 1:
                Input: A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

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

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

```
        </rule>

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

        <rule id="digits">
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
```

```
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Currency (Divisa)

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

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

        <rule id="digits">
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
```

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

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

    <rule id="above_twenty">
        <tag>out.tens = 0;</tag>
        <one-of>
            <item>twenty<tag>out.tens+=20;</tag></item>
            <item>thirty<tag>out.tens+=30;</tag></item>
            <item>forty<tag>out.tens+=40;</tag></item>
            <item>fifty<tag>out.tens+=50;</tag></item>
            <item>sixty<tag>out.tens+=60;</tag></item>
```

```
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

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


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

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

```
</grammar>
```

Fecha, dd/mm

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

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

        <rule id="months">
           <one-of>
             <item>january<tag>out="january";</tag></item>
             <item>february<tag>out="february";</tag></item>
             <item>march<tag>out="march";</tag></item>
             <item>april<tag>out="april";</tag></item>
             <item>may<tag>out="may";</tag></item>
             <item>june<tag>out="june";</tag></item>
             <item>july<tag>out="july";</tag></item>
             <item>august<tag>out="august";</tag></item>
             <item>september<tag>out="september";</tag></item>
             <item>october<tag>out="october";</tag></item>
             <item>november<tag>out="november";</tag></item>
```

```
            <item>december<tag>out="december";</tag></item>
            <item>jan<tag>out="january";</tag></item>
            <item>feb<tag>out="february";</tag></item>
            <item>aug<tag>out="august";</tag></item>
            <item>sept<tag>out="september";</tag></item>
            <item>oct<tag>out="october";</tag></item>
            <item>nov<tag>out="november";</tag></item>
            <item>dec<tag>out="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=1;</tag></item>
            <item>second<tag>out=2;</tag></item>
            <item>third<tag>out=3;</tag></item>
            <item>fourth<tag>out=4;</tag></item>
            <item>fifth<tag>out=5;</tag></item>
            <item>sixth<tag>out=6;</tag></item>
            <item>seventh<tag>out=7;</tag></item>
            <item>eighth<tag>out=8;</tag></item>
            <item>ninth<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>
```

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

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

Fecha, mm/yy

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

```
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

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

        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
               <item>december<tag>out.mon+="december";</tag></item>
               <item>jan<tag>out.mon+="january";</tag></item>
               <item>feb<tag>out.mon+="february";</tag></item>
               <item>aug<tag>out.mon+="august";</tag></item>
               <item>sept<tag>out.mon+="september";</tag></item>
               <item>oct<tag>out.mon+="october";</tag></item>
               <item>nov<tag>out.mon+="november";</tag></item>
```

```
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

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

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

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

    <rule id="thousands">
        <!-- <item>
            <ruleref uri="#digits"/>
            <tag>out = (1000 * rules.digits);</tag>
            thousand
        </item> -->
```

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

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

</grammar>
```

Fecha, dd/mm /aaaa

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

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item repeat="1-10">
                 <one-of>
```

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

        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
               <item>december<tag>out.mon+="december";</tag></item>
               <item>jan<tag>out.mon+="january";</tag></item>
               <item>feb<tag>out.mon+="february";</tag></item>
               <item>aug<tag>out.mon+="august";</tag></item>
               <item>sept<tag>out.mon+="september";</tag></item>
               <item>oct<tag>out.mon+="october";</tag></item>
               <item>nov<tag>out.mon+="november";</tag></item>
               <item>dec<tag>out.mon+="december";</tag></item>
```

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

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

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

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

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

 </grammar>
```

Números, dígitos

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

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

        <rule id="singleDigit">
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
```

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

Números ordinales

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

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

```
        </rule>

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

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

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

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

</grammar>
```

Agente

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

```
        <rule id="main" scope="public">
            <tag>out=""</tag>
            <ruleref uri="#text"/><tag>out = rules.text</tag>
        </rule>

        <rule id="text">
          <one-of>
             <item>Can I talk to the agent<tag>out="You will be trasnfered to the
 agent in a while"</tag></item>
             <item>talk to an agent<tag>out="You will be trasnfered to the agent in a
 while"</tag></item>
          </one-of>
        </rule>
</grammar>
```

Saludo

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

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>hey<tag>out="Greeting"</tag></item>
              <item>hi<tag>out="Greeting"</tag></item>
              <item>Hi<tag>out="Greeting"</tag></item>
              <item>Hey<tag>out="Greeting"</tag></item>
              <item>Hello<tag>out="Greeting"</tag></item>
              <item>hello<tag>out="Greeting"</tag></item>
           </one-of>
         </rule>
```

```
</grammar>
```

Vacilación

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

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>Hmm<tag>out="Waiting for your input"</tag></item>
              <item>Mmm<tag>out="Waiting for your input"</tag></item>
              <item>Can you please wait<tag>out="Waiting for your input"</tag></item>
           </one-of>
         </rule>
</grammar>
```

Tipos de slot compuestos

Un slot compuesto es una combinación de dos o más slot que capturan varios datos en una sola entrada de usuario. Por ejemplo, puede configurar el bot para que obtenga la ubicación solicitando la «ciudad y estado o código postal». Por el contrario, cuando la conversación se configura para utilizar distintos tipos de slots, se genera una experiencia de conversación rígida («¿Cuál es la ciudad?» seguido de «¿Cuál es el código postal?»). Con un slot compuesto, puede capturar toda la información a través de un solo slot. Un slot compuesto es una combinación de slots denominados subslots, como la ciudad, el estado y el código postal.

Puede usar una combinación de los tipos de slots de Amazon Lex disponibles (integrados) y sus propios slots (slots personalizados). Puede diseñar expresiones lógicas para capturar información dentro de los subslots requeridos. Por ejemplo: ciudad y estado o código postal.

El tipo de ranura compuesta solo está disponible en EE. UU. y en lugares con funciones limitadas a través de la IA generativa.

Crear un tipo de slot compuesto

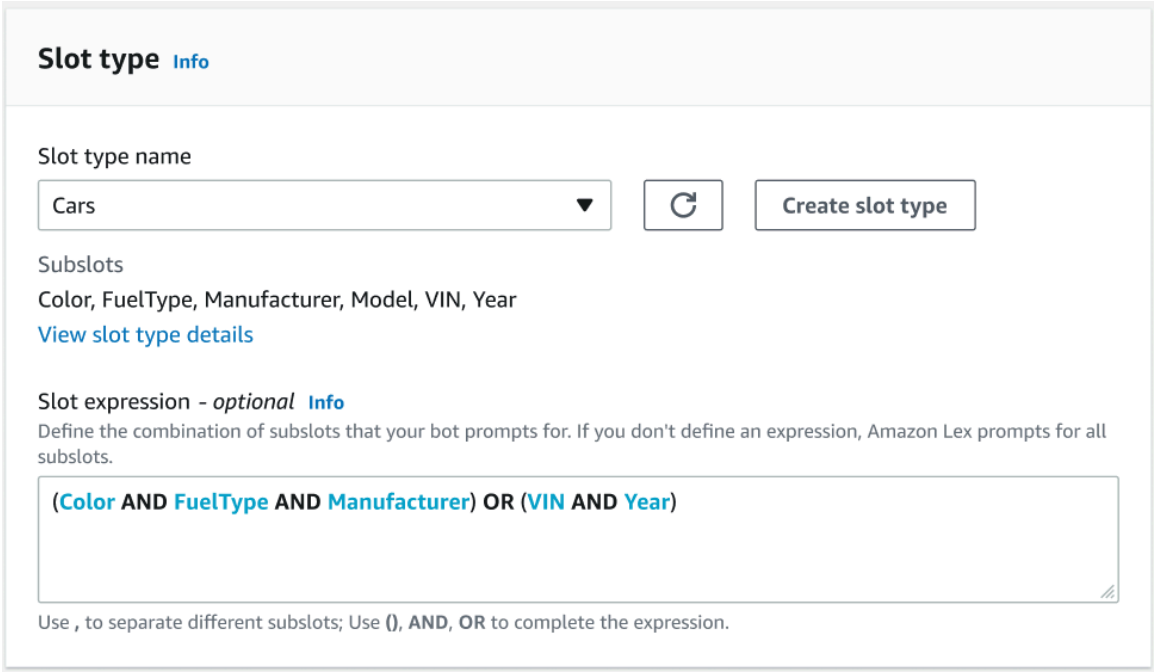
Para utilizar subslots dentro de un slot compuesto, primero debe configurar el tipo de slot compuesto. Para ello, use los pasos de la consola para agregar un tipo slot o la operación de la API. Una vez que haya elegido el nombre y la descripción del tipo de slot compuesto, debe proporcionar información sobre los subslots. Para obtener más información acerca de cómo agregar un tipo de slot, consulte Adición de tipos de ranura

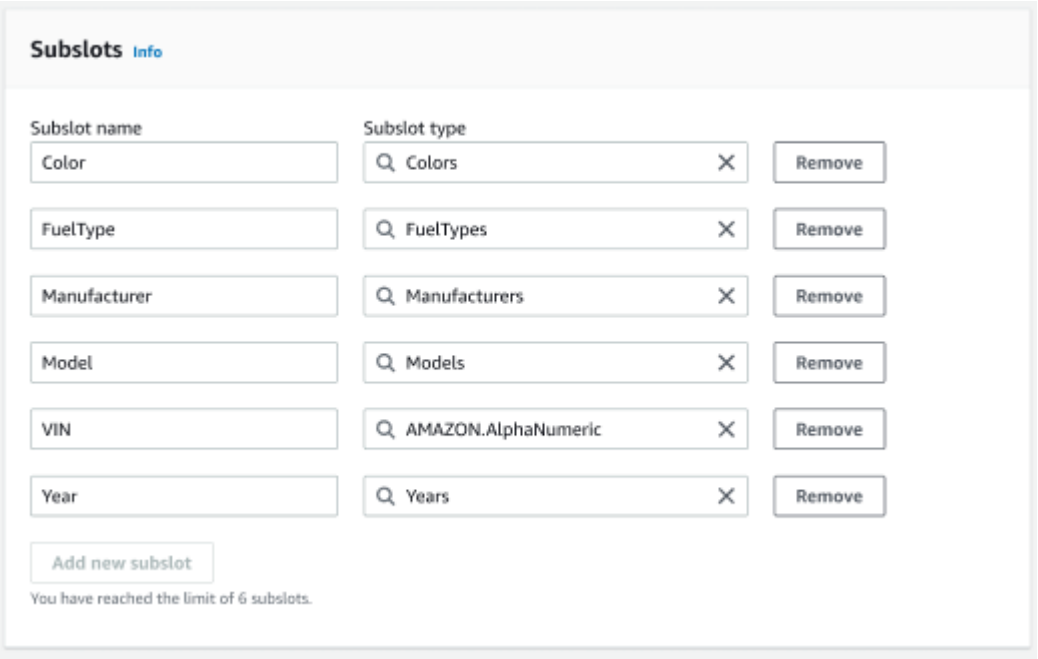
Subslots

Un tipo de slot compuesto requiere la configuración de los slots subyacentes, denominados subslots. Si desea obtener varios datos de un cliente en una sola solicitud, configure una combinación de subslots. Por ejemplo: ciudad, estado y código postal. Puede agregar hasta 6 subslots a un slot compuesto.

Se pueden usar slots de tipos de slots individuales para añadir subslots al tipo de slot compuesto. Sin embargo, no puede utilizar un tipo de slot compuesto como tipo de slot para un subslot.

En las imágenes siguientes se muestra una imagen de una ranura compuesta llamada «Coche», que es una combinación de subranuras: color FuelType, fabricante, modelo, VIN y año.





Generador de expresiones

Para impulsar el cumplimiento de un slot compuesto, puede utilizar opcionalmente el generador de expresiones. Con el generador de expresiones, puede diseñar una expresión de slot lógico para capturar los valores de subslot necesarios en el orden deseado. Como parte de la expresión booleana, puede utilizar operadores como Y y O. Según la expresión diseñada, cuando se cumplen los subslots requeridos, se considera que se cumple el slot compuesto.

Usar un tipo de slot compuesto

Para algunas intenciones, es posible que desee capturar diferentes slots como parte de un solo slot. Por ejemplo, un bot para programar el mantenimiento del coche podría tener una intención con el siguiente enunciado:

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

La intención es que el slot compuesto {de coche} contenga una lista de los slots, que incluyan los detalles del coche. Por ejemplo, «Toyota Camry 2021 blanco».

El slot compuesto se diferencia de un slot con varios valores. El slot compuesto se compone de varios slots, cada uno con su propio valor. Mientras que un slot con varios valores es un slot individual que puede contener una lista de valores. Para obtener más información sobre los slots con valores múltiples, consulte Uso de valores múltiples en un slot.

En el caso de un slot compuesto, Amazon Lex devuelve un valor para cada subslot en respuesta a la operación `RecognizeText` o `RecognizeUtterance`. La siguiente es la información sobre las franjas horarias que aparece al pronunciar el mensaje: «Quiero programar una revisión para mi «Toyota Camry 2021 blanco» por parte del bot. CarService

```
"slots": {
    "CarType": {
        "value": {
            "originalValue": "White Toyota Camry 2021",
            "interpretedValue": "White Toyota Camry 2021",
            "resolvedValues": [
                "white Toyota Camry 2021"
            ]
        },
        "subSlots": {
            "Color": {
                "value": {
                    "originalValue": "White",
                    "interpretedValue": "White",
                    "resolvedValues": [
                        "white"
                    ]
                },
                "shape": "Scalar"
            },
            "Manufacturer": {
                "value": {
                    "originalValue": "Toyota",
                    "interpretedValue": "Toyota",
                    "resolvedValues": [
                        "Toyota"
                    ]
                },
                "shape": "Scalar"
            },
            "Model": {
                "value": {
                    "originalValue": "Camry",
                    "interpretedValue": "Camry",
                    "resolvedValues": [
                       "Camry"
                    ]
                },
```

```
                "shape": "Scalar"
            },
            "Year": {
                "value": {
                    "originalValue": "2021",
                    "interpretedValue": "2021",
                    "resolvedValues": [
                        "2021"
                    ]
                },
                "shape": "Scalar"
            }
        }
    },
    ...
}
```

Se puede elegir un slot compuesto en el primer turno o en el enésimo turno de una conversación. En función de los valores de entrada suministrados, el slot compuesto puede generar el resto de subslots requeridos.

Los slots compuestos siempre devuelven un valor para cada subslot. Cuando el enunciado no contiene un valor reconocible para un subslot determinado, no se devuelve ninguna respuesta para ese subslot en particular.

Los slots compuestos funcionan tanto con la entrada de texto como con la de voz.

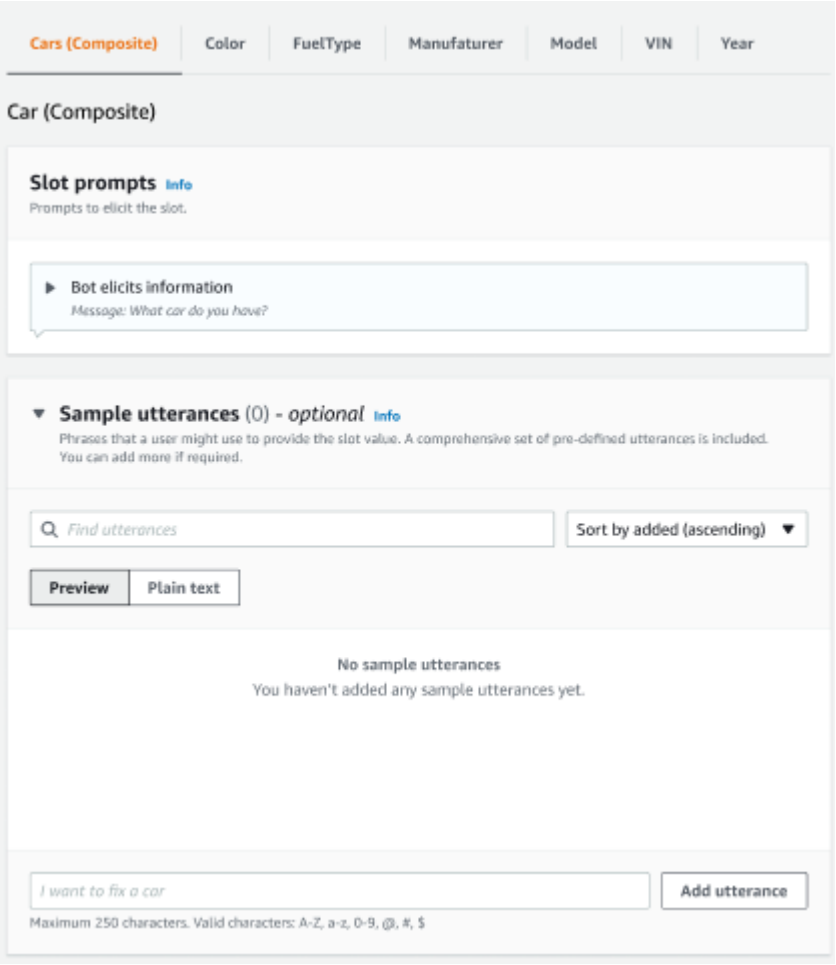
Al agregar un slot a un objeto, un slot compuesto solo está disponible como tipo de slot personalizado.

Puede usar slots compuestos en las indicaciones. Por ejemplo, puede establecer el mensaje de confirmación para una intención.

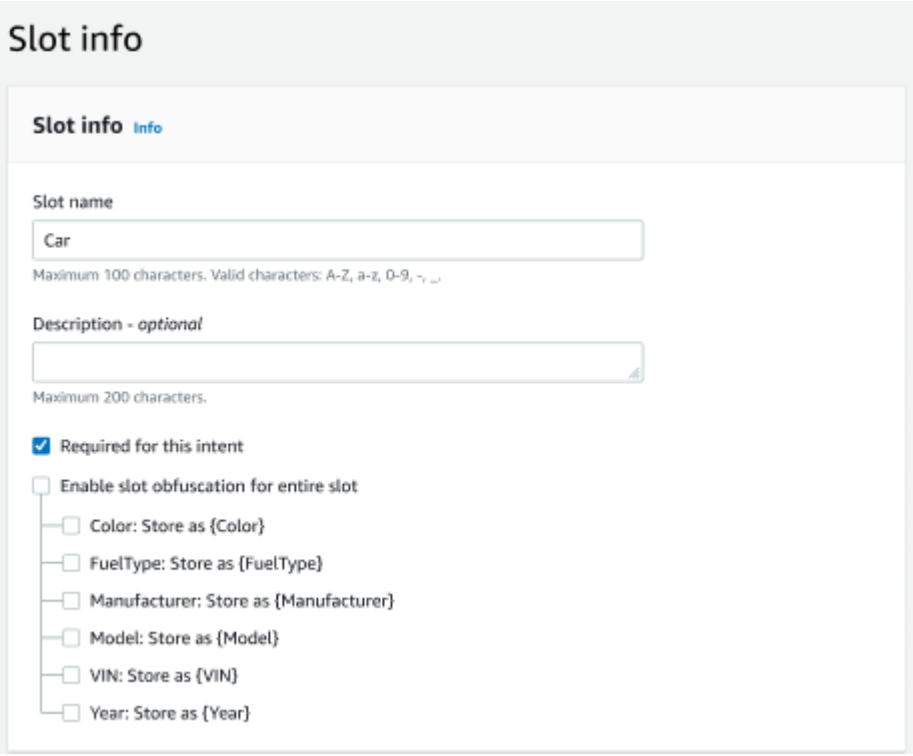
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

Cuando Amazon Lex envía el mensaje al usuario, dice: «¿Quieres que programe el servicio para tu Toyota Camry blanco de 2021?»

Cada subslot está configurado como un slot. Puede añadir mensajes de slots para obtener el subslot y los enunciados de muestra. Puede activar la opción de esperar y continuar para un subslot, así como los valores predeterminados. Para obtener más información, consulte Uso de los valores de ranura predeterminados en las intenciones de su bot Lex V2



Puede utilizar la ofuscación de slots para ocultar todo el slot compuesto en los registros de conversaciones. Tenga en cuenta que la ofuscación de slots se aplica a nivel de slot compuesta y, cuando está habilitada, se ocultan los valores de los subslots que pertenecen a un slot compuesto. Cuando elige ofuscar los valores de slot, el valor de cada valor de slot se reemplaza con el nombre del slot. Para obtener más información, consulte Cómo ocultar valores de ranura en registros de conversación de Lex V2.



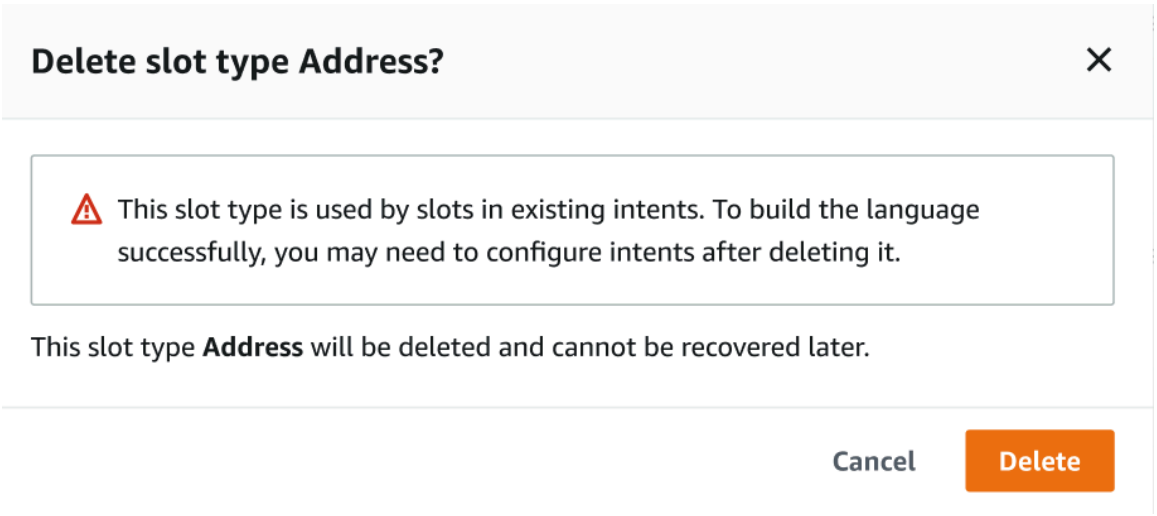
Editar un tipo de slot compuesto

Puede editar un subslot desde la configuración de slot compuesto para modificar el nombre y el tipo de slot secundario. Sin embargo, cuando se utilice un slot compuesto por una intención, tendrá que editar la intención antes de modificar el subslot.

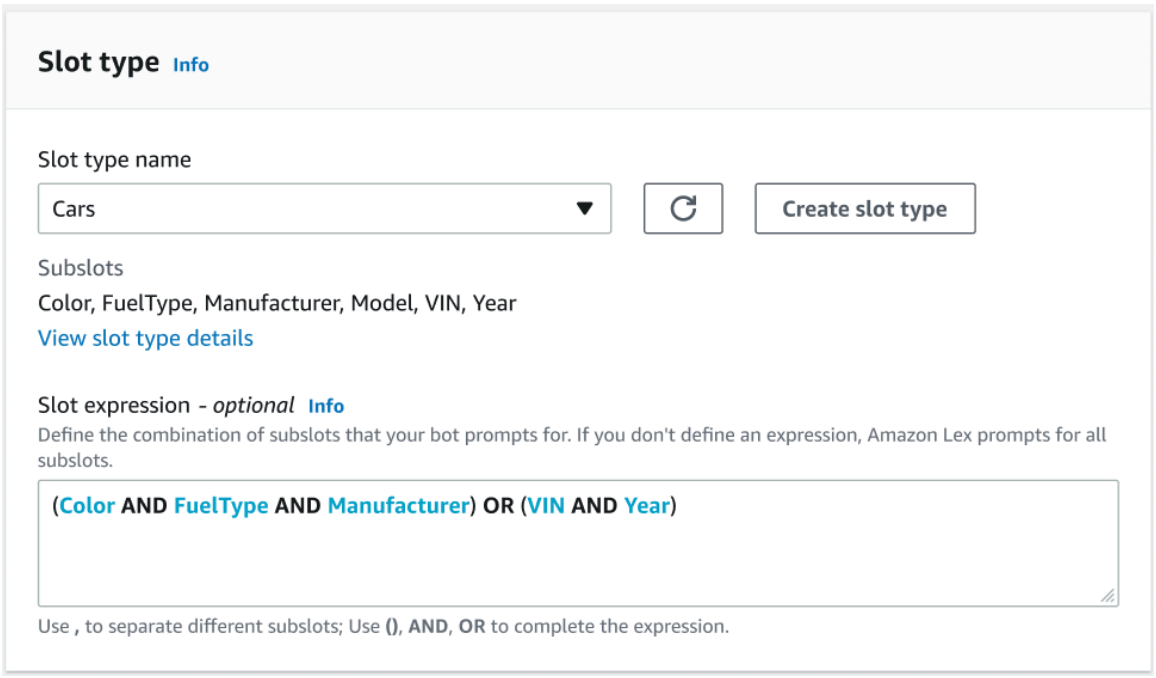
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

Eliminar un tipo de slot compuesto

Puede eliminar un subslot desde la configuración de slots compuestos. Tenga en cuenta que cuando se utiliza un subslot dentro de una intención, los subslots siguen siendo eliminados de esa intención.



La expresión de slot del generador de expresiones proporciona una alerta para informar sobre los subslots eliminados.



Probar un bot con la consola

La consola de Amazon Lex V2 contiene una ventana de prueba que puede utilizar para probar la interacción con su bot. Utilice la ventana de prueba para mantener una conversación de prueba con su bot y ver las respuestas que su aplicación recibe del bot.

Hay dos tipos de pruebas que se pueden realizar con su bot. La primera, la prueba rápida, le permite probar su bot con las frases exactas que utilizó para crearlo. Por ejemplo, si agregó el enunciado «Quiero recoger flores» a su intención, puede probar el bot con esa frase exacta.

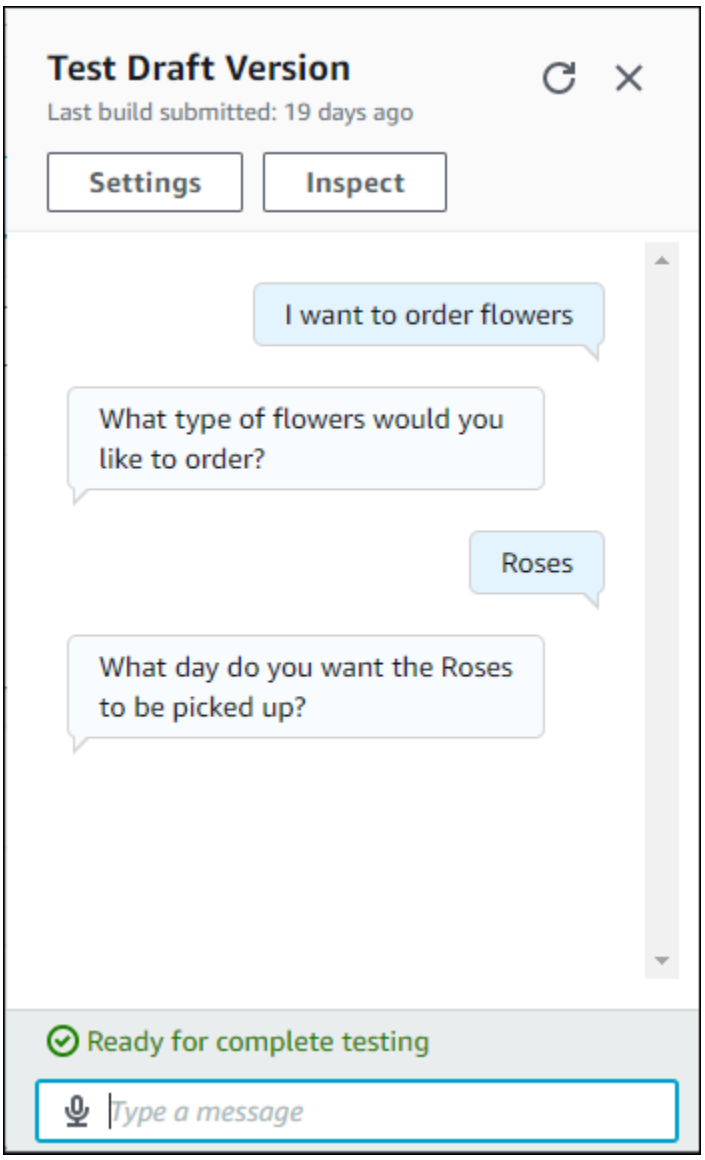
El segundo tipo, la prueba completa, le permite probar el bot con frases relacionadas con los enunciados que ha configurado. Por ejemplo, puede usar la frase «¿Puedo pedir flores?» para iniciar una conversación con su bot.

Para probar un bot, utilice un alias y un idioma específicos. Si está probando la versión de desarrollo del bot, utilice el alias `TestBotAlias` para realizar las pruebas.

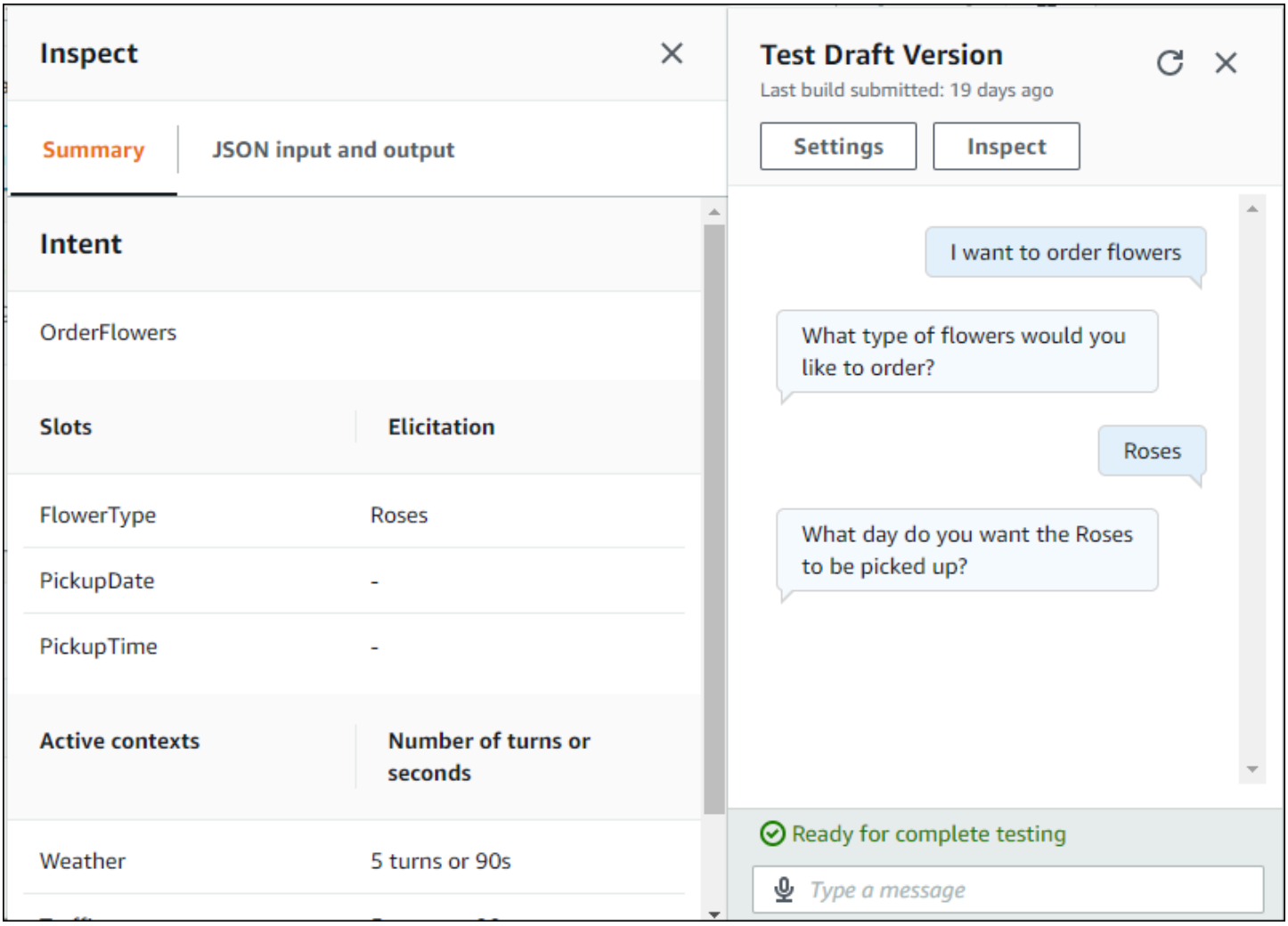
Abrir la ventana de prueba

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot para probar de la lista de bots.
3. En el menú izquierdo, seleccione Alias.
4. De la lista de alias, seleccione el alias que desea probar.
5. En Idiomas, seleccione el botón de radio del idioma que quiere probar y, a continuación, seleccione Probar.

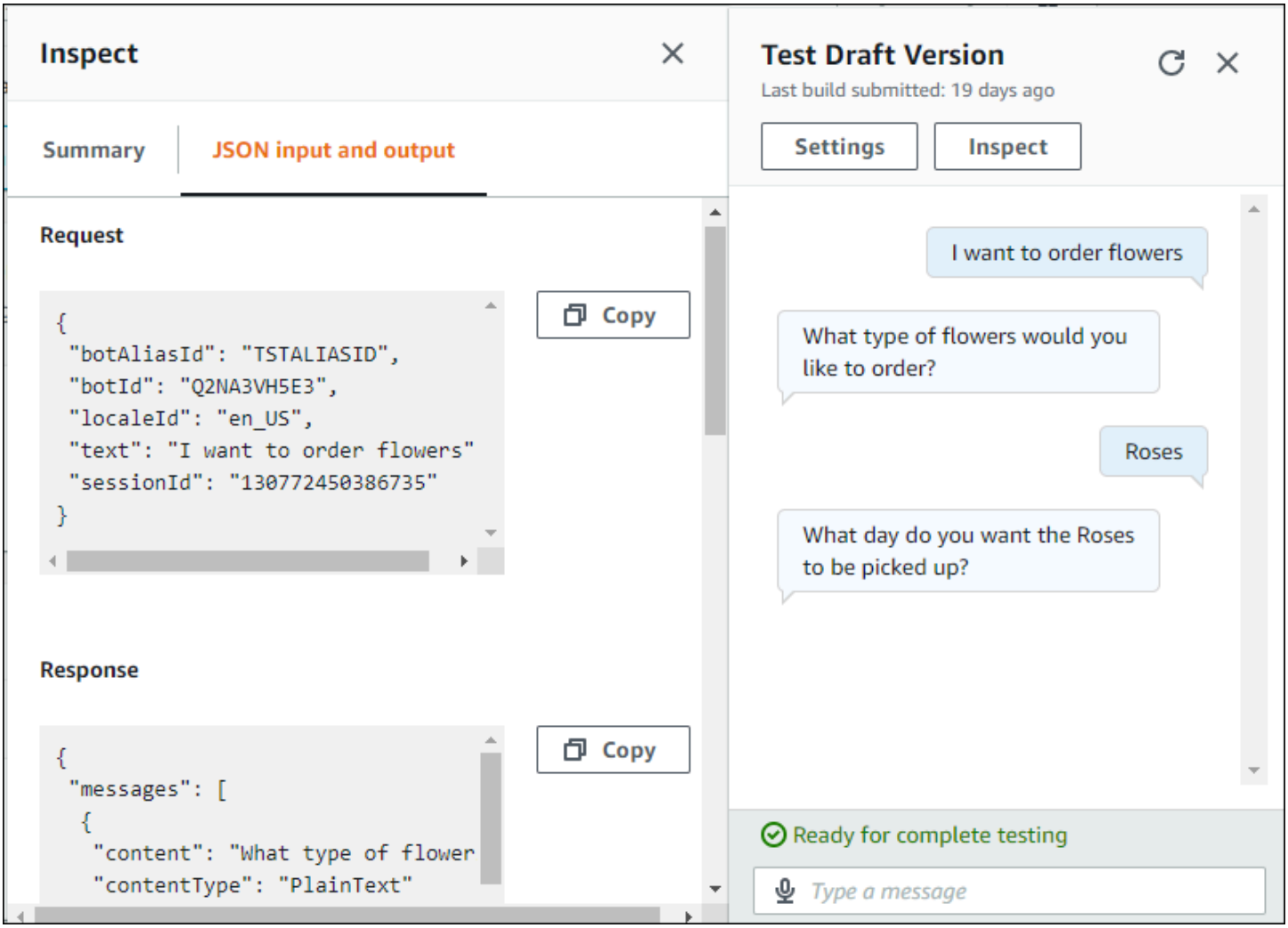
Tras seleccionar Probar, se abre la ventana de prueba en la consola. También puede usar la ventana de prueba para interactuar con su bot, tal como se muestra en el siguiente gráfico.



Además de la conversación, también puede seleccionar Inspeccionar en la ventana de prueba para ver las respuestas devueltas por el bot. La primera vista muestra un resumen de la información devuelta por el bot a la ventana de prueba.



También puede usar la ventana de inspección de pruebas para ver las estructuras JSON que se envían entre el bot y la ventana de prueba. Puede ver tanto la solicitud desde la ventana de prueba como la respuesta de Amazon Lex V2.


Inspect
Summary
JSON input and output
Request
{
"botAliasId": "TSTALIASID",
"botId": "Q2NA3VH5E3",
"localeId": "en_US",
"text": "I want to order flowers"
"sessionId": "130772450386735"
}
Copy
Response
{
"messages": [
{
"content": "What type of flower
"contentType": "PlainText"
Copy
Test Draft Version
Last build submitted: 19 days ago
Settings
Inspect
I want to order flowers
What type of flowers would you like to order?
Roses
What day do you want the Roses to be picked up?
Ready for complete testing
Type a message

Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa

Aproveche las capacidades de la IA generativa de Amazon Bedrock para automatizar y acelerar el proceso de creación de bots de Amazon Lex V2. Puede llevar a cabo los siguientes procesos con ayuda de Amazon Bedrock.

> **Note**
>
> Estas características utilizan IA generativa. Cuando utilice el servicio, recuerde que puede dar respuestas inexactas o inapropiadas. Para obtener más información, consulte AWS Responsible AI Policy.
> Desarrollado por Amazon Bedrock: AWS implementa la detección automática de abusos. Dado que las características de IA generativa de Amazon Lex V2 se basan en Amazon Bedrock, los usuarios heredan los controles implementados en Amazon Bedrock para garantizar la protección, la seguridad y el uso responsable de la IA.

- Crear nuevos bots y completarlos con intenciones y tipos de slots relevantes de manera eficiente utilizando una descripción en lenguaje natural.
- Generar automáticamente enunciados de ejemplo para las intenciones de su bot.
- Mejorar el rendimiento de resolución de los slots de sus bots.
- Crear una intención que ayude a responder a las preguntas de sus clientes.
- Utilice los agentes de Amazon Bedrock y las bases de conocimiento de Amazon Bedrock para responder a las preguntas de sus clientes.

Puede activar las capacidades de IA generativa para Amazon Lex V2 mediante la consola o la API.

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.

2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Using the console

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. Seleccione el bot y la configuración regional del bot cuyas capacidades de IA generativa desea activar.
3. En la sección Configuraciones de IA generativa, seleccione Configurar.
4. Active el botón Habilitado para cada característica que desee activar. Seleccione el modelo y la versión que desee utilizar para esa característica. Habilitar una característica puede conllevar cargos adicionales. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock. Para obtener más información sobre una característica, seleccione el tema correspondiente en la lista siguiente. Seleccione Guardar después de activar las características que desea activar. Aparece un aviso en verde para confirmar que las capacidades están activadas.

Using the API

1. Para habilitar las capacidades generativas de IA para un bot nuevo, utilice la CreateBotoperación para crear un bot nuevo.
2. Envía una CreateBotLocalesolicitud modificando el `generativeAISettings` objeto según sea necesario. Si estás habilitando las capacidades de un bot existente, envía una UpdateBotLocalesolicitud en su lugar.

    a. Para habilitar el uso del generador de bots descriptivo, modifique el objeto `descriptiveBotBuilder`. Especifique el modelo fundacional que desea utilizar en el campo `modelArn` y defina el valor `enabled`como `True`.

    b. Para habilitar la mejora de la resolución de slots, modifique el objeto `slotResolutionImprovement`. Especifique el modelo fundacional que desea utilizar en el campo `modelArn` y defina el valor `enabled`como `True`.

c. Para habilitar la generación de enunciados de ejemplo, modifique el objeto `sampleUtteranceGeneration`. Especifique el modelo fundacional que desea utilizar en el campo `modelArn` y defina el valor `enabled`como `True`.

Temas

- Uso de una descripción para crear un bot en Lex V2 con el generador de bots descriptivo
- Uso de la generación de enunciados para generar ejemplos de enunciados para el reconocimiento de intenciones
- Uso de la resolución asistida de ranuras para aclarar los valores de las ranuras en Amazon Lex V2
- Uso BedrockAgentIntent de un agente de Amazon Bedrock en Amazon Lex V2
- Mejora de la clasificación de intenciones y la resolución de ranuras en Lex V2 con la NLU asistida
- Resuelva las entradas ambiguas de los usuarios con la desambiguación de intenciones
- Optimice el bot con un analizador de bots impulsado por IA
- Amazon.qn AIntent

Uso de una descripción para crear un bot en Lex V2 con el generador de bots descriptivo

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.
> 2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

El generador de bots descriptivo le permite aprovechar el acceso de Amazon Bedrock a modelos de lenguaje de gran tamaño para mejorar la eficiencia del proceso de creación de bots. Se proporciona un mensaje en lenguaje natural que incluye el propósito del bot y las acciones que debe realizar. Amazon Lex V2 aprovecha las capacidades de Amazon Bedrock para generar intenciones y tipos

de slots relevantes para el bot en función de la descripción. Cuando haya elegido las intenciones y los tipos de slots que desea conservar, puede iterar el bot para modificarlo según el caso de uso específico. El generador de bots descriptivo ahorra tiempo, ya que evita la necesidad de crear manualmente intenciones y tipos de slots para el bot.

El generador de bots descriptivo está disponible en configuraciones regionales en inglés (consulte las configuraciones regionales que comiencen por en_ en la tabla de Idiomas y configuraciones regionales compatibles con Amazon Lex V2).

Antes de crear el bot, haga lo siguiente:

1. Compruebe que su rol tiene los permisos correctos revisando los pasos que se indican enPermisos necesarios para crear un bot con una descripción de lenguaje natural en Lex V2.
2. Decida la descripción que desea utilizar. Puede consultar ejemplos de descripciones de bot en Ejemplos de descripciones de bots para un generador de bots descriptivo.

Cree un bot utilizando lenguaje natural para describir lo que debería hacer el bot. Amazon Lex V2 invoca modelos de Amazon Bedrock para generar intenciones y tipos de slots que se adaptan al caso de uso de su bot. Puede crear el bot con la consola o la API.

Console

Creación de un bot utilizando el generador de bots descriptivo

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En la página Bots, seleccione Crear bot.
3. Como Método de creación, seleccione Generador de bots descriptivo - GenAI.
4. Asigne un nombre a su bot y una descripción opcional, configure los permisos de IAM y elija si su bot está sujeto a COPPA. A continuación, seleccione Siguiente.
5. Seleccione un idioma para crear el bot, una voz para el bot y un umbral de confianza para la clasificación de intenciones (para obtener más información, consulte Uso de puntuaciones de confianza en la intención para mejorar la selección de intenciones con Lex V2).
6. En Generador de bots descriptivos - GenAI, proporcione una descripción del bot que desea crear. La descripción debe ser a la vez detallada y precisa para ayudar a generar intenciones adecuadas y suficientes para el bot. Incluya una lista de acciones para mejorar el proceso de creación de intenciones.

7. Seleccione un proveedor de modelos y un modelo en Seleccionar modelo.
8. Para crear el bot en otra configuración regional, seleccione Agregar otro idioma. Cuando haya terminado de agregar idiomas, seleccione Listo. Amazon Lex V2 crea el bot y el generador de bots descriptivo genera intenciones y slots para él. Cuando se ha generado la configuración regional, el aviso pasa de azul a verde. Seleccione Revisar para ver las intenciones generadas y los tipos de slots.

> **Note**
>
> Por el momento, el generador de bots descriptivo solo está disponible en configuraciones regionales en inglés. Sin embargo, puede copiar un bot a una configuración regional que no sea en inglés después de crearlo.

Revisión de las intenciones generadas y los tipos de slots y su adición al bot

1. Si hay intenciones y tipos de slots suficientes que son aplicables al caso de uso de su bot, puede revisar las intenciones generadas.
   a. Revise las Intenciones generadas.
      i. Seleccione una casilla situada junto a una intención para eliminarla de la lista de intenciones que se agregan al bot.
      ii. Elija un nombre de intención para ver los Ejemplos de enunciados y los Slots generados para la intención.
      iii. Todos los enunciados y los slots están seleccionados de manera predeterminada. Seleccione una casilla para eliminar ese elemento de la intención. Seleccione Agregar a la selección para mantener los elementos marcados en la intención.
   b. Revise los Tipos de slots generados.
      i. Seleccione una casilla junto a un tipo de slot para eliminarlo de la lista de intenciones que se agregan al bot.
      ii. Puede agregar valores a un tipo de slot después de agregarlo al bot
2. Cuando esté satisfecho con las intenciones y los tipos de slots, seleccione Agregar intenciones y tipos de slots en la parte superior de la página para agregar las intenciones y los tipos de slots a su bot.

3. Cuando terminen de agregarse los recursos, aparecerá un aviso de confirmación en verde. Vaya a Intenciones y Tipos de slots para editar los generados y agregar más valores.
4. Si las Intenciones generadas y los Tipos de slots generados son prácticamente inaplicables al bot que desea crear, siga estos pasos.

   a. Seleccione Nueva generación en la sección Detalles del generador de bots descriptivo.

   b. Vuelva a escribir el mensaje y seleccione Volver a generar para generar nuevas intenciones y tipos de slots. Los resultados varían si se utiliza un modelo diferente.

   > **Important**
   >
   > No se garantiza que se generen las mismas intenciones y slots. Se le cobrará cada vez que vuelva a generar las intenciones y los tipos de slots.

API

Creación del bot utilizando una descripción en lenguaje natural

Cuando se utiliza el generador de bots descriptivo a través de la API, se crea una definición de bot en un archivo.zip en un bucket de Amazon S3. Descargue este archivo e importe la definición del bot en Amazon Lex V2 para crear el bot.

1. Envíe una CreateBotsolicitud para crear un bot nuevo. A continuación, envía una CreateBotLocalesolicitud para crear una configuración regional para el bot.
2. Envía una StartBotResourceGenerationsolicitud especificando el ID, la versión y la configuración regional del bot. Puede utilizar `DRAFT` para la versión del bot. Proporcione su mensaje en el campo `generationInputPrompt`. La descripción debe ser a la vez detallada y precisa para ayudar a generar intenciones adecuadas y suficientes para el bot. Incluya una lista de acciones para mejorar el proceso de creación de intenciones.
3. Anote el `generationId` en la respuesta.
4. Envía una DescribeBotResourceGenerationsolicitud utilizando `generationId` lo que recibiste en la `StartBotResourceGeneration` respuesta. Incluya el ID, la versión y la configuración regional del bot.
5. Si el `generationStatus` de la respuesta de `DescribeBotResourceGeneration` es `Complete`, el campo `generatedBotLocaleUrl` también se rellenará. Utilice este URI

de Amazon S3 para descargar la definición del bot siguiendo los pasos que se indican en Descargar un objeto.

Comprobación e importación de la definición generada del bot

1. Utilice este URI de Amazon S3 del `generationStatus` en la respuesta de `DescribeBotResourceGeneration` para descargar la definición del bot siguiendo los pasos que se indican en Descargar un objeto.
2. Puede modificar directamente el contenido generado para el caso de uso específico de su bot editando el archivo. También puede enviar otra solicitud de `StartBotResourceGeneration` para volver a generar intenciones y slots.

   > **Important**
   >
   > No se garantiza que se generen las mismas intenciones y slots. Se le cobrará cada vez que vuelva a generar las intenciones y los tipos de slots.

3. Para importar la definición del bot, siga los pasos que se indican en Importación de bots en Lex V2.
4. Tras la importación, puede modificar las intenciones y los espacios generados mediante las UpdateSlotTypeoperaciones UpdateIntentUpdateSlot, y.

Para enumerar los metadatos de todos los elementos generados para una configuración regional de bots, utilice la ListBotResourceGenerationsoperación. Utilice cualquiera de los valores de `generationId` devueltos en una solicitud de `DescribeBotResourcGeneration` para recuperar el URI de Amazon S3 para una definición de bot generada.

Temas

- Ejemplos de descripciones de bots para un generador de bots descriptivo
- Permisos necesarios para crear un bot con una descripción de lenguaje natural en Lex V2

Ejemplos de descripciones de bots para un generador de bots descriptivo

Estos son algunos ejemplos útiles de descripciones de bots que puede utilizar con el generador de bots descriptivos de Amazon Lex V2.

Industry	Mensaje de ejemplo
Servicios financieros	Somos un servicio de tarjetas financieras que ayuda a los usuarios a realizar tareas cuando reciben una nueva tarjeta, como activar la tarjeta, enviar un PIN por correo electróni co o por correo postal o verificar una nueva tarjeta (mediante un código postal). También les ayudamos en tareas relacionadas con su tarjeta actual, como solicitar información sobre las ventajas de una tarjeta de crédito, denunciar la pérdida de una tarjeta, solicitar una nueva, restablecer el PIN o pagar la factura de una tarjeta.
Servicios de alimentación	Quiero un bot que ayude a los clientes a encargar comida (utilizando ID de producto, cantidad y tamaño), comprobar el estado de un pedido y cancelar un pedido. Utilice ID de pedido para indexar los pedidos.
Aerolíneas	Somos un dominio de aerolíneas que ayuda a los usuarios a reservar billetes de avión, comprobar los detalles de una reserva, obtener el recibo de un vuelo reservado, consultar el estado del vuelo, reprogramar vuelos reservados, obtener detalles de vuelo y cancelar vuelos reservados. También se pueden generar intenciones adicionales si sirven de apoyo a funciones en la descripción del dominio.
Seguros	Objetivo: somos una compañía de seguros que vende pólizas de seguros de automóvil , hogar y rentas. Quiero un bot que pueda comprobar el estado de las reclamaciones, presentar una reclamación, realizar pagos y

Industry	Mensaje de ejemplo
	cancelar una póliza. Usamos policy_id y los cuatro últimos números del SSN para identific ar y validar la cuenta. Espero que el bot tenga al menos las siguientes intenciones y espacios: autenticación: policy_id, últimos 4 SSNpolicy tipos: coche, vivienda, anualidadEstado de la póliza: comprobar el saldo, comprobar la fecha de vencimiento, comprobar la cobertura realizar un pago: pago único, cuotas, importe
Gestión de vehículos	Estamos creando un bot de búsqueda de vehículos retirados por la grúa que ayuda a los conductores a averiguar dónde se encuentra su automóvil. Este bot debería preguntar la dirección o la ubicación inicial del automóvil y detalles sobre este, como matrícula y marca, modelo y año de fabricación. El bot debería responder con la ubicación del estacionamiento donde se encuentra el automóvil y su horario.
Turismo	Soy agente de viajes y quiero un bot que ayude a mis clientes a reservar un viaje a Disney. Disney cuenta con varios parques de atracciones en todo el mundo, además de hoteles, restaurantes y espectáculos especiale s que se pueden reservar. Los usuarios del bot deberían poder modificar o cancelar su reserva. Las reservas deben incluir como mínimo el parque, las fechas y el hotel. Incluir comidas o espectáculos es opcional y se puede agregar o cambiar más adelante.

Permisos necesarios para crear un bot con una descripción de lenguaje natural en Lex V2

- Para acceder a esta característica en la consola de Amazon Lex V2, asegúrese de que su rol de consola tiene permisos de `bedrock:ListFoundationModels` y `bedrock:ListInferenceProfiles`.
- El rol de IAM asociado al bot debe tener el permiso `bedrock:InvokeModel`. Si habilita la característica con la consola de Amazon Lex, la política se agregará automáticamente al rol de bot siempre que el bot utilice un rol vinculado a un servicio generado por Amazon Lex.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-1::foundation-model/model-id"
            ]
        }
    ]
}
```

Uso de la generación de enunciados para generar ejemplos de enunciados para el reconocimiento de intenciones

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.

2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Utilice la generación de enunciados para automatizar la creación de enunciados de ejemplo según su intención. En lugar de introducir enunciados de ejemplo manualmente, Amazon Lex V2 los genera basándose en el nombre de la intención, la descripción y los enunciados de ejemplo existentes, para que pueda reducir el tiempo y el esfuerzo que dedica a detectar y escribir sus propios enunciados de ejemplo. Una vez que Amazon Lex V2 genere los enunciados, podrá editarlos y eliminarlos. Utilice esta herramienta para agilizar la creación de enunciados de ejemplo para el proceso de reconocimiento de intenciones.

Para permitir la generación de enunciados, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa para activar las capacidades de la IA generativa.

Para acceder a esta función en la consola Amazon Lex V2, asegúrese de que su rol de consola tenga `bedrock:ListFoundationModels` `bedrock:InvokeModel` permisos y permisos. `bedrock:ListInferenceProfiles`

Puede generar enunciados mediante la consola o la API.

Console

1. Vaya a la sección Ejemplos de enunciados de cualquier intención de su bot (en el generador visual de conversaciones, que se encuentra en el bloque Inicio).
2. Seleccione el botón Generar enunciados para generar 5 ejemplos de enunciados. Si su intención tiene más de 25 ejemplos de enunciados, el botón Generar enunciados se deshabilita.
3. Los enunciados generados se muestran con un aviso en verde que diferencia los enunciados generados de los enunciados existentes.
4. Pase el ratón sobre un enunciado para ver las opciones para editar, eliminar y ordenar los enunciados generados.

API

1. Envía una GenerateBotElementsolicitud introduciendo la intención y el ID del bot, la versión y la configuración regional para las que deseas generar expresiones de muestra.
2. La respuesta devuelve una lista de SampleUtteranceobjetos, cada uno de los cuales contiene un enunciado generado.
3. Para añadir los enunciados a la intención, envíe una UpdateIntentsolicitud y añada los enunciados al campo. `sampleUtterances`

Uso de la resolución asistida de ranuras para aclarar los valores de las ranuras en Amazon Lex V2

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.
> 2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Puede mejorar la precisión de algunos slots integrados en el flujo de conversación del bot mediante la resolución asistida de slots. La resolución asistida de franjas horarias utiliza los modelos lingüísticos extensos de Amazon Bedrock (LLMs) para mejorar el reconocimiento de algunas franjas horarias integradas, lo que se traduce en una mejor interpretación de las respuestas de los clientes durante la selección de franjas horarias. En el caso de los enunciados que no se hayan podido resolver normalmente, Amazon Lex V2 intentará resolverlos por segunda vez con Amazon Bedrock.

La resolución asistida de slots permite utilizar la potencia de los modelos fundacionales de Amazon Bedrock para mejorar la precisión de los siguientes slots integrados:

- `AMAZON.Alphanumeric` sin compatibilidad con expresiones regulares
- `AMAZON.City`

- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

Puede habilitar la resolución asistida de ranuras para cualquier intención que utilice las ranuras integradas enumeradas. La resolución asistida de slots no se aplica a los slots personalizados ni a los slots integrados de Amazon que no figuran en la lista anterior.

Puede recopilar datos sobre las mejoras de precisión después de habilitar la resolución asistida de ranuras en su bot Amazon Lex V2 mediante el uso de registros y métricas de conversaciones.

- Registros de conversación: las interpretaciones tendrán `Bedrock` como `interpretationSource` si se ha utilizado Amazon Bedrock para resolver el slot.
- CloudWatch métricas: las métricas se publicarán en las dimensiones que figuran en la CloudWatch métrica. Para obtener más información, consulte Supervisión de Amazon Lex con Amazon CloudWatch.

Para utilizar el generador de bots descriptivo, asegúrese de que su rol de IAM tiene los permisos adecuados siguiendo los pasos que se indican en Se necesitan permisos en Lex V2 para la resolución asistida de ranuras.

Temas

- Ejemplos de resolución asistida de ranuras utilizadas en Lex V2
- Habilitación de la resolución asistida de slots en la pantalla de configuración de IA generativa
- Habilitación de la resolución asistida de ranuras en la configuración de ranuras en Lex V2
- Se necesitan permisos en Lex V2 para la resolución asistida de ranuras

Ejemplos de resolución asistida de ranuras utilizadas en Lex V2

A continuación se muestran algunos ejemplos en los que la resolución asistida de slots puede resolver de forma inteligente los enunciados del usuario en un valor.

AMAZON.Number

Vertical	slotType	slotName	slotPrompt	utterance	Valor resuelto
Turismo	AMAZON.Number	numberOfNightsMequedé	¿Cuántas noches se hospedó durante el viaje?	Una semana entera, 7 noches.	7
Banca	AMAZON.Number	numberOfPeopleOnTheAccount	¿Cuántas personas hay en la cuenta?	Mi mujer y yo.	2
Turismo	AMAZON.Number	numberOfStops	¿Cuántas paradas?	Una en Japón. Una en Los Ángeles.	2

AMAZON. AlphaNumeric

Vertical	slotType	slotName	slotPrompt	utterance	Valor resuelto
Alquiler de automóviles	AMAZON.Alphanumeric	transactionId	¿Cuál es su identificador de transacción?	Creo que era alfa whiskey eco ocho tres cuatro nueve romeo julieta.	AWE8349RJ
Travel	AMAZON.Alphanumeric	confirmationCode	¿Cuál es el número de confirmación de su reserva?	El número de confirmación es BLT2 UE.	BLT2UE

AMAZON.Date

Vertical	slotType	slotName	slotPrompt	utterance	Valor resuelto	currentDate
Alquiler de automóviles	AMAZON.Date	dueDate	¿Cuándo vence el contrato de alquiler?	El contrato de arrendamiento vence el 1 del mes que viene.	2023-12-01	2023-11-09
Turismo	AMAZON.Date	returnDate	¿Cuándo regresará?	Hoy más tarde, alrededor de las 7.	2023-11-09	2023-11-09

AMAZONA. PhoneNumber

Vertical	slotType	slotName	slotPrompt	utterance	Valor resuelto
Seguros	AMAZONA. PhoneNumber	policyHolder	¿Cuál es el número de teléfono del titular de la póliza?	El número de teléfono del titular de la póliza es 123-456-7890.	1234567890
Venta minorista	AMAZONA. PhoneNumber	phoneLookup	¿Cuál es su número de teléfono para que pueda localizar su cuenta?	Creo que el número es 413-570-9617, deje que lo compruebe.	4135709617

AMAZON.Country

Vertical	slotType	slotName	slotPrompt	utterance	Valor resuelto
Turismo	AMAZON.Country	nativeCountry	¿Cuál es su país de origen?	Soy de la India.	India
Banca	AMAZON.Country	countryItinerary	¿A qué países viajará con su tarjeta de débito?	Viajaré a Nueva Delhi.	India

AMAZON.City

Vertical	Tipo de slot	Intención	Pregunta	Respuesta	Valor resuelto
Seguros	AMAZON.City	policyHolderCity	¿En qué ciudad reside el titular de la póliza?	Vivo en Springfield.	Springfield
Turismo	AMAZON.City	destinationCity	¿A qué ciudad va a viajar?	Voy a viajar a Tokio.	Tokio

AMAZON.Confirmation

Vertical	slotType	slotName	slotPrompt	utterance	Valor resuelto
Seguros	AMAZON.Confirmation	policyExpired	¿Ha caducado la póliza del seguro?	Sí, lamentablemente ha caducado.	Sí
Banca	AMAZON.Confirmation	hasInvestments	¿Tiene alguna inversión?	Aún no he invertido en nada.	No

Habilitación de la resolución asistida de slots en la pantalla de configuración de IA generativa

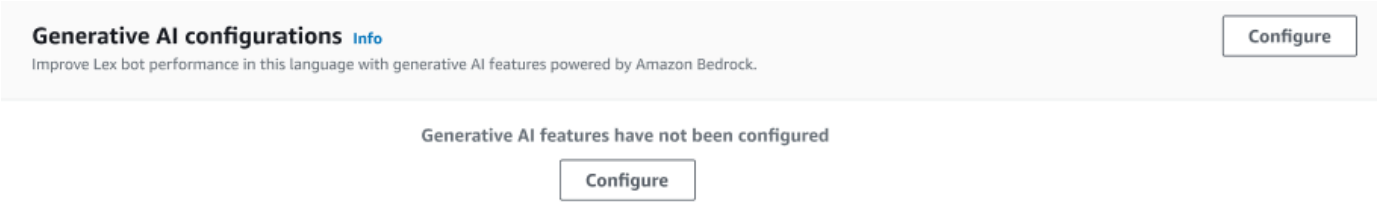
Para habilitar la resolución asistida de slots en slots integrados compatibles, vaya a la pantalla de IA generativa.

Si el slot es un slot integrado compatible, tendrá la opción de activar la resolución asistida des slot en el nivel de slot.

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En el panel de navegación, en Bots, seleccione el bot que quiera utilizar para la resolución asistida de slots.
3. Seleccione el idioma Inglés (EE. UU.) para el bot que desea habilitar.
4. Vaya a la sección de Configuración de IA generativa en la pantalla.
5. Seleccione Ir a Amazon Bedrock para registrarse y habilitar la característica, si no se ha habilitado.

   > **Note**
   >
   > Si no tiene acceso a los modelos fundacionales de Amazon Bedrock, consulte Ir a Amazon Bedrock. Haga clic en Ir a Amazon Bedrock para ir a la página de Amazon Bedrock, donde puede registrarse para acceder a los modelos fundacionales. La resolución asistida de ranuras actualmente es compatible con Anthropic Claude. Sugerimos usar Anthropic Claude para obtener mejores resultados.

6. Si ya tiene acceso a los modelos fundacionales de Amazon Bedrock, debería aparecer el botón Configurar. Haga clic en este botón para ir a la página de configuración de IA generativa y activar las características de IA generativa en Lex.



7. En la esquina superior derecha del cuadro, mueva el control deslizante hacia la derecha para seleccionar el ajuste Habilitado.

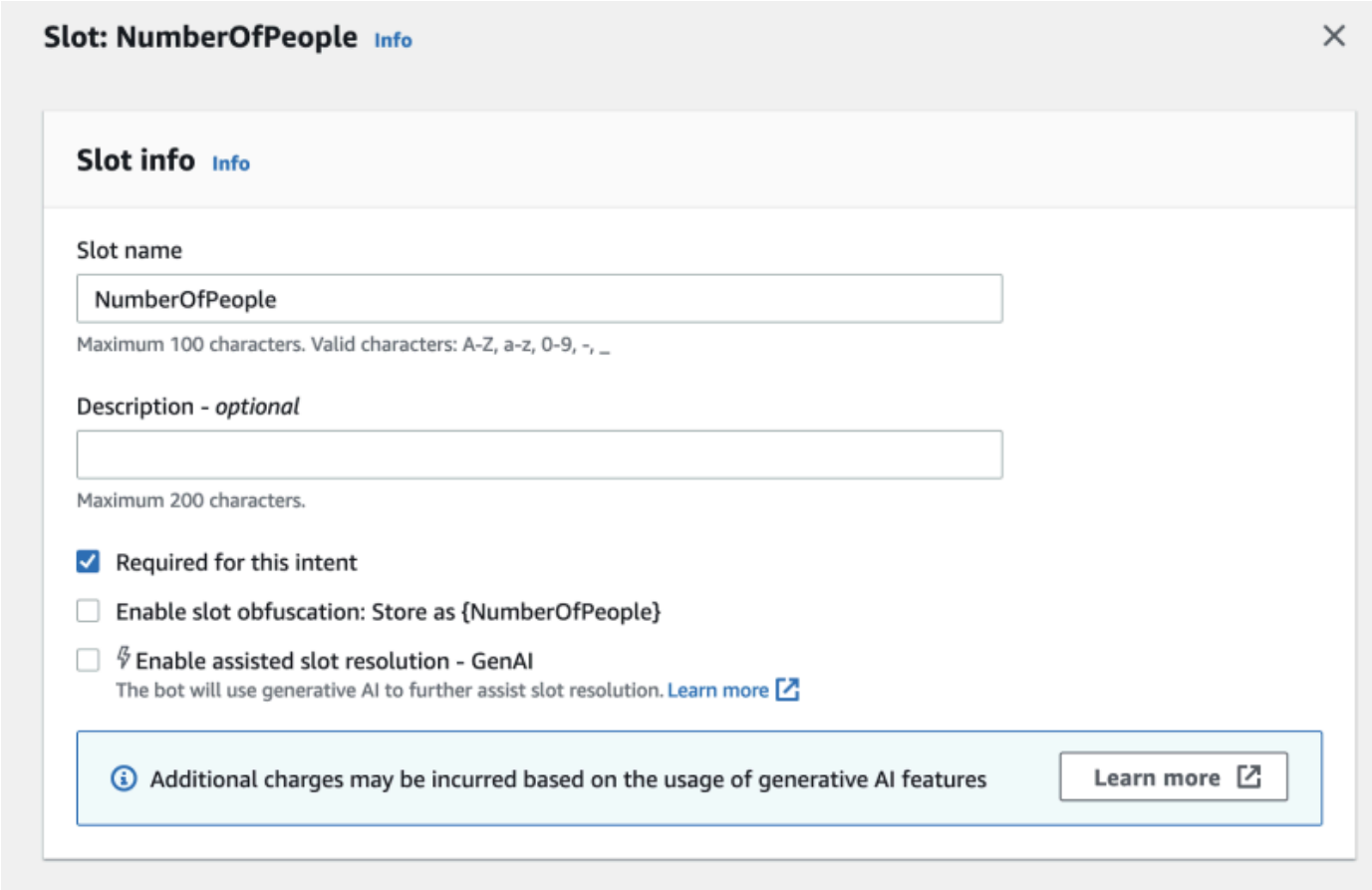
8. Elija el botón Habilitar para activar la resolución asistida de slots para los slots seleccionados.
9. Puede deshabilitar la resolución asistida de slots seleccionando los slots de la lista y seleccionando el botón Deshabilitar.

Habilitación de la resolución asistida de ranuras en la configuración de ranuras en Lex V2

Puede habilitar la resolución asistida de slots para slots integrados compatibles navegando hasta el nivel de slot de cada intención que tenga slots. Los slots deben slots integrados compatibles indicados anteriormente para tener la opción de activar la resolución asistida de slots. Si el slot no tiene la opción de activar la resolución asistida de slots, la opción aparecerá atenuada.

> **Note**
>
> Primero debe activar la característica de resolución asistida de slots en el panel de IA generativa para activar la característica para slots individuales.

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/ casa.
2. En el panel de navegación, en Bots, seleccione el bot que quiera utilizar para la resolución asistida de slots.
3. En Todos los idiomas, seleccione Inglés (EE. UU.) para ampliar la lista.
4. En el panel lateral izquierdo, seleccione Intenciones para ver una lista de intenciones del bot que ha seleccionado.
5. En la pantalla Intenciones, elija la intención que contiene los slots que desea modificar.
6. Seleccione el nombre de la intención para ver los slots correspondientes a esa intención.
7. Seleccione el botón Opciones avanzadas en la sección Slots.
8. Seleccione la casilla Habilitar resolución asistida de slots para habilitar la característica.



9. Seleccione el botón Actualizar ranura en la esquina inferior derecha de la pantalla. De este modo se activará la resolución asistida de los slots que haya elegido.

Puede habilitar la resolución asistida de slots para slots integrados compatibles realizando llamadas a la API.

- Siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa para habilitar la resolución asistida de slots en la configuración regional de su bot.
- Envíe una UpdateSlotsolicitud especificando la ranura para la que desea habilitar la resolución asistida de ranuras. En el campo `slotResolutionSetting`, defina el valor de `slotResolutionStrategy` como `EnhancedFallback`. Para crear una nueva ranura con la resolución asistida de ranuras habilitada, envíe una CreateSlotsolicitud en su lugar.

Se necesitan permisos en Lex V2 para la resolución asistida de ranuras

- Para acceder a esta característica en la consola de Amazon Lex V2, asegúrese de que su rol de consola tiene permisos de `bedrock:ListFoundationModels` y `bedrock:ListInferenceProfiles`.
- El rol de IAM asociado al bot debe tener el permiso `bedrock:InvokeModel`. Al activar la función con la consola Amazon Lex V2, la política se añadirá automáticamente al rol de bot siempre que el bot utilice un rol vinculado a un servicio generado por Amazon Lex V2.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-1::foundation-model/modelId"
            ]
        }
    ]
}
```

Uso BedrockAgentIntent de un agente de Amazon Bedrock en Amazon Lex V2

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.

2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Puede aprovechar los agentes de Amazon Bedrock para gestionar las cargas de trabajo complejas solicitadas por los clientes sin tener que pasar por un proceso exhaustivo de definición de tareas. Amazon Lex V2 ofrece una `AMAZON.BedrockAgentIntent` integrada que puede agregar a su bot. Esta intención aprovecha las capacidades generativas de IA de Amazon Bedrock al reconocer las solicitudes de los clientes, analizarlas, razonarlas y, finalmente, responder. También tiene la capacidad de hacer cualquier pregunta de seguimiento para llevar a cabo la tarea necesaria (por ejemplo, imagine que usted define un agente minorista que puede consultar el estado del pedido de un cliente. Cuando el cliente pregunte por el estado del pedido, primero solicitará el `customerId` o el `emailId` asociado para obtener los detalles y, finalmente, responderá con el estado correcto del pedido). También puede decidir integrar su AMAZON. BedrockAgentIntent con una base de conocimientos de Bedrock para responder directamente a cualquier consulta de los clientes.

Asegúrese de que su función de IAM tenga los permisos adecuados para acceder a AMAZON. BedrockAgentIntent siguiendo los pasos que se indican en Permisos para AMAZON. BedrockAgentIntent

Para aprovechar el AMAZON. BedrockAgentIntent debe haber creado uno de los siguientes almacenes de conocimientos.

- Agentes de Amazon Bedrock - Para obtener más información, consulte Creación de agentes de Bedrock.
- Base de conocimiento de Amazon Bedrock - Para obtener más información, consulte Creación de una base de conocimiento.

Para usar AMAZON. BedrockAgentIntent, asegúrese de que su función de IAM tenga los permisos adecuados siguiendo los pasos de Permisos necesarios en Lex V2 para Bedrock Agent Intent.

Temas

- Habilitación de la intención del agente de Bedrock en la pantalla de configuración de IA generativa
- Habilitación de la intención del agente de Bedrock agregando una intención integrada a su bot
- Se necesitan permisos en Lex V2 para la intención del agente de Bedrock

- Ejemplo de solicitud con los atributos de la sesión

Habilitación de la intención del agente de Bedrock en la pantalla de configuración de IA generativa

Para activar la intención del agente de Bedrock, vaya a la pantalla de IA generativa.

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En el panel de navegación, en Bots, seleccione el bot que quiera utilizar para la intención del agente de Bedrock.
3. Seleccione el idioma del bot que quiere activar.
4. Ve a la sección de configuración de IA generativa de la pantalla y selecciona Configurar.
5. En la sección BedrockAgentIntent de configuración, selecciona Crear BedrockAgent intención.

Habilitación de la intención del agente de Bedrock agregando una intención integrada a su bot

Puede habilitar la intención del agente de Bedrock agregando una intención integrada a su bot de Amazon Lex V2.

> **Note**
>
> Primero debe activar la característica Intención del agente de Bedrock en el panel de IA generativa para activar la característica para bots individuales.

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/ casa.
2. En el panel de navegación, en Bots, seleccione el bot que quiere utilizar para la intención del agente de Bedrock.
3. En Todos los idiomas, seleccione Inglés (EE. UU.) para ampliar la lista.
4. Seleccione Agregar intención y seleccione Usar intención integrada en el menú desplegable.
5. Para obtener más información sobre las configuraciones de AMAZON. BedrockAgentIntent, consulteAMAZON.BedrockAgentIntent.

Se necesitan permisos en Lex V2 para la intención del agente de Bedrock

- Para acceder a esta característica en la consola de Amazon Lex V2, asegúrese de que su rol de consola tiene permisos de `bedrock:ListFoundationModels` y `bedrock:ListInferenceProfiles`.
- El rol de IAM asociado al bot debe tener los siguientes permisos necesarios para `AMAZON.BedrockAgentIntent`.

  El rol de bot debe tener permisos para llamar a `bedrock:InvokeAgent` si usa agentes de Amazon Bedrock, y `bedrock:InvokeModel` y `bedrock:RetrieveAndGenerate` si usa bases de conocimiento de Amazon Bedrock en la intención. También deberá asociar una instrucción para cada agente que especifique en la `AMAZON.BedrockAgentIntent` de sus bots (consulte la instrucción Permisos para acceder al agente de Amazon Bedrock o Permisos para acceder a la instrucción de las bases de conocimiento de Amazon Bedrock en la política que aparece a continuación).

  Si habilita la característica con la consola de Amazon Lex, la política se agregará automáticamente al rol de bot siempre que el bot utilice un rol vinculado a un servicio generado por Amazon Lex.

Ejemplo de solicitud con los atributos de la sesión

El siguiente ejemplo muestra cómo invocar el AMAZON. BedrockAgentIntent y muestra los atributos de sesión y solicitud que se rellenan en la respuesta. Estos atributos contienen los datos de respuesta del agente de Bedrock y se pueden utilizar para acceder a los resultados del agente, a los orígenes de la base de conocimiento de Amazon Bedrock y a los detalles de invocación de los grupos de acciones.

```
{
    "sessionId": "user-session-123",
    "messages": [{
        "content": "Your order #12345 is currently being processed and will ship within
 2-3 business days. You will receive a tracking number via email once it ships.",
        "contentType": "PlainText"
    }],
    "sessionState": {
        "sessionAttributes": {
            "x-amz-lex:bedrock-agent-search-response": "Your order #12345 is currently
 being processed and will ship within 2-3 business days. You will receive a tracking
 number via email once it ships.",
```

```
            "x-amz-lex:bedrock-knowledge-base-search-response-source": "[{\"title\":
 \"Order Processing Guide\", \"uri\": \"s3://knowledge-base/orders/processing.pdf\",
 \"excerpt\": \"Standard orders typically ship within 2-3 business days...\"}]",
            "x-amz-lex:bedrock-agent-action-group-invocation-input":
 "{\"actionGroupName\": \"OrderLookup\", \"function\": \"getOrderStatus\", \"parameters
\": {\"orderId\": \"12345\", \"customerId\": \"67890\"}}",
            "x-amz-lex:bedrock-agent-knowledge-base-lookup-input": "{\"knowledgeBaseId
\": \"KB123456\", \"query\": \"order status processing time\", \"numberOfResults\": 3}"
        },
        "intent": {
            "name": "BedrockAgentIntent",
            "slots": {},
            "state": "Fulfilled",
            "confirmationState": "None"
        },
        "dialogAction": {
            "type": "ElicitIntent"
        }
    },
    "interpretations": [{
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        },
        "interpretationSource": "Lex"
    }],
    "requestAttributes": {
        "x-amz-lex:channels:platform": "Web",
        "x-amz-lex:accept-content-types": "PlainText",
        "x-amz-lex:bedrock-agent-search-response": "Your order #12345 is currently
 being processed and will ship within 2-3 business days. You will receive a tracking
 number via email once it ships.",
        "x-amz-lex:bedrock-knowledge-base-search-response-source": "[{\"title\":
 \"Order Processing Guide\", \"uri\": \"s3://knowledge-base/orders/processing.pdf\",
 \"excerpt\": \"Standard orders typically ship within 2-3 business days...\"}]",
        "x-amz-lex:bedrock-agent-action-group-invocation-input": "{\"actionGroupName
\": \"OrderLookup\", \"function\": \"getOrderStatus\", \"parameters\": {\"orderId\":
 \"12345\", \"customerId\": \"67890\"}}",
        "x-amz-lex:bedrock-agent-knowledge-base-lookup-input": "{\"knowledgeBaseId\":
 \"KB123456\", \"query\": \"order status processing time\", \"numberOfResults\": 3}"
    }
}
```

En este ejemplo, los atributos de sesión muestran cómo BedrockAgentIntent se rellenan los datos de respuesta, incluida la respuesta del agente, las fuentes de Amazon Bedrock Knowledge Base utilizadas, las invocaciones de grupos de acción y los detalles de búsqueda de Amazon Bedrock Knowledge Base que se utilizaron para generar la respuesta.

Para obtener más información, consulte AMAZON. BedrockAgentIntent.

Mejora de la clasificación de intenciones y la resolución de ranuras en Lex V2 con la NLU asistida

La NLU asistida es una función que utiliza modelos de lenguaje grandes (LLMs) para mejorar las capacidades de clasificación por intenciones y resolución de ranuras de Amazon Lex V2. Mejora la precisión y, al mismo tiempo, se mantiene dentro de las ranuras e intenciones configuradas por el bot. La función no genera ni modifica el contenido de ningún bot. Esta función ayuda a mejorar la precisión general del sistema NLU, lo que resulta en una experiencia de conversación más fluida y eficaz para los usuarios.

La función NLU asistida está disponible en los idiomas inglés, español, portugués, catalán, francés, italiano y alemán. En concreto, admite configuraciones regionales que comiencen por,, `pt_` (pt_BR `en_es_`, pt_PT), (ca_ES), `ca_` (fr_CA, fr_FR), `fr_` (it_IT), (de_AT, de_DE), `it_` (zh_CN, zh_HKde_) y. `zh_` `ja_JP` `ko_KR` Para ver la lista Idiomas y configuraciones regionales compatibles con Amazon Lex V2 completa de las configuraciones regionales compatibles, consulte la tabla de.

Utilice la NLU asistida para mejorar la clasificación de intenciones y la resolución de franjas horarias. Amazon Lex V2 invoca modelos de Amazon Bedrock para ayudar a clasificar las intenciones y resolver los tipos de ranura que se adaptan al caso de uso de su bot. Puedes activar la NLU asistida para tu bot con la consola.

Modo NLU asistido

En el modo principal, Lex utilizará de forma predeterminada el LLM como el medio principal de procesamiento de la entrada del usuario para determinar la intención del usuario, así como para llenar los valores de los espacios.

En el modo alternativo, Lex utilizará el LLM para determinar la intención del usuario si la puntuación de confianza determinada por la NLU es inferior al umbral configurado o se dirige de otro modo hacia él FallbackIntent, así como para determinar los valores de las ranuras a partir de las entradas del usuario si la NLU tradicional no captura un valor.

Console

Uso de la NLU asistida con su bot de Amazon Lex V2

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En la página de bots, seleccione el bot que desee utilizar con la NLU asistida.
3. En la página Configuración regional del bot, haz clic en Configurar en la sección NLU asistida.
4. En la sección de funciones de IA generativa en tiempo de ejecución, puedes ver la función de NLU asistida. Usa el botón de alternancia situado junto a ella para activar la función de NLU asistida por LLM. A continuación, puede seleccionar el modo principal o el modo alternativo y hacer clic en Guardar.
5. Compruebe que la función NLU asistida por LLM esté habilitada en la sección NLU asistida de la página de configuración regional del bot.
6. Cree el bot para ver cómo los cambios se reflejan en su bot en tiempo de ejecución.
7. Una vez finalizada la creación del bot, puedes utilizar el panel de pruebas de la consola o ejecutar un conjunto de pruebas para comprobar las mejoras tras activar la función NLU asistida por LLM.

Guía para mejorar la precisión de tu bot al utilizar la función de NLU asistida por LLM

Las siguientes prácticas recomendadas pueden ayudarte a maximizar la eficacia de la función de NLU asistida:

1. Haga que los nombres de las intenciones se expliquen por sí mismos: utilice nombres que expresen inmediatamente la acción o el propósito de la intención. Por ejemplo, si está creando una intención para reservar vuelos, simplemente llámela «BookFlight».
2. Mantén los nombres limpios y sencillos: evita añadir prefijos, sufijos o palabras innecesarias a tu intención y a los nombres de los espacios. Elementos adicionales como «Dev» o «Test» pueden confundir el LLM y hacer que el propósito sea menos claro.
3. Proporcione descripciones detalladas: para cada intención y espacio personalizados, incluya una descripción breve pero informativa. Esto ayuda a explicar su uso y contexto específicos, lo que facilita que tanto los humanos como el LLM comprendan su propósito.

> **Note**
>
> Al habilitar esta función, es posible que sus datos se procesen en todas las regiones de AWS. Para obtener más información sobre la inferencia entre regiones, consulte. https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html

> **Important**
>
> Habilite esta función en una versión preliminar del bot. Pruébelo antes de usarlo en un alias de producción.

Desactivación de la NLU asistida

Para deshabilitar la función de NLU asistida, sigue estos pasos:

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En la página de bots, seleccione su bot.
3. En la página Configuración regional del bot, haz clic en Configurar en la sección NLU asistida.
4. En la sección Funciones de IA generativa en tiempo de ejecución, desactive la función NLU asistida y haga clic en Guardar.
5. Crea el bot para aplicar los cambios.

Resuelva las entradas ambiguas de los usuarios con la desambiguación de intenciones

La desambiguación de intenciones es una mejora de la NLU asistida que ayuda a resolver las entradas ambiguas de los usuarios cuando pueden coincidir varias intenciones. Cuando está activado, el sistema presenta preguntas aclaratorias a los usuarios, que les ayudan a especificar su intención exacta para mejorar la precisión de la conversación. El sistema utiliza un modelo lingüístico amplio (LLM) que analiza los nombres y descripciones de las intenciones en función del contexto y, en función de la ambigüedad del enunciado del usuario, devuelve las intenciones más probables que coincidan. El LLM evalúa si la entrada del usuario coincide claramente con una o

varias intenciones y si es lo suficientemente ambigua como para requerir una desambiguación. A continuación, proporciona las intenciones candidatas.

La función de desambiguación por intención está disponible en los idiomas inglés, español, portugués, catalán, francés, italiano, alemán, chino, japonés y coreano. Específicamente, admite configuraciones regionales que comiencen por `en_es_`, `pt_` (pt_BR, pt_PT), `ca_` (ca_ES), `fr_` (fr_CA, fr_FR), `it_` (it_IT), `de_` (de_AT, de_DE), `zh_` (zh_CN, zh_HK) `yko_KR`. `ja_JP` Para ver la lista Idiomas y configuraciones regionales compatibles con Amazon Lex V2 completa de las configuraciones regionales compatibles, consulte la tabla de.

Puede configurar las siguientes opciones para la desambiguación por intención:

Número de opciones de intención

Configure el número máximo de intenciones (de 2 a 5) para presentarlas a los usuarios cuando sea necesaria la desambiguación. Esta configuración determina cuántas opciones de intención se mostrarán a los usuarios cuando el sistema detecte una entrada ambigua. El valor predeterminado es 3, lo que proporciona un buen equilibrio entre ofrecer a los usuarios opciones suficientes y, al mismo tiempo, mantener la selección manejable.

Mensaje de desambiguación

Proporcione un mensaje personalizado que se mostrará antes de presentar las opciones de desambiguación a los usuarios. Este mensaje ayuda a establecer el contexto para los usuarios y se puede personalizar para que coincida con el tono y la marca de tu bot. Si no se especifica, se utilizará un mensaje predeterminado.

Intención: nombres para mostrar

Configure nombres para mostrar fáciles de usar según sus necesidades y así mejorar la experiencia de desambiguación. Esto se recomienda cuando los nombres de tus intenciones son técnicos o no son adecuados para mostrarlos a los usuarios finales. Los nombres para mostrar se mostrarán a los usuarios durante la desambiguación en lugar del nombre de intención técnica.

Console

Uso de la desambiguación de intenciones con su bot Amazon Lex V2

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.

2. En la página de bots, seleccione el bot que desee usar con Intent Disambiguation.
3. En la página Configuración regional del bot, haz clic en Configurar en la sección NLU asistida.
4. Activa la NLU asistida y selecciona el modo principal o el modo alternativo (la desambiguación por intención funciona en ambos modos).
5. En la sección de desambiguación por intención de la configuración de la NLU asistida, utilice el botón de alternancia para activar la función de desambiguación por intención.
6. Configure los siguientes ajustes opcionales:

   - Número de opciones de intención: seleccione el número máximo de intenciones (de 2 a 5) que se presentarán a los usuarios durante la desambiguación. El valor predeterminado es 3.
   - Mensaje de desambiguación: proporciona un mensaje personalizado que se mostrará al presentar las opciones de intención. Si no se especifica, se utilizará un mensaje predeterminado.
7. Haga clic en Guardar para aplicar la configuración.
8. Si lo desea, configure los nombres mostrados por intención para mejorar la experiencia de usuario:

   a. Navega hasta cada intención de tu bot que quieras configurar.
   b. En la página del editor de intenciones, busca el campo Nombre para mostrar.
   c. Introduzca un nombre fácil de usar que se muestre a los usuarios durante la desambiguación en lugar del nombre de la intención.
9. Cree el bot para ver los cambios reflejados en su bot en tiempo de ejecución.

Guía para mejorar la eficacia de tu bot al utilizar la función de desambiguación de intenciones

Las siguientes prácticas recomendadas pueden ayudarte a maximizar la eficacia de la función de desambiguación de intenciones:

1. Nombres y descripciones de intenciones claros: asegúrese de que los nombres y descripciones de las intenciones sean limpios, claros y no se superpongan con otras intenciones, ya que estas son las principales entradas que se proporcionan al LLM para la desambiguación.
2. Nombres descriptivos para mostrar: si los nombres de las intenciones actuales son técnicos, utilice nombres descriptivos para mostrar que comuniquen claramente el propósito de la

intención. Los nombres para mostrar deben parecerse a los nombres de las intenciones o coincidir con ellos.

3. Intenciones máximas adecuadas: establece el número máximo de opciones de intención en función de tus preferencias y de las pruebas.
4. Mensajes personalizados: crea mensajes de desambiguación concisos que reconozcan los comentarios del usuario y dirijan a las opciones de intención.
5. Escenarios de prueba: prueba con expresiones ambiguas para asegurarte de que la experiencia de desambiguación resulte natural con los nombres de las intenciones o los nombres mostrados con las intenciones y los mensajes personalizados, y comprueba que se presentan las opciones de intención correctas durante la solicitud de desambiguación.

> **Important**
>
> Activa esta función en una versión preliminar del bot. Pruébelo antes de usarlo en un alias de producción.

Desactivar la intención (desambiguación)

Para deshabilitar la función de desambiguación de intenciones, sigue estos pasos:

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En la página de bots, seleccione su bot.
3. En la página Configuración regional del bot, haz clic en Configurar en la sección NLU asistida.
4. En la sección Desambiguación por intención de la configuración de la NLU asistida, desactive la función de desambiguación por intención y haga clic en Guardar.
5. Cree el bot para aplicar los cambios.

Optimice el bot con un analizador de bots impulsado por IA

Analice la configuración de sus bots de Amazon Lex V2 comparándola con las prácticas recomendadas de AWS mediante recomendaciones basadas en la IA. Bot Analyzer utiliza las capacidades de IA generativa de Amazon Bedrock para identificar los problemas de configuración

y proporcionar una guía práctica para mejorar la clasificación de intenciones y el rendimiento de la resolución de franjas horarias.

Bot Analyzer evalúa automáticamente las configuraciones de intención de su bot y ofrece recomendaciones para:

- Mejore la separación de intenciones: identifique y resuelva las intenciones genéricas que agrupan varios conceptos
- Elimine la superposición de intenciones: detecte significados y frases similares entre las intenciones que provocan errores de enrutamiento
- Optimice el uso de las ranuras: recomiende el uso adecuado de las ranuras para combinar intenciones similares y mejorar la extracción de entidades
- Mejore la calidad de las expresiones: analice la cobertura y la diversidad de las expresiones de los ejemplos para una mejor clasificación por intenciones

Requisitos previos

Antes de usar Bot Analyzer, asegúrate de lo siguiente:

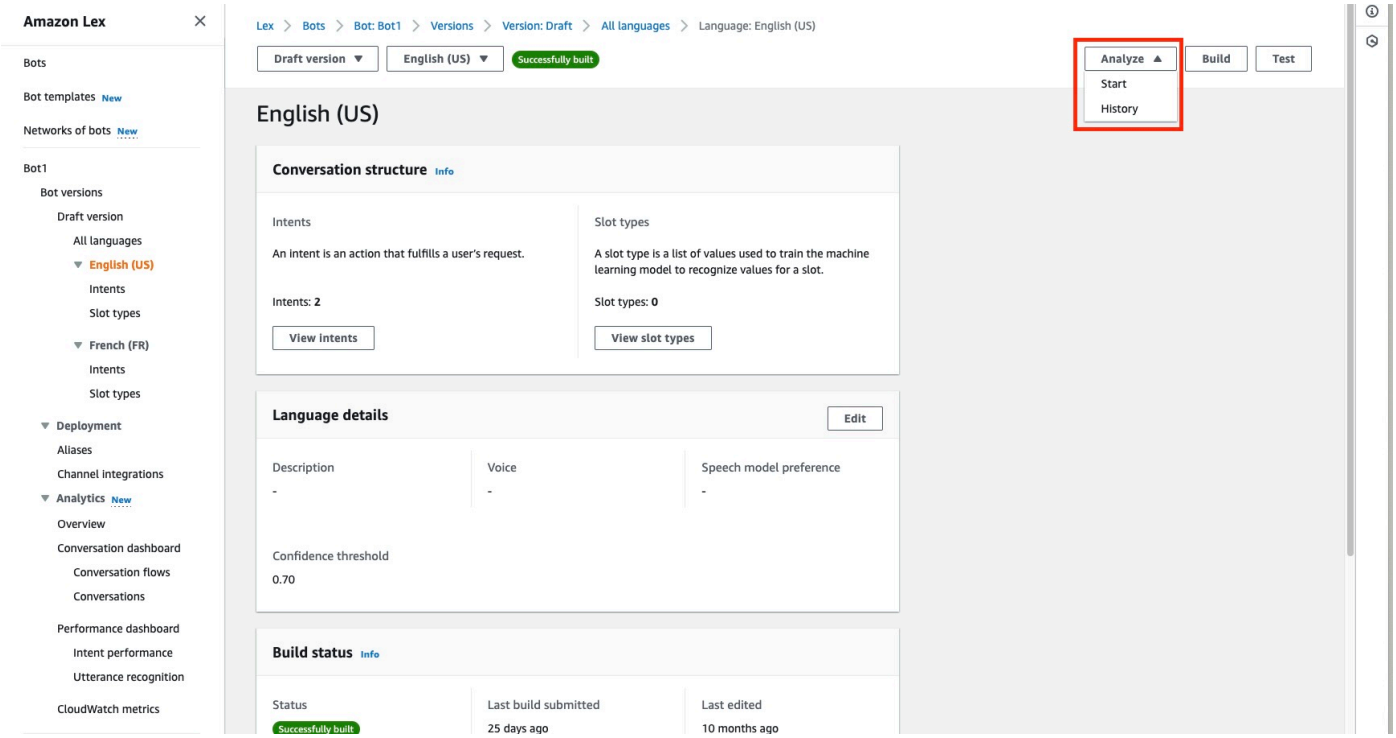
- La configuración regional de su bot se creó correctamente
- La versión de tu bot para el análisis es `DRAFT`
- La configuración regional de tu bot es una de las configuraciones regionales en inglés compatibles:`en_AU`,,`en_GB`,`en_IN`, `en_US` `en_ZA`

Puedes usar Bot Analyzer con la consola o la API.

Console

Inicie un análisis

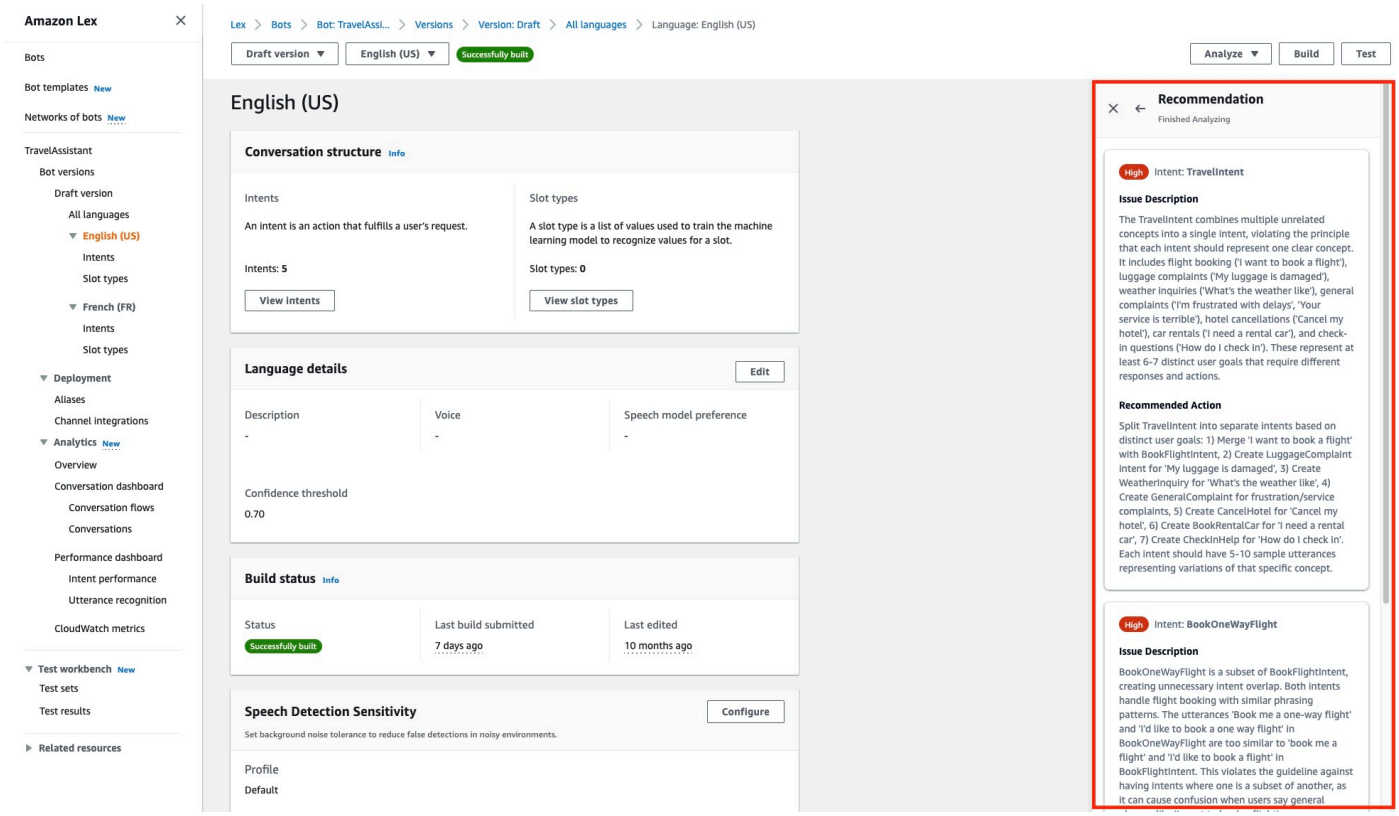
1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. Seleccione su bot y navegue hasta la configuración regional del bot que desee analizar.
3. En el editor de configuración regional del bot, haz clic en el menú desplegable Analizar.
4. Selecciona Iniciar para iniciar el análisis.



Por lo general, el análisis se completa en cuestión de minutos. Durante el análisis, el botón de inicio cambia a Detener el análisis si es necesario cancelarlo.

Ver recomendaciones

Una vez finalizado el análisis, las recomendaciones aparecen en el panel de recomendaciones de la parte derecha.



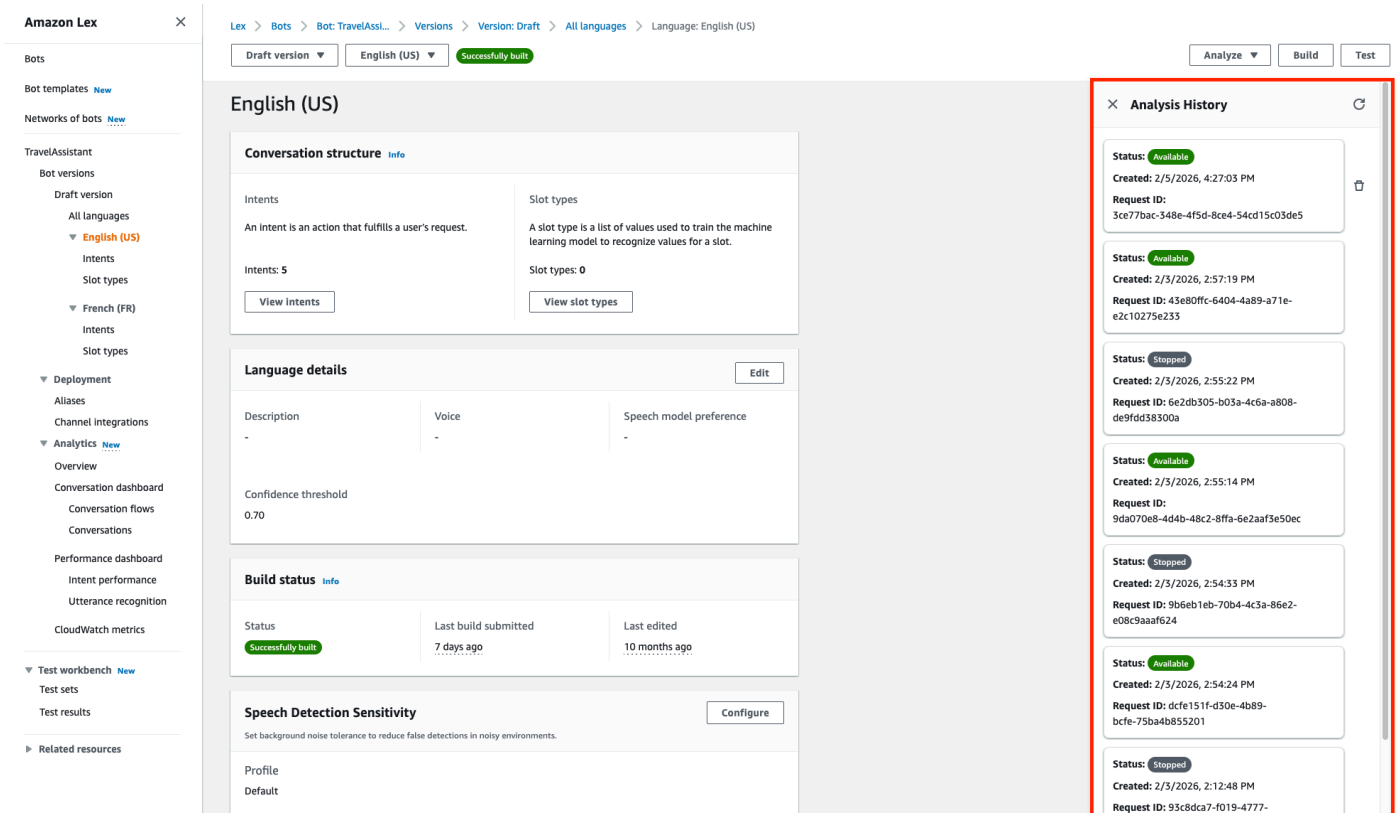
Cada recomendación incluye:

- Prioridad: gravedad alta, media o baja
- Ubicación del problema: la intención específica afectada
- Descripción del problema: ¿Qué problema de configuración se detectó
- Solución propuesta: pasos prácticos para resolver el problema

Ver el historial de análisis

Para ver los análisis anteriores:

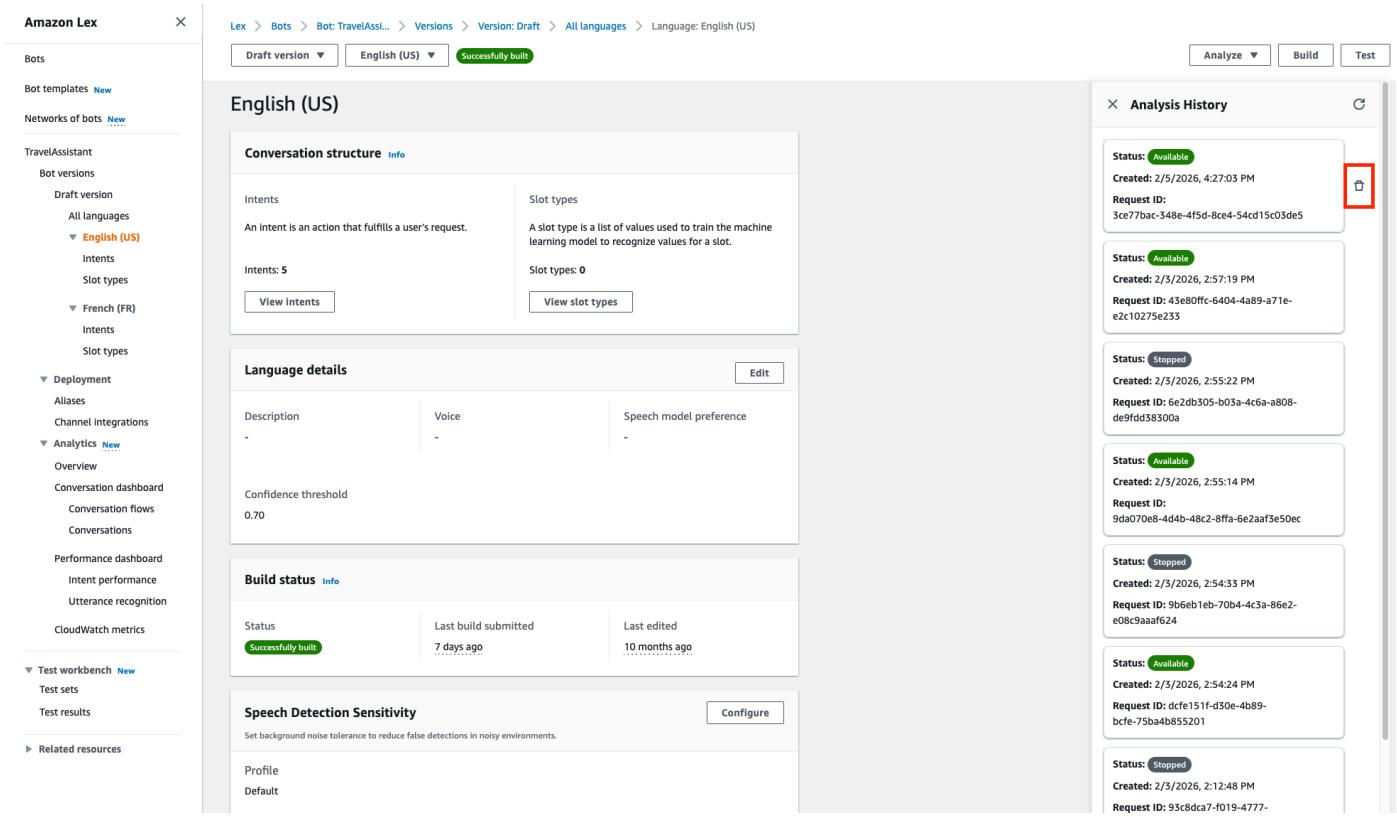
1. Haga clic en el menú desplegable Analizar.
2. Selecciona Historial.
3. El panel Historial de análisis muestra las solicitudes de análisis anteriores con su estado y fecha.



Eliminar recomendaciones

Para eliminar los resultados del análisis:

1. Haz clic en el menú desplegable Analizar.
2. Seleccione Eliminar.
3. Confirme la eliminación de las recomendaciones actuales.



API

Inicie un análisis

Envía una `StartBotAnalyzer` solicitud para iniciar el análisis de la configuración regional de tu bot. La respuesta devuelve un estado HTTP 202 con un`botAnalyzerRequestId`. Tome nota de este ID y lo necesitará para comprobar el estado del análisis y recuperar las recomendaciones.

Compruebe el estado del análisis y recupere las recomendaciones

Envíe una `DescribeBotAnalyzerRecommendation` solicitud siguiendo `botAnalyzerRequestId` el paso anterior. Incluya el `botId` en la ruta de la solicitud.

Si `botAnalyzerStatus` la respuesta es`Available`, el análisis se ha completado y el `botAnalyzerRecommendationList` campo se rellenará con recomendaciones. Cada recomendación incluye:

- `issueLocation`- La ubicación en la que se detectó el problema
- `priority`- Gravedad alta, media o baja
- `issueDescription`- Detalles sobre el problema de configuración

- `proposedFix`- Guía práctica para resolver el problema

Detenga un análisis continuo

Si necesita cancelar un análisis en curso, envíe una `StopBotAnalyzer` solicitud con la tecla `botId` y`botAnalyzerRequestId`.

Ver el historial de análisis

Para recuperar una lista de los análisis anteriores de la configuración regional de un bot, envía una `ListBotAnalyzerHistory` solicitud. `localeId`Especifica `botId` y consulta todas las solicitudes de análisis anteriores con su estado y fecha.

Elimine las recomendaciones

Para eliminar los resultados del análisis, envíe una `DeleteBotAnalyzerRecommendation` solicitud con las teclas `botId` y`botAnalyzerRequestId`. De este modo, se eliminan permanentemente las recomendaciones asociadas a ese análisis.

> **Note**
>
> Las recomendaciones se eliminan automáticamente después de 15 días.

Temas relacionados

- Mejores prácticas para crear modelos de interacción de Amazon Lex
- Mejora de la clasificación de intenciones y la resolución de ranuras en Lex V2 con la NLU asistida
- Uso de la resolución asistida de ranuras para aclarar los valores de las ranuras en Amazon Lex V2

Amazon.qn AIntent

> **Note**
>
> Para aprovechar las características de la IA generativa debe cumplir los siguientes requisitos previos
>
> 1. Para obtener información sobre el precio de uso de Amazon Bedrock, consulte Precios de Amazon Bedrock.

2. Active las capacidades de IA generativa para la configuración regional de su bot. Para ello, siga los pasos que se indican en Optimización de la creación y el rendimiento de bots Lex V2 con IA generativa.

Puede aprovechar Amazon Bedrock FMs para responder a las preguntas de los clientes en una conversación de bots. Amazon Lex V2 ofrece una `AMAZON.QnAIntent` integrada que puede agregar a su bot. Esta intención aprovecha las capacidades de la IA generativa de Amazon Bedrock al reconocer las preguntas de los clientes y buscar una respuesta en las siguientes bases de conocimientos (por ejemplo, **`Can you provide me details on the baggage limits for my international flight?`**). Esta característica reduce la necesidad de configurar preguntas y respuestas utilizando un diálogo orientado a tareas en las intenciones de Amazon Lex V2. Esta intención también reconoce las preguntas de seguimiento (por ejemplo, **`What about domestic flight?`**) en función del historial de conversaciones y proporciona la respuesta correspondiente.

Asegúrese de que su rol de IAM tiene los permisos adecuados para acceder a `AMAZON.QnAIntent` siguiendo los pasos que se indican en Permisos para la AMAZON.QnAIntent.

Para aprovechar la `AMAZON.QnAIntent` debe haber configurado una de las siguientes bases de conocimientos.

- Base OpenSearch de datos de Amazon Service: para obtener más información, consulte Creación y gestión de dominios OpenSearch de Amazon Service.
- Índice de Amazon Kendra: para obtener más información, consulte Creación de un índice.
- Base de conocimiento de Amazon Bedrock: para obtener más información, consulte Creación de una base de conocimiento.

Puede configurar la `AMAZON.QnAIntent` de dos formas:

Para configurarla mediante configuraciones de IA generativa

1. En la consola de Amazon Lex V2, seleccione Bots en el panel de navegación izquierdo y elija el bot al que desee agregar la intención en la sección Bots.
2. En el panel de navegación izquierdo, seleccione el idioma para el que desea agregar la intención.
3. En la sección Configuraciones de IA generativa, seleccione Configurar.
4. En la sección Configuraciones de QnA, seleccione Crear intención de QnA.

Para configurarla agregando una intención integrada a su bot

1. En la consola de Amazon Lex V2, seleccione Bots en el panel de navegación izquierdo y elija el bot al que desee agregar la intención en la sección Bots.
2. En el panel de navegación izquierdo, seleccione Intenciones para el que desea agregar la intención.
3. Seleccione Agregar intención y seleccione Usar intención integrada en el menú desplegable.
4. Para obtener más información sobre la configuración de la `AMAZON.QnAIntent`, consulte AMAZON.QnAIntent.

> **Note**
>
> La `AMAZON.QnAIntent` se activa cuando un enunciado no está clasificado en ninguna de las otras intenciones presentes en el bot. Esta intención se activa cuando un enunciado no está clasificado en ninguna de las otras intenciones presentes en el bot. Tenga en cuenta que esta intención no se activará en el caso de enunciados omitidos cuando se obtenga un valor de slot. Una vez reconocida, la `AMAZON.QnAIntent` utiliza el modelo de Amazon Bedrock especificado para buscar en la base de conocimientos configurada y responder a la pregunta del cliente.

Temas

- Permisos para la AMAZON.QnAIntent

Permisos para la AMAZON.QnAIntent

Para acceder a esta característica en la consola de Amazon Lex V2, asegúrese de que su rol de consola tiene permisos de `bedrock:ListFoundationModels` y `bedrock:ListInferenceProfiles`.

El rol de IAM asociado al bot debe tener los siguientes permisos necesarios para `AMAZON.QnAIntent`. El rol de bot debe tener permisos para realizar llamadas a `bedrock:InvokeModel`. También deberá asociar una instrucción para cada almacén de datos que especifique en la `AMAZON.QnAIntent` de sus bots (consulte las instrucciones `Permissions to access Amazon Kendra index`, `Permissions to access OpenSearch Service index` y `Permissions to access knowledge base in Amazon Bedrock` de la política que aparece

a continuación). Cuando habilita la característica con la consola de Amazon Lex, las políticas se agregarán automáticamente al rol de bot siempre que el bot utilice un rol vinculado a un servicio generado por Amazon Lex.

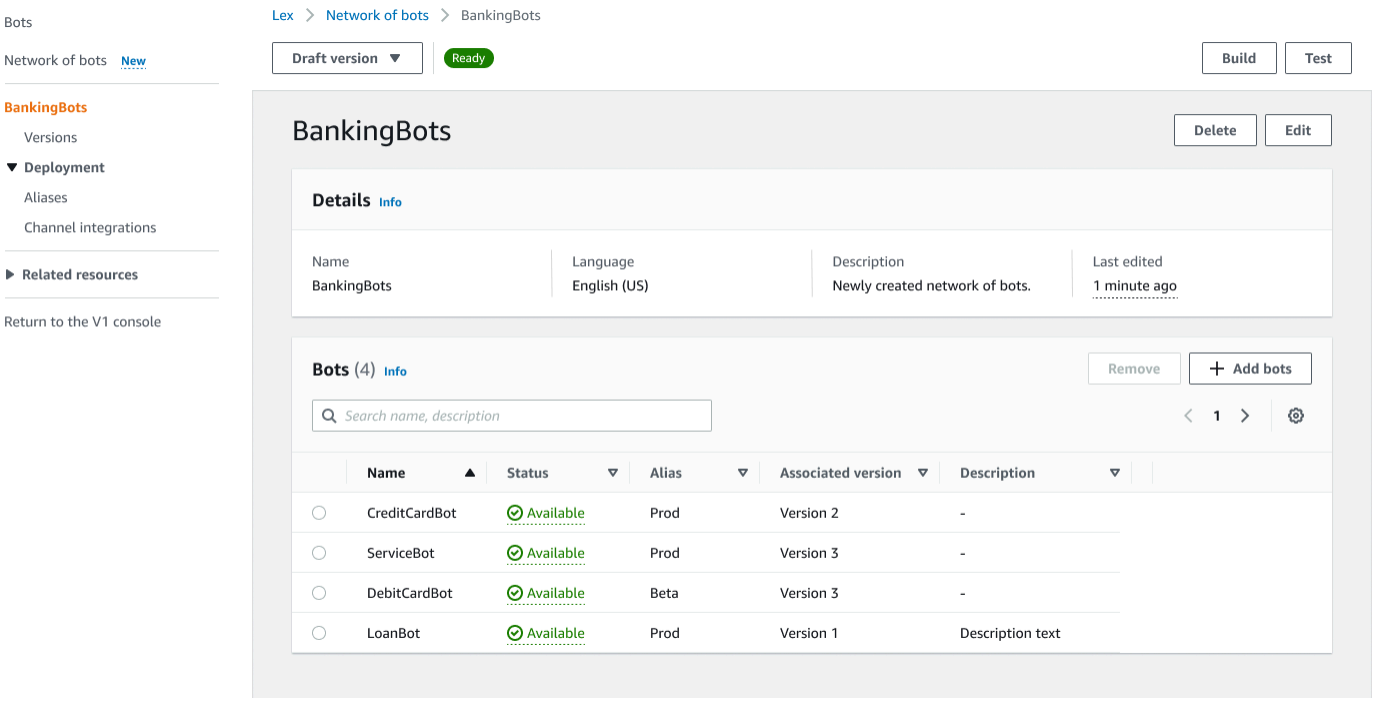
Creación de una red de bots para sus bots Lex V2

Una red de bots permite a las empresas ofrecer una experiencia de usuario unificada en varios bots. Con una red de bots, las empresas pueden añadir varios bots a una sola red para permitir una gestión flexible e independiente del ciclo de vida de los bots. La red ofrece una única interfaz unificada al usuario final y enruta la solicitud al bot correspondiente en función de las entradas del usuario.

Los equipos pueden colaborar para crear una red de bots que satisfaga diversas necesidades empresariales mediante el mantenimiento y la adición de bots a la red a medida que se implementan bots mejorados en la producción. Los desarrolladores pueden simplificar y acelerar la implementación y las mejoras integrando varios bots en una sola red.

En la actualidad, la red de bots solo está disponible en el idioma en-US.

> **Note**
>
> Actualmente, una red de bots está limitada a una cuenta. No puede añadir bots de otras cuentas.



Creación de una red de bots para sus bots Lex V2

Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/home. Seleccione Red de bots en el menú lateral. Debe haber creado al menos un bot para poder crear una red de bots.

Paso 1: Configurar los ajustes de la red de bots

1. En la sección Detalles, introduzca el nombre de su red y dele una descripción opcional.
2. En la sección de permisos de IAM, elija un rol de AWS Identity and Access Management (IAM) que otorgue permiso a Amazon Lex V2 para acceder a otros servicios de AWS, como Amazon. CloudWatch Puede hacer que Amazon Lex V2 cree el rol o puede elegir un rol existente con CloudWatch permisos. Para obtener más información, consulte Administración de identidad y acceso para Amazon Lex V2.
3. En la sección Ley de Protección de la Privacidad en Línea para Niños (COPPA), seleccione la respuesta adecuada. Para obtener más información, consulte DataPrivacy.
4. En la sección Tiempo de espera de la sesión inactivo, seleccione el tiempo que Amazon Lex V2 mantiene abierta una sesión con un usuario. Amazon Lex V2 mantiene las variables de sesión durante la sesión para que el bot pueda reanudar una conversación con las mismas variables. Para obtener más información, consulte Establecer tiempo de espera de la sesión.
5. En la sección Añadir configuración de idioma, seleccione una voz para que su bot interactúe con los usuarios. Puede escribir una frase en el Ejemplo de voz y seleccionar Reproducir para escucharla.
6. En la sección de Configuración avanzada, si lo desea, añada etiquetas que ayuden a identificar el bot. Las etiquetas pueden usarse para controlar el acceso y monitorear recursos. Para obtener más información, consulte Recursos de etiquetado.
7. Seleccione Siguiente para crear la red de bots y pasar a añadir bots.

Paso 2: Añadir bots

1. En la sección Bots, seleccione + Añadir bots.
2. Aparecerá el modo Añadir bots. Seleccione un bot para añadirlo en el menú desplegable del bot y el alias del bot que quiere usar en el menú desplegable Alias.

   El alias debe apuntar a una versión numerada del bot y no a la versión preliminar. Puede agregar hasta 5 bots. Se puede añadir un bot a un máximo de 25 redes diferentes.

3. Seleccione + Añadir bot para añadir más bots a su red. Para eliminar un bot, seleccione Eliminar junto al bot que desee eliminar. Cuando haya terminado de añadir bots, seleccione Guardar para cerrar el modo.
4. Seleccione Guardar para terminar de crear su red.

Administración de su red de bots para sus bots Lex V2

Después de crear su red de bots, llegará a una página donde podrá administrar y crear su red. También puede acceder a esta página seleccionando Red de bots en el menú lateral y seleccionando el nombre de la red que quiere gestionar.

1. Para editar la información de su red, seleccione Editar en la parte superior de la sección de Detalles. Para eliminar la red, seleccione Eliminar por encima de la sección de Detalles.
2. En la sección Bots, puede añadir más bots seleccionando + Añadir bots. También puede añadir bots si navega a la página Bots del menú lateral de la consola de Amazon Lex V2. Active el botón de radio situado junto al bot que quiere añadir y seleccione Añadir a una red de bots en el menú desplegable Acciones.

   En el menú desplegable Red de bots del modo que aparece, seleccione la red a la que quiere añadir el bot. A continuación, seleccione el alias del bot que quiera usar del menú desplegable Alias de bot. Seleccione Añadir para añadir el bot a la red que haya elegido.
3. Para eliminar los bots de la red, mueva el botón de la radio situado junto al bot y seleccione Eliminar.
4. Cuando haya terminado de configurar la red, seleccione Crear en la esquina superior derecha para crear la red. La creación puede tardar unos minutos. Si la creación se realiza correctamente, aparece un banner verde de confirmación en la parte superior de la página.
5. Una vez creada la red, puede seleccionar Probar en la esquina superior derecha para que aparezca una ventana de chat en la esquina inferior derecha. Puede usar esta ventana de chat para conversar con los bots de su red y asegurarse de que los flujos de conversación y las transiciones estén configurados correctamente.

> **Note**
>
> Si agrega, elimina o actualiza bots en su red, debe volver a crear la red.

Versiones de su red de bots para Lex V2

Puede crear diferentes versiones de su red de bots. Para gestionar las versiones, seleccione su red en el menú lateral de la consola de Amazon Lex V2 y seleccione Versiones.

1. Seleccione Crear versión para crear una nueva versión de su red de bots. También puede agregar una descripción opcional. Seleccione Crear para crear la política.
2. Cuando mueve el botón de radio situado junto a una versión de su red de bots, puede seleccionar Asociar el alias a la versión para apuntar un alias a esa versión.
3. Para administrar una versión de la red, seleccione el nombre de la versión en la sección Versiones. En la página siguiente, puede editar los detalles de la versión y administrar los bots de la versión y su alias asociado.

Alias para su red de bots para Lex V2

Puede usar alias para implementar sus redes. Para gestionar los alias, seleccione su red en el menú lateral de la consola de Amazon Lex V2 y seleccione Alias.

1. Seleccione Crear alias para crear un nuevo alias.
2. Dele un nombre al alias y, si lo desea, una Descripción en la sección de Detalles del alias. Puede elegir una versión para asociar el alias a la sección Asociar a una versión y añadir etiquetas en la sección Etiquetas. Seleccione Crear para crear el alias.
3. Para administrar un alias para su red, seleccione el nombre del alias en la sección Alias. En la página siguiente, puede editar los detalles del alias y administrar sus etiquetas, las integraciones de canales y la política basada en los recursos. También puede ver el historial de su asociación con versiones de la red.

Integraciones de canales para su red de bots Lex V2

Para integrar su red de bots con una plataforma de mensajería, seleccione su red de bots en el menú lateral de la consola de Amazon Lex V2. A continuación, seleccione Integraciones de canales.

1. Seleccione Añadir canal para integrar la red con un canal nuevo.
2. En la sección Plataforma, elija la plataforma en la que quiere implementar su bot en Seleccionar plataforma. Se creará un rol de IAM. Seleccione una clave del menú desplegable situada debajo de la clave de KMS para proteger su información.

3. En el canal de configuración de la integración, introduzca el nombre y una descripción opcional. Seleccione un Alias en el menú desplegable:
4. Obtenga el SID de su cuenta y el token de autenticación de la plataforma y complete los campos SID de cuenta y token de autenticación. Para obtener más información, consulte el tema Integrar sus bots.
5. Seleccione Crear para completar la integración del canal.

> **Note**
>
> La red de bots no está disponible actualmente en la voz o el chat de Connect Customer.

Implementación de bots Lex V2 para su entorno de producción

Tras crear y probar su bot, estará listo para su implementación e interactuará con sus clientes. En esta sección, aprenda a crear versiones de su bot cuando haya realizado una actualización. Use alias para apuntar a diferentes versiones de su bot cuando estén listas para su implementación. Aprenda a integrar sus bots con plataformas de mensajería, aplicaciones móviles y sitios web.

Temas

- Control de versiones y alias con su bot Lex V2
- Uso de una aplicación Java para interactuar con un bot de Amazon Lex V2
- Uso de la resiliencia global para implementar bots en otras regiones
- Integrar un bot de Amazon Lex V2 con una plataforma de mensajería
- Integración de un bot de Amazon Lex V2 con un centro de contacto

Control de versiones y alias con su bot Lex V2

Amazon Lex V2 admite la creación de versiones y alias de bots y redes de bots para que pueda controlar la implementación que utilizan las aplicaciones de sus clientes. Una versión actúa como una instantánea numerada de su trabajo. Puede asignar un alias a la versión de su bot que quiere que esté disponible para sus clientes. Entre la creación de versiones, puede seguir actualizando la versión `Draft` de su bot sin que ello afecte a la experiencia del usuario.

Versiones

Amazon Lex V2 admite la creación de versiones de bots para que pueda controlar la implementación que utilizan las aplicaciones de sus clientes. Una versión es una instantánea numerada de su trabajo que puede crear para su uso en diferentes partes del flujo de trabajo, como, por ejemplo, el desarrollo, la implementación beta y la producción.

La versión de borrador de su bot Lex V2

Al crear un bot de Amazon Lex V2 solo hay una versión: la versión `Draft`.

`Draft` es la copia de trabajo de su bot. Solo puede actualizar la versión `Draft` y, hasta que cree la primera versión, `Draft` es la única versión del bot con la que cuenta.

La versión `Draft` de su bot está asociada a `TestBotAlias`. El `TestBotAlias` solo debería utilizarse para la realización de pruebas manuales. Amazon Lex V2 limita el número de solicitudes de tiempo de ejecución que puede realizar al alias `TestBotAlias` del bot.

Creación de una versión para su bot Lex V2

Cuando define una versión de un bot de Amazon Lex V2, crea una instantánea numerada del bot para utilizarlo tal y como estaba en el momento de crear la versión. Después de crear una versión numérica, esta no cambiará mientras continúa trabajando en la versión preliminar de su aplicación.

Al crear una versión, puede elegir las opciones regionales que se van a incluir en la versión. No es necesario elegir todas las opciones de configuración regional de un bot. Además, al crear una versión, puede elegir una configuración regional de una versión anterior. Por ejemplo, si tiene tres versiones de un bot, puede elegir una configuración regional de la versión `Draft` y otra de la versión dos al crear la versión cuatro.

Si elimina una configuración regional de la versión `Draft`, no se eliminará de una versión numerada.

Si no se utiliza una versión de bot durante seis meses, Amazon Lex V2 marcará la versión como inactiva. Cuando una versión está inactiva, no se pueden utilizar las operaciones de tiempo de ejecución con el bot. Para activar el bot, vuelva a compilar todos los idiomas asociados a la versión.

Actualizar un bot de Amazon Lex V2

Solo puede actualizar la versión `Draft` de un bot de Amazon Lex V2. Las versiones publicadas no se pueden cambiar. Puede crear una nueva versión en cualquier momento después de actualizar un recurso en la consola o durante la CreateBotVersionoperación.

Eliminar una versión o un bot de Amazon Lex V2

Amazon Lex V2 permite eliminar un bot o versión mediante la consola o una de las operaciones API:

- DeleteBot
- DeleteBotVersion

Alias de su bot Lex V2

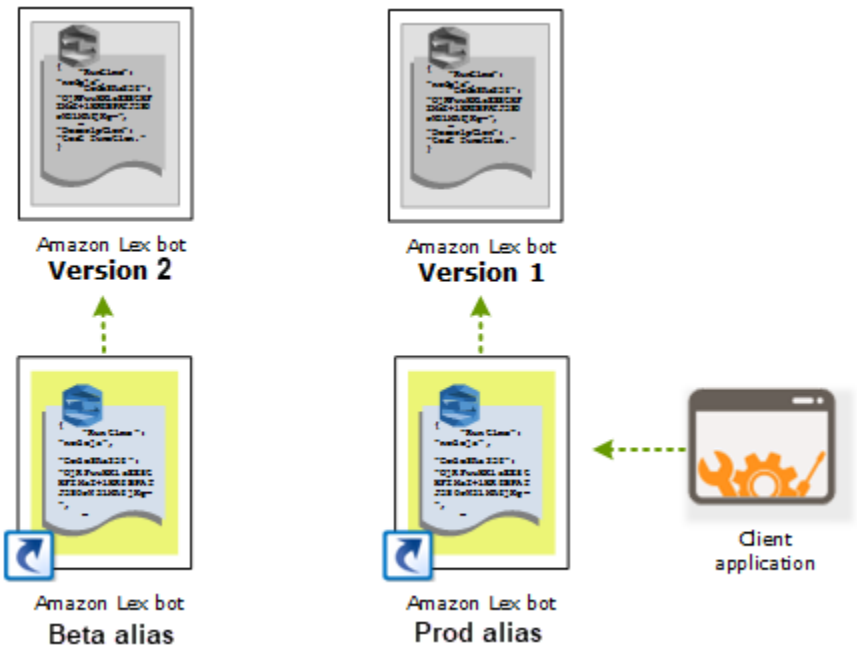
Los bots de Amazon Lex V2 admiten alias. Un alias es un puntero hacia una versión específica de un bot. Con un alias, puede actualizar fácilmente la versión que usan las aplicaciones cliente.

Por ejemplo, puede apuntar un alias hacia la versión 1 de su bot. Cuando esté listo para actualizar el bot, puede crear 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.

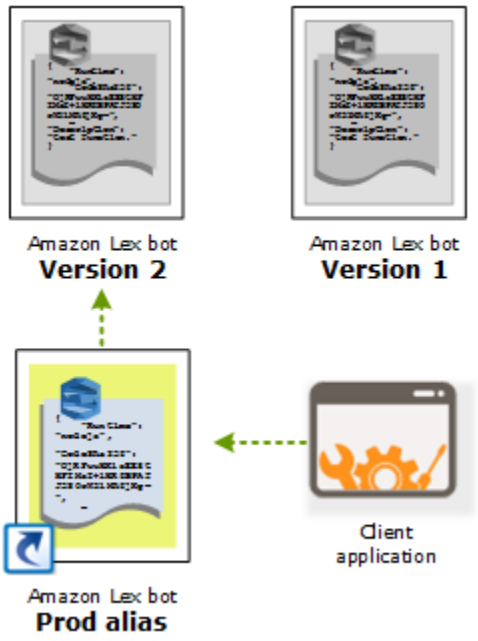
Un alias es un puntero a una versión específica de un bot de Amazon Lex V2. Utilice un alias para permitir que las aplicaciones cliente utilicen una versión específica del bot sin necesidad de que la aplicación realice un seguimiento de la versión de que se trata.

Al crear un bot, Amazon Lex V2 crea un alias llamado `TestBotAlias` que puede usar para probar el bot. El alias `TestBotAlias` de su bot siempre está asociado a la versión `Draft` de su bot. Solo debería usar el alias `TestBotAlias` con fines de prueba, Amazon Lex V2 limita el número de solicitudes de tiempo de ejecución que puede realizar al alias.

El siguiente ejemplo muestra dos versiones de un bot de Amazon Lex V2, 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 UpdateBotAlias. 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.



Cuando configura las aplicaciones de sus clientes para que llamen a Amazon Lex Runtime V2 y permita APIs a los clientes interactuar con su bot, utiliza el alias que indica la versión que quiere que usen sus clientes.

> **Note**
>
> Aunque puede probar la versión `Draft` de un bot en la consola, le recomendamos que, al integrar un bot con la aplicación cliente, primero cree 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 V2 utilizará la versión actual para todas las sesiones en curso. Las nuevas sesiones utilizan la nueva versión.

Uso de una aplicación Java para interactuar con un bot de Amazon Lex V2

La versión 2.0 de AWS SDK para Java proporciona una interfaz que puede utilizar desde sus aplicaciones Java para interactuar con sus bots. Utilice SDK para Java para crear aplicaciones de cliente para usuarios.

La siguiente aplicación interactúa con el OrderFlowers bot en Ejercicio 1: Crear un chatbot a partir de una plantilla el que creaste. Utiliza el SDK `LexRuntimeV2Client` del SDK for Java para llamar a la RecognizeTextoperación y mantener una conversación con el bot.

La salida de la conversación es como se muestra a continuación:

```
User : I would like to order flowers
Bot  : What type of flowers would you like to order?
User : 1 dozen roses
Bot  : What day do you want the dozen roses to be picked up?
User : Next Monday
Bot  : At what time do you want the dozen roses to be picked up?
User : 5 in the evening
Bot  : Okay, your dozen roses will be ready for pickup by 17:00 on 2021-01-04.  Does
 this sound okay?
User : Yes
Bot  : Thanks.
```

Para ver las estructuras JSON que se envían entre la aplicación del cliente y el bot de Amazon Lex V2, consulte Ejercicio 2: Revisar el flujo de la conversación.

Para ejecutar la aplicación, necesita proporcionar la siguiente información:

- ID de bot: el identificador único asignado al bot cuando se creó. Puede ver el ID del bot en la consola de Amazon Lex V2 en la página Configuración del bot.
- botAliasId — El identificador asignado al alias del bot cuando lo creaste. Puede ver el ID del alias del bot en la consola de Amazon Lex V2 en la página Alias. Si no puede ver el ID de alias en la lista, seleccione el icono con forma de engranaje situado en la esquina superior derecha y active el ID de alias.
- ID de configuración regional: el identificador de la configuración regional que utilizó para su bot. Para obtener una lista de las configuraciones regionales, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.
- Clave de acceso y clave secreta: las claves de autenticación de su cuenta. Si no tiene un conjunto de claves, créelas con la consola AWS Identity and Access Management.
- ID de sesión: un identificador de la sesión con el bot de Amazon Lex V2. En este caso, el código usa un UUID aleatorio.
- Región: si su bot no está en la región Este de EE. UU. (Norte de Virginia), asegúrese de cambiar de región.

La aplicación utiliza una función llamada `getRecognizeTextRequest` para crear solicitudes individuales al bot. La función crea una solicitud con los parámetros necesarios para enviarla a Amazon Lex V2.

```
package com.lex.recognizetext.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2Client;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextRequest;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextResponse;

import java.net.URISyntaxException;
import java.util.UUID;


/**
 * This is a sample application to interact with a bot using RecognizeText API.
 */
public class OrderFlowersSampleApplication {

    public static void main(String[] args) throws URISyntaxException,
 InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "en_US";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.US_EAST_1; // pick an appropriate region

        AwsBasicCredentials awsCreds = AwsBasicCredentials.create(accessKey,
 secretKey);
        AwsCredentialsProvider awsCredentialsProvider =
 StaticCredentialsProvider.create(awsCreds);

        LexRuntimeV2Client lexV2Client = LexRuntimeV2Client
                .builder()
                .credentialsProvider(awsCredentialsProvider)
                .region(region)
                .build();

        // utterance 1
        String userInput = "I would like to order flowers";
```

```
        RecognizeTextRequest recognizeTextRequest = getRecognizeTextRequest(botId,
botAliasId, localeId, sessionId, userInput);
        RecognizeTextResponse recognizeTextResponse =
lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 2
        userInput = "1 dozen roses";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 3
        userInput = "next monday";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 4
        userInput = "5 in evening";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 5
```

```
        userInput = "Yes";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
 sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });
    }

    private static RecognizeTextRequest getRecognizeTextRequest(String botId, String
 botAliasId, String localeId, String sessionId, String userInput) {
        RecognizeTextRequest recognizeTextRequest = RecognizeTextRequest.builder()
                .botAliasId(botAliasId)
                .botId(botId)
                .localeId(localeId)
                .sessionId(sessionId)
                .text(userInput)
                .build();
        return recognizeTextRequest;
    }
}
```

Uso de la resiliencia global para implementar bots en otras regiones

La resiliencia global le permite replicar un bot en una región secundaria. La región secundaria se puede activar con la replicación automática del bot del usuario en ambas regiones. Dispondrá de una región de respaldo en caso de que se produzca una interrupción regional. Una vez que la resiliencia global esté activa, los nuevos bots creados se replicarán en una segunda región de AWS.

> **Note**
>
> Esta característica solo está disponible para las instancias de Amazon Connect y Amazon Lex V2 creadas en las regiones Este de EE. UU. (Norte de Virginia) y Oeste de EE. UU. (Oregón), además de Oeste de Europa (Londres) y Europa central (Fráncfort).

Para obtener acceso a esta característica, contacte con su arquitecto de soluciones o administrador técnico de cuentas de Amazon Connect.

Tras activar esta función, puede automatizar la replicación de los bots de Amazon Lex V2 y sus recursos, versiones y alias en una región de AWS emparejada prácticamente en tiempo real. Con esta característica, puede monitorizar el número de versión del bot original y de la réplica para asegurarse de que la réplica del bot permanece sincronizada con el bot original. Cuando habilitas la replicación, puedes activar la región de AWS predeterminada en la que deseas que se replique el bot (las regiones se basan en pares predeterminados). Cualquier actualización del bot de origen en la región de origen se actualiza automáticamente en el bot replicado de la segunda región.

Note

Cuando la Resiliencia global está activada para un bot, todos los alias existentes y sus versiones asociadas se replican en la región de réplica. Las versiones que no estaban asociadas a un alias antes de permitir la replicación se replican cuando se asocian a un alias. Todas las versiones y alias creados después de habilitar la replicación se replican automáticamente. Los usuarios pueden usar `ListBotVersionReplicas` y `ListBotAliasReplicas` revisar el estado de la replicación de cada versión y alias individuales. Las mutaciones de los bots son unidireccionales desde el bot hasta su réplica. Los usuarios no pueden modificar el bot de réplica porque siempre está sincronizado con el bot.

Información adicional sobre el uso de la resiliencia global:

- Actualmente, la resiliencia global solo funciona con pares de regiones predeterminados.

us-east-1	us-west-2
eu-west-2	eu-central-1

- Cuando la Resiliencia global está activada para un bot, todos los alias existentes y sus versiones asociadas se replican en la región de réplica. Las versiones que no estaban asociadas a un alias antes de permitir la replicación, se replican cuando se asocian a un alias. Todas las versiones y alias creados después de habilitar la replicación se replican automáticamente. Los usuarios pueden usar `ListBotVersionReplicas` y `ListBotAliasReplicas` conocer el estado de la

replicación de cada versión y alias individuales. Las mutaciones de los bots son unidireccionales desde el bot hasta su réplica. Los usuarios no pueden modificar el bot de réplica porque siempre está sincronizado con el bot.

- Cualquier alias se puede asociar a cualquier versión. Si la versión aún no está replicada, se replicará durante la asociación con el alias.

Limitaciones:

- La resiliencia global no replica los bots creados con espacios que utilizan LLM, como CFAQ y Utterance Generation.
- La resiliencia global no replica una red de bots, pero cualquier bot que forme parte de esta red se puede replicar individualmente.

Temas

- Permisos para replicar bots y administrar réplicas de bots en Lex V2
- Implementación de la resiliencia global con su bot Lex V2

Permisos para replicar bots y administrar réplicas de bots en Lex V2

Si un rol de IAM tiene la AmazonLexFullAccesspolítica asociada, puede crear y administrar réplicas de bots.

Si prefiere crear un rol con permisos mínimos para la resiliencia global, utilice la siguiente política, que contiene las siguientes declaraciones.

- Permisos para acceder al rol vinculado a un servicio de Amazon Lex V2 para la replicación de bots.
- Permisos que permiten a Amazon Lex V2 crear un rol vinculado a un servicio para la replicación de bots en su nombre.
- Permisos para llamar a la replicación del bot. APIs

Puede restringir aún más los permisos modificándolos de la siguiente manera.

- *Sustitúyalos por un bot o un alias de bot específicos IDs para limitar los permisos a bots o alias de bots específicos.

- Use un subconjunto de las acciones de `lex BotReplica` para restringir el rol a acciones específicas.

Para ver un ejemplo, consulta Cómo permitir a los usuarios crear y ver réplicas de bots, pero no eliminarlas.

Implementación de la resiliencia global con su bot Lex V2

La resiliencia global le permite replicar un bot en una región secundaria. La región secundaria se puede activar con la replicación automática del bot del usuario en ambas regiones. Dispondrá de una región de respaldo en caso de que se produzca una interrupción regional. Una vez que la resiliencia global esté activa, los nuevos bots creados se replicarán en una segunda región de AWS.

El panel de información sobre resiliencia global muestra detalles sobre sus implementaciones

Puede acceder a la siguiente información en el panel Resiliencia global:

- Detalles del origen: información sobre la región de origen del bot, el tipo de réplica, la fecha de activación de la replicación y la última versión creada. Use esta información para realizar un seguimiento de las iteraciones de su bot.
- Detalles de la replicación: después de crear la réplica del bot, puede realizar un seguimiento de la región replicada, el tipo de réplica, la fecha de sincronización de la última versión y la última versión replicada. Use esta información para realizar un seguimiento de la sincronización de la réplica de su bot.
- Región de origen: la región en la que está habilitada la resiliencia global. Puede realizar cambios en la región de origen para replicar el bot en ambas regiones.
- Tipo de réplica: indica si el bot es de solo lectura o si puede leer y escribir en función de la región.
- Región de réplica: la región secundaria que se utiliza para replicar el bot de origen para lograr una resiliencia global. Actualmente, Global Resiliency solo funciona con IAD/PDX pares LDN/FRA regionales.
- Fecha de activación de la replicación: fecha y hora en que se habilitó la réplica del bot.
- Última versión creada: la última versión del bot asociada a la réplica en la región de origen.

Activación de la resiliencia global para sus bots Lex V2

Antes de activar la resiliencia global en la consola de Amazon Lex V2, debe asegurarse de que el usuario que habilita la replicación de bots tenga permiso para crear roles vinculados a servicios (SLR). Global Resiliency utilizará estas credenciales de FAS para crear una SLR en la cuenta habilitada cuando se CreateReplica invoque. Para obtener más información sobre la configuración de la SLR para la resiliencia global en Amazon Lex V2, consulte la política administrada de AWS:. AmazonLexFullAccess

> **Note**
>
> Esta característica solo está disponible para las instancias de Amazon Connect y Amazon Lex V2 creadas en las regiones Este de EE. UU. (Norte de Virginia) y Oeste de EE. UU. (Oregón), además de Oeste de Europa (Londres) y Europa central (Fráncfort).
> Para obtener acceso a esta característica, contacte con su arquitecto de soluciones o administrador técnico de cuentas de Amazon Connect.

Active la resiliencia global y configure la replicación de bots para una segunda región:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija el bot que quiera replicar en la barra de navegación de bots situada en el panel de navegación de la izquierda.
3. Seleccione Implementación > Resiliencia global.
4. Seleccione el botón Crear réplica en la esquina superior derecha de la ventana para crear una versión preliminar del bot.

   > **Note**
   >
   > Asegúrese de que no tiene ningún bot en la región secundaria con el mismo nombre que el bot que quiere replicar (el bot debe tener un nombre único).

5. Diríjase a Resiliencia global y haga clic en Crear réplica. Esta acción crea una versión preliminar de su bot (no es necesario volver a la pestaña Resiliencia global excepto para revisar el estado o ver los detalles de las futuras versiones).

> **Note**
>
> También puede crear un bot de alias para replicarlo en la resiliencia global. Para ello, vaya a Alias y seleccione Crear un nuevo alias para la resiliencia global. Todos los alias se replicarán, incluso si se crearon antes de que se habilitara la replicación.

6. Vaya a Alias > Crear un nuevo alias en el bot que tenga habilitada la resiliencia global. Todos los alias se replicarán, incluso si se crearon antes de que se habilitara la replicación.
7. Vaya a Versión > Crear una nueva versión en el bot que tenga habilitada la resiliencia global. Se replicará cualquier versión asociada a un alias, incluso si se creó antes de que se habilitara la replicación.

> **Note**
>
> Los clientes siguen teniendo el control total de la administración de sus políticas y etiquetas basadas en los recursos para los bots replicados. Las funciones Lambda y CloudWatch los grupos de registros deberán implementarse en ambas regiones con los mismos identificadores. Los usuarios no tendrán que volver a asociar la función de Lambda en la región de réplica.

Desactivación de la resiliencia global

Puede deshabilitar la resiliencia global en cualquier momento seleccionando el botón Deshabilitar Resiliencia global. Esta acción impide que el bot de origen y cualquier alias y versión asociados a él se repliquen en otras regiones.

APIs Utilízalo con Global Resiliency para tus bots Lex V2

Puede realizar llamadas a la API en Global Resiliency de la siguiente manera. APIs Puede encontrar información adicional sobre Global Resiliency APIs y Amazon Lex V2 en la Guía de API de Amazon Lex V2.

- CreateBotReplica

  Habilita la resiliencia global y crea un bot replicado. Requiere `replicaRegion`.

  Para obtener más información, consulte CreateBotReplicala Guía de API de Amazon Lex V2.

- DeleteBotReplica

  Deshabilita la resiliencia global y elimina el bot replicado. Requiere `replicaRegion` y `botId`.

  Para obtener más información, consulte DeleteBotReplicala Guía de API de Amazon Lex V2.
- ListBotReplicas

  Enumera los bots replicados en la zona secundaria. Requiere `botId`.

  Para obtener más información, consulte ListBotReplicasla Guía de API de Amazon Lex V2.
- DescribeBotReplica

  Resumen de la información del bot replicado. Requiere `replicaRegion` y `botId`.

  Para obtener más información, consulte DescribeBotReplicala Guía de API de Amazon Lex V2.

Integrar un bot de Amazon Lex V2 con una plataforma de mensajería

En esta sección se explica cómo integrar bots de Amazon Lex V2 con las plataformas de mensajería de Facebook, Slack y Twilio. Si aún no tiene un bot de Amazon Lex V2, cree uno. En este tema, asumimos que está utilizando el bot que ha creado en Ejercicio 1: Crear un chatbot a partir de una plantilla. Sin embargo, puede utilizar cualquier bot.

> **Note**
>
> Al almacenar sus configuraciones de Facebook, Slack o Twilio, Amazon Lex V2 utiliza una AWS KMS key para cifrar la información. La primera vez que cree un canal para una de estas plataformas de mensajería, Amazon Lex V2 crea una clave administrada por el cliente predeterminada (`aws/lex`) en su cuenta de AWS o puede seleccionar su propia clave administrada por el cliente. Amazon Lex V2 solo admite claves simétricas. Para obtener más información, consulte la Guía para desarrolladores de AWS Key Management Service.

Cuando una plataforma de mensajería envía una solicitud a Amazon Lex V2, incluye información específica de la plataforma como atributo de la solicitud para la función de Lambda. Utilice este atributo para personalizar el comportamiento del bot. Para obtener más información, consulte Cómo establecer atributos de solicitud para su bot Lex V2.

Atributos de solicitud comunes para las plataformas de mensajería

Atributo	Description (Descripción)
x-amz-lex:channels:platform	Uno de los valores siguientes: • `Facebook` • `Slack` • `Twilio`

Integración de un bot de Amazon Lex V2 con Facebook Messenger

Puede alojar su bot Amazon Lex V2 en Facebook Messenger. Cuando lo haga, los usuarios de Facebook podrán interactuar con su bot para cumplir sus intenciones.

Antes de empezar, debes registrarte para obtener una cuenta de desarrollador de Facebook en https://developers.facebook.com.

Debe realizar los pasos siguientes:

Temas

- Paso 1: Crear una aplicación para Facebook
- Paso 2: Integrar Facebook Messenger con el bot de Amazon Lex V2
- Paso 3: Completar la integración con Facebook con su bot Lex V2
- Paso 4: Probar la integración de Facebook Messenger.

Paso 1: Crear una aplicación para Facebook

En el portal de desarrolladores de Facebook, cree una aplicación para Facebook y una página de Facebook.

Crear una aplicación para Facebook

1. Abre https://developers.facebook.com/apps
2. Seleccione Crear una aplicación.
3. En la página Crear una aplicación, seleccione Empresa y, a continuación, Siguiente.

4. En los campos Añadir nombre de la aplicación, Correo electrónico de contacto para la aplicación y Cuenta empresarial, seleccione las opciones adecuadas para su aplicación. Seleccione Crear aplicación para continuar.
5. En Agregar productos a su aplicación, seleccione Configurar en el icono de Messenger.
6. En la sección Tokens de acceso, seleccione Añadir o eliminar páginas.
7. Seleccione una página para usarla con su aplicación y, a continuación, seleccione Siguiente.
8. En Qué puede hacer la aplicación, deje los valores predeterminados y seleccione Listo.
9. En la página de confirmación, seleccione OK.
10. En la sección Tokens de acceso, seleccione Generar token y copie el token. Introduzca este token en la consola de Amazon Lex V2.
11. En el menú de la izquierda, seleccione Configuración y, a continuación, Básica.
12. En Clave secreta de la aplicación, seleccione Mostrar y, a continuación, copie la clave secreta. Introduzca este token en la consola de Amazon Lex V2.

Siguiente paso

Paso 2: Integrar Facebook Messenger con el bot de Amazon Lex V2

Paso 2: Integrar Facebook Messenger con el bot de Amazon Lex V2

En este paso, vincule su bot de Amazon Lex V2 con Facebook.

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. De la lista de bots, seleccione el bot de Amazon Lex V2 que creó.
3. En el menú de la izquierda, seleccione Integraciones de canales y, a continuación, seleccione Añadir canal.
4. En Crear un canal, haga lo siguiente:

    a. En Plataforma, seleccione Facebook.

    b. Para las políticas de identidad, elija la AWS KMS clave para proteger la información del canal. Amazon Lex V2 proporciona la clave predeterminada.

    c. Para Configurar la integración, asigne un nombre al canal y, si lo desea, una descripción. Seleccione el alias dirigido a la versión del bot que se va a utilizar y seleccione el idioma compatible con el canal.

d. En Configuración adicional, haga lo siguiente:

- Alias: cadena que identifica la aplicación que llama a Amazon Lex V2. Puede usar cualquier cadena. Grabe esta cadena e introdúzcala en la consola de desarrolladores de Facebook.
- Token de acceso a la página: el token de acceso a la página que copió de la consola de desarrolladores de Facebook.
- Clave secreta de la aplicación: la clave secreta que copiaste de la consola de desarrolladores de Facebook.

e. Seleccione Crear.

f. Amazon Lex V2 muestra la lista de canales de su bot. En la lista, seleccione el canal que acaba de crear.

g. En la URL de devolución de llamada, grabe la URL de devolución de llamada. Introduzca esta URL en la consola de desarrolladores de Facebook.

Siguiente paso

Paso 3: Completar la integración con Facebook con su bot Lex V2

Paso 3: Completar la integración con Facebook con su bot Lex V2

En este paso, utilice la consola de desarrolladores de Facebook para completar la integración con un bot de Amazon Lex V2.

Para completar la integración con Facebook Messenger

1. Abre https://developers.facebook.com/apps
2. En la lista de aplicaciones, seleccione la aplicación que va a integrar con Facebook Messenger.
3. En el menú de la izquierda, seleccione Messenger y, a continuación, Configuración.
4. En la sección Webhooks:

   a. Seleccione Añadir URL de devolución de llamada.

   b. En Editar URL de devolución de llamada, introduzca lo siguiente:

   - URL de devolución de llamada: introduzca la URL de devolución de llamada que grabó desde la consola de Amazon Lex V2.
   - Token de verificación: introduzca el alias que introdujo en la consola de Amazon Lex V2.

c. Seleccione Verificar y guardar.

d. Seleccione Agregar suscripciones en Webhooks junto a su página.

e. En la ventana que aparece, seleccione `messages` y, a continuación, haz clic en Guardar.

Siguiente paso

Paso 4: Probar la integración de Facebook Messenger.

Paso 4: Probar la integración de Facebook Messenger.

Ahora puede iniciar una conversación desde Facebook Messenger con su bot de Amazon Lex V2.

Probar la integración entre Facebook Messenger y un bot de Amazon Lex V2

1. Abra la página de Facebook que asoció a su bot en el paso 1.
2. En la ventana de Messenger, utilice los enunciados de prueba que se facilitaron en Ejercicio 1: Crear un chatbot a partir de una plantilla.

Integración de un bot de Amazon Lex V2 con Slack

Este tema proporciona instrucciones para la integración de un bot de Amazon Lex V2 con la aplicación de mensajería Slack. Debe realizar los pasos siguientes:

Temas

- Paso 1: Registrarse en Slack y crear un equipo de Slack
- Paso 2: Crear una aplicación de Slack
- Paso 3: Integrar la aplicación de Slack con el bot de Amazon Lex V2
- Paso 4: Finalizar la integración con su bot Lex V2
- Paso 5: Comprobar la integración entre su bot Lex V2 y Slack

Paso 1: Registrarse en Slack y crear 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 de Slack, que pueda instalar cualquier equipo de Slack.

Siguiente paso

Paso 2: Crear una aplicación de Slack

Paso 2: Crear una aplicación de Slack

En esta sección, hará lo siguiente:

1. Cree una aplicación de 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, clave secreta de cliente y token de verificación). En el siguiente paso, utilizará esta información para integrar el bot en la consola de Amazon Lex V2.

Para crear una aplicación de Slack

1. Inicia sesión en la consola de API de Slack en https://api.slack.com.
2. Cree una aplicación.

   Si ha creado la aplicación correctamente, Slack muestra la página Información básica 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 Solicitar 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.

     - Seleccione Guardar cambios.
4. En el menú izquierdo, en Configuración, seleccione Información básica. Registre las siguientes credenciales de la aplicación:

- ID de cliente
- Clave secreta del cliente
- Token de verificación

Siguiente paso

Paso 3: Integrar la aplicación de Slack con el bot de Amazon Lex V2

Paso 3: Integrar la aplicación de Slack con el bot de Amazon Lex V2

En esta sección, integre la aplicación de Slack que creó con el bot de Amazon Lex V2 que creó mediante integraciones de canales.

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. De la lista de bots, seleccione el bot de Amazon Lex V2 que creó.
3. En el menú de la izquierda, seleccione Integraciones de canales y, a continuación, seleccione Añadir canal.
4. En Crear un canal, haga lo siguiente:

   a. En Plataforma, seleccione Slack.

   b. Para las políticas de identidad, elija la AWS KMS clave para proteger la información del canal. Amazon Lex V2 proporciona la clave predeterminada.

   c. Para Configurar la integración, asigne un nombre al canal y, si lo desea, una descripción. Seleccione el alias dirigido a la versión del bot que se va a utilizar y seleccione el idioma compatible con el canal.

   > **Note**
   >
   > Si su bot está disponible en varios idiomas, debe crear un canal diferente y una aplicación diferente para cada idioma.

   d. En Configuración adicional, haga lo siguiente:

   - ID de cliente: introduzca el ID de cliente de Slack.
   - Clave secreta de cliente: introduzca la clave secreta de cliente de Slack.

- Token de verificación: introduzca el token de verificación de Slack.
- URL de la página de éxito: la URL de la página que Slack debe abrir cuando el usuario se autentica. Normalmente, puede dejarlo en blanco.

5. Seleccione Crear para crear el canal.
6. Amazon Lex V2 muestra la lista de canales de su bot. En la lista, seleccione el canal que acaba de crear.
7. En la URL de devolución de llamada, registre el punto final y el OAuth punto final.

Siguiente paso

Paso 4: Finalizar la integración con su bot Lex V2

Paso 4: Finalizar la integración con su bot Lex V2

En esta sección, utilice la consola de la API de Slack para completar la integración de la aplicación de Slack.

1. Inicia sesión en la consola de la API de Slack en. https://api.slack.com Seleccione la aplicación que ha creado en el Paso 2: Crear una aplicación de Slack.
2. Actualiza la función OAuth y permisos de la siguiente manera:

   a. En el menú de la izquierda, selecciona OAuth & Permisos.

   b. En la URLs sección Redirigir, añada el OAuth punto de conexión que Amazon Lex proporcionó en el paso anterior. Elija Add (Añadir) y, a continuación, elija Save (Guardar) URLs.

   c. En la sección Ámbitos de los tokens de bots, añada dos permisos con el botón Añadir un OAuth ámbito. Filtre la lista con el siguiente texto:

      - **`chat:write`**
      - **`team:read`**

3. Actualice la función Interactividad y atajos mediante la actualización del valor de Solicitar URL con el punto de conexión que Amazon Lex proporcionó en el paso anterior. Introduzca el punto de conexión que guardó en el paso 3 y, a continuación, seleccione Guardar cambios.
4. Suscríbase a la función Suscripciones de eventos de la siguiente manera:

   - Habilite los eventos mediante la opción On.

- Defina el valor de Solicitar URL en el punto de conexión que Amazon Lex ha facilitado en el paso anterior.
- En la sección Suscribirse a eventos de bot, suscríbase a Añadir evento para usuarios de bot y añada el evento del bot de **`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.
6. Seleccione Administrar distribución en Configuración. Seleccione Añadir a Slack para instalar la aplicación. Si está autenticado en varios espacios de trabajo, seleccione primero el espacio de trabajo correcto en la esquina superior derecha de la lista desplegable. A continuación, seleccione Permitir para autorizar al bot a responder a los mensajes.

> **Note**
>
> Si realiza algún cambio en la configuración de la aplicación de Slack más adelante, debe volver a realizar este subpaso.

Siguiente paso

Paso 5: Comprobar la integración entre su bot Lex V2 y Slack

Paso 5: Comprobar la integración entre su bot Lex V2 y Slack

Ahora utilice una ventana de navegador para probar la integración de Slack con el bot de Amazon Lex V2.

Para probar las aplicaciones de Slack

1. Abra Slack. En el menú de la izquierda, en la sección Mensajes directos, seleccione su bot. Si no ve su bot, seleccione el icono más (+) junto a Mensajes directos para buscarlo.
2. Participe en un chat con su aplicación de Slack. Ahora su bot responde a los mensajes.

Si creó el bot usando Ejercicio 1: Crear un chatbot a partir de una plantilla, puede utilizar los ejemplos de conversaciones proporcionados en dicho ejercicio.

Integración de un bot de Amazon Lex V2 con Twilio SMS

Este tema proporciona instrucciones para integrar un bot de Amazon Lex V2 con el servicio de mensajería simple (SMS) de Twilio. Debe realizar los pasos siguientes:

Temas

- Paso 1: Crear una cuenta de Twilio SMS
- Paso 2: Integrar el punto de conexión del servicio de mensajería de Twilio con el bot de Amazon Lex V2
- Paso 3: Completar la integración de Twilio entre su bot Lex V2 y Twilio
- Paso 4: Comprobar la integración de su bot Lex V2 y Twilio

Paso 1: Crear 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 obtener instrucciones de registro, consulte https://www.twilio.com/console.

Siguiente paso

Paso 2: Integrar el punto de conexión del servicio de mensajería de Twilio con el bot de Amazon Lex V2

Paso 2: Integrar el punto de conexión del servicio de mensajería de Twilio con el bot de Amazon Lex V2

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. De la lista de bots, seleccione el bot de Amazon Lex V2 que creó.

3. En el menú de la izquierda, seleccione Integraciones de canales y, a continuación, seleccione Añadir canal.
4. En Crear un canal, haga lo siguiente:
   a. En Platform, seleccione Twilio.
   b. Para las políticas de identidad, elija la AWS KMS clave para proteger la información del canal. Amazon Lex V2 proporciona la clave predeterminada.
   c. Para Configurar la integración, asigne un nombre al canal y, si lo desea, una descripción. Seleccione el alias dirigido a la versión del bot que se va a utilizar y seleccione el idioma compatible con el canal.
   d. Para una Configuración adicional, introduzca el SID de la cuenta y el token de autenticación desde el panel de control de Twilio.
5. Seleccione Crear.
6. En la lista de canales, seleccione el canal que acaba de crear.
7. Copie la URL de devolución de llamada.

Siguiente paso

Paso 3: Completar la integración de Twilio entre su bot Lex V2 y Twilio

Paso 3: Completar la integración de Twilio entre su bot Lex V2 y Twilio

Utilice la consola de Twilio para completar la integración de un bot de Amazon Lex V2 con Twilio SMS.

1. Abre la consola de Twilio en https://www.twilio.com/console.
2. En el menú de la izquierda, seleccione Todos los productos y servicios y, a continuación, seleccione Número de teléfono.
3. Si tiene un número de teléfono, selecciónelo. Si no tiene un número de teléfono, seleccione Comprar un número para obtener uno.
4. En la sección Mensajes, en LLEGA UN MENSAJE, introduzca la URL de devolución de llamada de la consola de Amazon Lex V2.
5. Seleccione Guardar.

Siguiente paso

Paso 4: Comprobar la integración de su bot Lex V2 y Twilio

Paso 4: Comprobar la integración de su bot Lex V2 y Twilio

Utilice su teléfono móvil para probar la integración entre Twilio SMS y su bot. Use su teléfono móvil para enviar mensajes al número de Twilio.

Si creó el bot usando Ejercicio 1: Crear un chatbot a partir de una plantilla, puede utilizar los ejemplos de conversaciones proporcionados en dicho ejercicio.

Integración de un bot de Amazon Lex V2 con un centro de contacto

Puede integrar los bots de Amazon Lex V2 en sus centros de contacto para habilitar casos de uso de autoservicio mediante la API de transmisión de Amazon Lex V2. Utilice estos bots como agentes de respuesta de voz interactiva (IVR) en telefonía o como chatbot de texto integrado en su centro de contacto. Para obtener más información acerca de las API de transmisión, consulte Transmisión de conversaciones a un bot Amazon Lex V2.

Con las API de transmisión, puede habilitar las siguientes funciones:

- Interrupciones («barge-in»): las personas que llaman pueden interrumpir el bot y responder a una pregunta antes de que se complete el mensaje. Para obtener más información, consulte Cómo permitir que su usuario interrumpa a su bot Amazon Lex V2.
- Esperar y continuar: las personas que llamen pueden indicarle al bot que espere si necesitan tiempo para recuperar información adicional durante una llamada, como un número de tarjeta de crédito o un identificador de reserva. Para obtener más información, consulte Cómo permitir que el bot Amazon Lex V2 espere a que el usuario proporcione más información durante una pausa.
- Compatibilidad con DTMF: las personas que llaman pueden proporcionar información por voz o DTMF de forma intercambiable.
- Compatibilidad con SSML: puede configurar las indicaciones del bot de Amazon Lex V2 mediante etiquetas SSML para tener un mayor control sobre la generación de voz a partir del texto. Para obtener más información, consulte Generación de voz a partir de documentos SSML en la Guía para desarrolladores de Amazon Polly.
- Tiempos de espera configurables: puede configurar cuánto tiempo deben esperar a que los clientes terminen de hablar antes de que Amazon Lex V2 recopile su entrada de voz, como

responder a una pregunta de sí o no, o proporcionar una fecha o un número de tarjeta de crédito. Para obtener más información, consulte Configuración de los tiempos de espera para capturar entradas del usuario con un bot Lex V2.

- Actualizaciones del progreso del cumplimiento: puede configurar el bot para que responda con varios mensajes en función del estado del cumplimiento durante la ejecución de la lógica empresarial para el cumplimiento de la intención. Puede configurar el bot para que responda con mensajes cuando comience y finalice el cumplimiento, y para que proporcione actualizaciones periódicas para las funciones de Lambda de larga ejecución. Para obtener más información, consulte Configuración de las actualizaciones del progreso de cumplimiento para su bot Lex V2.

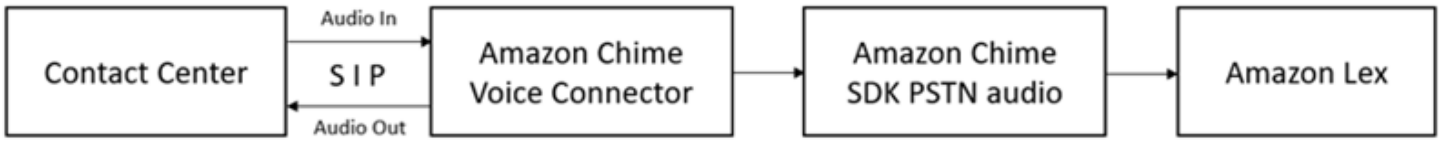
Amazon Chime SDK

Utilice el Amazon Chime SDK para añadir funciones de audio, vídeo, uso compartido de pantalla y mensajería en tiempo real a sus aplicaciones web o móviles. El Amazon Chime SDK proporciona un servicio de audio de red telefónica pública conmutada (PSTN) para que pueda crear aplicaciones de telefonía personalizadas con una función AWS Lambda .

El audio Amazon Chime PSTN está integrado en Amazon Lex V2. Puede utilizar esta integración para acceder a los bots de Amazon Lex V2 como sistemas de respuesta de voz interactiva (IVR) en los centros de contacto para interacciones de audio. Úselo para integrar Amazon Lex V2 mediante servicios de audio PSTN en los siguientes escenarios.

Integraciones de centros de contacto: puede utilizar el Amazon Chime Voice Connector y el servicio de audio Amazon Chime SDK PSTN para acceder a los bots de Amazon Lex V2. Úselos en cualquier aplicación de centro de contacto que utilice el protocolo de inicio de sesión (SIP) para las comunicaciones de voz. Esta integración añade experiencias de conversación de voz en lenguaje natural a su centro de contacto local o basado en la nube con compatibilidad con SIP. Para obtener una lista de las plataformas de centros de contacto compatibles, consulte los recursos de Amazon Chime Voice Connector.

El siguiente diagrama muestra la integración entre un centro de contacto mediante SIP y Amazon Lex V2.



Soporte de telefonía directa: puede crear soluciones IVR personalizadas para acceder directamente a los bots de Amazon Lex V2 mediante un número de teléfono proporcionado en el Amazon Chime SDK.

Para obtener más información, consulte los siguientes temas en la Guía de Amazon Chime SDK.

- Integración SIP mediante un Amazon Chime Voice Connector
- Uso del servicio de audio Amazon Chime SDK PSTN
- Integración del audio Amazon Chime PSTN en Amazon Lex V2

Cuando el Amazon Chime SDK envía una solicitud a Amazon Lex V2, incluye información específica de la plataforma en sus registros de conversaciones y funciones de Lambda. Utilice esta información para determinar la aplicación del centro de contacto que envía tráfico a su bot.

Atributos de solicitud comunes	Valor
x-amz-lex: canales: plataforma	`Amazon Chime SDK PSTN Audio`

Cliente de Connect

Connect Customer es un centro de contacto omnicanal en la nube. Puede configurar un centro de contacto en pocos pasos, agregar agentes ubicados en cualquier lugar y comenzar a interactuar con sus clientes. Para obtener más información, consulte Cómo empezar a usar Connect Customer en la guía del administrador de Connect Customer.

Puede crear experiencias personalizadas para sus clientes mediante comunicaciones omnicanal. Por ejemplo, puede ofrecer chat y contacto de voz en función de las preferencias del cliente y los tiempos de espera estimados. Los agentes pueden gestionar todos los clientes desde una sola interfaz. Por ejemplo, pueden hablar con los clientes y crear tareas o responder a ellas a medida que se les enrutan.

Puedes usar Connect Customer para interacciones de audio con tus clientes o Connect Customer Chat para interacciones de solo texto.

Para obtener más información, consulte los siguientes temas de la guía del administrador de Connect Customer.

- Qué es Connect Customer

- Agregar un bot de Amazon Lex
- Connect Customer obtiene el bloque de contactos de entrada del cliente

Cuando un centro de contacto envía una solicitud a Amazon Lex V2, incluye información específica de la plataforma como atributo de la solicitud para la función de Lambda y registros de conversación. Utilice esta información para determinar qué aplicación del centro de contacto envía tráfico a su bot.

Atributos de solicitud comunes para Amazon Connect

Atributo	Valor
x-amz-lex:channels:platform	Uno de los valores siguientes: • `Connect` • `Connect Chat`

Genesys Cloud

Genesys Cloud es un conjunto de servicios en la nube para la comunicación empresarial, la colaboración y la gestión de centros de contacto. Genesys Cloud se basa en AWS y utiliza un entorno de nube distribuido que proporciona acceso seguro a las organizaciones de todo el mundo.

Para obtener más información, consulte las siguientes páginas en el sitio web de Genesys Cloud.

- Acerca del centro de contacto de Genesys Cloud
- Acerca de la integración de Amazon Lex V2

Cuando un centro de contacto envía una solicitud a Amazon Lex V2, incluye información específica de la plataforma como atributo de la solicitud para la función de Lambda y registros de conversación. Utilice esta información para determinar qué aplicación del centro de contacto envía tráfico a su bot.

Atributos de solicitud comunes para Genesys Cloud

Atributo	Valor
x-amz-lex:channels:platform	• `Genesys Cloud`

Más información

- Potencie su centro de contacto con Amazon Lex y Genesys Cloud

Descripción de las conversaciones de los bots de Amazon Lex V2

Tras compilar un bot, debe integrar la aplicación cliente con las operaciones de tiempo de ejecución de Amazon Lex V2 para mantener conversaciones con el bot. Cuando un usuario inicia una conversación con un bot, Amazon Lex V2 crea una sesión. Una sesión encapsula la información intercambiada entre la aplicación y el bot. Para obtener más información, consulte Introducción a las sesiones de bots Amazon Lex V2.

Una conversación típica implica un flujo de ida y vuelta entre el usuario y un bot. Por ejemplo:

```
User : I'd like to make an appointment
Bot : What type of appointment would you like to schedule?
User : dental
Bot : When should I schedule your dental appointment?
User : Tomorrow
Bot : At what time do you want to schedule the dental appointment on 2021-01-01?
User : 9 am
Bot : 09:00 is available, should I go ahead and book your appointment?
User : Yes
Bot : Thank you. Your appointment has been set successfully.
```

Utilice las RecognizeTextoperaciones de la RecognizeUtteranceAPI para gestionar las conversaciones usted mismo. Utilice la operación de StartConversationAPI para permitir que Amazon Lex V2 gestione la conversación por usted.

Para administrar la conversación, debe enviar los enunciados de los usuarios al bot hasta que la conversación llegue a un final lógico. La conversación actual se captura en el estado de la sesión. El estado de la sesión se actualiza después de cada enunciado del usuario. El estado de la sesión contiene el estado actual de la conversación y lo devuelve el bot en respuesta a cada enunciado del usuario.

Una conversación puede tener uno de los siguientes estados:

- ElicitIntent— Indica que el bot aún no ha determinado la intención del usuario.
- ElicitSlot— Indica que el bot ha detectado la intención del usuario y está recopilando la información necesaria para cumplirla.

- ConfirmIntent— Indica que el bot está esperando a que el usuario confirme que la información recopilada es correcta.
- Cerrada: indica que la intención del usuario es completa y que la conversación con el bot ha llegado a un final lógico.

Un usuario puede especificar una nueva intención una vez completada la primera. Para obtener más información, consulte Contexto de conversación con sus bots Lex V2.

Una intención puede tener los siguientes estados:

- InProgress— Indica que el bot está recopilando la información necesaria para cumplir la intención. Esto se produce junto con el estado de la conversación `ElicitSlot`.
- En espera: indica que el usuario solicitó al bot que esperara cuando este solicitó información para un slot específico.
- Cumplida: indica que la lógica empresarial de una función de Lambda asociada a la intención se ejecutó correctamente.
- ReadyForFulfillment— Indica que el bot recopiló toda la información necesaria para cumplir la intención y que la aplicación cliente puede ejecutar una lógica empresarial de cumplimiento.
- Error: indica que se ha producido un error en una intención.

Consulte los siguientes temas para aprender a utilizar Amazon Lex V2 APIs para gestionar el contexto de las conversaciones y las sesiones entre el bot y los usuarios.

Temas

- Contexto de conversación con sus bots Lex V2
- Introducción a las sesiones de bots Amazon Lex V2

Contexto de conversación con sus bots Lex V2

El contexto de conversación es la información que el usuario, la aplicación cliente o una función de Lambda proporcionan a un bot de Amazon Lex V2 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

- Cómo establecer un contexto de intención para su bot Lex V2
- Uso de los valores de ranura predeterminados en las intenciones de su bot Lex V2
- Cómo establecer atributos de sesión para su bot Lex V2
- Cómo establecer atributos de solicitud para su bot Lex V2
- Establecer el tiempo de espera de la sesión
- Cómo compartir información entre intenciones con su bot Lex V2
- Cómo establecer atributos complejos en su bot Lex V2

Cómo establecer un contexto de intención para su bot Lex V2

Puede hacer que Amazon Lex V2 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 CreateIntent. Solo puede usar el contexto en la configuración regional inglesa (US) (en-US).

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 RecognizeText o RecognizeUtterance 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 V2 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 `RecognizeText` o `RecognizeUtterance`.

También puede establecer el contexto de una conversación con la función de Lambda para la intención. El contexto de salida de Amazon Lex V2 se envía al evento de entrada de la función Lambda. La función de Lambda puede enviar contextos en su respuesta. Para obtener más información, consulte Integración de una AWS Lambda función en su bot Amazon Lex V2.

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 V2 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 intenciones de salida para su bot Lex V2

Amazon Lex V2 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, consulteUso de los valores de ranura predeterminados en las intenciones de su bot Lex V2.

El contexto de salida se configura cuando crea una intención con la consola o con la operación CreateIntent. 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 RecognizeText o RecognizeUtterance.

Cuando define un contexto de salida, también define su tiempo de vida, el tiempo o el número de turnos que el contexto incluye en las respuestas de Amazon Lex V2. Un turno es una solicitud de su solicitud a Amazon Lex V2. 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».
- Devuelva el contexto de salida a Amazon Lex V2 como contexto de entrada para el siguiente enunciado. Si Amazon Lex V2 reconoce el enunciado como un intento de provocar una intención, utiliza el contexto para limitar las intenciones que se pueden devolver a las que tienen el contexto especificado.

Contexto de intenciones de entrada para su bot Lex V2

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 `CreateIntent`. Una intención puede tener más de un contexto de entrada.

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 RecognizeText, RecognizeUtterance 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 V2 reconoce una nueva intención pero no recibe un valor de espacio. 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 los valores de ranura predeterminados en las intenciones de su bot Lex V2.

Uso de los valores de ranura predeterminados en las intenciones de su bot Lex V2

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 CreateIntent 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": [
    {
        "botId": "string",
        "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 parámetro «StartDate» no está establecido, el valor del atributo de sesión «reservaStartDate» se utiliza como valor predeterminado.
3. Si no se utiliza ninguno de los dos primeros valores predeterminados, la ranura no tiene un valor predeterminado y Amazon Lex V2 generará un valor como de costumbre.

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

Cómo establecer atributos de sesión para su bot Lex V2

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 transfiere los atributos de sesión a todas las funciones de Lambda configuradas para un bot. Si una función de Lambda añade o actualiza los atributos de sesión, Amazon Lex V2 devuelve la nueva información a la aplicación cliente.

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 RecognizeText o RecognizeUtterance. 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, «Hey [FirstName], ¿qué ingredientes le gustaría añadir?» Si pasa el nombre del usuario como un atributo de sesión (`{"FirstName": "Vivian"}`), Amazon Lex lo reemplazará donde aparezca el marcador. A continuación, envía una pregunta personalizada al usuario, «Hola, Vivian, ¿qué ingredientes quieres?».

Los atributos de sesión persisten durante toda la sesión. Amazon Lex V2 los almacena en un almacén de datos cifrados hasta que finaliza la sesión. El cliente puede crear atributos de sesión en una solicitud llamando a la RecognizeUtteranceoperación RecognizeTexto con el `sessionAttributes` campo establecido 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 crean un atributo de sesión, el valor del atributo almacenado se utiliza siempre que la aplicación cliente no incluya un `sessionAttribute` campo en una solicitud a Amazon Lex V2.

Por ejemplo, supongamos que tiene dos atributos de sesión, `{"x": "1", "y": "2"}`. Si el cliente llama a la `RecognizeUtterance` operación `RecognizeText` o sin especificar el `sessionAttributes` campo, Amazon Lex V2 llama a la función Lambda con los atributos de sesión almacenados (){`"x": 1, "y": 2`}. Si la función Lambda no devuelve los atributos de sesión, Amazon Lex V2 devuelve los atributos de sesión almacenados a la aplicación cliente.

Si la aplicación cliente o una función Lambda pasan los atributos de sesión, Amazon Lex V2 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 los atributos de sesión a Amazon Lex V2, debe crear un mapa de los atributos cadena a cadena. A continuación, se muestra cómo asignar atributos de sesión:

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

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

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

Para la operación `RecognizeUtterance`, hay que codificar en base64 la asignación y luego enviarla como parte del encabezado de `x-amz-lex-session-state`.

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 Cómo establecer atributos complejos en su bot Lex V2.

Atributos de sesión predefinidos

Amazon Lex proporciona atributos de sesión predefinidos. Estos atributos administran la forma en que Amazon Lex procesa la información que se envía a su bot. Los atributos se conservan durante toda la sesión. Todos los atributos predefinidos se encuentran en el espacio de nombres `x-amz-lex:`.

Aumento del tiempo de espera de Lambda para el enlace de código

Para aumentar el tiempo de espera de Lambda del enlace de código en tu bot, que es de 30 segundos de forma predeterminada, usa el `x-amz-lex:codehook-timeout-ms` atributo session. Establece el valor del atributo de sesión en milisegundos. El tiempo de espera máximo es de 120 segundos.

Por ejemplo, si un usuario establece «x-amz-lex:codehook-timeout-ms»: «90000", el tiempo de espera del enlace de código Lambda será de 90 segundos.

Cómo establecer atributos de solicitud para su bot Lex V2

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 V2. 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 RecognizeUtterance o `requestAttributes` en una solicitud RecognizeText. Dado que los atributos de solicitud no persisten en las solicitudes como atributos de sesión, no se devuelven en las respuestas `RecognizeUtterance` o `RecognizeText`.

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

Cómo establecer atributos de solicitud definidos por el usuario para cada solicitud del bot Lex V2

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 `RecognizeUtterance` o en el campo `requestAttributes` de una solicitud `RecognizeText`.

Para enviar los atributos de la solicitud a Amazon Lex V2, debe crear un mapa de los atributos cadena a cadena. 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 Cómo establecer atributos complejos en su bot Lex V2.

Amazon Lex V2 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 esos 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 plano UTF-8 .
- `SSML`— El mensaje contiene texto formateado para la salida de voz.
- `CustomPayload`— El mensaje contiene un formato personalizado que ha creado para su cliente. Puede definir la carga para satisfacer las necesidades de su aplicación.

Amazon Lex V2 devuelve únicamente los mensajes con el tipo especificado en el campo Mensaje 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.

Establecimiento de atributos de solicitud predefinidos para su bot de Lex V2

Amazon Lex V2 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:`.

Desactivación de los interruptores de intención en su bot de Lex V2

Para controlar si los usuarios pueden cambiar de una intención a otra durante la confirmación de la intención o la selección de franjas horarias, utiliza el atributo `x-amz-lex:intent-switch` request. Si se establece en`DISABLE`, este atributo evita que los usuarios activen una intención diferente mientras están completando el flujo de intenciones actual.

Por ejemplo, si un usuario está reservando un vuelo y se le piden los detalles del vuelo, se ignorarán las peticiones del tipo “consulta el tiempo” o “reserva un hotel”, que normalmente podrían generar más intentos, para garantizar que la conversación se centre en el proceso de reserva actual.

Establecer el 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 Amazon Lex V2, se establece el tiempo de espera de la sesión al crear un bot. Con la interfaz de línea de comandos de AWS (AWS CLI) o la API, se establece el tiempo de espera al crear un bot con la CreateBotoperación configurando el SessionTTLInSeconds campo idle.

Cómo compartir información entre intenciones con su bot Lex V2

Amazon Lex V2 permite compartir información entre intenciones. Para compartir información entre intenciones, utilice atributos de sesión.

Para utilizar los contextos de salida, se define un contexto de salida al crear o actualizar una intención. Cuando se cumple la intención, las respuestas de Amazon Lex V2 contienen el contexto y los valores de slot de la intención como parámetros de contexto. Puede utilizar estos parámetros como valores predeterminados en intentos posteriores o en el código de la aplicación o en las funciones de Lambda.

Para usar los atributos de sesión, debe configurar los atributos en su Lambda o en el código de la aplicación. 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.

Cómo establecer atributos complejos en su bot Lex V2

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 RecognizeText y RecognizeUtterance.

Por ejemplo, si está utilizando JavaScript, puede utilizar la `JSON.stringify` operación para convertir un objeto en JSON y la `JSON.parse` operación para convertir el texto JSON en un JavaScript objeto:

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

Para enviar atributos con la `RecognizeUtterance` operación, debes codificar los atributos en base64 antes de añadirlos al encabezado de la solicitud, como se muestra en el siguiente JavaScript código:

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

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

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

Introducción a las sesiones de bots Amazon Lex V2

Cuando un usuario inicia una conversación con un bot, Amazon Lex V2 crea una sesión. La información que se intercambia entre la aplicación y Amazon Lex V2 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 `sessionId` de la operación RecognizeText o RecognizeUtterance.

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 tres formas de actualizar el estado de la sesión.

- Transfiera la información de la sesión en línea como parte de una llamada a la operación `RecognizeText` o `RecognizeUtterance`.
- Utilice la función de Lambda con la operación `RecognizeText` o `RecognizeUtterance` que se invoca después de cada turno de la conversación. Para obtener más información, consulte Integración de una AWS Lambda función en su bot Amazon Lex V2. La otra es utilizar la API en tiempo de ejecución de Amazon Lex V2 en la aplicación para realizar cambios en el estado de la sesión.
- Utilice operaciones que le permitan administrar la información de la sesión en una conversación con el bot. Las operaciones son la operación PutSession, la operación GetSession y la operación 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 sesión que se hayan establecido y los valores de slot de la intención actual y cualquier otra intención que Amazon Lex V2 haya identificado como posibles intenciones y coincidan con el enunciado del usuario.

La operación `PutSession` le permite manipular directamente la sesión actual. Puede configurar la sesión, incluido el tipo de acción de diálogo que el bot realizará a continuación y los mensajes que Amazon Lex V2 enviará al usuario. 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 V2 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.

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

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.

Iniciar 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.

Cambiar 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.

Reanudar una intención anterior

Si desea reanudar una intención anterior, utilice la operación `GetSession` para obtener el estado de la intención, lleve a cabo la interacción necesaria y utilice después la operación `PutSession` para establecer la intención en el estado del diálogo anterior.

- Llame a la operación `GetSession`. Guarde el estado de la intención.
- Realice otra interacción, como cumplir una intención diferente.
- Con la información guardada para la intención anterior, llame a 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.

Validar 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.

Integración de una AWS Lambda función en su bot Amazon Lex V2

Con las funciones de AWS Lambda, puede controlar mejor el comportamiento de su bot Amazon Lex V2 mediante funciones personalizadas definidas por usted. Amazon Lex V2 utiliza una función de Lambda por alias de bot por idioma en lugar de una función de Lambda para cada intención. Antes de empezar, determine de qué campos del evento de entrada quiere extraer información y qué campos de la respuesta quiere manipular y devolver desde la función de Lambda.

Para integrar una función de Lambda con el bot de Amazon Lex V2, lleve a cabo los siguientes pasos:

1. Crea una función AWS Lambda usando el lenguaje de programación que prefieras y redacta tu script.
2. Asegúrese de que la función devuelva una estructura que coincida con el formato de respuesta.
3. Implemente la función de Lambda.
4. Asocie la función de Lambda a un alias de bot de Amazon Lex V2 con las operaciones de la consola o de la API.
5. Seleccione las etapas de conversación en las que desee invocar la función de Lambda con las operaciones de la consola o de las operaciones de la API.
6. Compile su bot de Amazon Lex V2 y compruebe que la función de Lambda funciona según lo previsto. Depura tu función con la ayuda de Amazon CloudWatch.

Temas

- AWS Lambdaformato de evento de entrada para Lex V2
- AWS Lambdaformato de respuesta para Lex V2
- Estructuras comunes de una AWS Lambda función para Amazon Lex V2
- Creación de una AWS Lambda función para su bot Amazon Lex V2
- Depuración de una función Lambda mediante registros CloudWatch

AWS Lambdaformato de evento de entrada para Lex V2

El primer paso para integrar una función de Lambda en el bot de Amazon Lex V2 consiste en comprender los campos del evento de Amazon Lex V2 y determinar la información de estos campos que desea utilizar al escribir el script. El siguiente objeto JSON muestra el formato general de un evento de Amazon Lex V2 pasado a una función de Lambda:

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

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook | FulfillmentCodeHook",
    "inputMode": "DTMF | Speech | Text",
    "responseContentType": "audio/mpeg | audio/ogg | audio/pcm | text/plain;
 charset=utf-8",
    "sessionId": string,
    "inputTranscript": string,
    "invocationLabel": string,
    "bot": {
        "id": string,
        "name": string,
        "localeId": string,
        "version": string,
        "aliasId": string,
        "aliasName": string
    },
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "intent": {
                // see Intención for details about the structure
            },
            "nluConfidence": number,
            "sentimentResponse": {
                "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
                "sentimentScore": {
                    "mixed": number,
```

```
                    "negative": number,
                    "neutral": number,
                    "positive": number
                }
            }
        },
        ...
    ],
    "proposedNextState": {
        "dialogAction": {
            "slotToElicit": string,
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
        },
        "intent": {
            // see Intención for details about the structure
        },
        "prompt": {
            "attempt": string
        }
    },
    "requestAttributes": {
        string: string,
        ...
    },
    "sessionState": {
        // see Estado de la sesión for details about the structure
    },
    "transcriptions": [
        {
            "transcription": string,
            "transcriptionConfidence": number,
            "resolvedContext": {
                "intent": string
            },
            "resolvedSlots": {
                slot name: {
                    // see Ranuras for details about the structure
                },
                ...
            }
        },
        ...
    ]
```

```
}
```

Cada campo de evento se describe a continuación.

Versión del mensaje

La versión del mensaje que identifica el formato de los datos del evento que se van a pasar a la función de Lambda y el formato previsto de la respuesta de una función de Lambda.

> **Note**
>
> Puede configurar este valor a la hora de definir una intención. En la implementación actual, Amazon Lex V2 solo 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.

Origen de invocación

El enlace de código que llamó a la función de Lambda. Se admiten los siguientes valores:

`DialogCodeHook`: Amazon Lex V2 llamó a la función de Lambda después de la entrada del usuario.

`FulfillmentCodeHook`: Amazon Lex V2 llamó a la función de Lambda después de llenar todos los slots requeridos y la intención estaba lista para su cumplimiento.

Modo de entrada

El modo de enunciado del usuario. Los valores posibles son los siguientes:

`DTMF`: el usuario introdujo el enunciado mediante un teclado táctil (multifrecuencia de doble tono).

`Speech`: el usuario pronunció el enunciado.

`Text`: el usuario escribió el enunciado.

responseContentType

El modo de respuesta del bot al usuario. `text/plain; charset=utf-8` indica que se escribió el último enunciado, mientras que un valor que comience por `audio` indica que se pronunció el último enunciado.

ID de sesión

El identificador de sesión alfanumérico utilizado para la conversación.

Transcripción de entrada

Una transcripción de la entrada del usuario.

- Para la entrada de texto, este es el texto que escribió el usuario. Para la entrada DTMF, esta es la clave que introdujo el usuario.
- En el caso de entrada de voz, este es el texto en el que Amazon Lex V2 convierte el enunciado del usuario para invocar una intención o llenar un slot.

Etiqueta de invocación

Un valor que indica la respuesta que invocó la función de Lambda. Puede establecer etiquetas de invocación para la respuesta inicial, los slots y la respuesta de confirmación.

bot

Información sobre el bot que procesó la solicitud, que consta de los siguientes campos:

- id: el identificador único asignado al bot cuando se creó. Puede ver el ID del bot en la consola de Amazon Lex V2 en la página Configuración del bot.
- Nombre: el nombre que le dio al bot cuando lo creó.
- ID de configuración regional: el identificador de la configuración regional que utilizó para su bot. Para obtener una lista de las configuraciones regionales, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.
- Versión: versión del bot que ha procesado la solicitud.
- ID de alias: ID de alias del bot: el identificador único asignado al alias del bot cuando se creó. Puede ver el ID del alias del bot en la consola de Amazon Lex V2 en la página Alias. Si no puede ver el ID de alias en la lista, seleccione el icono con forma de engranaje situado en la esquina superior derecha y active el ID de alias.
- Nombre de alias: el nombre que le diste al alias del bot.

Interpretaciones

Lista de información sobre las intenciones que Amazon Lex V2 considera posibles coincidencias con el enunciado del usuario. Cada elemento es una estructura que proporciona información sobre la coincidencia del enunciado con una intención, con el siguiente formato:

```
{
    "intent": {
        // see Intención for details about the structure
    },
    "interpretationSource": "Bedrock | Lex",
    "nluConfidence": number,
    "sentimentResponse": {
        "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
        "sentimentScore": {
            "mixed": number,
            "negative": number,
            "neutral": number,
            "positive": number
        }
    }
}
```

Los campos de la estructura son los siguientes:

- Intención: estructura que contiene información sobre la intención. Consulte Intención para obtener más información sobre la estructura.
- Confianza de NLU: puntuación que indica la confianza de Amazon Lex V2 en que la intención coincide con la intención del usuario.
- Respuesta del sentimiento: un análisis del sentimiento de la respuesta, que contiene los siguientes campos:
  - Sentimiento: indica si el sentimiento del enunciado es`POSITIVE`,`NEGATIVE`, `NEUTRAL` o. `MIXED`
  - Puntuación del sentimiento: una estructura que asigna cada sentimiento a un número que indica la confianza de Amazon Lex V2 en que el enunciado transmite ese sentimiento.
- InterpretationSource: indica si Amazon Lex V2 o Amazon Bedrock resuelven un espacio.

proposedNextState

Si la función de Lambda establece la `dialogAction` del `sessionState` en `Delegate`, este campo aparece y muestra la propuesta de Amazon Lex V2 para el siguiente paso de la conversación. En caso contrario, el siguiente estado depende de la configuración que devuelva en la respuesta de su función de Lambda. Esta estructura solo está presente si las dos afirmaciones siguientes son verdaderas:

1. El valor `invocationSource` es `DialogCodeHook`.
2. La predicción `type` de `dialogAction` es `ElicitSlot`.

Puede usar esta información para agregar `runtimeHints` en el momento correcto de la conversación. Consulte Mejora del reconocimiento de los valores de las ranuras con sugerencias de tiempo de ejecución en la conversación para obtener más información. `proposedNextState` es una estructura que contiene los siguientes campos:

La estructura de `proposedNextState` es la siguiente:

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Intención for details about the structure
    },
    "prompt": {
        "attempt": string
    }
}
```

- Acción de diálogo: contiene información sobre el siguiente paso que propone Amazon Lex V2. Los campos de la estructura son los siguientes:
  - slotToElicit— El espacio a obtener a continuación, según lo propuesto por Amazon Lex V2. Este campo solo aparece si el valor de `type` es `ElicitSlot`.
  - Tipo: el siguiente paso de la conversación propuesto por Amazon Lex V2. Se admiten los siguientes valores:

    `Delegate`: Amazon Lex V2 determina la siguiente acción.

`ElicitIntent`: la siguiente acción es obtener una intención por parte del usuario.

`ElicitSlot`: la siguiente acción es obtener un slot por parte del usuario.

`Close`: finaliza el proceso de cumplimiento de la intención e indica que no habrá respuesta por parte del usuario.

`ConfirmIntent`: la siguiente acción consiste en preguntar al usuario si los slots son correctos y si la intención está lista para ser cumplida.

- intención: la intención que el bot ha determinado que el usuario intenta cumplir. Consulte Intención para obtener más información sobre la estructura.
- Solicitud: una estructura que contiene el campo `attempt`, que se asigna a un valor que especifica cuántas veces Amazon Lex V2 ha pedido al usuario que pase al siguiente slot. Los valores posibles son `Initial` para el primer intento y `Retry1`, `Retry2`, `Retry3`, `Retry4` y `Retry5` para los intentos posteriores.

Atributos de solicitud

Una estructura que contiene 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, consulte Cómo establecer atributos de solicitud para su bot Lex V2.

Estado de la sesión

El estado actual de la conversación entre el usuario y el bot de Amazon Lex V2. Consulte Estado de la sesión para obtener más información sobre la estructura.

Transcripciones

Lista de transcripciones que Amazon Lex V2 considera posibles coincidencias con el enunciado del usuario. Para obtener más información, consulte Uso de las puntuaciones de confianza en la transcripción de voz para mejorar las conversaciones con su bot Lex V2. Cada elemento es un objeto con el siguiente formato, que contiene información sobre una posible transcripción:

```
{
    "transcription": string,
    "transcriptionConfidence": number,
```

```
    "resolvedContext": {
        "intent": string
    },
    "resolvedSlots": {
        slot name: {
            // see Ranuras for details about the structure
        },
        ...
    }
}
```

Los siguientes campos se describen a continuación:

- Transcripción: una transcripción que Amazon Lex V2 considera una posible coincidencia con el enunciado de audio del usuario.
- Confianza de la transcripción: puntuación que indica la confianza de Amazon Lex V2 en que la intención coincide con la intención del usuario.
- Contexto resuelto: estructura que contiene el campo `intent`, que corresponde a la intención a la que pertenece el enunciado.
- Slots resueltos: estructura cuyas claves son los nombres de cada slot que se resuelve mediante el enunciado. El nombre de cada slot se asigna a una estructura que contiene información sobre ese slot. Consulte Ranuras para obtener más información sobre la estructura.

AWS Lambdaformato de respuesta para Lex V2

El segundo paso para integrar una función de Lambda en el bot de Amazon Lex V2 consiste en comprender los campos de la respuesta de la función de Lambda y determinar qué parámetros desea manipular. El siguiente objeto JSON muestra el formato general de una respuesta de Lambda que se devuelve a Amazon Lex V2:

```
{
    "sessionState": {
        // see Estado de la sesión for details about the structure
    },
    "messages": [
        {
            "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
            "content": string,
            "imageResponseCard": {
```

```
                "title": string,
                "subtitle": string,
                "imageUrl": string,
                "buttons": [
                    {
                        "text": string,
                        "value": string
                    },
                    ...
                ]
            }
        },
        ...
    ],
    "requestAttributes": {
        string: string,
        ...
    }
}
```

Cada campo en la respuesta se describe a continuación.

Estado de la sesión

El estado de la conversación entre el usuario y el bot de Amazon Lex V2 que desea devolver. Consulte Estado de la sesión para obtener más información sobre la estructura. Este campo siempre es obligatorio.

Mensajes

Lista de mensajes que Amazon Lex V2 devuelve al cliente para la siguiente sesión de la conversación. Si el `contentType` que ha proporcionado es `PlainText`, `CustomPayload` o `SSML`, escriba en el campo `content` el mensaje que desea devolver al cliente. Si el `contentType` que ha proporcionado es `ImageResponseCard`, proporcione los detalles de la tarjeta en el campo `imageResponseCard`. Si no proporciona mensajes, Amazon Lex V2 utiliza el mensaje apropiado definido cuando se creó el bot.

El campo `messages` es obligatorio si `dialogAction.type` es `ElicitIntent` o `ConfirmIntent`.

Cada elemento de la lista es una estructura con el siguiente formato, que contiene información sobre un mensaje para devolverlo al usuario. A continuación se muestra un ejemplo:

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
                "value": string
            },
            ...
        ]
    }
}
```

A continuación, se proporciona una descripción de cada campo:

- Tipo de contenido: el tipo de mensaje que se va a utilizar.

  `CustomPayload`: una cadena de respuesta que se puede personalizar para incluir datos o metadatos para la aplicación.

  `ImageResponseCard`: una imagen con botones que el cliente puede seleccionar. Para obtener más información, consulte ImageResponseCard.

  `PlainText`: una cadena de texto simple.

  `SSML`: una cadena que incluye el lenguaje de marcado de síntesis de voz para personalizar la respuesta de audio.
- Contenido: el mensaje que se va a enviar al usuario. Utilice este campo si el tipo de mensaje es `PlainText`, `CustomPayload`, o `SSML`.
- imageResponseCard— Contiene la definición de la tarjeta de respuesta que se mostrará al usuario. Utilice este campo si el tipo de mensaje es `ImageResponseCard`. Corresponde a una estructura que contiene los siguientes campos:
  - título: el título de la tarjeta de respuesta.
  - subtítulo: la solicitud para que el usuario seleccione un botón.
  - URL de imagen: enlace a una imagen de la tarjeta.

- botones: lista de estructuras que contienen información sobre un botón. Cada estructura contiene un campo `text` con el texto que se debe mostrar y un campo `value` con el valor que se debe enviar a Amazon Lex V2 si el cliente selecciona ese botón. Puede incluir hasta tres botones.

Atributos de solicitud

Estructura que contiene atributos específicos de la solicitud para la respuesta al cliente. Para obtener más información, consulte Cómo establecer atributos de solicitud para su bot Lex V2. Este campo es opcional.

Campos obligatorios en la respuesta

Como mínimo, la respuesta de Lambda debe incluir un objeto `sessionState`. Dentro de eso, proporcione un objeto `dialogAction` y especifique el campo `type`. En función del `type` de la `dialogAction` que proporcione, es posible que haya otros campos obligatorios para la respuesta de Lambda. Estos requisitos se describen a continuación, junto con algunos ejemplos prácticos mínimos:

Delegado

El delegado permite a Amazon Lex V2 determinar el siguiente paso. No hay más campos obligatorios.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntentpide al cliente que exprese su intención. Debe incluir al menos un mensaje en el campo `messages` para que se manifieste una intención.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
```

```
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlotpide al cliente que proporcione un valor de espacio. Debe incluir el nombre del slot en el campo `slotToElicit` del objeto `dialogAction`. También debe incluir el `name` de `intent` en el objeto `sessionState`.

```
{`
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
    }
}
```

ConfirmIntent

ConfirmIntentconfirma los valores de las franjas horarias del cliente y si la intención está lista para cumplirse. Debe incluir el `name` del `intent` en el objeto `sessionState` y los `slots` que deben confirmarse. También debe incluir al menos un mensaje en el campo `messages` para solicitar al usuario que confirme los valores de los slots. Su mensaje debería arrojar una respuesta de «sí» o «no». Si el usuario responde «sí», Amazon Lex V2 establece el `confirmationState` de la intención en `Confirmed`. Si el usuario responde «no», Amazon Lex V2 establece el `confirmationState` de la intención en `Denied`.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
```

```
        "intent": {
            "name": "BookFlight",
            "slots": {
                "DepartureDate": {
                    "value": {
                        "originalValue": "tomorrow",
                        "interpretedValue": "2023-05-09",
                        "resolvedValues": [
                            "2023-05-09"
                    ]
                 }
                },
                "DestinationCity": {
                    "value": {
                        "originalValue": "sf",
                        "interpretedValue": "sf",
                        "resolvedValues": [
                            "sf"
                        ]
                    }
                },
                "OriginCity": {
                    "value": {
                        "originalValue": "nyc",
                        "interpretedValue": "nyc",
                        "resolvedValues": [
                            "nyc"
                        ]
                    }
                }
            }
        },
        "messages": [
            {
                "contentType": PlainText,
                "content": "Okay, you want to fly from {OriginCity} to \
                {DestinationCity} on {DepartureDate}. Is that correct?"
            }
        ]
}
```

Cerrar

Cerrar finaliza el proceso de cumplimiento de la intención e indica que no se espera ninguna otra respuesta del usuario. Debe incluir el `name` y el `state` de la `intent` en el objeto `sessionState`. Los estados de intención compatibles son `Failed`, `Fulfilled` y `InProgress`.

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Estructuras comunes de una AWS Lambda función para Amazon Lex V2

Dentro de la respuesta Lambda, hay una serie de estructuras que se repiten. En esta sección se proporcionan detalles sobre estas estructuras comunes.

Intención

```
"intent": {
    "confirmationState": "Confirmed | Denied | None",
    "name": string,
    "slots": {
        // see Ranuras for details about the structure
    },
    "state": "Failed | Fulfilled | FulfillmentInProgress | InProgress |
 ReadyForFulfillment | Waiting",
    "kendraResponse": {
        // Only present when intent is KendraSearchIntent. For details, see
// https://docs.aws.amazon.com/kendra/latest/dg/API_Query.html#API_Query_ResponseSyntax
       }
}
```

El campo `intent` se asigna a un objeto con los siguientes campos:

Estado de confirmación

Indica si el usuario ha confirmado los slots para la intención y si la intención está lista para su cumplimiento. Se admiten los siguientes valores:

`Confirmed`: el usuario confirma que los valores de los slots son correctos.

`Denied`: el usuario indica que los valores de los slots son incorrectos.

`None`: el usuario aún no ha llegado a la fase de confirmación.

name

El nombre de la intención.

slots

Información sobre los slots necesarios para cumplir con la intención. Consulte Ranuras para obtener más información sobre la estructura.

estado

Indica el estado de cumplimiento de la intención. Se admiten los siguientes valores:

`Failed`: el bot no cumplió con la intención.

`Fulfilled`: el bot ha completado el cumplimiento de la intención.

`FulfillmentInProgress`: el bot está a punto de cumplir su intención.

`InProgress`: el bot está a punto de obtener los valores de slot necesarios para cumplir su intención.

`ReadyForFulfillment`: el bot ha obtenido todos los valores de slot disponibles para esa intención y está preparado para cumplirla.

`Waiting`: el bot espera la respuesta del usuario (limitado a la transmisión de conversaciones).

Respuesta de Kendra

Contiene información sobre los resultados de la consulta de búsqueda de Kendra. Este campo solo aparece si la intención es una `KendraSearchIntent`. Consulte la sintaxis de respuesta en la llamada a la API de consulta para Kendra para obtener más información.

Ranuras

El campo `slots` existe dentro de una `intent` estructura y está asignado a una estructura cuyas claves son los nombres de los slots correspondientes a esa intención. Si el slot no es un slot con varios valores (consulte Uso de valores múltiples en un slot para obtener más información), se asigna a una estructura con el siguiente formato. Tenga en cuenta que `shape` es `Scalar`.

```
{
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    }
}
```

Si el slot es un slot con varios valores, el objeto al que se asigna contiene otro campo denominado `values`, que se asigna a una lista de estructuras, cada una de las cuales contiene información sobre el slot que forma el slot con varios valores. El formato de cada objeto de la lista coincide con el del objeto al que se asigna un slot normal. Tenga en cuenta que `shape` es `List`, pero la `shape` de los slots de los componentes que se encuentran por debajo de `values` es `Scalar`.

```
{
    slot name: {
    "shape": "List",
    "value": {
        "originalValue": string,
        "interpretedValue": string,
        "resolvedValues": [
            string,
            ...
        ]
    },
    "values": [
        {
            "shape": "Scalar",
```

```
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        ...
    ]
}
```

Los campos del objeto del slot se describen a continuación:

Forma

La forma del slot. Este valor es `List` si hay varios valores en el slot (consulte Uso de valores múltiples en un slot para obtener más información) y es `Scalar` en caso contrario.

valor

Un objeto que contiene información sobre el valor que el usuario proporcionó para una ranura y la interpretación de Amazon Lex V2, en el siguiente formato:

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
        ...
    ]
```

```
}
```

Los campos se describen a continuación:

- OriginalValue: la parte de la respuesta del usuario a la selección de franjas horarias que Amazon Lex V2 determina que es relevante para el valor de la ranura.
- InterpretedValue: el valor que Amazon Lex V2 determina para la ranura, según lo introducido por el usuario.
- ResolvedValues: lista de valores que Amazon Lex V2 determina que son posibles resoluciones para la entrada del usuario.

valores

Una lista de objetos que contiene información sobre los slots que componen el slot con valores múltiples. El formato de cada objeto coincide con el de un slot normal, con los campos `shape` y `value` descritos anteriormente. `values` solo aparece si el slot consta de varios valores (consulte Uso de valores múltiples en un slot para obtener más información). El siguiente objeto JSON muestra dos slots de componentes:

```
"values": [
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
```

```
        }
    },
    ...
]
```

Estado de la sesión

El campo `sessionState` está asignado a un objeto que contiene información sobre el estado de la conversación con el usuario. Los campos reales que aparecen en el objeto dependen del tipo de acción de diálogo. Consulte Campos obligatorios en la respuesta los campos obligatorios en una respuesta de Lambda. El formato del objeto `sessionState` es el siguiente:

```
"sessionState": {
    "activeContexts": [
        {
            "name": string,
            "contextAttributes": {
                string: string
            },
            "timeToLive": {
                "timeToLiveInSeconds": number,
                "turnsToLive": number
            }
        },
        ...
    ],
    "sessionAttributes": {
        string: string,
        ...
    },
    "runtimeHints": {
        "slotHints": {
            intent name: {
                slot name: {
                    "runtimeHintValues": [
                        {
                            "phrase": string
                        },
                        ...
                    ]
                },
                ...
            },
```

```
            ...
        }
    },
    "dialogAction": {
        "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Intención for details about the structure
    },
    "originatingRequestId": string
}
```

Los siguientes campos se describen a continuación:

Contextos activos

Una lista de objetos que contiene información sobre un contexto que un usuario utiliza en una sesión. Utilice los contextos para facilitar y controlar el reconocimiento de intenciones. Para obtener más información sobre contextos, consulte Cómo establecer un contexto de intención para su bot Lex V2. Cada objeto tiene el siguiente formato:

```
{
    "name": string,
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

Los siguientes campos se describen a continuación:

- Nombre: el nombre del contexto.
- Atributos del contexto: objeto que contiene los nombres de los atributos del contexto y los valores a los que están asignados.
- timeToLive— Un objeto que especifica cuánto tiempo permanece activo el contexto. Este objeto puede contener uno o los dos siguientes campos:

- timeToLiveInSeconds— El número de segundos que el contexto permanece activo.
- turnsToLive— El número de turnos durante los que el contexto permanece activo.

sessionAttributes

Un mapa de key/value pares que representa información contextual específica de la sesión. Para obtener más información, consulte Cómo establecer atributos de sesión para su bot Lex V2. El objeto tiene el siguiente formato:

```
{
    string: string,
    ...
}
```

Sugerencias en tiempo de ejecución

Proporciona sugerencias sobre las frases que es probable que un cliente utilice en un slot para mejorar el reconocimiento de audio. Los valores que proporcione en las sugerencias aumentan el reconocimiento de audio de esos valores en comparación con palabras que suenan similares. El formato del objeto `runtimeHints` es el siguiente:

```
{
    "slotHints": {
        intent name: {
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

El campo `slotHints` se asigna a un objeto cuyos campos son los nombres de las intenciones del bot. Cada nombre de intención se asigna a un objeto cuyos campos son los nombres de los slots

correspondientes a esa intención. El nombre de cada slot se asigna a una estructura con un solo campo, `runtimeHintValues`, que es una lista de objetos. Cada objeto contiene un campo `phrase` que se asigna a una sugerencia.

Acción de diálogo

Determina la siguiente acción que debe realizar Amazon Lex V2. El formato del objeto es el siguiente:

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

Los siguientes campos se describen a continuación:

- slotElicitationStyle— Determina cómo Amazon Lex V2 interpreta la entrada de audio del usuario si `dialogAction` es `type` `ElicitSlot` de. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación. Se admiten los siguientes valores:

  `Default`: Amazon Lex V2 interpreta la entrada de audio de la manera predeterminada para ocupar un slot.

  `SpellByLetter`: Amazon Lex V2 escucha la ortografía del usuario sobre el valor del slot.

  `SpellByWord`: Amazon Lex V2 escucha la ortografía del valor del slot por parte del usuario mediante palabras asociadas a cada letra (por ejemplo, «m como en manzana»).
- slotToElicit— Define el espacio que se debe obtener del usuario si el `type` de `dialogAction` es. `ElicitSlot`
- Tipo: define la acción que debe ejecutar el bot. Se admiten los siguientes valores:

  `Delegate`: permite a Amazon Lex V2 determinar el siguiente paso.

  `ElicitIntent`: pide al cliente que exprese una intención.

  `ConfirmIntent`: confirma los valores del slot del cliente y si la intención está lista para cumplirse.

  `ElicitSlot`: pide al cliente que indique un valor de slot para una intención.

`Close`: finaliza el proceso de cumplimiento de la intención.

Intención

Para obtener información sobre la estructura del campo `intent`, consulte Intención.

originatingRequestId

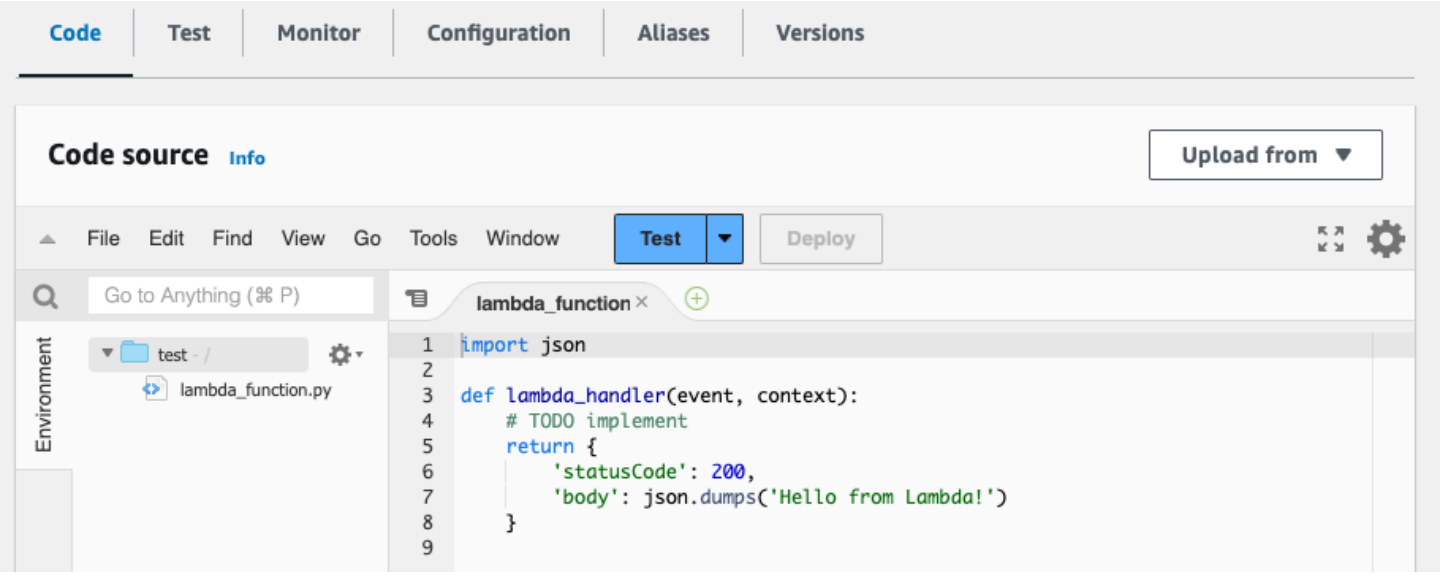
Un identificador único para la solicitud. Este campo es opcional para la respuesta de Lambda.

Creación de una AWS Lambda función para su bot Amazon Lex V2

Para crear una función Lambda para su bot de Amazon Lex V2, acceda AWS Lambda desde la consola de administración de AWS y cree una función nueva. Puede consultar la guía para AWS Lambda desarrolladores para obtener más información al respectoAWS Lambda.

1. Inicie sesión en la consola de administración de AWS y abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. Seleccione Funciones en la barra lateral izquierda.
3. Seleccione Crear función.
4. Puede seleccionar Autor desde cero para empezar con un código mínimo, Usar un esquema para seleccionar de una lista código de muestra para casos de uso comunes, o Imagen de contenedor para seleccionar una imagen de contenedor para implementarla en su función. Si selecciona Autor desde cero, continúe con los siguientes pasos:
    a. Dele a la función un Nombre de función significativo para describir lo que hace.
    b. Seleccione un idioma en el menú desplegable de Tiempo de ejecución para escribir la función.
    c. Seleccione una Arquitectura de un conjunto de instrucciones para la función.
    d. Por defecto, Lambda crea un rol con permisos básicos. Para usar un rol existente o crear un rol mediante plantillas de políticas de AWS, expanda el menú Cambiar el rol de ejecución predeterminado y seleccione una opción.
    e. Amplíe el menú de Configuración avanzada para configurar más opciones.
5. Seleccione Crear función.

La siguiente imagen muestra lo que se ve al crear una nueva función desde cero:



La función del controlador de Lambda varía según el idioma que utilice. Como mínimo, toma un objeto `event` JSON como argumento. Puede ver los campos del `event` que Amazon Lex V2 proporciona en AWS Lambdaformato de evento de entrada para Lex V2. Modifique la función del controlador para, en última instancia, devolver un objeto `response` JSON que coincida con el formato descrito en AWS Lambdaformato de respuesta para Lex V2.

- Cuando termine de escribir la función, seleccione Implementar para permitir que se use la función.

Recuerde que puede asociar cada alias de bot como máximo a una función de Lambda. Sin embargo, puede definir tantas funciones como necesite para su bot en el código Lambda y llamar a estas funciones en la función de controlador de Lambda. Por ejemplo, si bien todas las intenciones del mismo alias de bot deben llamar a la misma función de Lambda, puede crear una función de router que active una función independiente para cada intención. A continuación, se muestra un ejemplo de función de router que puede utilizar o modificar para 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
```

Cuándo utilizar las AWS Lambda funciones en las conversaciones de los bots de Amazon Lex V2

Puede utilizar funciones de Lambda en los siguientes momentos de una conversación con un usuario:

- En la respuesta inicial después de reconocer la intención. Por ejemplo, después de que el usuario diga que quiere pedir una pizza.
- Después de obtener un valor de slot del usuario. Por ejemplo, después de que el usuario le diga al bot el tamaño de pizza que quiere pedir.
- Entre cada reintento para obtener un slot. Por ejemplo, si el cliente no usa un tamaño de pizza reconocido.
- Al confirmar una intención. Por ejemplo, al confirmar un pedido de pizza.
- Para cumplir una intención. Por ejemplo, para hacer un pedido de pizza.
- Una vez cumplida la intención y antes de que su bot cierre la conversación. Por ejemplo, para cambiar a una intención de pedir una bebida.

Temas

- Adjunte una AWS Lambda función a un bot de Amazon Lex V2 mediante la consola
- Adjunte una AWS Lambda función a un bot de Amazon Lex V2 mediante operaciones de API

Adjunte una AWS Lambda función a un bot de Amazon Lex V2 mediante la consola

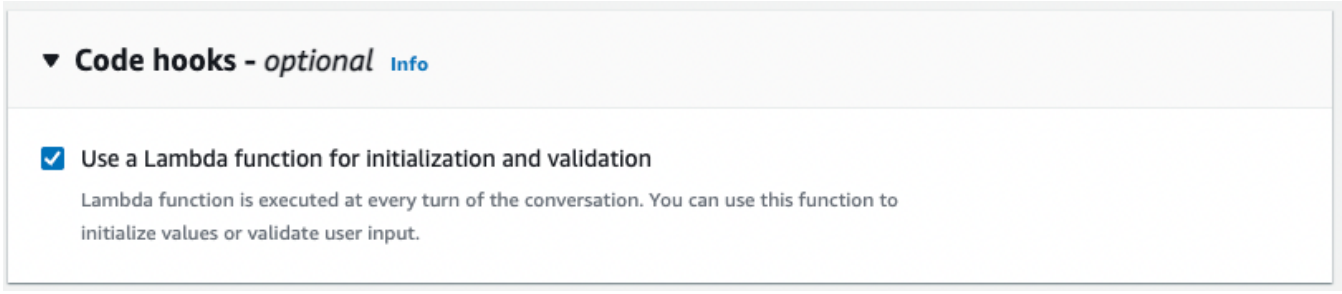
Primero debe adjuntar una función Lambda a su alias de bot de Amazon Lex V2 antes de poder invocarla. Solo puede asociar una función de Lambda a cada alias de bot. Realice estos pasos para asociar la función de Lambda con la consola de AWS.

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione Bots en el panel lateral izquierdo y, en la lista de bots, seleccione el nombre del bot al que desee adjuntar una función de Lambda.
3. En el panel lateral izquierdo, seleccione Alias en el menú de implementación.
4. En la lista de alias, seleccione el nombre del alias al que quiera adjuntar una función de Lambda.
5. En el panel Idiomas, seleccione el idioma en el que desee que aparezca una función de Lambda. Seleccione Administrar idiomas en alias para añadir un idioma si no está presente en el panel.
6. En el menú desplegable Origen, seleccione el nombre de la función de Lambda que quiera adjuntar.
7. En el menú desplegable Versión o alias de la función de Lambda, seleccione la versión o el alias de la función de Lambda que quiera utilizar. A continuación, seleccione Guardar. La misma función de Lambda se utiliza a todos los efectos en un idioma compatible con el bot.

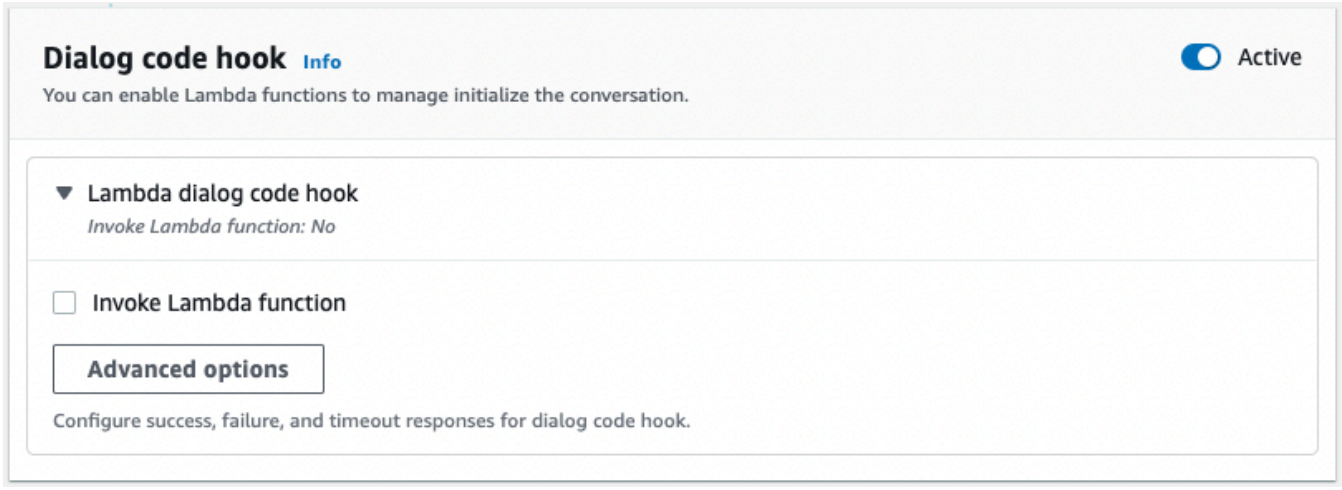
Establecer la intención de Amazon Lex V2 de invocar una función Lambda mediante la consola

1. Tras seleccionar un bot, seleccione Intenciones en el menú lateral de la izquierda, debajo del idioma del bot para el que desee invocar la función de Lambda.
2. Seleccione la intención con la que desea invocar la función de Lambda para abrir el editor de intenciones.
3. Existen dos opciones para configurar el enlace de código de Lambda:

    1. Para invocar la función de Lambda después de cada paso de la conversación, desplácese hasta la sección de Enlaces de código en la parte inferior del editor de intenciones y seleccione la casilla Usar una función de Lambda para la inicialización y la validación, como se muestra en la siguiente imagen:



2. Como alternativa, utilice la sección de Enlaces de código de diálogo en las etapas de la conversación en las que se invoca la función de Lambda. La sección del Enlace de código de diálogo aparece de la siguiente manera:



Existen dos formas de controlar la forma en que Amazon Lex V2 llama al enlace de código para obtener una respuesta:

- Active el botón Activar para marcarlo como activo o inactivo. Cuando un enlace de código esté activo, Amazon Lex V2 lo llamará. Cuando el enlace de código esté inactivo, Amazon Lex V2 no lo ejecutará.
- Amplíe la sección del enlace de código de diálogo Lambda y seleccione la casilla de verificación Invocar función de Lambda para marcarla como habilitada o deshabilitada. Solo puede activar o desactivar un enlace de código si está marcado como activo. Cuando está marcado como activado, el enlace de código se ejecuta con normalidad. Cuando está deshabilitado, no se llama al enlace de código y Amazon Lex V2 actúa como si el enlace de código se hubiera devuelto correctamente. Para configurar las respuestas después de que el enlace de código de diálogo se complete correctamente, falle o se agote el tiempo de espera, seleccione Opciones avanzadas

El enlace de código Lambda se puede invocar en las siguientes etapas de la conversación:

- Para invocar la función como respuesta inicial, desplácese hasta la sección Respuesta inicial, expanda la flecha situada junto a Respuesta para confirmar la solicitud del usuario y seleccione Opciones avanzadas. Busque la sección Enlace de código de diálogo en la parte inferior del menú que aparece.
- Para invocar la función después de obtener el slot, desplácese hasta la sección Slots, expanda la flecha situada junto a la Solicitud para slot correspondiente y seleccione Opciones avanzadas. Busque la sección Enlace de código de diálogo cerca de la parte inferior del menú que aparece, justo encima de los Valores predeterminados.

  También puedes invocar la función después de cada activación. Para ello, expanda la Información sobre la obtención del bot en la sección de Solicitudes de slots, seleccione Más opciones de solicitudes y active la casilla de verificación situada junto a Invocar el enlace de código Lambda después de cada obtención.
- Para invocar la función para confirmar la intención, desplácese hasta la sección Confirmación, expanda la flecha situada junto a Solicitudes para confirmar la intención y seleccione Opciones avanzadas. Busque la sección Enlace de código de diálogo en la parte inferior del menú que aparece.
- Para invocar la función para cumplir una intención, desplácese hasta la sección Cumplimiento. Active el botón Activo para establecer el enlace de código en activo. Expanda la flecha situada junto a Cumplimiento en proceso correctamente y seleccione Opciones avanzadas. Seleccione la casilla de verificación situada junto a Utilizar una función de Lambda para el cumplimiento en la sección Enlace de código Lambda de cumplimiento para establecer el enlace de código como activado.

4. Una vez que haya establecido las etapas de conversación en las que se invocará la función de Lambda, vuelva a Compilar el bot para probar la función.

Adjunte una AWS Lambda función a un bot de Amazon Lex V2 mediante operaciones de API

Primero debe adjuntar una función Lambda a su alias de bot de Amazon Lex V2 antes de poder invocarla. Recuerde que puede asociar cada alias de bot como máximo a una función de Lambda. Realice estos pasos para asociar la función de Lambda mediante operaciones de API.

Si va a crear un nuevo alias de bot, utilice la CreateBotAliasoperación para adjuntar una función Lambda. Para adjuntar una función Lambda a un alias de bot existente, utilice la

UpdateBotAliasoperación. Modifique el campo `botAliasLocaleSettings` para que contenga la configuración correcta:

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. El campo `botAliasLocaleSettings` se asigna a un objeto cuyas claves son las configuraciones regionales en las que desea adjuntar la función de Lambda. Consulte Idiomas y configuraciones regionales compatibles para obtener una lista de las configuraciones regionales compatibles y los códigos que son claves válidas.
2. `lambdaARN`Para buscar la función Lambda, abre la AWS Lambda consola en https://console.aws.amazon.com/lambda/casa, selecciona Funciones en la barra lateral izquierda y selecciona la función que deseas asociar al alias del bot. En el lado derecho de la Descripción de la función, busque el `lambdaARN` de la Función de ARN. Debe contener una región, un ID de cuenta y el nombre de la función.
3. Para permitir que Amazon Lex V2 invoque la función de Lambda para el alias, defina el campo `enabled` en `true`.

Establecer la intención de Amazon Lex V2 de invocar una función Lambda mediante operaciones de API

Para configurar la invocación de la función Lambda durante una intención, utilice la CreateIntentoperación si está creando una nueva intención o la UpdateIntentoperación si está invocando la función en una intención existente. Los campos que controlan la invocación de la función de Lambda en las operaciones de intención son `dialogCodeHook`, `initialResponseSetting`, `intentConfirmationSetting` y `fulfillmentCodeHook`.

Si invoca la función durante la generación de una ranura, utilice la CreateSlotoperación si va a crear una nueva ranura o utilice la UpdateSlotoperación para invocar la función en una ranura existente. El campo que controla la invocación de la función de Lambda en las operaciones de slot es la `slotCaptureSetting` del objeto `valueElicitationSetting`.

1. Para configurar el enlace de código del cuadro de diálogo de Lambda para que se ejecute después de cada turno de la conversación, defina el `enabled` campo del siguiente DialogCodeHookSettingsobjeto del `dialogCodeHook` campo en: `true`

   ```
   "dialogCodeHook": {
       "enabled": boolean
   }
   ```

2. Como alternativa, puede configurar el enlace de código del diálogo Lambda para que se ejecute solo en puntos específicos de las conversaciones modificando el `codeHook` and/or `elicitationCodeHook` campo dentro de las estructuras que corresponden a las etapas de la conversación en las que desea invocar la función. Para utilizar el enlace de códigos del cuadro de diálogo Lambda para el cumplimiento de la intención, utilice el `fulfillmentCodeHook` campo de la operación CreateIntento UpdateIntent. Las estructuras y los usos de estos tres tipos de enlaces de código son los siguientes:

enlace de código

El campo `codeHook` define los ajustes para que el enlace de código se ejecute en una etapa determinada de la conversación. Es un DialogCodeHookInvocationSettingobjeto con la siguiente estructura:

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- Cambie el campo `active` a `true` para que Amazon Lex V2 llame al enlace de código en ese momento de la conversación.

- Cambie el campo `enableCodeHookInvocation` a `true` para que Amazon Lex V2 permita que el enlace de código se ejecute con normalidad. Si lo marca como `false`, Amazon Lex V2 actúa como si el enlace de código se hubiera devuelto correctamente.
- La `invocationLabel` indica el paso de diálogo desde el que se invoca el enlace de código.
- Utilice el campo `postCodeHookSpecification` para especificar las acciones y los mensajes que se producen después de que el enlace de código se ejecute correctamente, falle o se agote el tiempo de espera.

elicitationCodeHook

El campo `elicitationCodeHook` define la configuración del enlace de código que se ejecutará en caso de que sea necesario volver a activar una o varios slots. Este escenario puede ocurrir si la no se logra la obtención de slots o si se deniega la confirmación de intención. El `elicitationCodeHook` campo es un ElicitationCodeHookInvocationSettingobjeto con la siguiente estructura:

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- Cambie el campo `enableCodeHookInvocation` a `true` para que Amazon Lex V2 permita que el enlace de código se ejecute con normalidad. Si lo marca como `false`, Amazon Lex V2 actúa como si el enlace de código se hubiera devuelto correctamente.
- La `invocationLabel` indica el paso de diálogo desde el que se invoca el enlace de código.

fulfillmentCodeHook

El campo `fulfillmentCodeHook` define la configuración para que el enlace de código se ejecute para cumplir la intención. Se asigna al siguiente FulfillmentCodeHookSettingsobjeto:

```
"fulfillmentCodeHook": {
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- Cambie el campo `active` a `true` para que Amazon Lex V2 llame al enlace de código en ese momento de la conversación.
- Cambie el campo `enabled` a `true` para que Amazon Lex V2 permita que el enlace de código se ejecute con normalidad. Si lo marca como `false`, Amazon Lex V2 actúa como si el enlace de código se hubiera devuelto correctamente.
- Utilice el campo `fulfillmentUpdatesSpecification` para especificar los mensajes que parecen para actualizar al usuario durante el cumplimiento de la intención y tiempo asociado a ellos.
- Utilice el campo `postFulfillmentStatusSpecification` para especificar los mensajes y las acciones que se producen después de que el enlace de código se ejecute correctamente, falle o se agote el tiempo de espera.

Puede invocar el enlace de código Lambda en los siguientes puntos de una conversación configurando los campos `active` y `enableCodeHookInvocation`/`enabled` en `true`:

Durante la respuesta inicial

Para invocar la función Lambda en la respuesta inicial después de reconocer la intención, utilice `codeHook` la estructura del campo de `initialResponse` CreateIntentla UpdateIntentoperación o. El `initialResponse` campo se asigna al siguiente objeto: InitialResponseSetting

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

Después de la obtención de slots o durante la reactivación de slots

Para invocar la función Lambda después de obtener un valor de ranura, utilice `slotCaptureSetting` el campo dentro del campo de `valueElicitation` CreateSlotla operación o. UpdateSlot El `slotCaptureSetting` campo se asigna al siguiente objeto: SlotCaptureSetting

```
"slotCaptureSetting": {
    "captureConditional": ConditionalSpecification object,
    "captureNextStep": DialogState object,
    "captureResponse": ResponseSpecification object,
    "codeHook": {
        "active": true,
        "enableCodeHookInvocation": true,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
    "failureResponse": ResponseSpecification object
}
```

- Para invocar la función de Lambda después de que la obtención de slots se haya realizado correctamente, utilice el campo `codeHook`.
- Para invocar la función de Lambda después de que se produzca un error en la obtención de slots y Amazon Lex V2 intente volver a intentar la obtención de slots, utilice el campo `elicitationCodeHook`.

Tras la confirmación o denegación de la intención

Para invocar la función Lambda al confirmar una intención, utilice `intentConfirmationSetting` el campo de CreateIntentla UpdateIntentoperación o. El `intentConfirmation` campo se asigna al siguiente objeto: IntentConfirmationSetting

```
"intentConfirmationSetting": {
    "active": boolean,
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "confirmationConditional": ConditionalSpecification object,
    "confirmationNextStep": DialogState object,
```

```
    "confirmationResponse": ResponseSpecification object,
    "declinationConditional": ConditionalSpecification object,
    "declinationNextStep": FulfillmentUpdatesSpecification object,
    "declinationResponse": PostFulfillmentStatusSpecification object,
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
    "failureResponse": ResponseSpecification object,
    "promptSpecification": PromptSpecification object
}
```

- Para invocar la función de Lambda después de que el usuario confirme la intención y sus slots, utilice el campo `codeHook`.
- Para invocar la función de Lambda después de que el usuario rechace la confirmación de la intención y Amazon Lex V2 intente volver a intentar la obtención de slots, utilice el campo `elicitationCodeHook`.

Durante el cumplimiento de la intención

Para invocar la función Lambda para cumplir una intención, utilice `fulfillmentCodeHook` el campo de la operación o. CreateIntentUpdateIntent El `fulfillmentCodeHook` campo se asigna al siguiente objeto: FulfillmentCodeHookSettings

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. Una vez que haya establecido las etapas de conversación en las que se invocará la función de Lambda, utilice la operación `BuildBotLocale` para volver a compilar el bot para probar la función.

Depuración de una función Lambda mediante registros CloudWatch

Amazon CloudWatch Logs es una herramienta de seguimiento de las llamadas y métricas de la API que puede utilizar para depurar las funciones de Lambda. Cuando pruebas tu bot en la consola o con llamadas a la API, CloudWatch registra cada paso de la conversación. Si utiliza una función de impresión en el código Lambda, también la CloudWatch muestra.

Para ver CloudWatch los registros de la función Lambda

1. Inicie sesión en la consola de administración de AWS y abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. En el menú Registros de la barra lateral izquierda, seleccione Grupos de registros.
3. Seleccione el grupo de registros de funciones de Lambda, que debe tener el formato `/aws/lambda/`*`function-name`*.
4. La lista de Flujos de registro contiene un registro para cada sesión con un bot. Seleccione el flujo de registros que desea ver.
5. En la lista de Eventos de registro, seleccione la flecha derecha situada junto a la Marca de tiempo para ampliar los detalles de ese evento. Todo lo que imprima desde el código Lambda aparecerá como un evento de registro. Utilice esta información para depurar el código.
6. Tras depurar el código, recuerde Implementar la función de Lambda y, si utiliza la consola, volver a cargar la ventana de Prueba antes de volver a probar el compartimento del bot.

Personalización de las interacciones de los bots con usuarios en Lex V2

Obtenga información sobre las siguientes funciones que puede usar para personalizar las interacciones de los bots con sus usuarios ampliando y ajustando su comportamiento predeterminado:

Temas

- Análisis del sentimiento de los enunciados de los usuarios en las conversaciones con el bot
- Uso de puntuaciones de confianza para mejorar la precisión de la conversación
- Personalización de las transcripciones de voz para usarlas con su bot Lex V2
- Uso del relleno de audio para mejorar la capacidad de respuesta de los bots

Análisis del sentimiento de los enunciados de los usuarios en las conversaciones con el bot

Puede utilizar el análisis de emociones para determinar los sentimientos expresados por un usuario. Con la información de los sentimientos, puede administrar el flujo de la conversación o realizar análisis tras la llamada. Por ejemplo, si el sentimiento del usuario es negativo, 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 el sentimiento se haya detectado correctamente.

Para habilitar el análisis de sentimientos de un bot, utilice la consola o la API de Amazon Lex. Se habilita el análisis de sentimientos en un alias del bot. En la consola de Amazon Lex:

1. Seleccione un alias.
2. En Detalles, seleccione Editar.
3. Seleccione Habilitar análisis del sentimiento para activarlo o desactivarlo.
4. Para guardar los cambios, seleccione Confirmar.

Si está utilizando la API, llame a la operación CreateBotAlias con el campo `detectSentiment` establecido en `true`.

Cuando el análisis de opiniones está activado, la respuesta de las RecognizeUtteranceoperaciones RecognizeTexty devuelve un campo llamado `sentimentResponse` en la `interpretations` estructura junto con otros metadatos. El campo `sentimentResponse` tiene a su vez dos campos: `sentiment` y `sentimentScore`, que contienen el resultado del análisis de sentimientos. 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 `RecognizeText` o `RecognizeUtterance`.

```
sentimentResponse {
    "sentimentScore": {
        "mixed": 0.030585512690246105,
        "positive": 0.94992071056365967,
        "neutral": 0.0141543131828308,
        "negative": 0.00893945890665054
    },
    "sentiment": "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 sentimientos de Amazon Comprehend, consulte Determinar el sentimiento en la Guía para desarrolladores de Amazon Comprehend.

Uso de puntuaciones de confianza para mejorar la precisión de la conversación

Amazon Lex V2 utiliza dos pasos para determinar lo que dice un usuario. El primero, el reconocimiento automático de voz (ASR), crea una transcripción del enunciado de audio del usuario. El segundo, la comprensión del lenguaje natural (NLU), determina el significado del enunciado del usuario para reconocer su intención o el valor de los slots.

De forma predeterminada, Amazon Lex V2 devuelve el resultado más probable de ASR y NLU. En algunos casos, puede resultar difícil para Amazon Lex V2 determinar el resultado más probable. En ese caso, devuelve varios resultados posibles junto con una puntuación de confianza que indica la probabilidad de que el resultado sea correcto. Una puntuación de confianza es una calificación que proporciona Amazon Lex V2 y que muestra la confianza relativa que tiene en un resultado. Las puntuaciones de confianza están comprendidas entre 0,0 y 1,0.

Puede utilizar sus conocimientos del dominio con la puntuación de confianza para ayudar a determinar la interpretación correcta del resultado de la ASR o la NLU.

La puntuación de confianza, ASR, o transcripción, es una valoración sobre el grado de confianza de Amazon Lex V2 en que una transcripción concreta es correcta La puntuación de confianza en la NLU, o intención, es una valoración de la confianza que tiene Amazon Lex V2 en cuanto a que la intención especificada en la transcripción superior es correcta. Utilice la puntuación de confianza que mejor se adapte a su aplicación.

Temas

- Uso de puntuaciones de confianza en la intención para mejorar la selección de intenciones con Lex V2
- Uso de las puntuaciones de confianza en la transcripción de voz para mejorar las conversaciones con su bot Lex V2

Uso de puntuaciones de confianza en la intención para mejorar la selección de intenciones con Lex V2

Cuando un usuario hace un enunciado, Amazon Lex V2 utiliza la comprensión del lenguaje natural (NLU) para entender la solicitud del usuario y devolverle la intención correcta. De forma predeterminada, Amazon Lex V2 devuelve la intención más probable definida por su bot.

En algunos casos, puede resultar difícil para Amazon Lex V2 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 la NLU en una lista de interpretaciones. Una puntuación de confianza es una calificación que proporciona Amazon Lex V2 y que muestra el grado de confianza en que una intención es la correcta.

Para determinar la diferencia entre dos intenciones en una interpretación, 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 V2 son valores comparativos. No debe confiar en ellos como puntuación absoluta. Los valores pueden cambiar en función de las mejoras de Amazon Lex V2.

Amazon Lex V2 devuelve la intención más probable y hasta 4 intenciones alternativas con sus puntuaciones asociadas en la estructura de `interpretations` de cada respuesta. El siguiente código JSON muestra la `interpretations` estructura de la respuesta de la RecognizeTextoperación:

```
"interpretations": [
   {
      "intent": {
         "confirmationState": "string",
         "name": "string",
         "slots": {
            "string" : {
               "value": {
                  "interpretedValue": "string",
                  "originalValue": "string",
                  "resolvedValues": [ "string" ]
               }
            }
         },
         "state": "string"
      },
      "nluConfidence": number
   }
]
```

AMAZON. FallbackIntent

Amazon Lex V2 devuelve `AMAZON.FallbackIntent` al ser la intención principal en dos situaciones:

1. Si las puntuaciones de confianza de todas las intenciones posibles son inferiores al umbral de confianza. Puede utilizar el umbral predeterminado o puede definir el suyo propio. Si tiene `AMAZON.KendraSearchIntent` configurado, Amazon Lex V2 también lo devuelve en esta situación.
2. Si la confianza en la interpretación `AMAZON.FallbackIntent` es superior a la confianza en la interpretación de todos los demás intentos.

Tenga en cuenta que Amazon Lex V2 no muestra una puntuación de confianza para `AMAZON.FallbackIntent`.

Establecer y cambiar el umbral de confianza

El umbral de confianza debe ser un número entre 0,00 y 1,00. Puede definir el umbral para cada idioma de su bot de las siguientes maneras:

Usar la consola de Amazon Lex V2

- Para establecer el umbral al añadir un idioma a su bot con Añadir idioma, puede insertar el valor que desee en el panel del Umbral de puntuación de confianza.
- Para actualizar el umbral, puede seleccionar Editar en el panel de Detalles del idioma en el idioma de su bot. A continuación, introduzca el valor que desee en el panel del Umbral de puntuación de confianza.

Usar operaciones de API

- Para establecer el umbral, defina el `nluIntentConfidenceThreshold` parámetro de la CreateBotLocaleoperación.
- Para actualizar el umbral de confianza, defina el `nluIntentConfidenceThreshold` parámetro de la UpdateBotLocaleoperación.

Administración de sesiones

Para cambiar la intención que Amazon Lex V2 utiliza en una conversación con el usuario, puede utilizar la respuesta de la función Lambda del enlace de código del cuadro de diálogo o puede utilizar la administración de sesiones APIs en su aplicación personalizada.

Uso de una función de Lambda con su bot Lex V2

Cuando utiliza una función de Lambda, Amazon Lex V2 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 V2 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 V2 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 Integración de una AWS Lambda función en su bot Amazon Lex V2.

Uso de la API de administración de sesiones con su bot Lex V2

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 V2, 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 Introducción a las sesiones de bots Amazon Lex V2.

Uso de las puntuaciones de confianza en la transcripción de voz para mejorar las conversaciones con su bot Lex V2

Cuando un usuario hace un enunciado de voz, Amazon Lex V2 utiliza el reconocimiento de voz automático (ASR) para transcribir la solicitud del usuario antes de interpretarla. De forma predeterminada, Amazon Lex V2 utiliza la transcripción más probable del audio para la interpretación.

En algunos casos, es posible que haya más de una transcripción del audio. Por ejemplo, un usuario puede hacer un enunciado con un sonido ambiguo, como «Me llamo John», que podría entenderse

como «Me llamo Juan». En este caso, puede utilizar técnicas de desambiguación o combinar sus conocimientos del dominio con la puntuación de confianza de la transcripción para determinar qué transcripción de una lista de transcripciones es la correcta.

Amazon Lex V2 incluye la transcripción superior y hasta dos transcripciones alternativas para que el usuario las introduzca en la solicitud a la función de enlace de códigos de Lambda. Cada transcripción contiene una puntuación de confianza que indica que es la transcripción correcta. Cada transcripción también incluye cualquier valor de slot inferido de la entrada del usuario.

Puede comparar las puntuaciones de confianza de dos transcripciones para determinar si hay ambigüedad entre ellas. Por ejemplo, si una transcripción tiene una puntuación de confianza de 0,95 y la otra tiene una puntuación de confianza de 0,65, es probable que la primera transcripción sea correcta y que la ambigüedad entre ambas sea baja. Si las dos transcripciones tienen puntuaciones de confianza de 0,75 y 0,72, la ambigüedad entre ellas es alta. Es posible que pueda discriminar entre ellas utilizando sus conocimientos de dominio.

Por ejemplo, si los valores de slot inferidos en dos transcripciones con una puntuación de confianza de 0,75 y 0,72 son «John» y «Juan», puede consultar a los usuarios de su base de datos para comprobar la existencia de estos nombres y eliminar una de las transcripciones. Si «John» no es un usuario de la base de datos y «Juan» sí lo es, puede usar el enlace de códigos del diálogo para cambiar el valor de slot inferido para el nombre a «Juan».

Las puntuaciones de confianza que devuelve Amazon Lex V2 son valores comparativos. No confíe en ellas como puntuación absoluta. Los valores pueden cambiar en función de las mejoras de Amazon Lex V2.

Las puntuaciones de confianza en la transcripción de audio solo están disponibles en los idiomas inglés (GB) (en_GB) e inglés (US) (en_US). Las puntuaciones de confianza solo se admiten para entradas de audio de 8 kHz. Las puntuaciones de confianza de la transcripción no se proporcionan para la entrada de audio desde la ventana de prueba de la consola de Amazon Lex V2 porque utiliza una entrada de audio de 16 kHz.

> **Note**
>
> Para poder utilizar las puntuaciones de confianza en la transcripción de audio con un bot existente, primero debe volver a compilar el bot. Las versiones existentes de un bot no admiten las puntuaciones de confianza en la transcripción. Debe crear una nueva versión del bot para usarlas.

Puede usar las puntuaciones de confianza para varios patrones de diseño de conversación:

- Si la puntuación de confianza más alta cae por debajo de un umbral debido a un entorno ruidoso o a una mala calidad de la señal, puede preguntar al usuario con la misma pregunta para capturar un audio de mejor calidad.
- Si varias transcripciones tienen puntuaciones de confianza similares para los valores de los slots, como «John» y «Juan», puede comparar los valores con una base de datos preexistente para eliminar las entradas, o puede pedirle al usuario que seleccione uno de los dos valores. Por ejemplo, «di 1 para John o di 2 para Juan».
- Si su lógica empresarial requiere un cambio de intención en función de palabras clave específicas en una transcripción alternativa con una puntuación de confianza cercana a la transcripción superior, puede cambiar la intención mediante el enlace de código de diálogo, la función de Lambda o mediante las operaciones de administración de sesiones. Para obtener más información, consulte Gestión de sesiones.

Amazon Lex V2 envía la siguiente estructura JSON con hasta tres transcripciones para que el usuario introduzca en la función de enlace de código de Lambda:

```
"transcriptions": [
    {
        "transcription": "string",
        "rawTranscription": "string",
        "transcriptionConfidence": "number",
        },
        "resolvedContext": {
            "intent": "string"
        },
        "resolvedSlots": {
            "string": {
                "shape": "List",
                "value": {
                    "originalValue": "string",
                    "resolvedValues": [
                        "string"
                    ]
                },
                "values": [
                    {
                        "shape": "Scalar",
```

```
                        "value": {
                            "originalValue": "string",
                            "resolvedValues": [
                                "string"
                            ]
                        }
                    },
                    {
                        "shape": "Scalar",
                        "value": {
                            "originalValue": "string",
                            "resolvedValues": [
                                "string"
                            ]
                        }
                    }
                ]
            }
        }
    }
]
```

La estructura JSON contiene el texto de la transcripción, la intención que se resolvió para el enunciado y los valores de los slots detectados en el enunciado. En cuanto al texto introducido por el usuario, las transcripciones contienen una sola transcripción con una puntuación de confianza de 1.0.

El contenido de las transcripciones depende del giro de la conversación y de la intención reconocida.

Para el primer turno, la obtención de intenciones, Amazon Lex V2 determina las tres transcripciones principales. En el caso de la transcripción superior, devuelve la intención y cualquier valor de slot inferido en la transcripción.

En los turnos subsiguientes, es decir, los resultados dependen de la intención inferida para cada una de las transcripciones, tal y como se indica a continuación.

- Si la intención inferida de la transcripción superior es la misma que en el turno anterior y todas las demás transcripciones tienen la misma intención, entonces
  - Todas las transcripciones contienen valores de slot inferidos.

- Si la intención inferida de la transcripción superior es diferente a la del turno anterior y todas las demás transcripciones tienen la intención anterior, entonces
  - La transcripción superior contiene los valores de slot inferidos para la nueva intención.
  - Otras transcripciones tienen la intención anterior y los valores de slot inferidos para la intención anterior.

- Si la intención inferida de la transcripción superior es diferente a la del turno anterior, una transcripción es la misma que la de la intención anterior y una transcripción es una intención diferente, entonces
  - La transcripción superior la nueva intención inferida y cualquier valor de slot inferido en el enunciado.
  - La transcripción que tiene la intención inferida anterior contiene los valores de slot inferidos para esa intención.
  - La transcripción con la intención diferente no tiene un nombre de intención inferido ni valores de slot inferidos.

- Si la intención inferida de la transcripción superior es diferente a la del turno anterior y todas las demás transcripciones tienen intenciones diferentes, entonces
  - La transcripción superior la nueva intención inferida y cualquier valor de slot inferido en el enunciado.
  - Otras transcripciones no contienen intenciones inferidas ni valores de slot inferidos.

- Si la intención inferida de las dos transcripciones superiores es la misma y diferente a la del turno anterior, y la tercera transcripción es una intención diferente, entonces
  - Las dos transcripciones superiores contienen la nueva intención inferida y cualquier valor de slot inferido en el enunciado.
  - La tercera transcripción no tiene ningún nombre de intención ni valores de slot resueltos.

Gestión de sesiones

Para cambiar la intención que Amazon Lex V2 utiliza en una conversación con el usuario, utilice la respuesta del enlace de código del cuadro de diálogo (función de Lambda). O bien, puede usar la administración de sesiones APIs en su aplicación personalizada.

Uso de una función de Lambda con su bot Lex V2

Cuando utiliza una función de Lambda, Amazon Lex V2 la llama con una estructura JSON que contiene la entrada a la función. La estructura JSON contiene un campo denominado `transcriptions` que contiene las posibles transcripciones que Amazon Lex V2 ha determinado para el enunciado. El campo `transcriptions` contiene de una a tres posibles transcripciones, cada una con una puntuación de confianza.

Para utilizar la intención de una transcripción alternativa, debe especificarla en la `ConfirmIntent` o la acción de diálogo `ElicitSlot` de la función de Lambda. Para utilizar un valor de slot de una transcripción alternativa, defina el valor en el campo `intent` de la respuesta de la función de Lambda. Para obtener más información, consulte Integración de una AWS Lambda función en su bot Amazon Lex V2.

Ejemplo de código con Lambda con Lex V2

El siguiente ejemplo de código es una función de Lambda de Python que utiliza transcripciones de audio para mejorar la experiencia de conversación del usuario.

Para utilizar el código de ejemplo, debe disponer de lo siguiente:

- Un bot con un idioma, inglés (GB) (en_GB) o inglés (US) (en_US).
- Una intención, OrderBirthStone. Asegúrese de que la opción Utilizar una función de Lambda para la inicialización y la validación esté seleccionada en la sección Enlaces de códigos de la definición de intención.
- La intención debe tener dos espacios, «BirthMonth» y «Nombre», ambos de tipo`AMAZON.AlphaNumeric`.
- Un alias con la función de Lambda definida. Para obtener más información, consulte Creación de una AWS Lambda función para su bot Amazon Lex V2.

```
import time
import os
import logging
```

```
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)


# --- Helpers that build all of the responses ---

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit, message):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'ElicitSlot',
                'slotToElicit': slot_to_elicit
            },
            'intent': {
                'name': intent_request['sessionState']['intent']['name'],
                'slots': slots,
                'state': 'InProgress'
            },
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'e3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'sessionId': intent_request['sessionId'],
        'messages': [message],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': '3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'messages': [message],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
```

```
    }


def delegate(intent_request, session_attributes):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'Delegate'
            },
            'intent': intent_request['sessionState']['intent'],
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'abc'
        },
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']

    return {}


def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']


""" --- Functions that control the behavior of the bot --- """


def order_birth_stone(intent_request):
    """
    Performs dialog management and fulfillment for ordering a birth stone.
    Beyond fulfillment, the implementation for this intent demonstrates the following:
    1) Use of N best transcriptions to re prompt user when confidence for top
 transcript is below a threshold
    2) Overrides resolved slot for birth month from a known fixed list if the top
 transcript
    is not accurate.
    """
```

```
    transcriptions = intent_request['transcriptions']

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Disambiguate if there are multiple transcriptions and the top transcription
        # confidence is below a threshold (0.8 here)
        if len(transcriptions) > 1 and transcriptions[0]['transcriptionConfidence'] <
 0.8:
            if transcriptions[0]['resolvedSlots'] is not {} and 'Name' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['Name'] is not None:
                return prompt_for_name(intent_request)
            elif transcriptions[0]['resolvedSlots'] is not {} and 'BirthMonth' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['BirthMonth'] is not None:
                return validate_month(intent_request)

    return continue_conversation(intent_request)


def prompt_for_name(intent_request):
    """
    If the confidence for the name is not high enough, re prompt the user with the
 recognized names
    so it can be confirmed.
    """
    resolved_names = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'Name' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['Name'] is not None:
            resolved_names.append(transcription['resolvedSlots']['Name']['value']
['originalValue'])
    if len(resolved_names) > 1:
        session_attributes = get_session_attributes(intent_request)
        slots = get_slots(intent_request)
        return elicit_slot(session_attributes, intent_request, slots, 'Name',
                           {'contentType': 'PlainText',
                            'content': 'Sorry, did you say your name is {} ?'.format("
 or ".join(resolved_names))})
    else:
        return continue_conversation(intent_request)
```

```
def validate_month(intent_request):
    """
    Validate month from an expected list, if not valid looks for other transcriptions
 and to see if the month
    recognized there has an expected value. If there is, replace with that and if not
 continue conversation.
    """

    expected_months = ['january', 'february', 'march']
    resolved_months = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'BirthMonth' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['BirthMonth'] is not None:
            resolved_months.append(transcription['resolvedSlots']['BirthMonth']
['value']['originalValue'])

    for resolved_month in resolved_months:
        if resolved_month in expected_months:
            intent_request['sessionState']['intent']['slots']['BirthMonth']
['resolvedValues'] = [resolved_month]
            break

    return continue_conversation(intent_request)


def continue_conversation(event):
    session_attributes = get_session_attributes(event)

    if event["invocationSource"] == "DialogCodeHook":
        return delegate(event, session_attributes)


# --- Intents ---


def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """

    logger.debug('dispatch sessionId={},
 intentName={}'.format(intent_request['sessionId'],
```

```
 intent_request['sessionState']['intent']['name']))

    intent_name = intent_request['sessionState']['intent']['name']

    # Dispatch to your bot's intent handlers
    if intent_name == 'OrderBirthStone':
        return order_birth_stone(intent_request)

    raise Exception('Intent with name ' + intent_name + ' not supported')


# --- Main handler ---


def lambda_handler(event, context):
    """
    Route the incoming request based on intent.
    The JSON body of the request is provided in the event slot.

    """
    # By default, treat the user request as coming from the America/New_York time
 zone.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()
    logger.debug('event={}'.format(event))

    return dispatch(event)
```

Uso de la API de administración de sesiones para elegir un valor de ranura o intención diferente

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 V2, puede utilizar la operación `PutSession` para cambiar las intenciones. De esta forma, la siguiente intención con la que interactúe el usuario será la que usted haya seleccionado.

También puede usar la operación `PutSession` para cambiar el valor del slot en la estructura `intent` y usar un valor de una transcripción alternativa.

Para obtener más información, consulte Introducción a las sesiones de bots Amazon Lex V2.

Personalización de las transcripciones de voz para usarlas con su bot Lex V2

El comportamiento predeterminado de su bot a veces puede provocar transcripciones de voz inexactas. Las siguientes funciones están disponibles para ayudar a su bot a reconocer palabras o nombres que son menos comunes o que se confunden fácilmente.

Temas

- Configuración de las preferencias del modelo de reconocimiento de voz
- Mejora del reconocimiento de voz con un vocabulario personalizado
- Configuración de la sensibilidad de detección de actividad de voz
- Uso de la diarización del altavoz para centrarse en el altavoz principal
- Mejora del reconocimiento de los valores de las ranuras con sugerencias de tiempo de ejecución en la conversación
- Captura de valores de ranura con estilos de ortografía durante la conversación

Configuración de las preferencias del modelo de reconocimiento de voz

Amazon Lex V2 ofrece diferentes modelos de reconocimiento de voz entre los que puede elegir para optimizar la precisión y el rendimiento de las capacidades de reconocimiento de voz de su bot. Puede configurar las preferencias del modelo de voz para seleccionar el modelo más adecuado para su caso de uso.

Tipos de modelos de reconocimiento de voz

Amazon Lex V2 es compatible con los siguientes modelos de reconocimiento de voz:

Modelo estándar

El modelo de reconocimiento de voz estándar proporciona un rendimiento de reconocimiento de voz fiable para casos de uso general. Este modelo ofrece una precisión uniforme en una amplia gama de condiciones de audio y es adecuado para la mayoría de las aplicaciones de IA conversacional.

Modelo neuronal

El modelo de reconocimiento neuronal de voz proporciona una mayor precisión y un mejor manejo de los patrones naturales del habla, los acentos y el ruido de fondo. Este modelo utiliza

arquitecturas de redes neuronales avanzadas para mejorar el rendimiento del reconocimiento, especialmente en entornos de audio exigentes.

Deepgram

Deepgram proporciona una API pública de conversión de voz a texto (STT) para los usuarios que crean una cuenta y una clave de API. Consulte https://deepgram.com/para obtener información sobre sus ofertas públicas.

Configuración de las preferencias del modelo de voz

Puede configurar las preferencias del modelo de voz al crear o actualizar la configuración regional de un bot. La configuración de preferencias del modelo de voz determina qué modelo de reconocimiento utiliza Amazon Lex V2 para procesar la entrada de audio de su bot.

Para configurar las preferencias del modelo de voz:

1. En la consola Amazon Lex V2, navegue hasta el bot y seleccione la configuración regional que desee configurar.
2. En la configuración regional del bot, localice la sección de configuración del reconocimiento de voz.
3. En la preferencia del modelo de voz, selecciona una de las siguientes opciones:
   - Estándar: utilice el modelo de reconocimiento de voz estándar para obtener un rendimiento fiable en todos los casos de uso generales.
   - Neural: utilice el modelo de reconocimiento de voz neuronal para aumentar la precisión y gestionar mejor los patrones naturales del habla.
   - Deepgram: utilice la API Listen de Deepgram para el reconocimiento de voz. Para obtener instrucciones de configuración, consulte. the section called “Configuración de la preferencia del modelo de voz de Deepgram”
4. Guarde los cambios para aplicar la preferencia del modelo de voz a la configuración regional del bot.

> **Note**
>
> Si no especifica una preferencia de modelo de voz, Amazon Lex V2 utiliza el modelo estándar de forma predeterminada.

Elegir el modelo de voz correcto

Ten en cuenta los siguientes factores a la hora de elegir un modelo de reconocimiento de voz para tu bot:

- Calidad de audio: si tu bot procesa el audio con ruido de fondo, con una calidad de audio variable o en condiciones acústicas difíciles, el modelo neuronal puede ofrecer una mayor precisión.
- Diversidad de altavoces: si tu bot interactúa con usuarios que tienen distintos acentos o patrones de voz, las capacidades mejoradas de procesamiento del lenguaje natural del modelo neuronal pueden mejorar el rendimiento del reconocimiento.
- Requisitos de rendimiento: el modelo estándar proporciona un rendimiento uniforme y puede ser suficiente para aplicaciones con entornos de audio controlados y una entrada de voz clara.

Puede probar ambos modelos con su caso de uso específico para determinar cuál ofrece el mejor equilibrio entre precisión y rendimiento para su aplicación.

Configuración de la preferencia del modelo de voz de Deepgram

Deepgram es un servicio de reconocimiento de voz de terceros que proporciona speech-to-text capacidades avanzadas impulsadas por IA con soporte para el procesamiento por lotes y en tiempo real. Deepgram ofrece una precisión mejorada en diversas condiciones de audio, varios idiomas y modelos especializados para diferentes casos de uso. Para obtener más información sobre las ofertas de Deepgram, consultehttps://deepgram.com/. Para usar Deepgram como modelo de reconocimiento de voz preferido, debe completar un proceso de configuración único para configurar su clave de API de Deepgram y almacenarla de forma segura en AWS Secrets Manager.

> **Important**
>
> Deepgram es un servicio de terceros y es posible que no cumpla con determinados marcos normativos, como el RGPD, FedRAMP u otros estándares de conformidad a los que se adhieran los servicios de AWS. Revise la documentación de conformidad de Deepgram y los requisitos de su organización antes de utilizar esta integración.

Selección de puntos finales regionales

Amazon Lex V2 selecciona automáticamente el punto de enlace de la API de Deepgram adecuado en función de su región de AWS para optimizar el rendimiento y la localidad de los datos:

- Regiones de la UE: para los bots desplegados en las regiones de AWS con el `eu-` prefijo (como`eu-west-1`, `eu-central-1`)`eu-west-2`, Amazon Lex V2 utiliza el punto de enlace Deepgram EU (`api.eu.deepgram.com`).
- Todas las demás regiones: para los bots desplegados en todas las demás regiones de AWS, Amazon Lex V2 utiliza el punto de conexión global de Deepgram (`api.deepgram.com`).

La selección de este punto final es automática y actualmente no se puede personalizar. La misma clave de API de Deepgram funciona con ambos puntos finales. Amazon Lex V2 no admite el `apiTokenRegion` parámetro que Amazon Connect proporciona para la configuración de puntos de conexión personalizada. Para obtener más información sobre las opciones de configuración de puntos de conexión de Amazon Connect, consulte Puntos de conexión y regiones para proveedores de STT de terceros en la Guía del administrador de Amazon Connect.

Crear una clave de API de Deepgram

Antes de poder utilizar Deepgram con Amazon Lex V2, debe obtener una clave de API de Deepgram.

Para crear una clave de API de Deepgram:

1. Inicie sesión en la consola de Deepgram enhttps://console.deepgram.com/.
2. En el panel de navegación izquierdo, elija API Keys.
3. Selecciona Crear una nueva clave de API.
4. Sigue las instrucciones para crear la clave de API y cópiala para utilizarla en la siguiente sección.

> **Important**
>
> Guarde su clave de API de forma segura. La necesitará para configurar AWS Secrets Manager en la siguiente sección.

Almacenamiento de la clave de API en AWS Secrets Manager

Debe almacenar su clave de API de Deepgram en AWS Secrets Manager para Amazon Lex V2 para acceder a ella de forma segura. El secreto debe contener un único par clave-valor `apiToken` como clave y su clave de API de Deepgram como valor.

> **Important**
>
> Debe crear una clave KMS simétrica para usarla con el secreto. La clave de KMS administrada por AWS predeterminada no funcionará con Amazon Lex V2.

Para almacenar tu clave API de Deepgram en Secrets Manager:

1. Abra la consola de AWS Secrets Manager en https://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.
3. En Secret type (Tipo de secreto), elija Other type of secret (Otro tipo de secreto).
4. Configure el secreto mediante uno de los siguientes métodos:
   - Método de pares clave/valor: en Pares clave/valor, añada un único par clave-valor como clave y su clave de API de Deepgram `apiToken` como valor.
   - Método de texto sin formato: en Texto sin formato, introduzca un objeto JSON con la siguiente estructura:

     ```
     {
       "apiToken": "your-deepgram-api-key-here"
     }
     ```
5. Elija Siguiente.
6. Introduce un nombre para tu secreto y selecciona Siguiente.
7. (Opcional) Configura la rotación secreta si es necesario y, a continuación, selecciona Siguiente.
8. Revisa tu configuración secreta y selecciona Almacenar.
9. Una vez creado el secreto, navega hasta tu secreto y copia el ARN. Necesitarás este ARN al configurar tu bot.

Configuración de la política de recursos para Secrets Manager

Para permitir que Amazon Lex V2 recupere su clave de API de Deepgram, debe adjuntar una política de recursos a su secreto.

El siguiente es un ejemplo de política de recursos que permite a Amazon Lex V2 recuperar el secreto:

```
{
  "Version": "2012-10-17"       ,
  "Statement": [
    {
      "Sid": "LexTrust",
      "Effect": "Allow",
      "Principal": {
        "Service": "lex.amazonaws.com"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "<YOUR_ACCOUNT_ID>"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:lex:us-east-1:<YOUR_ACCOUNT_ID>:bot-alias/*/*"
        }
      }
    }
  ]
}
```

<YOUR_ACCOUNT_ID>Sustitúyalo por el ID de su cuenta de AWS real y ajuste la región en el patrón de ARN según sea necesario para su implementación.

¿Está configurando su bot para que utilice Deepgram

Tras configurar la clave API de Deepgram en Secrets Manager, puede configurar su bot Amazon Lex V2 para que utilice Deepgram para el reconocimiento de voz.

Para configurar Deepgram para tu bot:

1. En la consola Amazon Lex V2, navegue hasta el bot y seleccione la configuración regional que desee configurar.
2. Para preferir el modelo de voz, selecciona Deepgram.
3. Aparecerán campos adicionales para la configuración de Deepgram:

- ID de modelo (opcional): especifique un ID de modelo de Deepgram si desea utilizar un modelo específico. Para ver los modelos disponibles, consulte la documentación del modelo de Deepgram. Si se deja en blanco, se utilizará el modelo predeterminado de la API.
- ARN secreto (obligatorio): introduzca el ARN del secreto que creó en AWS Secrets Manager que contiene su clave de API de Deepgram.

4. Guarde los cambios para aplicar la preferencia del modelo de voz de Deepgram a la configuración regional de su bot.

Su bot ahora está configurado para usar Deepgram para el reconocimiento de voz. Pruebe su bot para asegurarse de que el reconocimiento de voz funciona como se espera con la integración de Deepgram.

Solución de problemas de integración con Deepgram

Si tiene problemas con su integración con Deepgram, compruebe lo siguiente:

- Validez de la clave de API: asegúrese de que su clave de API de Deepgram sea válida y no haya caducado.
- Configuración secreta: compruebe que su secreto en AWS Secrets Manager contenga el nombre de clave (`apiToken`) y el valor de clave de API correctos.
- Política de recursos: confirme que la política de recursos de su secreto permite a Amazon Lex V2 acceder a él con el ID de cuenta y el patrón de ARN correctos.
- Clave de KMS: asegúrese de utilizar una clave de KMS simétrica gestionada por el cliente, no la clave gestionada por AWS predeterminada.
- ID de modelo: si especificó un ID de modelo, compruebe que es un identificador de modelo de Deepgram válido.

Para obtener asistencia adicional, consulte los CloudWatch registros de Amazon Lex V2 o póngase en contacto con AWS Support.

Mejora del reconocimiento de voz con un vocabulario personalizado

Puede proporcionar a Amazon Lex V2 más información sobre cómo procesar las conversaciones de audio con un bot creando un vocabulario personalizado en un idioma específico. Un vocabulario personalizado es una lista de palabras específicas que desea que Amazon Lex V2 reconozca en la

entrada de audio. Por lo general, se trata de nombres propios o palabras de dominios específicos que Amazon Lex V2 no reconoce.

Por ejemplo, supongamos que dispone de un bot de soporte técnico. Puede añadir «contexto» a un vocabulario personalizado para ayudar al bot a transcribir correctamente el audio como «contexto», incluso cuando el audio suene como «contesto». Un vocabulario personalizado también puede ayudar a reconocer palabras raras en el audio, como «solvencia» para servicios financieros o nombres propios, como «Cognito» o «Monitron».

Conceptos básicos de vocabulario personalizado

- Un vocabulario personalizado funciona con la transcripción de la entrada de audio a un bot. Debe proporcionar ejemplos de enunciados para reconocer una intención o un valor de slot.
- Un vocabulario personalizado es exclusivo de un idioma específico. Debe configurar los vocabularios personalizados de forma independiente para cada idioma. Se admiten vocabularios personalizados en los siguientes idiomas: inglés (Reino Unido), inglés (EE. UU.), coreano (Corea), italiano (Italia), alemán (Alemania), francés (Canadá), francés (Francia), español (España), español (EE. UU.), japonés (Japón), mandarín (PRC), sueco (Suecia), árabe del Golfo (Emiratos Árabes Unidos)), hindi (India), neerlandés (Países Bajos), noruego (Noruega), polaco (Polonia), finés (Finlandia), portugués (Portugal), portugués (Brasil) y catalán (España).
- Los vocabularios personalizados están disponibles con integraciones de centros de contacto compatibles con Amazon Lex V2. La ventana de pruebas de la consola de Amazon Lex V2 admite vocabularios personalizados para todos los bots de Amazon Lex V2 creados a partir del 31 de julio de 2022. Si tiene problemas con los vocabularios personalizados en la ventana de prueba, vuelva a compilar el bot e inténtelo de nuevo.

Amazon Lex V2 utiliza vocabularios personalizados para obtener intenciones y slots. Se utiliza el mismo archivo de vocabulario personalizado para las intenciones y los slots. Puede desactivar de forma selectiva la función de vocabulario personalizado para un slot al añadir un tipo de slot.

Obtener una intención: puede crear un vocabulario personalizado para obtener una intención. Estas frases se utilizan para transcribir cuando su bot determina la intención del usuario. Por ejemplo, si configuró la frase «contexto» en su vocabulario personalizado, Amazon Lex V2 transcribe la entrada del usuario a «¿puedes darme más contexto?» —incluso cuando el audio suene como «¿puedes darme más contesto?». Puede especificar el grado de refuerzo de cada frase configurando un peso de 0, 1, 2 o 3. También puede especificar una representación alternativa para la frase en la salida final del discurso a texto añadiendo un campo `displayAs`.

Las frases de vocabulario personalizadas que se utilizan para mejorar la transcripción durante la obtención de la intención no afectan a las transcripciones al obtener slots. Para obtener más información acerca de la creación de un vocabulario personalizado para obtener intenciones, consulte Crear un vocabulario personalizado para descubrir intenciones y slots.

Obtener slots personalizados: puede usar un vocabulario personalizado para mejorar el reconocimiento de los slots en las conversaciones de audio. Para mejorar la capacidad de su bot de Amazon Lex V2 para reconocer los valores de los slots, cree un slot personalizado y añada los valores de los slots al slot personalizado y, a continuación, seleccione Utilizar valores de los slots como vocabulario personalizado. Algunos ejemplos de valores de slots son los nombres de productos, los catálogos o los nombres propios. No debe usar palabras o frases comunes como «sí» y «no» en los vocabularios personalizados.

Una vez añadidos los valores de los slots, estos valores se utilizan para mejorar el reconocimiento de los slots cuando el bot espera introducir datos para el slot personalizado. Estos valores no se utilizan para la transcripción cuando se suscita una intención. Para obtener más información, consulte Adición de tipos de ranura.

Prácticas recomendadas para crear un vocabulario personalizado

Obtener una intención

- Los vocabularios personalizados funcionan mejor cuando se utilizan para palabras o frases específicas. Añada palabras a un vocabulario personalizado únicamente si Amazon Lex V2 no las reconoce fácilmente.
- Decida cuánto peso darle a una palabra en función de la frecuencia con la que no se reconoce la palabra en la transcripción y de su rareza en la entrada. Las palabras difíciles de pronunciar requieren un peso mayor.
- Utilice un conjunto de pruebas representativo para determinar si un peso es apropiado. Puede recopilar un conjunto de pruebas de audio activando el registro de audio en los registros de conversaciones.
- Evite usar palabras cortas como «así», «eso», «para», «sí», «no» en un vocabulario personalizado.

Crear un slot personalizado

- Añada los valores al tipo de slot personalizado que espera que se reconozca. Añada todos los valores de slot posibles para el tipo de slot personalizado, independientemente de lo común o raro que sea el valor del slot.
- Active la opción solo cuando el tipo de slot personalizado contenga una lista de valores de catálogo o entidades, como nombres de productos o fondos de inversión colectiva.
- Desactive esta opción si el tipo de slot se utiliza para capturar frases genéricas como «sí», «no», «no sé», «quizás» o palabras genéricas como «uno», «dos», «tres».
- Limite el número de valores y sinónimos de los slots a 500 o menos para obtener el mejor rendimiento.

Escriba separado mediante puntos o espacios los acrónimos u otras palabras cuyas letras deban pronunciarse por separado. No utilice letras individuales a menos que formen parte de una frase, como «J. P. Morgan» o «A. W. S.» Puede usar letras mayúsculas o minúsculas para definir un acrónimo.

Crear un vocabulario personalizado para descubrir intenciones y slots

Puede utilizar la consola de Amazon Lex V2 para crear y gestionar un vocabulario personalizado, o puede utilizar las operaciones de la API de Amazon Lex V2. Hay dos formas de crear un vocabulario personalizado a través de la consola:

Consola

Importar vocabulario personalizado en la consola:

1. Abre la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa
2. De la lista de bots, seleccione el bot al que desea añadir el vocabulario personalizado.
3. En la página de detalles del bot, en la sección Añadir idiomas, seleccione Ver idiomas.
4. De la lista de idiomas, seleccione el idioma al que desea añadir el vocabulario personalizado.

Cree un nuevo vocabulario personalizado directamente desde la consola:

1. Haga clic en Crear en la sección Vocabulario personalizado de la página de detalles del idioma. Se abrirá una ventana de edición sin ningún vocabulario personalizado.
2. Añada entradas para la frase DisplayAs y el peso según sea necesario. También puede realizar modificaciones integradas en los elementos agregados actualizando sus campos o eliminándolos de la lista.

3. Haga clic en Guardar. Tenga en cuenta que el nuevo vocabulario personalizado solo se guarda en su bot después de hacer clic en Guardar.
4. Puede seguir haciendo modificaciones en línea en esta página y hacer clic en Guardar cuando termine.
5. Esta página también le permite importar, exportar y eliminar un archivo de vocabulario personalizado del menú desplegable de la parte superior derecha.

API

Use la API de **`ListCustomVocabularyItems`** para ver las entradas de vocabulario personalizadas:

1. Use la operación de `ListCustomVocabularyItems` para ver las entradas de vocabulario personalizadas. El resultado tendrá este aspecto:

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. Tenga en cuenta que los campos `maxResults` y `nextToken` son opcionales para el cuerpo de la solicitud.
3. La respuesta de la operación `ListCustomVocabularyItems` tiene el siguiente aspecto:

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

Use la API de **`BatchCreateCustomVocabularyItem`** para ver las entradas de vocabulario personalizadas:

1. Si la configuración regional de tu bot aún no ha creado un vocabulario personalizado, sigue los pasos para utilizarla y StartImportcrear un vocabulario personalizado.
2. Una vez creado el vocabulario personalizado, use la operación `BatchCreateCustomVocabularyItem` para crear nuevas entradas de vocabulario personalizadas. El resultado tendrá este aspecto:

```
{
   "customVocabularyItemList": [
        {
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Tenga en cuenta que los campos `weight` y `displayAs` son opcionales para el cuerpo de la solicitud.
4. La salida del `BatchCreateCustomVocabularyItem` tendrá este aspecto:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
```

```
    ]
}
```

5. Como se trata de una operación por lotes, la solicitud no fallará si uno de los elementos no se crea. La lista de errores contendrá información sobre el motivo por el que se produjo un error en la operación en esa entrada específica. La lista de recursos contendrá todas las entradas que se crearon correctamente.
6. Para `BatchCreateCustomVocabularyItem`, puede esperar ver estos tipos de errores:
   - `RESOURCE_DOES_NOT_EXIST`: el vocabulario personalizado no existe. Siga los pasos para crear un vocabulario personalizado antes de realizar esta operación.
   - `DUPLICATE_INPUT`: la lista de entradas contiene frases duplicadas.
   - `RESOURCE_ALREADY_EXISTS`: la frase indicada para la entrada ya existe en su vocabulario personalizado.
   - `INTERNAL_SERVER_FAILURE`: se ha producido un error en el servidor al procesar su solicitud. Esto puede indicar una interrupción del servicio u otro problema.

Use la API de **`BatchDeleteCustomVocabularyItem`** para eliminar las entradas de vocabulario personalizadas existentes:

1. Si la configuración regional de tu bot aún no ha creado un vocabulario personalizado, sigue los pasos de Usa el StartImportpara crear un vocabulario personalizado para crear uno.
2. Una vez creado el vocabulario personalizado, use la operación `BatchDeleteCustomVocabularyItem` para eliminar nuevas entradas de vocabulario personalizadas. El resultado tendrá este aspecto:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. La salida del `BatchDeleteCustomVocabularyItem` tendrá este aspecto:

```
{
    "botId": "string",
```

```
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

4. Como se trata de una operación por lotes, la solicitud no fallará si uno de los elementos no se elimina. La lista de errores contendrá información sobre el motivo por el que se produjo un error en la operación en esa entrada específica. La lista de recursos contendrá todas las entradas que se eliminaron correctamente.
5. Para `BatchDeleteCustomVocabularyItem`, puede esperar ver estos tipos de errores:
   - `RESOURCE_DOES_NOT_EXIST`: la entrada de vocabulario personalizada que está intentando eliminar no existe.
   - `INTERNAL_SERVER_FAILURE`: se ha producido un error en el servidor al procesar su solicitud. Esto puede indicar una interrupción del servicio u otro problema.

Use la API de **`BatchUpdateCustomVocabularyItem`** para actualizar las entradas de vocabulario personalizadas existentes:

1. Si la configuración regional de tu bot aún no ha creado un vocabulario personalizado, sigue los pasos de Usa el StartImportpara crear un vocabulario personalizado para crear un vocabulario personalizado.
2. Una vez creado el vocabulario personalizado, use la operación `BatchUpdateCustomVocabularyItem` para actualizar nuevas entradas de vocabulario personalizadas. El resultado tendrá este aspecto:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Tenga en cuenta que los campos `weight` y `displayAs` son opcionales para el cuerpo de la solicitud.
4. La salida del `BatchUpdateCustomVocabularyItem` tendrá este aspecto:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Como se trata de una operación por lotes, la solicitud no fallará si uno de los elementos no se elimina. La lista de errores contendrá información sobre el motivo por el que se produjo un error en la operación en esa entrada específica. La lista de recursos contendrá todas las entradas que se actualizaron correctamente.
6. Para `BatchUpdateCustomVocabularyItem`, puede esperar ver estos tipos de errores:

- `RESOURCE_DOES_NOT_EXIST`: la entrada de vocabulario personalizada que está intentando actualizar no existe.
- `DUPLICATE_INPUT`: la lista de entradas contiene Id de elementos duplicadas.
- `RESOURCE_ALREADY_EXISTS`: la frase indicada para la entrada ya existe en su vocabulario personalizado.
- `INTERNAL_SERVER_FAILURE`: se ha producido un error en el servidor al procesar su solicitud. Esto puede indicar una interrupción del servicio u otro problema.

Crear un archivo de vocabulario personalizado

Un archivo de vocabulario personalizado es una lista de valores separados por tabulaciones que contiene la frase que se debe reconocer, un peso que la potencia y un campo `displayAs` que sustituirá a la frase en la transcripción del discurso. Es más probable que se usen frases con un valor de refuerzo más alto cuando aparecen en la entrada de audio.

El archivo de vocabulario personalizado debe tener un nombre **`CustomVocabulary.tsv`** y comprimirse en un archivo zip antes de poder importarlo. El archivo zip debe tener menos de 300 MB. El número máximo de frases en un vocabulario personalizado es 500.

- frase: de 1 a 4 palabras que deben reconocerse. Separe las palabras de la frase con espacios. No puede tener frases duplicadas en el archivo. El campo «frase» es obligatorio.
- peso: el grado en que debe potenciarse el reconocimiento de la frase. El valor es un número entero 0, 1, 2 o 3. Si no se especifica un peso, el valor predeterminado es 1. Decida el peso en función de la frecuencia con la que no se reconoce la palabra en la transcripción y de su rareza en la entrada. El peso 0 significa que no se potenciará nada y que la entrada solo se utilizará para realizar sustituciones utilizando el campo `displayAs`.
- DisplayAS: define el aspecto que desea que tenga la frase en el resultado de la transcripción. Se trata de un campo opcional en el vocabulario personalizado.

El archivo de vocabulario personalizado debe contener una fila de encabezados con los encabezados «frase», «peso» y «DisplayAs». Los encabezados pueden estar en cualquier orden, pero deben seguir la nomenclatura anterior.

A continuación se muestra un ejemplo de un archivo CSV personalizado. El carácter de tabulación necesario para separar la frase, el peso y la pantalla se representa mediante el texto «[TAB]». Si utiliza este ejemplo, sustituya el texto por un carácter de tabulación.

```
phrase[TAB]weight[TAB]displayAs
Newcastle[TAB]2
Hobart[TAB]2[TAB]Hobart, Australia
U. Dub[TAB]1[TAB]University of Washington, Seattle
W. S. U.[TAB]3
Issaquah
Kennewick
```

Configuración de la sensibilidad de detección de actividad de voz

La detección de actividad de voz (VAD) es una tecnología que determina si la voz está presente en una señal de audio. Amazon Lex V2 utiliza VAD para optimizar la precisión del reconocimiento de voz al distinguir entre la voz real y el ruido de fondo. Puede configurar el nivel de sensibilidad del VAD para mejorar el rendimiento del bot en diferentes entornos acústicos.

Comprender los niveles de sensibilidad del VAD

Amazon Lex V2 ofrece tres niveles de sensibilidad de VAD que puede configurar para la configuración regional de su bot:

Predeterminado

: El nivel de sensibilidad estándar adecuado para la mayoría de los entornos con niveles de ruido de fondo típicos. Esta es la configuración recomendada para casos de uso general.

HighNoiseTolerance

: Mayor tolerancia al ruido de fondo moderado. Usa esta configuración cuando tu bot funcione en entornos con niveles de ruido consistentes pero moderados, como oficinas concurridas o entornos comerciales.

MaximumNoiseTolerance

: Tolerancia máxima a niveles altos de ruido de fondo. Use esta configuración para entornos muy ruidosos, como centros de llamadas, plantas de fabricación o ubicaciones exteriores con un ruido ambiental significativo.

> **Note**
>
> Los niveles más altos de tolerancia al ruido pueden hacer que el sistema sea más permisivo con respecto a lo que considera voz, lo que podría provocar falsos negativos en entornos

muy silenciosos. Elija el nivel de sensibilidad que mejor se adapte al entorno acústico esperado.

Configuración de la sensibilidad del VAD

Puede configurar la sensibilidad de VAD al crear o actualizar la configuración regional de un bot mediante la consola Amazon Lex V2, la AWS CLI o. SDKs

Usar la consola de Amazon Lex V2

Para configurar la sensibilidad del VAD en la consola

1. Abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/.
2. Elige tu bot de la lista.
3. En el panel de navegación izquierdo, selecciona Idiomas del bot.
4. Elige el idioma que quieres configurar o selecciona Añadir idioma para añadir uno nuevo.
5. En la sección Sensibilidad de la detección de voz, selecciona una de las siguientes opciones:
   - Predeterminado: sensibilidad estándar para entornos típicos
   - Alta tolerancia al ruido: para entornos moderadamente ruidosos
   - Máxima tolerancia al ruido: para entornos muy ruidosos
6. Seleccione Guardar para aplicar los cambios.

Mediante la AWS CLI o SDKs

Puede configurar la sensibilidad del VAD mediante el `speechDetectionSensitivity` parámetro en las siguientes operaciones de API:

- `CreateBotLocale`- Configura la sensibilidad del VAD al crear una nueva configuración regional de bot
- `UpdateBotLocale`- Modifique la sensibilidad del VAD para una configuración regional de bot existente
- `DescribeBotLocale`- Vea la configuración actual de sensibilidad del VAD

Example Configuración de la sensibilidad de VAD con AWS CLI

```
aws lexv2-models create-bot-locale \
    --bot-id "AIDACKCEVSQ6C2EXAMPLE" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --nlu-intent-confidence-threshold 0.40 \
    --speech-detection-sensitivity "HighNoiseTolerance"
```

Prácticas recomendadas para la configuración de VAD

- Realice pruebas en el entorno de destino: configure la sensibilidad del VAD en función de las condiciones acústicas reales en las que se desplegará el bot.
- Comience con la configuración predeterminada: comience con la configuración predeterminada y ajústela en función de las pruebas de rendimiento y los comentarios de los usuarios.
- Supervise el rendimiento: utilice los registros de conversaciones y análisis de Amazon Lex V2 para supervisar la precisión del reconocimiento de voz y ajustar la sensibilidad del VAD según sea necesario.
- Considere un caso práctico: los niveles de sensibilidad más altos son beneficiosos para entornos ruidosos, pero pueden no ser necesarios para entornos controlados, como los centros de servicio al cliente con auriculares.

Uso de la diarización del altavoz para centrarse en el altavoz principal

La diarización de los altavoces mantiene al bot en el altavoz principal (el más alto) durante una conversación de voz. Cuando otras voces o conversaciones de fondo llegan al micrófono, el bot se concentra en el altavoz principal en lugar de reaccionar al sonido circundante. Algunos ejemplos son un televisor en la habitación contigua, una conversación paralela en una oficina abierta o alguien que habla cerca de la persona que llama. El resultado son conversaciones de voz más fluidas, con menos turnos iniciados por voces de fondo y menos mensajes interrumpidos por ellas.

> **Note**
>
> La diarización de los altavoces se aplica a las conversaciones de voz en streaming que utilizan esta StartConversationoperación, como las llamadas de telefonía. Requiere audio

monocanal (mono) de 8 kHz. Amazon Lex V2 no aplica el registro diario de los altavoces a las conversaciones de texto ni a las operaciones de no transmisión RecognizeText. RecognizeUtterance Para obtener más información sobre la API de streaming, consulte. Transmisión de conversaciones a un bot Amazon Lex V2

La diarización de los ponentes está disponible en todas las regiones de AWS en las que Amazon Lex V2 está disponible. Para ver la lista de regiones, consulte los puntos de enlace y las cuotas de Amazon Lex V2 en la Referencia general de Amazon Web Services.

Cómo funciona la diarización de los altavoces

Amazon Lex V2 siempre utiliza la detección de actividad de voz (VAD) para determinar si hay voz en la transmisión de audio. El VAD responde a la pregunta «¿está hablando alguien?» pero no «¿quién habla?» , por lo que cualquier voz cercana que sea lo suficientemente alta puede iniciar un giro. La diarización de los altavoces es una configuración opcional que se activa para la configuración regional de un bot. Añade una segunda casilla de verificación en la parte superior del VAD, de forma que el inicio y el fin se inicien y finalicen en función de la actividad del orador principal. Para obtener más información sobre el VAD, consulte. the section called “Configuración de la sensibilidad de detección de actividad de voz”

Al activar la diarización de los altavoces, Amazon Lex V2 añade una segunda señal a esa decisión:

- Actividad del altavoz: un modelo de diarización de streaming etiqueta los altavoces que están activos en cada cuadro de audio de 100 milisegundos. Realiza el seguimiento de hasta cuatro altavoces simultáneos.
- Selección de altavoz principal: Amazon Lex V2 rastrea el volumen de cada altavoz detectado y trata al más alto como altavoz principal. Este enfoque supone que la persona que llama está más cerca del micrófono.
- Decisión combinada: cuando la diarización determina que está hablando un altavoz que no es el principal, Amazon Lex V2 trata ese audio como si no fuera voz. Este audio no inicia un giro, no interrumpe un mensaje ni alcanza el reconocimiento de voz. Cuando el altavoz principal está hablando y cuando la diarización no identifica a un altavoz de fondo distinto, Amazon Lex V2 utiliza la decisión VAD. La diarización se interrumpe cuando se inicia y finaliza la captura de voz. No filtra las voces del audio que Amazon Lex V2 ya ha capturado.

El modelo distingue un altavoz de otro por las características de la voz y no por lo que se dice. Por lo tanto, funciona para todas las configuraciones regionales e idiomas de los bots.

El altavoz principal cambia

El altavoz principal no cambia debido a un breve sonido fuerte. Un altavoz diferente pasa a ser el altavoz principal solo después de ser el altavoz con más volumen registrado, mientras está hablando activamente, durante varios fotogramas consecutivos. La tos, los portazos y las exclamaciones breves no se apoderan de la conversación.

Warm-up punto

El modelo necesita aproximadamente el primer segundo de audio para establecer el seguimiento del altavoz. Durante ese período, la diarización no suprime ninguna voz, por lo que el principio de la voz de la persona que llama nunca se interrumpe mientras el modelo se calienta.

Antes de que la persona que llama hable por primera vez

El orador principal es el que ha hecho más ruido hasta el momento de la conversación. Si un altavoz de fondo es la primera voz que graba el modelo, ese altavoz se convierte en el altavoz principal. La persona que llama toma el relevo después de hablar el tiempo suficiente para convertirse en el altavoz con más volumen registrado. La diarización de los altavoces no protege la conversación durante este período. Una conversación en segundo plano a un volumen alto aún puede iniciar un turno o interrumpir un mensaje antes de que la persona que llama haya hablado.

Recurso automático

Amazon Lex V2 monitorea el registro diario durante cada conversación. Si la diarización comienza a filtrar prácticamente toda la voz que detecta VAD, Amazon Lex V2 restablece primero el estado de diarización. Si la situación persiste, Amazon Lex V2 deja de aplicar la diarización durante el resto del turno de usuario actual y continúa solo con VAD. Amazon Lex V2 vuelve a activar la diarización al inicio del siguiente turno. Un problema de registro diario nunca provoca que una conversación falle o se interrumpa.

¿Qué es lo que no hace la diarización de los altavoces

- No identifica quién es el orador. No hay inscripción, ni nombres de los oradores ni asociación con las cuentas de usuario. El bot solo distingue entre el orador principal y los demás oradores.
- No separa el audio en canales por altavoz y no devuelve las etiquetas de los altavoces en la transcripción. El único efecto es el audio que Amazon Lex V2 trata como voz de la persona que llama.

- No elimina la superposición de la voz de fondo del audio capturado mientras el altavoz principal está hablando. El discurso que se superpone al turno del orador principal se sigue transcribiendo. La diarización finaliza la captura en cuanto se detiene el altavoz principal.
- No rastrea a los altavoces a lo largo de las conversaciones. Todas las conversaciones comienzan sin historial de oradores.
- No se usa sin VAD. La diarización refina la decisión sobre el VAD en lugar de sustituirla.
- No se aplica a los bots que utilizan un agente de IA generativa o un punto final de inferencia personalizado.
- La precisión puede disminuir cuando más de cuatro personas hablan al mismo tiempo.

Configuración de la diarización de los altavoces

La diarización de los altavoces es una configuración regional de los bots, por lo que puedes habilitarla para algunos idiomas de un bot y dejarla desactivada para otros. Puede configurarlo al crear o actualizar la configuración regional de un bot mediante la consola Amazon Lex V2 o la CLI y los SDK de AWS.

Using the console

1. Abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/.
2. Elige tu bot de la lista.
3. En el panel de navegación izquierdo, selecciona Idiomas del bot.
4. Elige el idioma que quieres configurar o selecciona Añadir idioma para añadir uno nuevo.
5. En la sección Diarización de los altavoces, selecciona Configurar.
6. Activa Activar la diarización de los altavoces.
7. Selecciona Guardar para aplicar los cambios.
8. Cree el idioma para que el cambio surta efecto en tiempo de ejecución.

   La página de detalles del idioma muestra el estado actual de la configuración como Activado o Desactivado.

Using the API

Puedes configurar la diarización de los altavoces mediante el `speakerDiarizationSettings` parámetro en las siguientes operaciones de la API:

- `CreateBotLocale`- Configura la diarización de los altavoces para una nueva configuración regional de bots.
- `UpdateBotLocale`- Modifique la diarización de los altavoces para una configuración regional de bot existente.
- `DescribeBotLocale`- Vea la configuración actual de diarización de los altavoces.

La estructura tiene un único miembro obligatorio:

`enabled`

Si Amazon Lex V2 restringe la detección de voz al altavoz principal de esta configuración regional del bot. Los valores válidos son `true` y `false`.

También puede incluir un archivo de importación `speakerDiarizationSettings` en la configuración regional del bot, de modo que la configuración se transfiera con el bot al exportarlo e importarlo.

Example Habilite la diarización de los oradores mediante la AWS CLI

```
aws lexv2-models update-bot-locale \
    --bot-id "ABCDE12345" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --nlu-intent-confidence-threshold 0.40 \
    --speaker-diarization-settings '{
        "enabled": true
    }'
```

Example Vea la configuración actual mediante la AWS CLI

```
aws lexv2-models describe-bot-locale \
    --bot-id "ABCDE12345" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --query "speakerDiarizationSettings"
```

`speakerDiarizationSettings`se devuelve por`DescribeBotLocale`, y no por`ListBotLocales`. Para auditar varios locales, llame a `DescribeBotLocale` cada uno de ellos.

> **Note**
>
> Tras cambiar la configuración, cree la configuración regional del bot con la `BuildBotLocale` operación. Si tu alias incluye una versión de bot numerada, crea una nueva versión a partir de esa `DRAFT` versión y actualiza el alias para que apunte a ella, como harías con cualquier otra configuración regional de bot. Para obtener más información, consulte the section called “Control de versiones y alias con su bot Lex V2”.

Cómo interpreta Amazon Lex V2 la configuración

`{"enabled": true}`

La detección de voz está restringida al altavoz principal.

`{"enabled": false}`

La diarización de los altavoces está desactivada. Amazon Lex V2 solo usa VAD.

No configurado

Desactivado para los bots que nunca hayan configurado la configuración.

`UpdateBotLocale`conserva el valor almacenado cuando lo `speakerDiarizationSettings` omites en la solicitud, por lo que una actualización que cambie la configuración regional de otros bots no desactiva la programación de los altavoces. Para desactivar la función en la configuración regional de un bot cuando esté habilitada, envíala de `{"enabled": false}` forma explícita en lugar de omitir el campo.

Mejores prácticas para la diarización de los oradores

- Actívela cuando el entorno de la persona que llama sea ruidoso: la ventaja proviene de la interferencia de otros altavoces, como los que llaman en casa con la televisión encendida, los que llaman con el altavoz, las oficinas abiertas y las salas de conferencias. En las llamadas individuales silenciosas, la función está diseñada para que no tenga ningún efecto perceptible.

- Espere que la voz más alta sea tratada como la de la persona que llama: el altavoz principal se elige por su volumen. Es posible que dos personas compartan la llamada, por ejemplo, una persona que llama y un familiar, o una persona que llama y un intérprete. En este caso, Amazon Lex V2 trata a la persona más silenciosa como segundo plano hasta que esa persona se convierte en el altavoz con más volumen durante un período prolongado. Deje desactivada la diarización del altavoz para los flujos de conversación en los que dos personas hablen por turno.
- Desactive la función de entrada cuando se abra durante mucho tiempo: la diarización de los altavoces no sirve de nada antes de que la persona que llama haya hablado. Si tu bot se abre con un mensaje largo y las personas que llaman suelen mantener una conversación en segundo plano constante, establece el atributo de `x-amz-lex:allow-interrupt:*:*` sesión en ese mensaje. `false` A continuación, se reproduce el mensaje y la diarización del altavoz se activa una vez que la persona que llama ha hablado. Para obtener más información, consulte the section called “Cómo funciona el comportamiento de interrupción en un bot Lex V2”.
- Mantén los ajustes de interrupción tal y como están: la programación de los altavoces reduce las interrupciones causadas por otras voces. No cambia el funcionamiento de las interrupciones de la persona que llama, así que ajusta los ajustes de entrada y de espera por separado.
- Realice pruebas con un audio representativo: valide el cambio con llamadas de prueba que contengan el ruido de fondo que realmente escucha su bot, como la televisión, la superposición de conversaciones y las conversaciones secundarias. Compara los resultados con las mismas llamadas con el ajuste desactivado antes de ponerlo en producción.
- Implemente un idioma a la vez: dado que la configuración se aplica a una única configuración regional de bot, puede habilitarla para un idioma, revisar las métricas y los registros de las conversaciones y, después, ampliarla a otros idiomas.

Mejora del reconocimiento de los valores de las ranuras con sugerencias de tiempo de ejecución en la conversación

Con las sugerencias en tiempo de ejecución, puede proporcionar a Amazon Lex V2 un conjunto de valores de slot según el contexto para obtener un mejor reconocimiento en las conversaciones de audio y mejorar las resoluciones de los slots. Puede utilizar las sugerencias en tiempo de ejecución para proporcionar una lista de frases en tiempo de ejecución que sean candidatas para la resolución de un valor de slot.

Por ejemplo, si un usuario que interactúa con un bot de reservas de vuelos viaja con frecuencia a San Francisco, Yakarta, Seúl y Moscú, puede configurar las sugerencias en tiempo de ejecución con

una lista de estas cuatro ciudades cuando busque el destino para mejorar el reconocimiento de las ciudades que viaja con más frecuencia.

Las sugerencias sobre el tiempo de ejecución solo están disponibles en inglés (EE. UU.) e inglés (Reino Unido). Se pueden usar con los siguientes tipos de slot:

- Tipos de slots personalizados
- AMAZON.City
- AMAZON.Country
- AMAZON.FirstName
- AMAZON.LastName
- AMAZON.State
- AMAZON.StreetName

Consejos básicos sobre el tiempo de ejecución

- Las sugerencias en tiempo de ejecución se utilizan únicamente cuando se obtiene un valor de slot por parte de un usuario.
- Cuando se utilizan sugerencias en tiempo de ejecución, se prefieren los valores de las sugerencias a los valores similares. Por ejemplo, en el caso de un bot que pide comida, puede configurar una lista de elementos del menú como sugerencias en tiempo de ejecución y, al mismo tiempo, buscar alimentos en un slot personalizado para preferir «filete» en lugar de «ribete» que suene similar.
- Si la entrada del usuario es diferente de los valores proporcionados en las sugerencias en tiempo de ejecución, se utilizará la entrada original del usuario para el slot.
- En el caso de los tipos de slot personalizados, los valores proporcionados como sugerencias en tiempo de ejecución se utilizarán para resolver el slot, incluso si no formaron parte del slot personalizado durante la creación del bot.
- Las sugerencias en tiempo de ejecución solo se admiten para entradas de audio de 8 kHz. Están disponibles con integraciones de centros de contacto compatibles con Amazon Lex V2. Las sugerencias en tiempo de ejecución no se proporcionan para la entrada de audio desde la ventana de prueba de la consola de Amazon Lex V2 porque utiliza una entrada de audio de 16 kHz.

> **Note**
>
> Antes de poder usar las sugerencias en tiempo de ejecución con un bot existente, primero debe volver a compilarlo. Las versiones existentes de un bot no admiten sugerencias en tiempo de ejecución. Debe crear una nueva versión del bot para usarlas.

Puede enviar sugerencias en tiempo de ejecución a Amazon Lex V2 mediante las operaciones PutSession, RecognizeText, RecognizeUtterance o StartConversatio. También puede añadir sugerencias en tiempo de ejecución mediante una función de Lambda.

Puede enviar sugerencias en tiempo de ejecución al principio de una conversación para configurar las sugerencias para cada slot utilizado en el bot, o puede enviar sugerencias como parte del estado de la sesión durante una conversación. El atributo `runtimeHints` asigna un slot a las sugerencias de ese slot.

Una vez que envíe una sugerencia de tiempo de ejecución a Amazon Lex V2, esta persiste durante cada turno de la conversación hasta que finalice la sesión. Si envía una estructura de `runtimeHints` nula, se utilizan las sugerencias existentes. Puede modificar las sugerencias de la siguiente manera:

- Enviar una nueva estructura de `runtimeHints` al bot. Los contenidos de la nueva estructura sustituyen a los existentes.
- Enviar una nueva estructura vacía de `runtimeHints` al bot. Esto borra las sugerencias en tiempo de ejecución para el bot.

Añadir los valores de los slots en su contexto

Añada contexto a su bot proporcionando los valores de slot esperados como sugerencias en tiempo de ejecución cuando su aplicación tenga información sobre el próximo enunciado probable del usuario. Añada un enlace de código de diálogo Lambda a su bot (consulte Integración de una AWS Lambda función en su bot Amazon Lex V2 para obtener más información) y utilice el campo ProposedNextState en AWS Lambdaformato de evento de entrada para Lex V2 para determinar las sugerencias en tiempo de ejecución que debe incluir para mejorar la conversación con el usuario.

Por ejemplo, en una aplicación bancaria, puede generar una lista de apodos de cuentas para un usuario específico y, a continuación, utilizarla para obtener la cuenta a la que el usuario quiere acceder.

Envíe sugerencias sobre el tiempo de ejecución al inicio de la conversación cuando disponga de contexto para ayudar al bot a interpretar las entradas de los usuarios. Por ejemplo, si tiene el número de teléfono del usuario, puede usar esta información para buscarlo y usar la operación `PutSession` o `StartConversation` para pasar sugerencias sobre su nombre y apellidos al bot si está buscando el nombre del usuario para validar sus credenciales.

Durante una conversación, es posible que recopile información sobre el valor de un slot que pueda ayudarle con el valor de otro slot. Por ejemplo, si tienes el número de cuenta del usuario en una aplicación de mantenimiento de coches, puede buscar los coches que son propiedad del cliente y pasarlos como sugerencias para encontrar otro slot.

Escriba separado mediante puntos o espacios los acrónimos u otras palabras cuyas letras deban pronunciarse por separado. No utilice letras individuales a menos que formen parte de una frase, como «J. P. Morgan» o «A.W.S». Puede utilizar letras mayúsculas o minúsculas para definir los acrónimos.

Añadir sugerencias a un slot

Para añadir sugerencias en tiempo de ejecución a un slot, se utiliza la estructura de `runtimeHints` que forma parte de la estructura de `sessionState`. A continuación se muestra un ejemplo de la estructura de `runtimeHints`. Proporciona sugerencias para dos slots, «Nombre» y «Apellido» para la intención de «ConcertarCita».

```
{
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
```

```
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "Stiles"
                            },
                            {
                                "phrase": "Major"
                            }
                        ]
                    }
                }
            }
        }
    }
}
```

También puede utilizar una función de Lambda para añadir sugerencias sobre el tiempo de ejecución durante una conversación. Para añadir sugerencias en tiempo de ejecución, añada la estructura de `runtimeHints` al estado de sesión de la respuesta que la función de Lambda envía a Amazon Lex V2. Para obtener más información, consulte AWS Lambdaformato de respuesta para Lex V2.

Debe especificar un `intentName` y un `slotName` válidos en la solicitud; de lo contrario, Amazon Lex V2 devolverá un error de tiempo de ejecución.

Captura de valores de ranura con estilos de ortografía durante la conversación

Amazon Lex V2 incluye slots integrados para capturar información específica del usuario, como nombre, apellidos, direcciones de correo electrónico o identificadores alfanuméricos. Por ejemplo, puede usar el slot `AMAZON.LastName` para capturar apellidos como "Jackson" o "Garcia". Sin embargo, Amazon Lex V2 puede confundirse con apellidos difíciles de pronunciar o que no son comunes en un lugar, como «Xiulan». Para capturar estos nombres, puede pedirle al usuario que introduzca la entrada deletreando por letras o deletreando por palabras.

Amazon Lex V2 ofrece tres estilos de obtención de estilos para su uso. Al establecer un estilo de obtención de slots, cambia la forma en que Amazon Lex V2 interpreta las entradas del usuario.

Deletrear por letra: con este estilo, puede indicarle al bot que escuche la ortografía en lugar de la frase completa. Por ejemplo, para capturar un apellido como «Xiulan», puede decirle al usuario que

deletree su apellido letra por letra. El bot capturará la ortografía y resolverá las letras de una palabra. Por ejemplo, si el usuario dice «x i u l a n», el bot captura el apellido como «xiulan».

Deletrear por palabra: en las conversaciones de voz, especialmente por teléfono, hay algunas letras, como «t», «b», «p», que suenan de forma similar. Si al capturar valores alfanuméricos o al escribir nombres se obtiene un valor incorrecto, puede pedir al usuario que introduzca una palabra identificativa junto con la letra. Por ejemplo, si la respuesta de voz a una solicitud de identificador de reserva es «abp123», es posible que su bot reconozca la frase «abb123» en su lugar. Si el valor es incorrecto, puede pedirle al usuario que introduzca la entrada «a como en alfa, b como en boy, p como en peter, un dos tres». El bot resolverá la entrada como «abp123».

Puede usar los siguientes formatos para deletrear por palabra:

Para inglés (EE. UU.), inglés (Reino Unido) e inglés (Australia):

- «como en» (como en manzana)
- «para» (m para manzana)
- «como» (a como manzana)

Para alemán (de-DE):

- «vino» (p: pizza de vino)
- «für» (p para pizza)

Para francés (fr-FR):

- «comme» (p comme paris)
- «pour» (p pour paris)

Para español (es-419, es-US, es-ES):

- «como» (u como unión)
- «de» (u de unión)

Predeterminado: este es el estilo natural de capturar slots utilizando la pronunciación de las palabras. Por ejemplo, puede capturar nombres como «John Stiles» de forma natural. Si no se especifica un estilo de obtención, el bot utiliza el estilo predeterminado. Para los tipos de slot con código

`AMAZON.AlphaNumeric` y `AMAZON.UKPostal`, el estilo predeterminado admite el deletreo mediante la introducción de letras.

Si el nombre «Xiulan» se pronuncia con una combinación de letras y palabras, como «x como en rayos X i u l como en león a n», el estilo de obtención del slot debe configurarse para que se escriba por palabra. El estilo de deletreo por letra no lo reconocerá.

Debería crear una interfaz de voz que capture los valores de los slots con un estilo de conversación natural para una mejor experiencia. En el caso de entradas que no se capturen correctamente con el estilo natural, puede volver a preguntar al usuario y configurar el estilo de obtención de los slots como deletrear por letra o por palabra.

Puedes usar los estilos ortografía por palabra y ortografía por letra para los siguientes tipos de tragaperras en los idiomas inglés (EE. UU.), inglés (Reino Unido), inglés (Australia), alemán (de-DE), francés (fr-FR) y español (es-419, es-US, es-ES):

- AMAZON.AlphaNumeric
- AMAZON.EmailAddress
- AMAZON.FirstName
- AMAZON.LastName
- AMAZON.UKPostalCode(Solo inglés (EE. UU.), inglés (Reino Unido) e inglés (Australia))
- Tipos de slots personalizados

Habilitar deletreo

Se habilita el deletreo por letra y por palabra en tiempo de ejecución cuando se están obteniendo slots del usuario. Puede establecer el estilo ortográfico con la StartConversation operación PutSession RecognizeText RecognizeUtterance,, u. También puede activar el deletreo por letra por palabra mediante una función de Lambda.

El estilo de deletreo se establece utilizando el campo `dialogAction` del campo `sessionState` en la solicitud de una de las operaciones de API antes mencionadas o al configurar la respuesta de Lambda (consulte AWS Lambdaformato de respuesta para Lex V2 para obtener más información). Solo puede establecer el estilo si el tipo de acción del diálogo es `ElicitSlot` y si el slot que se va a generar es uno de los tipos de slots compatibles.

El siguiente código JSON muestra el campo `dialogAction` configurado para usar el estilo de deletreo por palabra:

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
    "type": "ElicitSlot"
}
```

El campo `slotElicitationStyle` se puede definir en `SpellByLetter`, `SpellByWord` o `Default`. Si no especifica un valor, el valor predeterminado es `Default`.

> **Note**
>
> No puede activar los estilos de obtención con deletreo por letra o por palabra en la consola.

Ejemplo de código con Lambda y Lex V2

Por lo general, se cambia el estilo de deletreo al intentar resolver por primera vez un valor de slot que no funcionó. El siguiente ejemplo de código es una función de Lambda de Python que utiliza el estilo de deletreo por palabra en el segundo intento de resolver un slot.

Para utilizar el código de ejemplo, debe disponer de lo siguiente:

- Un bot con un idioma, inglés (GB) (en_GB).
- Una intención, «CheckAccount» con un ejemplo de expresión: «Me gustaría comprobar mi cuenta». Asegúrese de que la opción Utilizar una función de Lambda para la inicialización y la validación esté seleccionada en la sección Enlaces de códigos de la definición de intención.
- La intención debe tener una ranura, "PostalCode«, del tipo `AMAZON.UKPostalCode` integrado.
- Un alias con la función de Lambda definida. Para obtener más información, consulte Creación de una AWS Lambda función para su bot Amazon Lex V2.

```
import json
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)
```

```
# --- Helpers that build all of the responses ---

def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']

def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']
    return {}

def get_slot(intent_request, slotName):
    slots = get_slots(intent_request)
    if slots is not None and slotName in slots and slots[slotName] is not None:
        logger.debug('resolvedValue={}'.format(slots[slotName]['value']
['resolvedValues']))
        return slots[slotName]['value']['resolvedValues']
    else:
        return None

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit,
 slot_elicitation_style, message):
    return {'sessionState': {'dialogAction': {'type': 'ElicitSlot',
                                              'slotToElicit': slot_to_elicit,
                                              'slotElicitationStyle':
 slot_elicitation_style
                                              },
                             'intent': {'name': intent_request['sessionState']
['intent']['name'],
                                        'slots': slots,
                                        'state': 'InProgress'
                                        },
                             'sessionAttributes': session_attributes,
                             'originatingRequestId': 'REQUESTID'
                             },
            'sessionId': intent_request['sessionId'],
            'messages': [ message ],
            'requestAttributes': intent_request['requestAttributes']
            if 'requestAttributes' in intent_request else None
            }

def build_validation_result(isvalid, violated_slot, slot_elicitation_style,
 message_content):
    return {'isValid': isvalid,
```

```
            'violatedSlot': violated_slot,
            'slotElicitationStyle': slot_elicitation_style,
            'message': {'contentType': 'PlainText',
            'content': message_content}
            }

def GetItemInDatabase(postal_code):
    """
    Perform database check for transcribed postal code. This is a no-op
    check that shows that postal_code can't be found in the database.
    """
    return None

def validate_postal_code(intent_request):

    postal_code = get_slot(intent_request, 'PostalCode')

    if GetItemInDatabase(postal_code) is None:
        return build_validation_result(
            False,
            'PostalCode',
            'SpellByWord',
            "Sorry, I can't find your information. " +
            "To try again, spell out your postal " +
            "code using words, like a as in apple."
        )
    return {'isValid': True}

def check_account(intent_request):
    """
    Performs dialog management and fulfillment for checking an account
    with a postal code. Besides fulfillment, the implementation for this
    intent demonstrates the following:
    1) Use of elicitSlot in slot validation and re-prompting.
    2) Use of sessionAttributes to pass information that can be used to
        guide a conversation.
    """
    slots = get_slots(intent_request)
    postal_code = get_slot(intent_request, 'PostalCode')
    session_attributes = get_session_attributes(intent_request)

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Validate the PostalCode slot. If any aren't valid,
        # re-elicit for the value.
```

```
        validation_result = validate_postal_code(intent_request)
        if not validation_result['isValid']:
            slots[validation_result['violatedSlot']] = None
            return elicit_slot(
                session_attributes,
                intent_request,
                slots,
                validation_result['violatedSlot'],
                validation_result['slotElicitationStyle'],
                validation_result['message']
            )

        return close(
            intent_request,
            session_attributes,
            'Fulfilled',
            {'contentType': 'PlainText',
             'content': 'Thanks'
             }
        )

def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': 'xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
        },
        'messages': [ message ],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }

# --- Intents ---

def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """
```

```
    intent_name = intent_request['sessionState']['intent']['name']
    response = None

    # Dispatch to your bot's intent handlers
    if intent_name == 'CheckAccount':
        response = check_account(intent_request)

    return response

# --- Main handler ---

def lambda_handler(event, context):
    """
    Route the incoming request based on the intent.

    The JSON body of the request is provided in the event slot.
    """

    # By default, treat the user request as coming from
    # Eastern Standard Time.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()

    logger.debug('event={}'.format(json.dumps(event)))
    response = dispatch(event)
    logger.debug("response={}".format(json.dumps(response)))

    return response
```

Uso del relleno de audio para mejorar la capacidad de respuesta de los bots

El relleno de audio reproduce un breve audio de fondo, como una melodía ligera o pulsaciones suaves, durante la pausa entre el final de la expresión del usuario y el inicio de la respuesta del bot. Esto oculta los retrasos en el procesamiento y hace que las conversaciones de voz parezcan naturales.

> **Note**
>
> En el momento del lanzamiento, el rellenador de audio está disponible para las configuraciones locales de bots que admiten interacciones de voz a voz y que estén

configuradasunifiedSpeechSettings. Support para modos de conversación adicionales se lanzará en los próximos meses.

Tipos de relleno de audio disponibles

Amazon Lex V2 incluye siete sonidos de relleno integrados, organizados en dos familias:

- Melody: Chipper Chime
- Melody: Curious Crawl
- Melodía: Rising Ripple
- Melody - Paciente Ping
- Melody - Reflexionando sobre Pong
- Escritura: teclas cinéticas
- Escritura: Qwerty silencioso

Utilice el botón Reproducir vista previa de audio de la consola Amazon Lex V2 para escuchar cada opción antes de guardarla en la configuración regional de un bot.

Parámetros de temporización

Puedes ajustar tres parámetros de temporización para controlar cuándo se reproduce el relleno de audio y cómo pasa a ser la respuesta del bot:

`startDelayInMilliseconds`

Tiempo de espera una vez finalizado el enunciado del usuario antes de iniciar la reproducción del audio Filler. El intervalo válido es `500` de `5000` milisegundos. El valor predeterminado es `1000`.

`minimumPlayDurationInMilliseconds`

Tiempo mínimo de reproducción del relleno de audio después de que se haya iniciado, incluso si la respuesta del bot está lista antes. El rango válido es `1000` de `5000` milisegundos. El valor predeterminado es `3000`.

`responseDeliveryDelayInMilliseconds`

Se inserta un retardo silencioso entre el final de la reproducción del relleno de audio y el inicio de la respuesta del bot. El rango válido es `200` de `1000` milisegundos. El valor predeterminado es `500`.

Configuración del relleno de audio

Puede configurar el relleno de audio al crear o actualizar la configuración regional de un bot a través de la consola Amazon Lex V2, el diseñador de IA conversacional de Amazon Connect o la CLI y los SDK de AWS.

Using the console

1. Abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/.
2. Elige tu bot de la lista.
3. En el panel de navegación izquierdo, selecciona Idiomas del bot.
4. Elige el idioma que quieres configurar o selecciona Añadir idioma para añadir uno nuevo.
5. En la sección Relleno de audio, selecciona Activar relleno de audio.
6. Elige un tipo de audio de la melodía o de las opciones de escritura. Usa Reproducir vista previa de audio para escuchar el relleno seleccionado.
7. Ajuste los controles deslizantes de temporización para ver el retardo de inicio, la duración mínima de la reproducción y el búfer de respuesta según sea necesario.
8. Selecciona Guardar para aplicar los cambios.

Using the Amazon Connect Conversational AI designer

1. Abre el sitio web de administración de Amazon Connect y navega hasta el diseñador de IA conversacional de tu bot.
2. Abra el idioma (configuración regional) que desee configurar.
3. En la sección Audio Filler, active el relleno de audio y elija un tipo de audio.
4. Ajuste el retardo de inicio, la duración mínima de reproducción y los valores del búfer de respuesta.
5. Guarde los cambios. El diseñador aplica lo mismo `audioFillerSettings` a la configuración regional del bot Amazon Lex V2 subyacente.

Using the API

Puede configurar el relleno de audio mediante el `audioFillerSettings` parámetro en las siguientes operaciones de la API:

- `CreateBotLocale`- Configura el relleno de audio para una nueva configuración regional de bots.
- `UpdateBotLocale`- Modifique el relleno de audio para una configuración regional de bot existente.
- `DescribeBotLocale`- Ver la configuración actual del relleno de audio.

Example Configure el rellenador de audio mediante la AWS CLI

```
aws lexv2-models update-bot-locale \
    --bot-id "bot-1234567890abcdef0" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --nlu-intent-confidence-threshold 0.40 \
    --audio-filler-settings '{
        "enabled": true,
        "audioType": "MELODY_CHIPPER_CHIME",
        "startDelayInMilliseconds": 1000,
        "minimumPlayDurationInMilliseconds": 3000,
        "responseDeliveryDelayInMilliseconds": 500
    }'
```

Relleno de audio con mensajes provisionales de un agente de IA

El relleno de audio funciona junto con los mensajes provisionales de los agentes de IA. Cuando un agente de IA envía un mensaje provisional a la persona que llama (por ejemplo, «Deja que te lo busque»), el temporizador de retardo de inicio se mide desde el final de ese mensaje provisional y no desde la invocación original. Esto evita que el relleno de audio se superponga con la voz del agente y garantiza que el retraso que experimenta la persona que llama se mida a partir del audio más reciente que haya escuchado.

Relleno de audio con ganchos de diálogo y código de cumplimiento

El relleno de audio también se reproduce durante el intervalo de procesamiento introducido por los ganchos de código de diálogo de Lambda y los ganchos de código de cumplimiento. Se aplican los mismos parámetros de tiempo, por lo que las personas que llaman disfrutan de una experiencia uniforme, independientemente de si tu bot delega el procesamiento en un agente de IA, en un agente de inteligencia artificial, en un enlace de códigos o en ambos al mismo tiempo.

Prácticas recomendadas para el relleno de audio

- Haga coincidir el relleno con la voz de su marca. Usa rellenadores de melodías para experiencias de consumo o venta minorista, y rellenadores de texto cuando los usuarios esperen que el bot esté trabajando activamente en una tarea.
- Ajusta el retraso de inicio a tu perfil de latencia. Si la mayoría de las respuestas de los bots son más rápidas que`startDelayInMilliseconds`, el relleno rara vez se reproducirá. Reduzca el retraso para las cargas de trabajo con alta latencia y auméntelo para los bots que respondan rápidamente.
- Mantén la duración mínima de juego corta para los bots rápidos. `minimumPlayDurationInMillisecondsE`n un robot rápido, si la respuesta es larga, se percibe la latencia al mantener pulsado el botón de relleno una vez que la respuesta esté lista.
- Realice una prueba con tráfico representativo. Valide la elección del relleno y el momento en conversaciones realistas antes de pasar a la fase de producción.

Supervisión del rendimiento de los bots en Lex V2

El monitoreo es importante para mantener la fiabilidad, disponibilidad y desempeño de los chatbots de Amazon Lex V2. En este tema se describe el uso de los registros de conversaciones para supervisar las conversaciones entre sus usuarios y sus chatbots, el uso de las estadísticas de enunciados para determinar los enunciados que los bots detectan y no detectan, y cómo utilizar Amazon CloudWatch Logs y AWS CloudTrail supervisar Amazon Lex V2. También describe las métricas de asociación de canales y el tiempo de ejecución de Amazon Lex V2.

Utilice estas herramientas y métricas para comprender qué instrucciones y acciones puede llevar a cabo para mejorar el rendimiento de sus bots.

Temas

- Medición del rendimiento empresarial con Analytics
- Cómo habilitar registros de conversaciones para sus bots Lex V2
- Registro de errores con registros de errores en Lex V2
- Monitoreo de métricas operativas en Lex V2
- Evaluación del rendimiento de los bots Lex V2 con Test Workbench

Medición del rendimiento empresarial con Analytics

Con Analytics, puede evaluar el rendimiento de su bot con métricas relacionadas con las tasas de éxito y fracaso de las interacciones de sus bots con los clientes. También puede visualizar los patrones de los flujos de conversación entre su bot y los clientes. Analytics optimiza su experiencia al resumir estas métricas en gráficos y tablas. Analytics proporciona herramientas que le ayudan a filtrar los resultados a fin de identificar cuestiones y problemas relacionados con las intenciones, los slots, las expresiones y las conversaciones. Puede usar estos datos para optimizar y mejorar su bot a fin de crear una mejor experiencia para el cliente.

> **Note**
>
> Para que un usuario pueda acceder a Analytics, debe adjuntarse a su rol de IAM la Política gestionada por AWS: AmazonLexFullAccess o una política personalizada que incluya permisos de API de análisis. Consulte Administración de permisos de acceso para análisis para obtener más información sobre cómo gestionar los permisos de usuario con una política

personalizada. Si la Política gestionada por AWS: AmazonLexReadOnly está asociada al rol de IAM de un cliente, aparecerá un mensaje de error con los permisos que faltan y que debe añadir al rol de IAM del usuario para que pueda acceder a los paneles de Analytics.

Para acceder a Analytics

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/home.
2. En el panel de navegación, en Bots, seleccione el bot que desea ver en Analytics.
3. Seleccione la sección de Analytics que desea ver.

Temas

- Definiciones clave
- Filtrado de resultados
- Información general: un resumen del rendimiento de su bot
- Panel de conversaciones: un resumen de las conversaciones de su bot
- Panel de rendimiento: un resumen de las métricas de intenciones y enunciados de su bot
- Uso de API para análisis
- Administración de permisos de acceso para análisis

Definiciones clave

En este tema se proporcionan las definiciones clave que le ayudarán a interpretar los análisis de sus bots. Estas definiciones están relacionadas con el rendimiento de su bot en cuatro contextos: Intenciones, Slots, Conversaciones y Enunciados. Los siguientes campos son relevantes para muchas de las métricas de rendimiento:

- El campo `state` del objeto `Intent`.
- El `type`campo del `dialogAction` objeto dentro del SessionStateobjeto.

Intenciones

Amazon Lex V2 clasifica las intenciones de las siguientes maneras:

- Exitosa: el bot cumplió satisfactoriamente la intención. Se cumple una de las siguientes condiciones:
  - El `state` de la intención es `ReadyForFulfillment` y el `type` de `dialogAction` es `Close`.
  - El `state` de la intención es `Fulfilled` y el `type` de `dialogAction` es `Close`.
- Fallida: el bot no cumplió con la intención. El estado de la intención. Se cumple una de las siguientes condiciones:
  - El `state` de la intención es `Failed` y el `type` de `dialogAction` es `Close` (por ejemplo, el usuario rechazó la solicitud de confirmación).
  - El bot cambia a la `AMAZON.FallbackIntent` antes de que se complete la intención.
- Cambiada: el bot reconoce una intención diferente y, en su lugar, cambia a esa intención, antes de que la intención original se clasifique como exitosa o fallida.
- Descartada: el cliente no responde antes de que la intención se clasifique como exitosa o fallida.

Ranuras

Amazon Lex V2 clasifica los slots de las siguientes maneras:

- Exitoso: el bot rellenó el slot y pasó correctamente a otro slot o al paso de confirmación.
- Fallido: el bot no pudo rellenar el slot, ni siquiera tras alcanzar el número máximo de reintentos.
- Descartado: el cliente no responde o cambia a otra intención antes de que el slot se clasifique como exitoso o fallido.

Conversaciones

Cuando un cliente realiza una llamada en tiempo de ejecución a Amazon Lex V2, proporciona un `sessionId` y Amazon Lex V2 genera un `originatingRequestId`. Si el cliente no responde en el tiempo de espera de la sesión (`idleSessionTTLInSeconds`) que ha establecido para el bot, la sesión caduca. Si un cliente vuelve a la sesión utilizando la misma `sessionId`, Amazon Lex V2 genera una nueva `originatingRequestId`.

Para el análisis, una conversación es una combinación única de una `sessionId` y una `originatingRequestId`. Amazon Lex V2 clasifica las conversaciones de las siguientes maneras:

- Exitosa: la intención final de la conversación se clasifica como un éxito.

- Fallida: la intención final de la conversación es fallida. La conversación también falla si Amazon Lex V2 utiliza de forma predeterminada la AMAZON.FallbackIntent.
- Descartada: el cliente no responde antes de que la intención se clasifique como exitosa o fallida.

Enunciados

Amazon Lex V2 clasifica los enunciados de las siguientes maneras:

- Detectado: Amazon Lex V2 reconoce el enunciado como un intento de invocar una intención configurada para un bot.
- Perdido: Amazon Lex V2 no reconoce el enunciado.

Filtrado de resultados

En la parte superior de cada página, puede filtrar los resultados de las analíticas de su bot.

Puede filtrar por los siguientes parámetros:

- Tiempo: puede filtrar los resultados por un rango de tiempo relativo o absoluto. Al seleccionar una hora de inicio y finalización, Amazon Lex V2 recupera las conversaciones que comenzaron después de la hora de inicio y finalizaron antes de la hora de finalización.
  - Intervalo relativo: seleccione 1d para ver los resultados del día anterior, 1s para la semana pasada o 1m para el mes anterior.

    Para ver más opciones, seleccione Personalizado y seleccione una duración en el menú Intervalo relativo. Para tener más control sobre la duración, seleccione Intervalo personalizado, introduzca un número en el campo Duración y seleccione una unidad de tiempo en el menú desplegable.
  - Intervalo absoluto: seleccione Personalizado y seleccione el menú Intervalo absoluto para filtrar las conversaciones dentro del rango de tiempo que especifique. Puede elegir una fecha de inicio y finalización en el calendario o escribirla en YYYY/MM/DD formato.

> **Note**
>
> El rango de tiempo de análisis tiene las siguientes restricciones:
>
> - La fecha de inicio debe situarse entre los últimos 365 días a partir de la fecha actual.

- La fecha de finalización no debe ser superior a 1 mes después de la fecha de inicio.

- Filtros de bots: para filtrar por configuración regional, alias y versión del bot, seleccione los menús desplegables denominados Todas las configuraciones regionales, Todos los alias y Todas las versiones.
- Modalidad: seleccione el icono de engranaje y seleccione el menú desplegable Modalidad para elegir si quiere que se muestren los resultados de Voz o Texto.
- Canal: seleccione el icono de engranaje y seleccione el menú desplegable Canal para elegir el canal en el que desea mostrar los resultados. Para obtener más información sobre la integración de canales, consulte Integrar un bot de Amazon Lex V2 con una plataforma de mensajería y los centros de contacto de Amazon Connect

Información general: un resumen del rendimiento de su bot

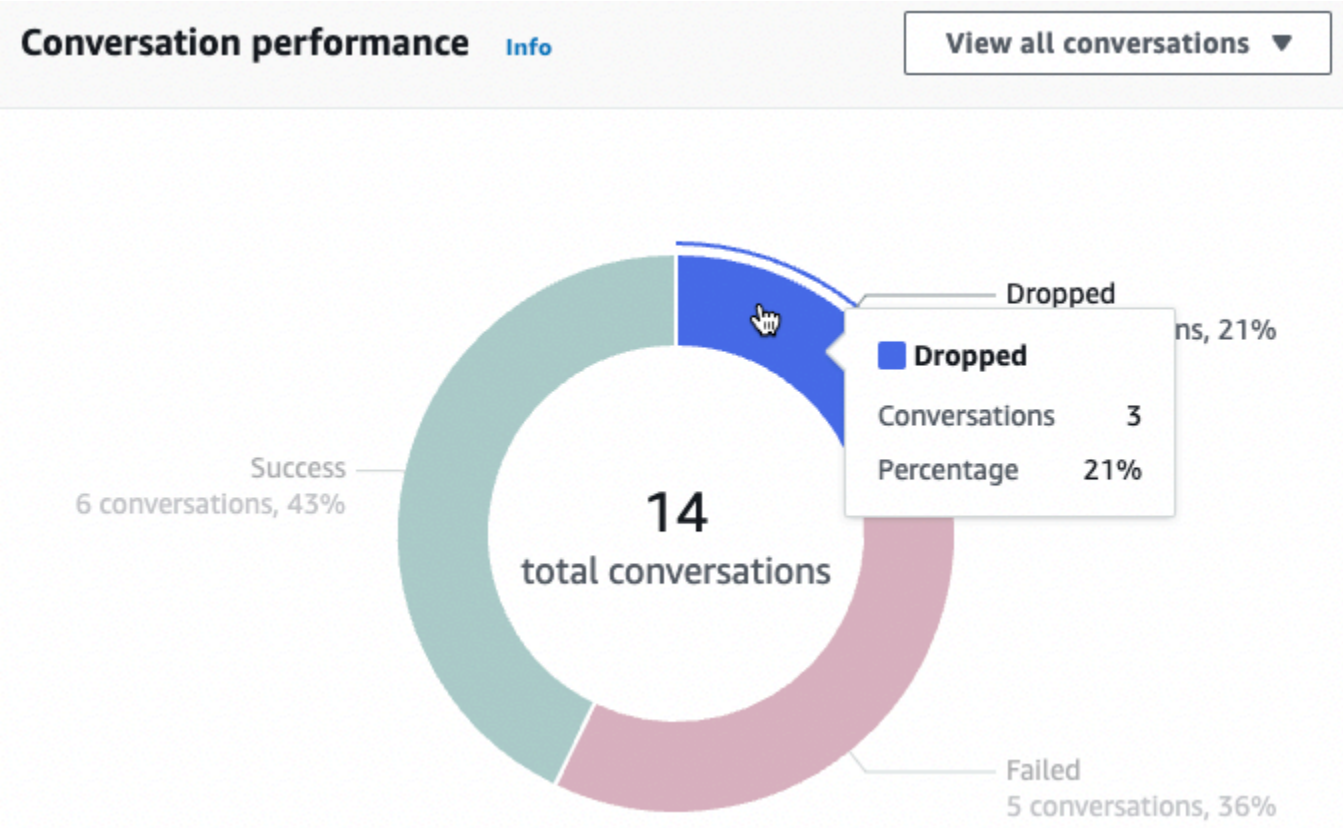
La página de información general resume el rendimiento de su bot en las conversaciones, el reconocimiento de enunciados y el uso de intenciones. Consta de las secciones siguientes:

- Rendimiento de las conversaciones
- Tasa de reconocimiento de enunciados
- Historial de rendimiento de las conversaciones
- Las 5 intenciones más utilizadas
- Las 5 intenciones fallidas más frecuentes

Rendimiento de las conversaciones

Use este gráfico para hacer un seguimiento del número y el porcentaje de conversaciones clasificadas como exitosas, fallidas o descartadas. Para acceder a una lista de conversaciones, seleccione Ver todas las conversaciones y verá un menú desplegable. Puede elegir ver una lista de todas las conversaciones de los usuarios con el bot o filtrar las conversaciones con un resultado específico (exitosas, fallidas o descartadas). Estos enlaces lo llevan a la subsección Conversaciones del Panel de conversaciones. Para obtener más información, consulte Conversaciones.

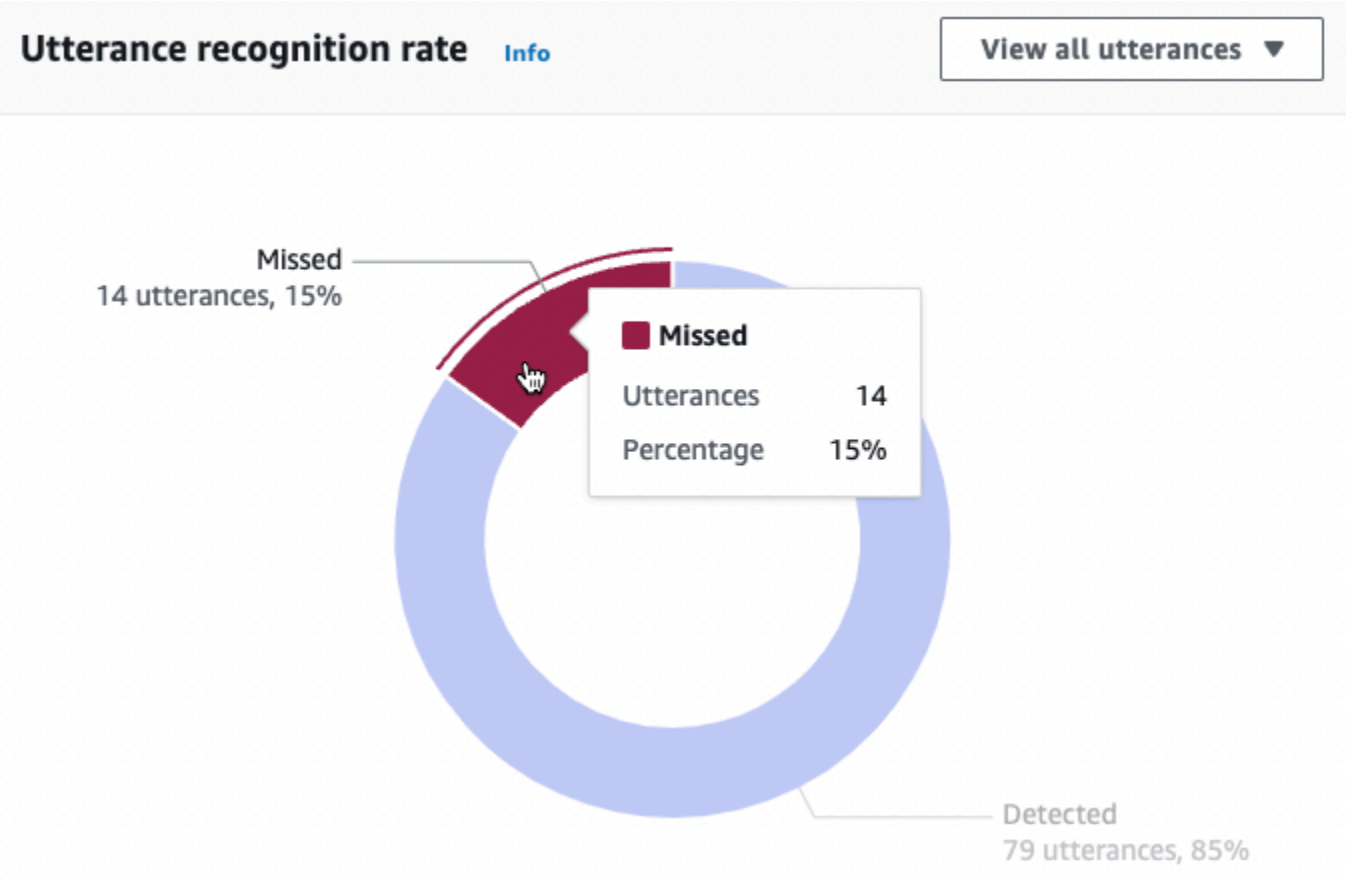
Para mostrar un cuadro con el recuento y el porcentaje de conversaciones con ese resultado, coloque el cursor sobre un segmento del gráfico, como en la imagen siguiente.



Tasa de reconocimiento de enunciados

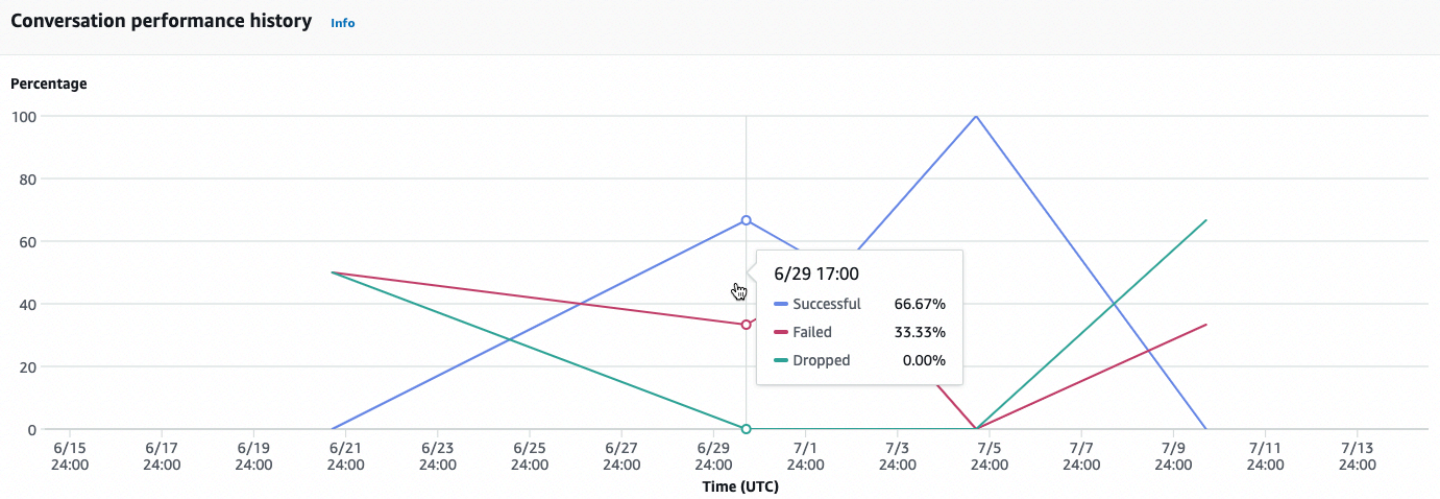
Use este gráfico para hacer un seguimiento del número y el porcentaje de enunciados que su bot detectó y no reconoció. Para acceder a una lista de enunciados, seleccione Ver enunciados y verá un menú desplegable. Puede elegir ver una lista de todos los enunciados de los usuarios o filtrar los enunciados con un resultado específico (perdidos o detectados). Estos enlaces lo llevan a la subsección de Reconocimiento de enunciados del Panel de rendimiento. Para obtener más información, consulte Ver enunciados para ir a Reconocimiento de enunciados.

Para mostrar un cuadro con el recuento y el porcentaje de enunciados, coloque el cursor sobre un segmento del gráfico, como se ve en la imagen siguiente.



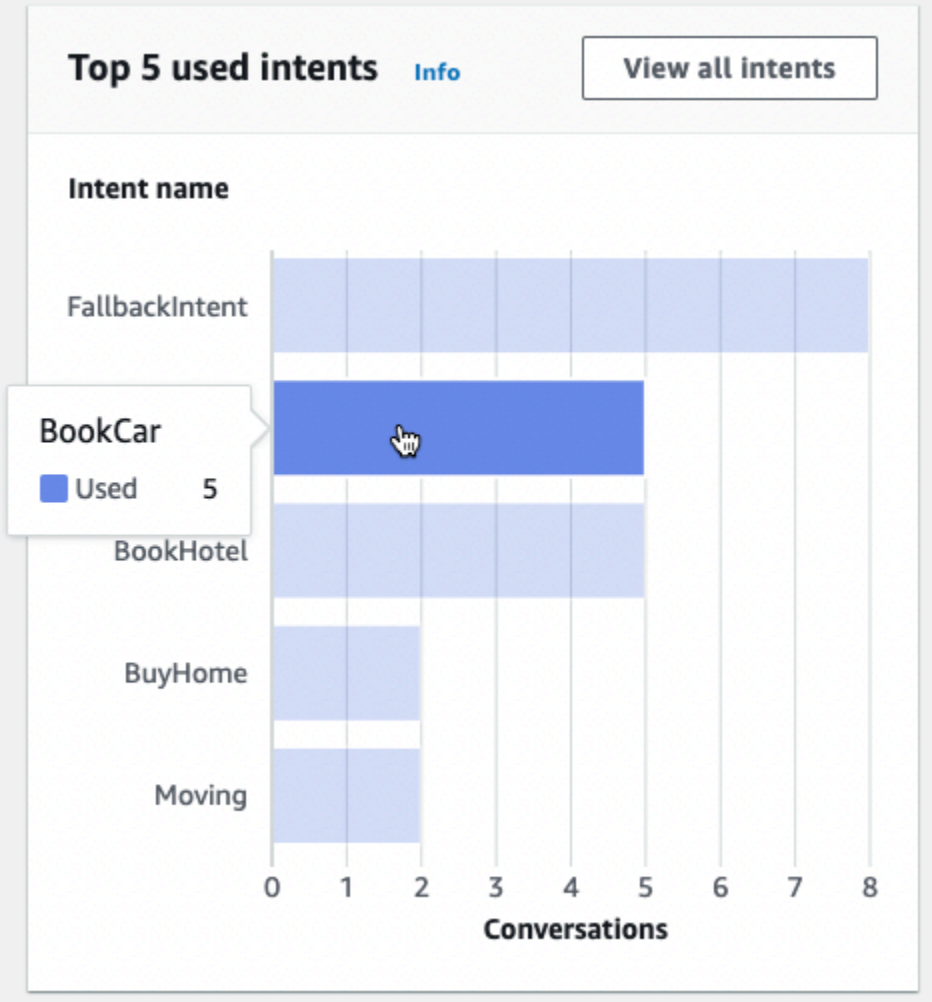
Historial de rendimiento de las conversaciones

Use este gráfico para hacer un seguimiento del porcentaje de conversaciones clasificadas como exitosas, fallidas o descartadas durante el intervalo de tiempo que haya establecido en los filtros. Para ver el porcentaje de conversaciones con un resultado específico en un intervalo de tiempo, coloque el cursor sobre ese intervalo, como se muestra en la siguiente imagen.



Las 5 intenciones más utilizadas

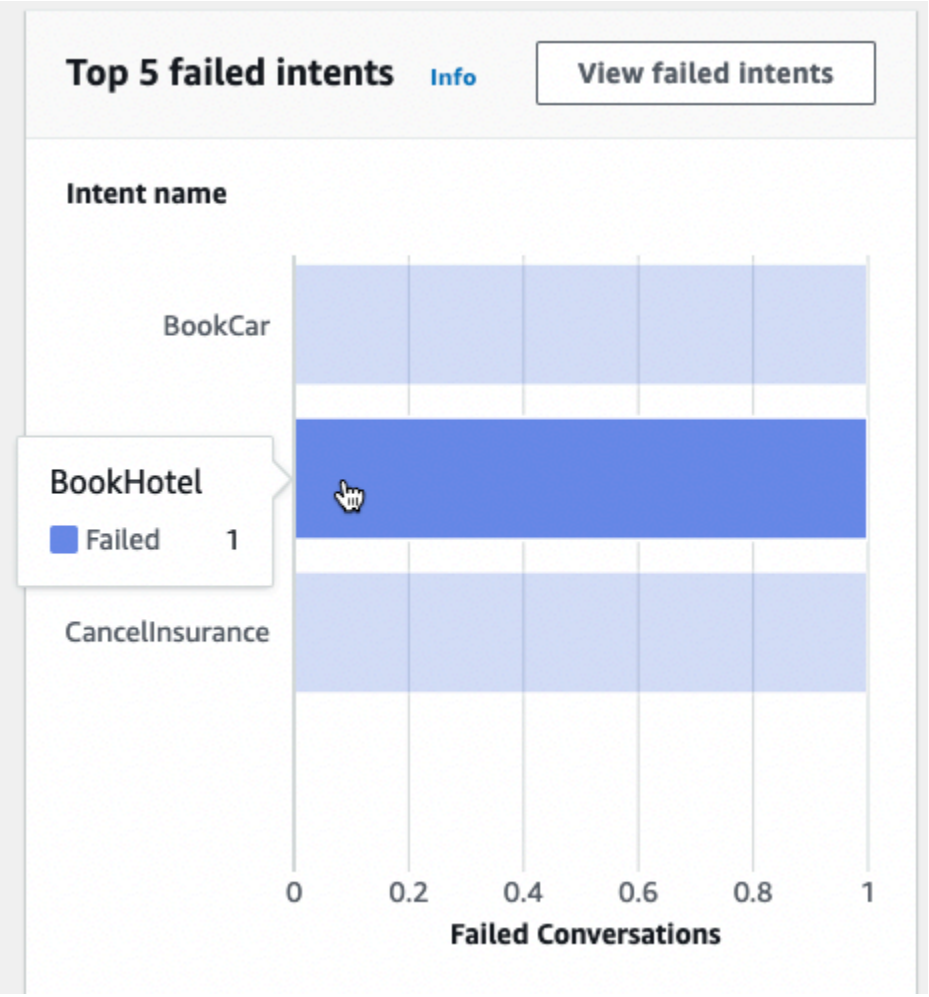
Use este gráfico para identificar las cinco intenciones principales que los clientes usaron con su bot. Coloque el cursor sobre una barra para ver el número de veces que su bot reconoció esa intención, como se muestra en la siguiente imagen.



Seleccione Ver todas las intenciones para ir a la subsección de Rendimiento de las intenciones del Panel de rendimiento, donde puede ver las métricas del rendimiento de su bot a la hora de cumplir las intenciones. Para obtener más información, consulte Rendimiento de las intenciones.

Las 5 intenciones fallidas más frecuentes

Use este gráfico para identificar las cinco intenciones fallidas más frecuentes de su bot (consulte Intenciones para ver la definición de intención fallida). Coloque el cursor sobre una barra para ver el número de veces que su bot falló esa intención, como se muestra en la siguiente imagen.



Seleccione Ver intenciones fallidas para ir a la subsección de Rendimiento de las intenciones del Panel de rendimiento, donde puede ver las métricas de las intenciones fallidas de su bot. Para obtener más información, consulte Rendimiento de las intenciones.

Panel de conversaciones: un resumen de las conversaciones de su bot

El panel de conversaciones visualiza las métricas de las conversaciones de los clientes (consulte Conversaciones para ver la definición de conversación) con su bot.

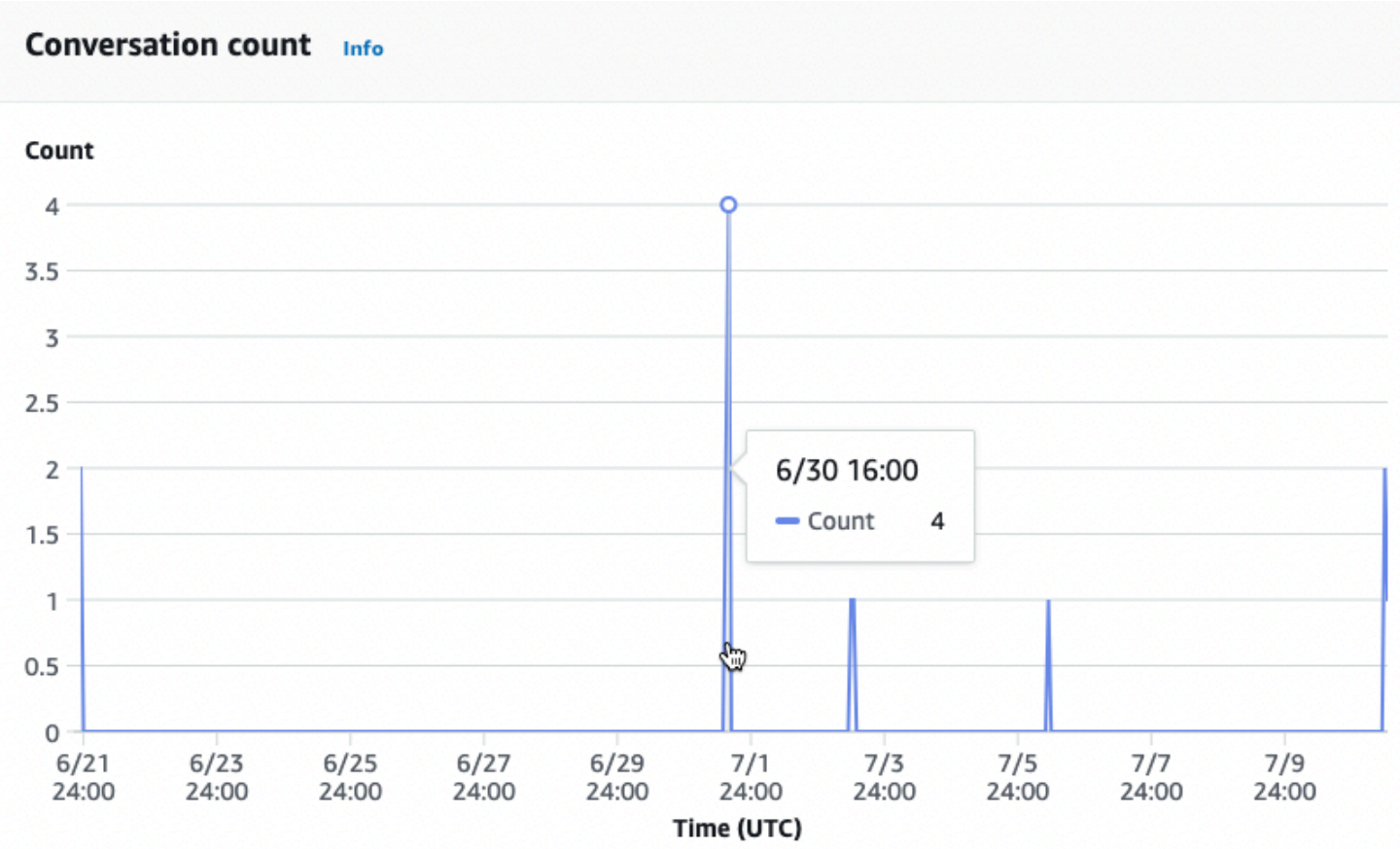
El Resumen contiene la siguiente información sobre las conversaciones de los usuarios con su bot. Las cifras se calculan en función de la configuración del filtro.

- Total de conversaciones: el número total de conversaciones con el bot.
- Duración media de las conversaciones: tiempo medio de las conversaciones de los usuarios con el bot en minutos y segundos. El formato es mm:ss.
- Media de turnos por conversación: número medio de turnos que dura una conversación.

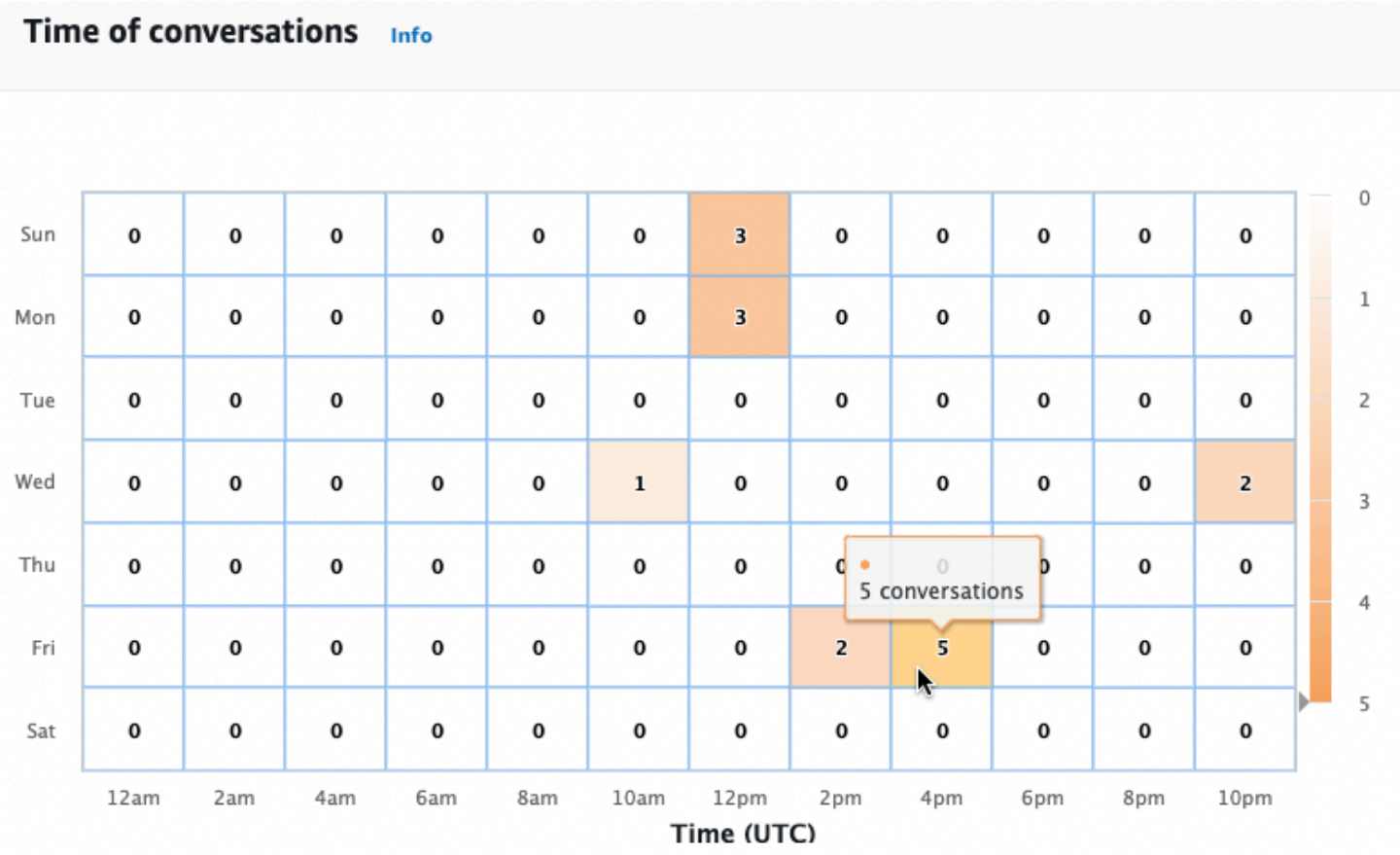
Las secciones Recuento de conversaciones y Recuento de mensajes contiene un gráfico que muestra el número de conversaciones y mensajes, respectivamente, en el intervalo de tiempo que especifique en los filtros. Coloque el cursor sobre un segmento de tiempo para ver el número de conversaciones o mensajes en ese segmento. El tamaño del segmento de tiempo depende del intervalo de tiempo que especifique:

- Menos de 1 semana: el recuento se muestra para cada hora.
- Una semana o más: el recuento se muestra para cada día.

En la imagen siguiente se muestra un ejemplo del comportamiento al pasar el cursor.



La sección Tiempo de las conversaciones presenta el número de conversaciones que tuvieron lugar entre su bot y los clientes en cada intervalo de dos horas de cada día de la semana, dentro del intervalo de tiempo que especifique en los filtros. Las celdas con sombras más oscuras indican los momentos en los que tuvieron lugar más conversaciones. Coloque el cursor sobre una celda para ver el número de conversaciones en las dos horas que comenzaron en esa franja horaria. Por ejemplo, la acción de la siguiente imagen muestra el número de conversaciones que tienen lugar entre las 16:00 y las 18:00 (UTC).



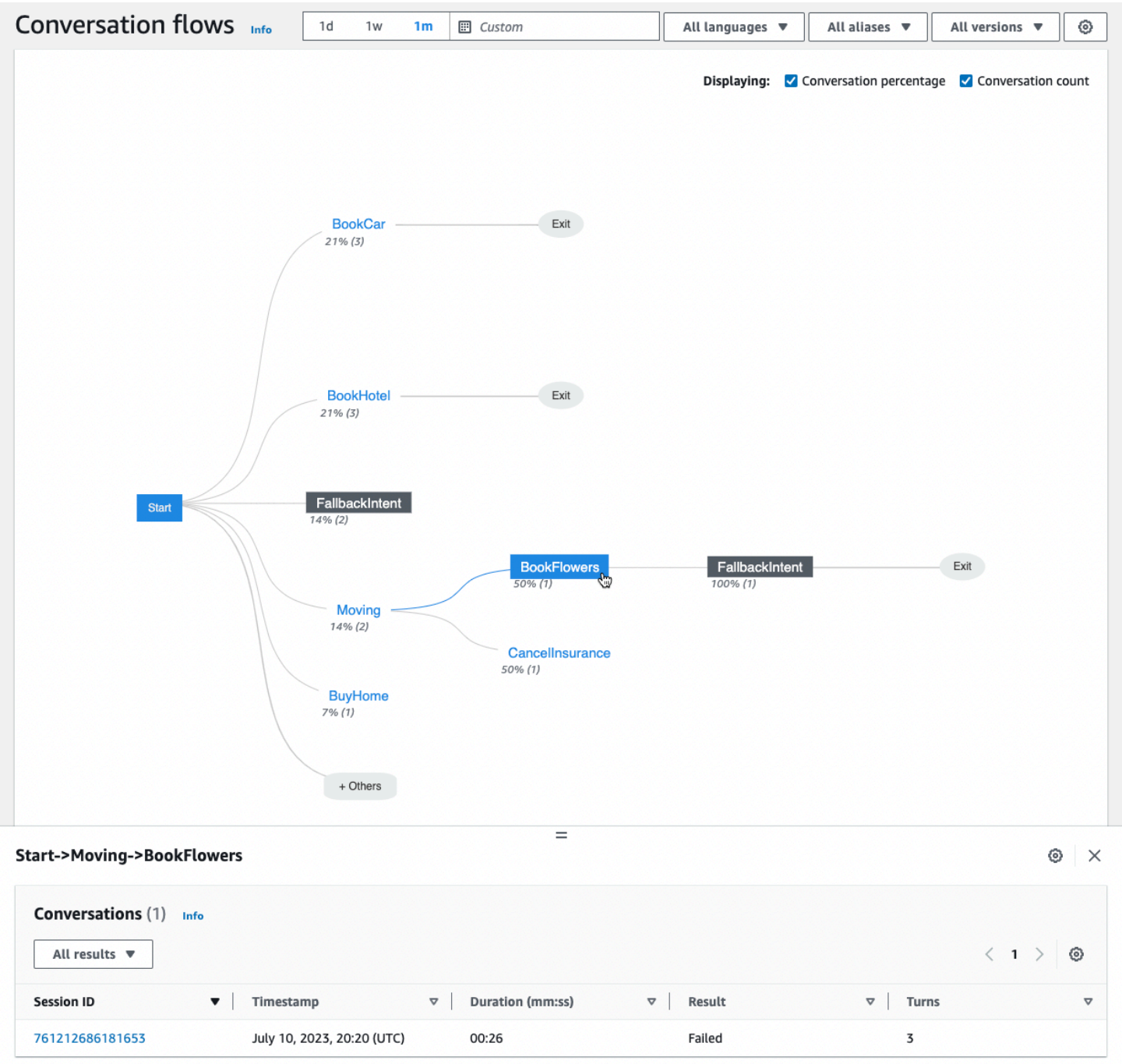
El Panel de conversaciones contiene dos herramientas: los Flujos de conversación y las Conversaciones. Para acceder a una herramienta, selecciónela en el Panel de conversaciones, en el panel de navegación izquierdo.

Flujos de conversación

Use los Flujos de conversación para visualizar el orden de intenciones que toman los clientes en las conversaciones con su bot. Debajo de cada intención se muestra el porcentaje y el recuento de conversaciones que invocaron esa intención en ese momento de la conversación. Para cambiar el porcentaje y la cuenta regresiva, seleccione Porcentaje de conversaciones y Recuento de conversaciones en la parte superior. De forma predeterminada, las cinco intenciones más comunes en ese momento de la conversación se muestran en orden descendente según su frecuencia. Seleccione + Otros para mostrar todas las intenciones.

Seleccione una intención para expandirla a una nueva columna de ramas que muestre una lista de las intenciones tomadas en ese momento de la conversación, ordenadas por frecuencia descendente.

Al seleccionar un nodo en el flujo de la conversación, puede expandir la ventana de abajo para mostrar una lista de las conversaciones que siguieron ese orden de intención. Seleccione el ID de sesión correspondiente a una conversación para ver los detalles de esa conversación. La siguiente imagen muestra un flujo de conversación y una ventana de Conversaciones ampliada en la parte inferior.



Conversaciones

La herramienta Conversaciones muestra una lista de las conversaciones de su bot. Puede seleccionar una columna para ordenar por esa columna en orden ascendente o descendente.

Para filtrar las conversaciones por resultado, seleccione Todos los resultados y seleccione Exitosa, Fallida o Descartada.

Para filtrar las conversaciones por duración

1. Seleccione la barra de búsqueda marcada como Filtrar conversaciones por duración
2. Defina el filtro de una de las siguientes formas:
   - Utilice las opciones predefinidas.
     a. Seleccione Duración.
     b. Seleccione entre los operadores = (igual a), > (mayor que) y < (menor que).
     c. Seleccione un período de tiempo.
   - Introduzca una entrada con el formato «Duración {operador} {número} segundos». Por ejemplo, para buscar todas las conversaciones que duren más de 30 segundos, introduzca **`Duration > 30 sec`**. Especifique el período de tiempo en segundos.

Para ver información detallada sobre la sesión, incluidos los metadatos, la intención de uso y una transcripción, seleccione el ID de sesión de una conversación.

> **Note**
>
> Como una conversación es una combinación única de una `sessionId` y una `originatingRequestId`, la misma `sessionId` puede aparecer varias veces en la tabla.

La página Detalles contiene los siguientes metadatos:

- Marca de tiempo: especifica la fecha y la hora de inicio de la conversación. La hora está en formato hh:mm:ss.
- Duración: especifica la duración de la conversación en formato mm:ss. La duración no incluye el tiempo de espera de la sesión (`idleSessionTTLInSeconds`).

- Resultado: especifica si la conversación se clasificó como exitosa, fallida o descartada. Para obtener detalles sobre estos resultados, consulte Conversaciones.
- Modo: especifica si la conversación fue `Speech`, `Text`, o `DTMF` (pulsaciones de teclado por tonos). Una conversación que consta de varios modos es `Multimode`.
- Canal: especifica el canal en el que tuvo lugar la conversación, si corresponde. Consulte Integrar un bot de Amazon Lex V2 con una plataforma de mensajería.
- Idioma: especifica el idioma del bot.

Las intenciones que el bot suscitó en la conversación se muestran en Detalles. Seleccione Ir a la intención para ir a esa intención en el editor de intenciones. Seleccione Ajustar a la transcripción para desplazar automáticamente la Transcripción hasta la primera instancia en la que el bot haya suscitado la intención.

Seleccione la flecha de la derecha situada junto al nombre de la intención para ver los detalles sobre los slots obtenidos para esa intención, incluidos los nombres de los slots, el valor que el bot obtuvo para cada slot y el número de veces que el bot intentó obtener cada slot.

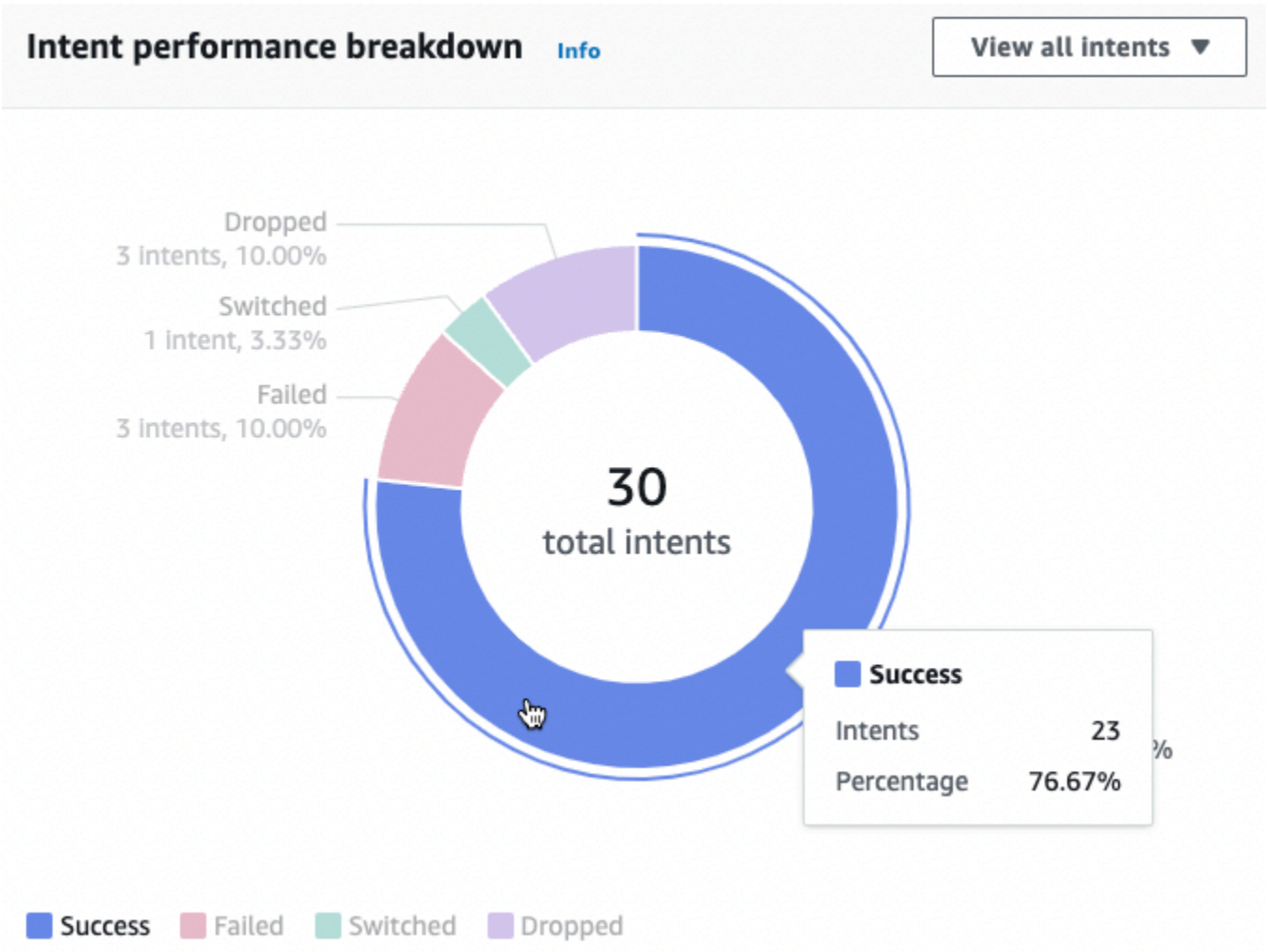
La Transcripción le permite revisar los enunciados de la conversación y el comportamiento de su bot al obtener las intenciones y los slots. Los enunciados de los usuarios se muestran a la izquierda y las del bot, a la derecha. Use la barra de búsqueda marcada como Filtrar transcripciones en esta sesión para buscar texto en la transcripción. Junto a Mostrar: debajo de cada turno de conversación se muestran tres datos que puede seleccionar para mostrarlos o no:

- Marca de tiempo: especifica la hora del enunciado.
- Estado de la intención: especifica la intención que suscita el bot durante un enunciado y el resultado de la intención, si corresponde. A continuación, se muestran los posibles estados de la intención:
  - Intención invocada: *`intent name`* — El bot identificó una intención que el cliente está invocando.
  - Intención cambiada: *`intent name`* — El bot ha cambiado a una intención diferente en función de la expresión.
  - *`intent name`*: Éxito: el bot ha cumplido la intención.
- Estado del slot: especifica el slot que suscita el bot durante un enunciado, si corresponde, y el valor que proporciona el cliente.

Panel de rendimiento: un resumen de las métricas de intenciones y enunciados de su bot

En el panel de rendimiento, puede ver los detalles sobre el rendimiento de su bot a la hora de cumplir las intenciones y reconocer los enunciados.

La sección de desglose del rendimiento de las intenciones muestra el número total de veces que su bot invocó una intención y desglosa el número y el porcentaje de veces que las intenciones se clasificaron como exitosas, fallidas, descartadas o cambiadas. Consulte Intenciones para obtener una explicación de estas definiciones. Coloque el cursor sobre un segmento del gráfico para mostrar un cuadro con el recuento y el porcentaje de conversaciones con ese resultado, como en la imagen siguiente.



Seleccione Ver todas las intenciones para que aparezca un menú desplegable en el que puede ver una lista de las intenciones suscitadas por el bot. También puede elegir ver las intenciones con un resultado específico (exitosa, fallida, descartada o cambiada). Estos enlaces lo llevan a la subsección

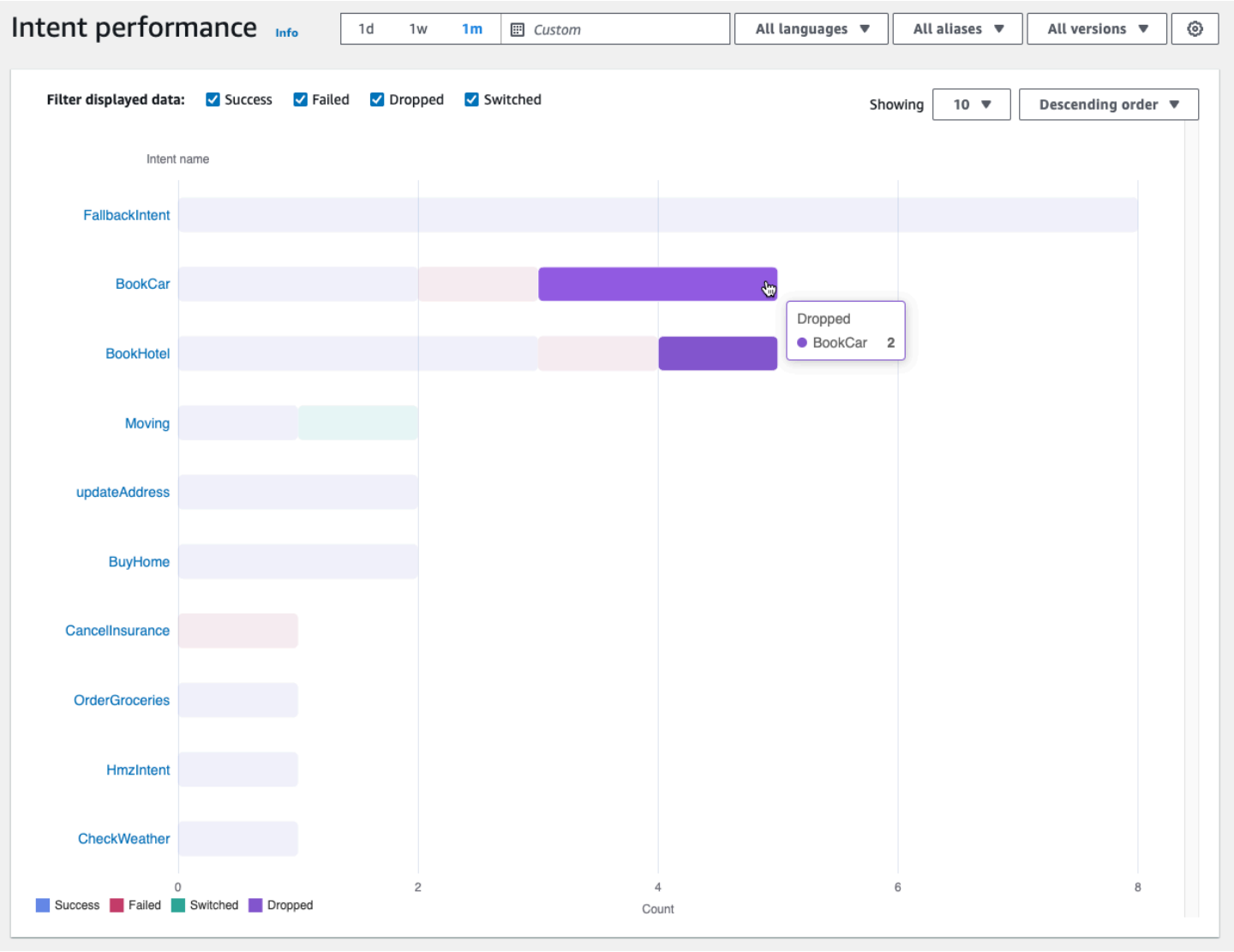
de Reconocimiento de enunciados del Panel de rendimiento. Para obtener más información, consulte Rendimiento de las intenciones.

La sección de Reconocimiento de enunciados resume la cantidad de enunciados que se perdieron y se detectaron. Seleccione Ver detalles para acceder a una lista de enunciados del bot. Seleccione el número que aparece en Enunciados perdidos para ver una lista de los enunciados que no se reconocieron y el número que aparece en Enunciados detectados para ver una lista de los enunciados detectados por el bot. Para obtener más información, consulte Reconocimiento de enunciados.

Seleccione Rendimiento de las intenciones y Reconocimiento de enunciados en el Panel de rendimiento de la barra lateral izquierda para ver los detalles sobre las intenciones y los enunciados de su bot.

Rendimiento de las intenciones

Este panel resume el rendimiento de las intenciones utilizadas con su bot en orden descendente según su frecuencia. La barra situada junto a cada intención muestra el número de veces que la intención se clasificó como exitosa, fallida, descartada o cambiada. Consulte Intenciones para obtener una explicación de estas definiciones. Coloque el cursor sobre un segmento de la barra para ver el número de conversaciones en las que se utilizó esa intención y se obtuvo ese resultado, como se muestra en la siguiente imagen:



> **Note**
>
> El panel de control muestra los 1000 resultados principales de un conjunto de ajustes de filtro. Para obtener resultados más específicos, configure los ajustes del filtro granular.

En la parte superior del gráfico, puede cambiar los estados de la intención que desee ver con las casillas Exitosa, Fallida, Descartada y Cambiada.

Seleccione los menús desplegables situados a la derecha de Mostrar para ajustar el número de intentos que se van a mostrar y si se deben mostrar en orden de frecuencia ascendente o descendente.

Seleccione un nombre de intención para ir a una página que muestra tres gráficos: el Desglose del rendimiento de las intenciones, Rendimiento de los slots y Cambios de intención.

La sección de Desglose del rendimiento de las intenciones muestra el número total de veces que el bot utilizó la intención y desglosa el número y el porcentaje de veces que las intenciones se clasificaron como exitosas, fallidas, descartadas o cambiadas. Consulte Intenciones para obtener una explicación de estas definiciones. Coloque el cursor sobre un segmento del gráfico para ver el recuento y el porcentaje de veces que el cumplimiento de la intención arrojó ese resultado.

La sección de Rendimiento de los slots muestra las métricas de los slots que pertenecen a la intención actual. Para ordenar por columna, seleccione esa columna una vez para ordenarla en orden ascendente y dos veces para ordenarla en orden descendente. Puede usar la barra de búsqueda para encontrar un slot específico o usar los botones de números de página para navegar por los slots.

> **Note**
>
> El panel de control muestra los 1000 resultados principales de un conjunto de ajustes de filtro. Para obtener resultados más específicos, configure los ajustes del filtro granular.

La sección Cambios de intención muestra los casos en los que el bot ha cambiado de la intención actual a otra con la siguiente información:

- Etapa: la etapa de la conversación en la que el bot cambió la intención.
- Intención cambiada a: la intención a la que el bot cambió la intención actual.
- Recuento de sesiones: el número de sesiones en las que se ha producido la combinación de Etapa y Intención cambiada a.

> **Note**
>
> El panel de control muestra los 1000 resultados principales de un conjunto de ajustes de filtro. Para obtener resultados más específicos, configure los ajustes del filtro granular.

Reconocimiento de enunciados

En esta página se enumeran todos los enunciados que su bot no ha reconocido y que ha detectado, y le proporciona herramientas para añadir ejemplos de enunciados a las intenciones para ayudar a entrenarlo. Consulte Enunciados para obtener una explicación de estas definiciones. Utilice las pestañas de la parte superior para cambiar entre una lista de Enunciados perdidos y una de Enunciados detectados.

> **Note**
>
> El panel de control muestra los 1000 resultados principales de un conjunto de ajustes de filtro. Para obtener resultados más específicos, configure los ajustes del filtro granular.

Para añadir enunciados a una intención:

1. Seleccione la casilla de verificación situada junto a los enunciados que quiera añadir como ejemplos de enunciados para una intención.
2. Seleccione Añadir a la intención y seleccione la intención a la que quiere añadir los enunciados en el menú desplegable, debajo de la Intención.
3. Seleccione Agregar.

Uso de API para análisis

En esta sección se describen las operaciones de la API que se utilizan para recuperar los análisis de un bot.

> **Note**
>
> Para usar ListUtteranceMetricsy ListUtteranceAnalyticsData, tu función de IAM debe tener permisos para realizar la ListAggregatedUtterancesoperación, lo que te da acceso a los análisis relacionados con las expresiones. Consulte Visualización de las estadísticas de enunciado de las conversaciones de Lex V2 para obtener información detallada y la política de IAM que se debe aplicar al rol de IAM.

- Las siguientes operaciones de la API recuperan las métricas resumidas de un bot:

  - ListSessionMetrics
  - ListIntentMetrics
  - ListIntentStageMetrics
  - ListUtteranceMetrics
- Las siguientes operaciones de la API recuperan una lista de metadatos para sesiones y enunciados:
  - ListSessionAnalyticsData
  - ListUtteranceAnalyticsData
- La ListIntentPathsoperación recupera métricas sobre el orden de intenciones que los clientes adoptan cuando conversan con un bot.

Filtrado de resultados

Las solicitudes de la API de Analytics requieren que especifique el `startTime` y el `endTime`. La API devuelve las sesiones, las intenciones, las etapas de intención o los enunciados que comenzaron después del `startTime` y finalizaron antes del `endTime`.

`filters` es un campo opcional en las solicitudes de la API de Analytics. Se asigna a una lista de AnalyticsSessionFilter, AnalyticsIntentFilterAnalyticsIntentStageFilter, u AnalyticsUtteranceFilterobjetos. En cada objeto, utilice los campos para crear una expresión por la que filtrar. Por ejemplo, si añade el siguiente filtro a la lista, el bot busca conversaciones que duren más de 30 segundos.

```
{
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

Recuperar las métricas de un bot

Utilice las operaciones `ListSessionMetrics`, `ListIntentMetrics`, `ListIntentStageMetrics` y `ListUtteranceMetrics` para recuperar métricas resumidas de las sesiones, las intenciones, las etapas de intención y los enunciados.

Para estas operaciones, rellene los siguientes campos obligatorios:

- Proporcione un `startTime` y un `endTime` para definir un intervalo de tiempo para el que desee recuperar los resultados.
- Especifique las métricas que desea calcular en `metrics` una lista de AnalyticsSessionMetric, AnalyticsIntentMetricAnalyticsIntentStageMetric, u AnalyticsUtteranceMetricobjetos. En cada objeto, utilice el campo `name` para especificar la métrica para calcular el campo `statistic` y especificar si se debe calcular el número `Sum`, `Average` o `Max`, y el campo `order` para especificar si se deben ordenar los resultados en un orden `Ascending` o `Descending`.

> **Note**
>
> Tanto los objetos `metrics` como `binBy` contienen un campo `order`. Puede especificar la clasificación `order` solo en uno de los dos objetos.

El resto de los campos de la solicitud son opcionales. Puede filtrar y organizar los resultados de las siguientes maneras:

- Filtrar los resultados: utilice el campo `filters` para filtrar los resultados. Consulte Filtrado de resultados para obtener más detalles.
- Agrupar los resultados por categoría: especifique el `groupBy` campo, una lista que contenga un único AnalyticsSessionResult, AnalyticsIntentResultAnalyticsIntentStageResult, o un AnalyticsUtteranceResultobjeto. En el objeto, especifique el campo `name` con la categoría por la que desee agrupar los resultados.

  Si especifica un `groupBy` campo en la solicitud, el `results` objeto de la respuesta contendrá `groupByKeys` una lista de AnalyticsSessionGroupByKey, AnalyticsIntentGroupByKey, u AnalyticsUtteranceGroupByKeyobjetos AnalyticsIntentStageGroupByKey, cada uno con los `name` que especificó en la solicitud y un miembro de esa categoría en el `value` campo.
- Clasificar los resultados por tiempo: especifique el `binBy` campo, una lista que contenga un único AnalyticsBinBySpecificationobjeto. En el objeto, especifique el campo `name` con `ConversationStartTime` para agrupar los resultados según el momento en que se inició la conversación o `UtteranceTimestamp` para agrupar los resultados según el momento en que tuvo lugar el enunciado. Especifique el intervalo de tiempo en el que desea agrupar los resultados en el campo `interval` y si desea ordenarlos por orden `Ascending` o `Descending` de tiempo en el campo `order`.

Si especificas un `binBy` campo en la solicitud, el `results` objeto de la respuesta contiene `binKeys` una lista de AnalyticsBinKeyobjetos, cada uno con los `name` que especificaste en la solicitud y el intervalo de tiempo que define ese intervalo en el `value` campo.

> **Note**
>
> Tanto los objetos `metrics` como `binBy` contienen un campo `order`. Puede especificar la clasificación `order` solo en uno de los dos objetos.

Utilice los siguientes campos para gestionar la visualización de la respuesta:

- Especifique un número entre 1 y 1000 en el campo `maxResults` para limitar el número de resultados que se devolverán en una sola respuesta.
- Si el número de resultados es mayor que el número que especificó en el campo `maxResults`, la respuesta contiene un `nextToken`. Vuelva a realizar la solicitud, pero utilice este valor en el campo `nextToken` para devolver el siguiente lote de resultados.

Si está utilizando `ListUtteranceMetrics`, puede especificar los atributos que se devolverán en el campo `attributes`. Este campo se asigna a una lista que contiene un único AnalyticsUtteranceAttributeobjeto. Especifique `LastUsedIntent` en el campo `name` para devolver la intención que utiliza Amazon Lex V2 en el momento del enunciado.

En la respuesta, el `results` campo se asigna a una lista de AnalyticsSessionResultAnalyticsIntentResult, AnalyticsIntentStageResult, u AnalyticsUtteranceResultobjetos. Cada objeto contiene un campo `metrics` que devuelve el valor de una estadística resumida de una métrica que haya solicitado, además de los contenedores o grupos creados a partir de los métodos que haya especificado.

Recuperar los metadatos de las sesiones y los enunciados de un bot

Utilice las ListUtteranceAnalyticsDataoperaciones ListSessionAnalyticsDatay para recuperar metadatos sobre sesiones y expresiones individuales.

Rellene los campos `startTime` y `endTime` obligatorios para definir el intervalo de tiempo para el que desee recuperar los resultados.

El resto de los campos de la solicitud son opcionales. Para filtrar y ordenar los resultados:

- Filtrar los resultados: utilice el campo `filters` para filtrar los resultados. Consulte Filtrado de resultados para obtener más detalles.
- Ordenar los resultados: ordena los resultados con el `sortBy` campo, que contiene un UtteranceDataSortByobjeto SessionDataSortBy o. Especifique el valor por el que desea ordenarlos en el campo `name` y si desea ordenarlos de forma `Ascending` o `Descending` en el campo `order`.

Utilice los siguientes campos para gestionar la visualización de la respuesta:

- Especifique un número entre 1 y 1000 en el campo `maxResults` para limitar el número de resultados que se devolverán en una sola respuesta.
- Si el número de resultados es mayor que el número que especificó en el campo `maxResults`, la respuesta contiene un `nextToken`. Vuelva a realizar la solicitud, pero utilice este valor en el campo `nextToken` para devolver el siguiente lote de resultados.

En la respuesta, el `utterances` campo `sessions` o se asigna a una lista de UtteranceSpecificationobjetos SessionSpecification o. Cada objeto contiene metadatos para una sola sesión o enunciado.

Recuperación de datos de análisis de la ruta de intención

Utilice la ListIntentPathsoperación para recuperar las métricas sobre el orden de intenciones que los clientes adoptan al conversar con un bot.

Para esta operación, rellene los siguientes campos obligatorios:

- Proporcione un `startTime` y un `endTime` para definir un intervalo de tiempo para el que desee recuperar los resultados.
- Proporcione un `intentPath` para definir un orden de intenciones para el que desea recuperar métricas. Separe las intenciones de la ruta con una barra diagonal. Por ejemplo, rellene el campo `intentPath` con **`/BookCar/BookHotel`** para ver información detallada sobre el número de veces que los usuarios han invocado las intenciones `BookCar` y `BookHotel` en ese orden.

Use el campo `filters` opcional para filtrar los resultados. Para obtener más información, consulte Filtrado de resultados.

Visualización de las estadísticas de enunciado de las conversaciones de Lex V2

Puede utilizar las estadísticas de enunciados para determinar los enunciados que los usuarios envían a su bot. Puede ver tanto los enunciados que Amazon Lex V2 detecta correctamente como los que no reconoce. Puede utilizar esta información para ajustar su bot.

Por ejemplo, si descubre que sus usuarios envían un enunciado que falta en Amazon Lex V2, puede añadir el enunciado a una intención. La versión preliminar de la intención se actualiza con el nuevo enunciado y puede probarlo antes de implementarlo en su bot.

Se detecta un enunciado cuando Amazon Lex V2 reconoce el enunciado como un intento de invocar una intención configurada para un bot. Se pierde un enunciado cuando Amazon Lex V2 no lo reconoce y, en su lugar, invoca `AMAZON.FallbackIntent`.

Las estadísticas del enunciado se pueden consultar mediante la API de `ListUtteranceMetrics` y la API de `ListAggregatedUtterance`.

Las estadísticas de los enunciados no se generan mediante la API de `ListUtteranceMetrics` en las siguientes condiciones:

- La configuración de la Ley de Protección de la Privacidad en Línea para Niños se estableció como Sí cuando se creó el bot con la consola, o el campo `childDirected` se estableció en verdadero cuando se creó el bot con la operación `CreateBot`.

La API de `ListUtteranceMetrics` ofrece funciones adicionales, entre las que se incluyen:

- Hay más información disponible, como la intención mapeada de los enunciados detectados.
- Más capacidad de filtrado (incluidos el canal y el modo).
- Intervalo de fechas de retención más prolongado (30 días).
- Puede usar la API incluso si ha optado por no almacenar datos. La funcionalidad de la consola para los enunciados perdidos o detectados dependerá de la API de `ListUtteranceMetrics`.

Las estadísticas de los enunciados no se generan mediante la API de `ListAggregatedUtterance` en las siguientes condiciones:

- La configuración de la Ley de Protección de la Privacidad en Línea para Niños se estableció como Sí cuando se creó el bot con la consola, o el campo `childDirected` se estableció en verdadero cuando se creó el bot con la operación `CreateBot`.

- Está utilizando la ofuscación de slots con uno o más slots.
- Ha optado por no participar en la mejora de Amazon Lex.

La API de `ListAggregatedUtterance` ofrece funciones como:

- Menos información detallada disponible (sin la intención mapeada de los enunciados).
- Capacidad de filtrado limitada (sin incluir el canal y el modo).
- Intervalo de fechas de retención más corto (15 días).

Con las estadísticas de enunciados, puede ver si se detectó o no se detectó un enunciado específico, así como la última vez que se utilizó el enunciado en una interacción con un bot.

Amazon Lex V2 almacena los enunciados continuamente mientras los usuarios interactúan con su bot. Puede consultar las estadísticas mediante la consola o la operación `ListAggregatedUtterances`. Tiene una retención de datos de 15 días y no está disponible si el usuario ha optado por no almacenarlos. Puede eliminar los enunciados mediante la operación `DeleteUtterances` u optando por no almacenar datos. Si cierra su cuenta de AWS, se eliminan todos los enunciados. Los enunciados almacenados se cifran con una clave gestionada por el servidor.

Cuando elimina una versión de bot, las estadísticas de enunciados de la versión estarán disponibles durante un máximo de 30 días con `ListUtteranceMetrics`, y 15 días usando `ListAggregatedUtterances`. No puede ver las estadísticas de la versión eliminada en la consola de Amazon Lex V2. Para ver las estadísticas de las versiones eliminadas, puede usar ambas operaciones `ListAggregatedUtterances` y `ListUtteranceMetrics`.

Con las API de `ListAggregatedUtterances` y `ListUtteranceMetrics`, los enunciados se agregan por el texto del enunciado. Por ejemplo, todas las instancias en las que el cliente utilizó la frase «Quiero pedir una pizza» se agrupan en la misma línea en una respuesta. Al utilizar la RecognizeUtteranceoperación, el texto utilizado es la transcripción de entrada.

Para usar las API de `ListAggregatedUtterances` y `ListUtteranceMetrics`, aplique la siguiente política a un rol.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
            "Resource": "*"
        }
    ]
}
```

Administración de permisos de acceso para análisis

Para proporcionar acceso a los análisis a un usuario, adjunte una política a un rol de IAM que le permita llamar a las operaciones de la API para realizar análisis. Puede asociar el Política gestionada por AWS: AmazonLexFullAccess al rol de IAM para proporcionar acceso completo a las operaciones de la API de Amazon Lex, o puede crear una política personalizada que conceda únicamente los permisos de análisis y asociarla a un rol de IAM.

Crear una política personalizada que contenga permisos para el análisis

1. Si primer necesita crear un rol de IAM, siga los pasos descritos en Creación de un rol para delegar permisos a un usuario de IAM.
2. Siga los pasos que se indican en Creación de políticas de IAM para crear una política con el siguiente objeto JSON. Para permitir el acceso analítico a bots específicos para el rol de IAM, añada el ARN de cada bot al campo `Resource`. Sustituya los valores *`regionaccount-id`*, y *`BOTID`* por los correspondientes a los bots. También puede sustituir el identificador de la sentencia*`AnalyticsActions`*, por el nombre que prefiera.
3. Adjunte la política que creó al rol al que desea conceder los permisos de análisis siguiendo los pasos que se indican en Añadir y eliminar permisos de identidad de IAM.
4. El rol ahora debería tener permisos para ver los análisis de los bots que especificó.

Cómo habilitar registros de conversaciones para sus bots Lex V2

Use los registros de conversaciones para almacenar las conversaciones de los usuarios con su bot. Revise estos registros para identificar problemas en las interacciones de su bot con los usuarios

y modifique el comportamiento del bot con esta información. En esta sección también se describe cómo ocultar los valores de los slots para proteger la privacidad de los usuarios.

Temas

- Registro de conversaciones con registros de conversaciones en Lex V2
- Cómo ocultar valores de ranura en registros de conversación de Lex V2
- Captura selectiva del registro de conversaciones en Lex V2

Registro de conversaciones con registros de conversaciones en Lex V2

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 RecognizeText. Puede registrar texto y audio para la operación RecognizeUtterance. 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 los CloudWatch registros. 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 UpdateBotAliasoperación CreateBotAliaso. Después de habilitar los registros de conversaciones para un alias, al usar la RecognizeUtteranceoperación RecognizeTexto para ese alias, se registran las expresiones de texto o audio en el grupo de registros de CloudWatch registros o en el depósito de S3 configurado.

Temas

- Políticas de IAM para registros de conversación
- Configuración de registros de conversaciones para su bot Lex V2
- Visualización de registros de texto en Amazon CloudWatch Logs desde Lex V2
- Acceder a los registros de audio en Amazon S3
- Supervisar el estado del registro de conversaciones con CloudWatch métricas

Políticas de IAM para registros de conversación

Según el tipo de registro que seleccione, Amazon Lex V2 necesitará permiso para usar Amazon CloudWatch Logs y los depósitos de Amazon Simple Storage Service (S3) para almacenar sus registros. Debe crear funciones y permisos de AWS Identity and Access Management para que Amazon Lex V2 pueda acceder a estos recursos.

Crear un rol de IAM y políticas para registros de conversación

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

Puede utilizar la consola de IAM, la API de IAM o la AWS Command Line Interface para crear el rol y las políticas. 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 (^).

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 V2 como entidad de confianza para el rol. Esto permite a Amazon Lex V2 asumir la función de entregar registros a los recursos configurados para los registros de conversaciones.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

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

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

A continuación, cree y adjunte una política al rol que permita a Amazon Lex V2 escribir en CloudWatch los registros.

Para crear una política de IAM para registrar el texto de una conversación en Logs CloudWatch

1. Cree un documento en el directorio actual llamado **`LexConversationLogsCloudWatchLogsPolicy.json`**, agregue la siguiente política de IAM y guárdelo.
2. En la AWS CLI, cree la política de IAM que concede permisos de escritura al grupo de CloudWatch registros Logs.

```
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 V2 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.

   JSON

```
{
  "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 conceda permisos 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

Si utiliza la consola AWS Command Line Interface, el o un SDK de AWS para especificar una función de IAM que se va a utilizar en los registros de conversaciones, el usuario que especifique la función de IAM de los registros de conversaciones debe tener permiso para transferir la función a Amazon Lex V2. Para permitir que el usuario pase el rol a Amazon Lex V2, debe conceder permiso `PassRole` al usuario, rol o grupo de IAM del usuario.

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 Otorgar permisos a un usuario para transferir un rol a un servicio de AWS y a un IAM y claves de contexto de AWS STS condición en la Guía del usuario de AWS Identity and Access Management.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lexv2.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN":
 "arn:aws:lex:region:123456789012:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

Configuración de registros de conversaciones para su bot Lex V2

Los registros de conversación se habilitan y deshabilitan mediante la consola o el campo `conversationLogSettings` de la operación `CreateBotAlias` o `UpdateBotAlias`. 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 CloudWatch registros de Amazon Logs 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 V2. Para obtener más información sobre la creación de un grupo de CloudWatch registros, consulte Trabajar con grupos de registros y transmisiones de CloudWatch registros en la Guía del usuario de Amazon Logs.

Para almacenar registros de audio, utilice un bucket de Amazon 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 V2. Para obtener más información acerca de la creación de un bucket de S3, consulte Crear un bucket en la Guía de introducción de Amazon Simple Storage Service.

Cuando gestiona los registros de conversaciones mediante la consola, la consola actualiza su rol de servicio para que tenga acceso al grupo de registros y al bucket de S3.

Si no está utilizando la consola, debe proporcionar un rol de IAM con políticas que permitan a Amazon Lex V2 escribir en el grupo de registro o en el bucket configurado. Si crea un rol vinculado a un servicio mediante el AWS Command Line Interface, debe añadir un sufijo personalizado al rol mediante la `custom-suffix` opción que se muestra en el siguiente ejemplo. Para obtener más información, consulte Crear un rol de IAM y políticas para registros de conversación.

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.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:

JSON

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

```
            "Resource": "arn:aws:iam::111122223333:role/role"
        }
    ]
}
```

Habilitar registros de conversación

Activar los registros mediante la consola

1. Abra la consola Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. En la lista, seleccione un bot.
3. En el menú izquierdo, seleccione Alias.
4. En la lista de alias, seleccione el alias para el que desea configurar los registros de conversación.
5. En la sección Registros de conversaciones, seleccione Administrar registros de conversaciones.
6. Para los registros de texto, selecciona Activar e introduce el nombre del grupo de CloudWatch registros de Amazon Logs.
7. Para los registros de audio, seleccione Habilitar e introduzca la información del bucket de S3.
8. Opcional. Para cifrar los registros de audio, elija la AWS KMS clave que se utilizará para el cifrado.
9. Seleccione Guardar para iniciar el registro de conversaciones. Si es necesario, Amazon Lex V2 actualizará su rol de servicio con permisos para acceder al grupo de CloudWatch registros y al bucket S3 seleccionado.

Como deshabilitar registros de conversación en Lex V2

Desactivar los registros mediante la consola

1. Abra la consola Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. En la lista, seleccione un bot.
3. En el menú izquierdo, seleccione Alias.
4. En la lista de alias, seleccione el alias para el que desea configurar los registros de conversación.
5. En la sección Registros de conversaciones, seleccione Administrar registros de conversaciones.
6. Deshabilite el registro de texto, el registro de audio o ambos para desactivar el registro.

7. Seleccione Guardar para detener el registro de conversaciones.

Visualización de registros de texto en Amazon CloudWatch Logs desde Lex V2

Amazon Lex V2 almacena los registros de texto de sus conversaciones en Amazon CloudWatch Logs. Para ver los registros, utilice la consola o la API de CloudWatch Logs. 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 Amazon CloudWatch Logs.

Ver los registros mediante la consola de Amazon Lex V2

1. Abra la consola Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. En la lista, seleccione un bot.
3. En el menú de la izquierda, selecciona Analytics y, a continuación, selecciona CloudWatch métricas.
4. Consulta las métricas de tu bot en la página de CloudWatch métricas.

También puedes usar la CloudWatch consola o la API para ver tus entradas de registro. Para buscar las entradas de registro, desplácese hasta el grupo de registros que configuró para el alias. Puede encontrar el prefijo de flujo de registro de sus registros en la consola Amazon Lex V2 o mediante la DescribeBotAliasoperación.

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:

versión-del-mensaje

Versión del esquema de mensajes.

bot

Detalles sobre el bot con el que interactúa el cliente.

mensajes

La respuesta que el bot envió al usuario.

Contexto del enunciado

Información sobre el procesamiento de este enunciado.

- `runtimeHints`—contexto de tiempo de ejecución utilizado para transcribir e interpretar la entrada del usuario. Para obtener más información, consulte Mejora del reconocimiento de los valores de las ranuras con sugerencias de tiempo de ejecución en la conversación.
- `slotElicitationStyle`—Estilo de obtención de slots utilizado para interpretar las entradas del usuario. Para obtener más información, consulte Captura de valores de ranura con estilos de ortografía durante la conversación.

Estado de la sesión

El estado actual de la conversación entre el usuario y el bot. Para obtener más información, consulte Descripción de las conversaciones de los bots de Amazon Lex V2.

Interpretaciones

Una lista de intenciones que Amazon Lex V2 determinó que podían satisfacer el enunciado del usuario. Uso de puntuaciones de confianza para mejorar la precisión de la conversación.

interpretationSource

Indica si Amazon Lex V2 o Amazon Bedrock resuelven una ranura. Valores: Lex | Bedrock

sessionId

El identificador de la sesión de usuario en la que se está manteniendo la conversación.

Transcripción de entrada

Una transcripción de la entrada del usuario.

- Para la entrada de texto, este es el texto que escribió el usuario. Para la entrada DTMF, esta es la clave que introdujo el usuario.
- En el caso de entrada de voz, este es el texto en el que Amazon Lex V2 convierte el enunciado del usuario para invocar una intención o llenar un slot.

Sin procesar InputTranscript

La transcripción sin procesar de la entrada del usuario antes de aplicar cualquier procesamiento de texto. Nota: El procesamiento de texto es solo para las configuraciones regionales en-US y en-GB.

Transcripciones

Una lista de posibles transcripciones de las entradas del usuario. Para obtener más información, consulte Uso de las puntuaciones de confianza en la transcripción de voz para mejorar las conversaciones con su bot Lex V2.

Transcripción sin procesar

Usar puntuaciones de confianza en la transcripción de voz. Para obtener más información, consulte Uso de las puntuaciones de confianza en la transcripción de voz para mejorar las conversaciones con su bot Lex V2.

Enunciado perdido

Indica si Amazon Lex V2 ha podido reconocer el enunciado del usuario.

ID de solicitud

Amazon Lex V2 generó el ID de solicitud para la entrada del usuario.

Marca de tiempo

La marca temporal de la entrada del usuario.

Anulación del desarrollador

Indica si el flujo de la conversación se actualizó mediante un enlace de código de diálogo. Para obtener más información sobre el uso de un enlace de código de diálogo, consulte Integración de una AWS Lambda función en su bot Amazon Lex V2.

Modo de entrada

Indica el tipo de campo de entrada. Puede ser audio, DTMF o texto.

Atributos de solicitud

Los atributos de solicitud utilizados al procesar la entrada del usuario.

Propiedades de audio

Si los registros de conversaciones de audio están habilitados y la entrada del usuario estaba en formato de audio, incluye la duración total de la entrada de audio, la duración de la voz y la duración del silencio en el audio. También incluye un enlace al archivo de audio.

BargeIn

Indica si la entrada del usuario interrumpió la respuesta anterior del bot.

Motivo de la respuesta

Motivo por el que se generó una respuesta. Puede ser uno de los siguientes:

- `UtteranceResponse` – respuesta a la entrada del usuario
- `StartTimeout` – respuesta generada por el servidor cuando el usuario no proporcionó información

- `StillWaitingResponse` – respuesta generada por el servidor cuando el usuario solicita al bot que espere
- `FulfillmentInitiated` – respuesta generada por el servidor de que el procesamiento está a punto de iniciarse
- `FulfillmentStartedResponse` – respuesta generada por el servidor en la que se indica que se ha iniciado el procesamiento
- `FulfillmentUpdateResponse` – respuesta periódica generada por el servidor mientras el procesamiento está en curso
- `FulfillmentCompletedResponse` – respuesta generada por el servidor cuando se completa el procesamiento.

Nombre de operación

La API utilizada para interactuar con el bot. Puede ser `PutSession`, `RecognizeText`, `RecognizeUtterance` o `StartConversation`.

```
{
    "message-version": "2.0",
    "bot": {
        "id": "string",
        "name": "string",
        "aliasId": "string",
        "aliasName": "string",
        "localeId": "string",
        "version": "string"
    },
    "messages": [
        {
            "contentType": "PlainText | SSML | CustomPayload | ImageResponseCard",
            "content": "string",
            "imageResponseCard": {
                "title": "string",
                "subtitle": "string",
                "imageUrl": "string",
                "buttonsList": [
                    {
                        "text": "string",
                        "value": "string"
                    }
```

```
                ]
            }
        }
    ],
    "utteranceContext": {
        "activeRuntimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
                            {
                                "phrase": "string"
                            }
                        ]
                    }
                }
            }
        },
        "slotElicitationStyle": "string"
    },
    "sessionState": {
        "dialogAction": {
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
            "slotToElicit": "string"
        },
        "intent": {
            "name": "string",
            "slots": {
                "string": {
                    "value": {
                       "interpretedValue": "string",
                       "originalValue": "string",
                       "resolvedValues": [ "string" ]
                    }
                 },
                "string": {
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "interpretedValue": "string",
                        "resolvedValues": [ "string" ]
```

```
                    },
                    "values": [
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        },
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        }
                    ]
                }
            },
            "kendraResponse": {
                // Only present when intent is KendraSearchIntent. For details, see
                // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                },
            "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
            "confirmationState": "Confirmed | Denied | None"
        },
        "originatingRequestId": "string",
        "sessionAttributes": {
            "string": "string"
        },
        "runtimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
                            {
                                "phrase": "string"
                            }
```

```
                        ]
                    }
                }
            }
        }
    },
   "dialogEventLogs": [
        {
   // only for conditional
     "conditionalEvaluationResult":[
      // all the branches until true

     {
     "conditionalBranchName": "string",
     "expressionString": "string",
     "evaluatedExpression": "string",
     "evaluationResult": "true | false"
     }
    ],
  "dialogCodeHookInvocationLabel": "string",
  "response": "string",
  "nextStep": {
        "dialogAction": {
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
            "slotToElicit": "string"
        },
       "intent": {
                          "name": "string",
           "slots": {
               }
        }
       }
    ]
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "nluConfidence": "string",
            "intent": {
                "name": "string",
                "slots": {
                    "string": {
                        "value": {
                            "originalValue": "string",
                            "interpretedValue": "string",
```

```
                            "resolvedValues": [ "string" ]
                        }
                    },
                    "string": {
                        "shape": "List",
                        "value": {
                            "interpretedValue": "string",
                            "originalValue": "string",
                            "resolvedValues": [ "string" ]
                        },
                        "values": [
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }
                            },
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }

                            }
                        ]
                    }
                },
                "kendraResponse": {
                    // Only present when intent is KendraSearchIntent. For details, see

                    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                    },
                "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
                "confirmationState": "Confirmed | Denied | None"
                },
            "sentimentResponse": {
                "sentiment": "string",
                "sentimentScore": {
                    "positive": "string",
```

```
                "negative": "string",
                "neutral": "string",
                "mixed": "string"
            }
        }
    }
],
"sessionId": "string",
"inputTranscript": "string",
"rawInputTranscript": "string",
"transcriptions": [
    {
        "transcription": "string",
        "rawTranscription": "string",
        "transcriptionConfidence": "number",
        },
        "resolvedContext": {
            "intent": "string"
        },
        "resolvedSlots": {
            "string": {
                "name": "slotName",
                "shape": "List",
                "value": {
                    "originalValue": "string",
                    "resolvedValues": [
                        "string"
                    ]
                }
            }
        }
    }
],
"missedUtterance": "bool",
"requestId": "string",
"timestamp": "string",
"developerOverride": "bool",
"inputMode": "DTMF | Speech | Text",
"requestAttributes": {
    "string": "string"
},
"audioProperties": {
    "contentType": "string",
    "s3Path": "string",
```

```
        "duration": {
            "total": "integer",
            "voice": "integer",
            "silence": "integer"
        }
    },
    "bargeIn": "string",
    "responseReason": "string",
    "operationName": "string"
}
```

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 la `messageVersion`. El código no debería devolver un error si hay nuevos campos.

Acceder a los registros de audio en Amazon S3

Amazon Lex V2 almacena registros de audio para sus conversaciones en un bucket de 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 V2 o en el campo `conversationLogSettings` en la respuesta de la operación `DescribeBotAlias`.

Supervisar el estado del registro de conversaciones con CloudWatch métricas

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

Amazon Lex V2 proporciona cuatro métricas en el espacio de nombres de `AWS/Lex` para los registros de conversación:

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

Las métricas de éxito muestran que Amazon Lex V2 ha escrito correctamente sus registros de audio o texto en sus destinos.

Las métricas de error muestran que Amazon Lex V2 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 V2 es una entidad de confianza para el rol de IAM.
- Para el registro de texto, asegúrate de que existe el grupo de CloudWatch registros. Para el registro de audio, asegúrese de que existe el bucket de S3.
- Asegúrese de que la función de IAM que Amazon Lex V2 utiliza para acceder al grupo de CloudWatch registros o al bucket de S3 tenga permiso de escritura para el grupo de registros o el bucket.
- Asegúrese de que el bucket de S3 existe en la misma región que el bot de Amazon Lex V2 y pertenece a su cuenta.

Cómo ocultar valores de ranura en registros de conversación de Lex V2

Amazon Lex V2 le permite ofuscar u ocultar el contenido de los slots para que el contenido no sea visible. Para proteger datos sensibles capturados como valores de slot, puede habilitar la ofuscación del slot para cubrir los valores para su registro.

Cuando elige ofuscar los valores de slot, Amazon Lex V2 reemplaza el valor del slot por el nombre del slot en los registros de conversación. Para un slot llamado `full_name`, el valor del slot se ofuscaría de la siguiente manera:

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

Si un enunciado contiene caracteres entre corchetes ({}), Amazon Lex V2 oculta los caracteres entre corchetes con dos barras invertidas (\\). Por ejemplo, el texto `{John Stiles}` se ofusca de la siguiente manera:

```
Before:
    My name is {John Stiles}
After:
    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 `RecognizeText` y `RecognizeUtterance`, y los valores de slot están disponibles para las funciones de Lambda de validación y cumplimiento. Si utiliza valores de slot en sus mensajes o respuestas, esos valores de slot no se ofuscan en los registros de conversación.

En el primer turno de una conversación, Amazon Lex V2 ofusca los valores de slot si reconoce un slot y un valor de slot en el enunciado. Si no se reconoce ningún valor de slot, Amazon Lex V2 no ofusca el enunciado.

En el segundo turno y los posteriores, Amazon Lex V2 sabe cuál es el slot que debe obtener y si el valor de slot debe ofuscarse. Si Amazon Lex V2 reconoce el valor del slot, el valor se ofusca. Si Amazon Lex V2 no reconoce un valor, se ofusca todo el enunciado. Los valores de slot en enunciados perdidos no se ofuscarán.

Amazon Lex V2 tampoco ofusca los valores de slot que almacena en atributos de solicitud o sesión. Si está almacenando valores de slot que deben ofuscarse como un atributo, debe cifrar u ofuscar el valor de otro modo.

Amazon Lex V2 no ofusca el valor del slot en el audio. Ofusca el valor del slot en la transcripción de audio.

Puede elegir qué slots se ofuscan usando la consola o usando la API de Amazon Lex V2. En la consola, seleccione Ofuscación de slot en la configuración de un slot. Si utilizas la API, establece el `obfuscationSetting` campo de la ranura en el que llamas a `DEFAULT_OBFUSCATION` la operación CreateSloto UpdateSlot.

Captura selectiva del registro de conversaciones en Lex V2

La captura selectiva del registro de conversaciones permite al usuario seleccionar cómo se capturan los registros de conversaciones con los datos de texto y audio de las conversaciones en directo.

Para habilitar y capturar el resultado de la función de captura selectiva del registro de conversaciones, debe activar la función en la consola de Amazon Lex V2 y habilitar los atributos de sesión necesarios en la configuración de la API para capturar el resultado seleccionado de los registros.

Puede seleccionar las siguientes opciones para la captura selectiva del registro de conversaciones:

- solo texto
- solo audio
- texto y audio

Puede capturar partes específicas de la conversación y elegir si desea capturar audio, texto o ambos para el registro de conversaciones.

> **Note**
>
> La captura selectiva del registro de conversaciones solo funciona para Amazon Lex V2.

Temas

- Gestionar la captura selectiva del registro de conversaciones

- Ejemplo de captura selectiva del registro de conversaciones

Gestionar la captura selectiva del registro de conversaciones

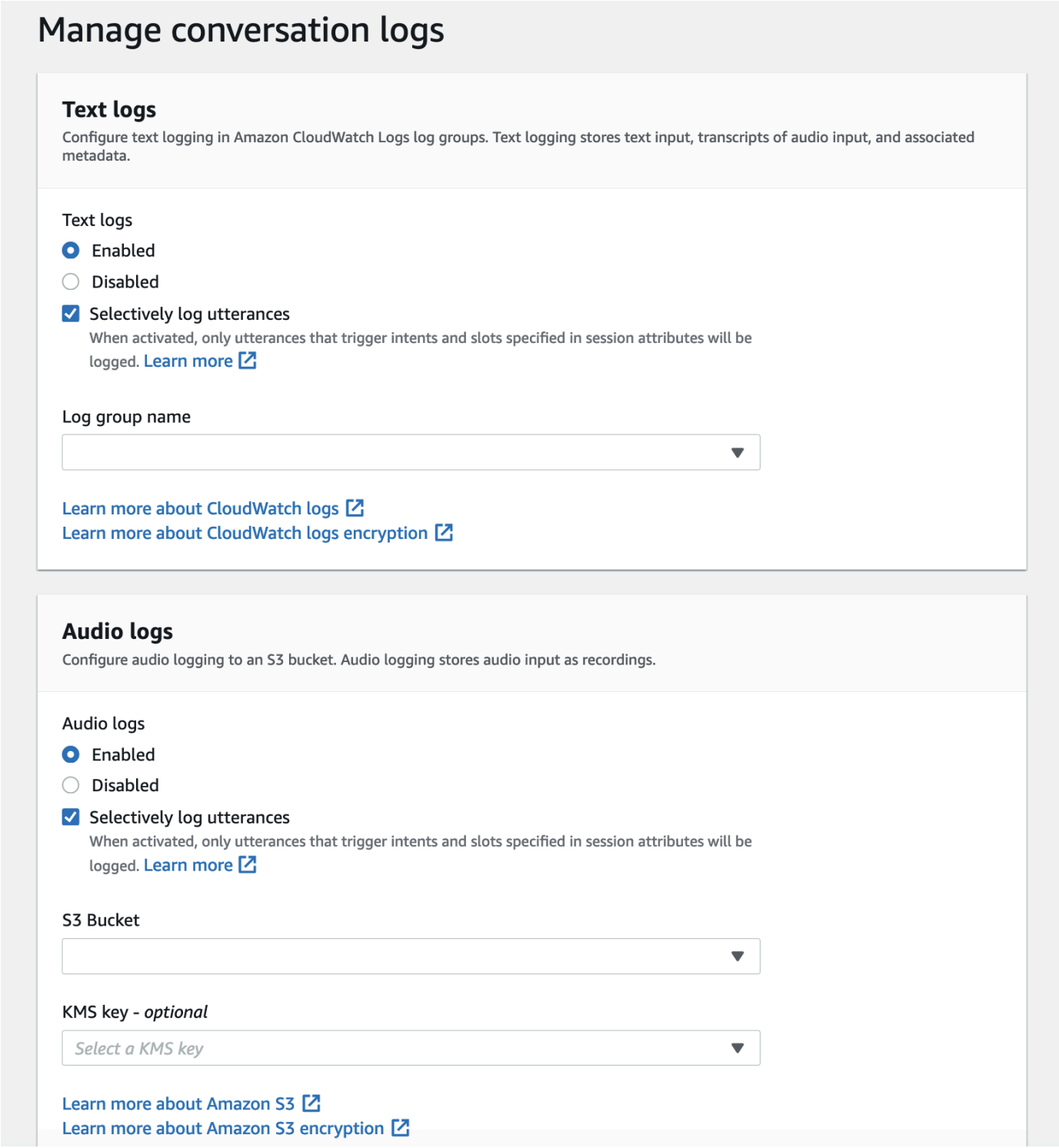
Con la consola de Lex, puede habilitar la configuración de captura selectiva del registro de conversaciones y elegir en qué slots desea habilitar la captura selectiva del registro de conversaciones.

Active la captura selectiva del registro de conversaciones en la consola de Amazon Lex V2:

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/home.
2. Seleccione Bots en los paneles laterales de la izquierda y seleccione el bot que desee para activar la captura selectiva del registro de conversaciones. Seleccione una API existente o cree una nueva.
3. Seleccione los alias para el bot seleccionado en la sección de Implementación del panel lateral izquierdo.
4. Seleccione el alias de su bot y, a continuación, seleccione Administrar registros de conversaciones.
5. En el panel Administrar registros de conversaciones, para Registros de texto, seleccione si los registros de texto están habilitados o deshabilitados pulsando el botón de radio. Si selecciona Habilitado para los registros de texto, tendrá que introducir un nombre de grupo de registros o elegir un nombre de grupo de registros existente en el menú desplegable. Seleccione la casilla Registrar enunciados de forma selectiva si va a registrar archivos de texto de forma selectiva.

   > **Note**
   >
   > Active los registros and/or de audio y texto marcando la casilla Registrar de forma selectiva las expresiones en la configuración de los registros de conversaciones (and/or audio de texto) en los ajustes de tiempo BotAliasde compilación. Debe configurar el grupo de CloudWatch registros y el bucket de Amazon S3 para seleccionar esta opción.

6. En la sección Registros de audio, seleccione si los registros de audio están habilitados o deshabilitados seleccionando el botón de radio. Si selecciona Habilitado para los registros de audio, debe especificar la ubicación del bucket de Amazon S3 y la clave de KMS (opcional) para cifrar los datos de audio. Seleccione la casilla Registrar enunciados de forma selectiva si va a registrar archivos de audio de forma selectiva.

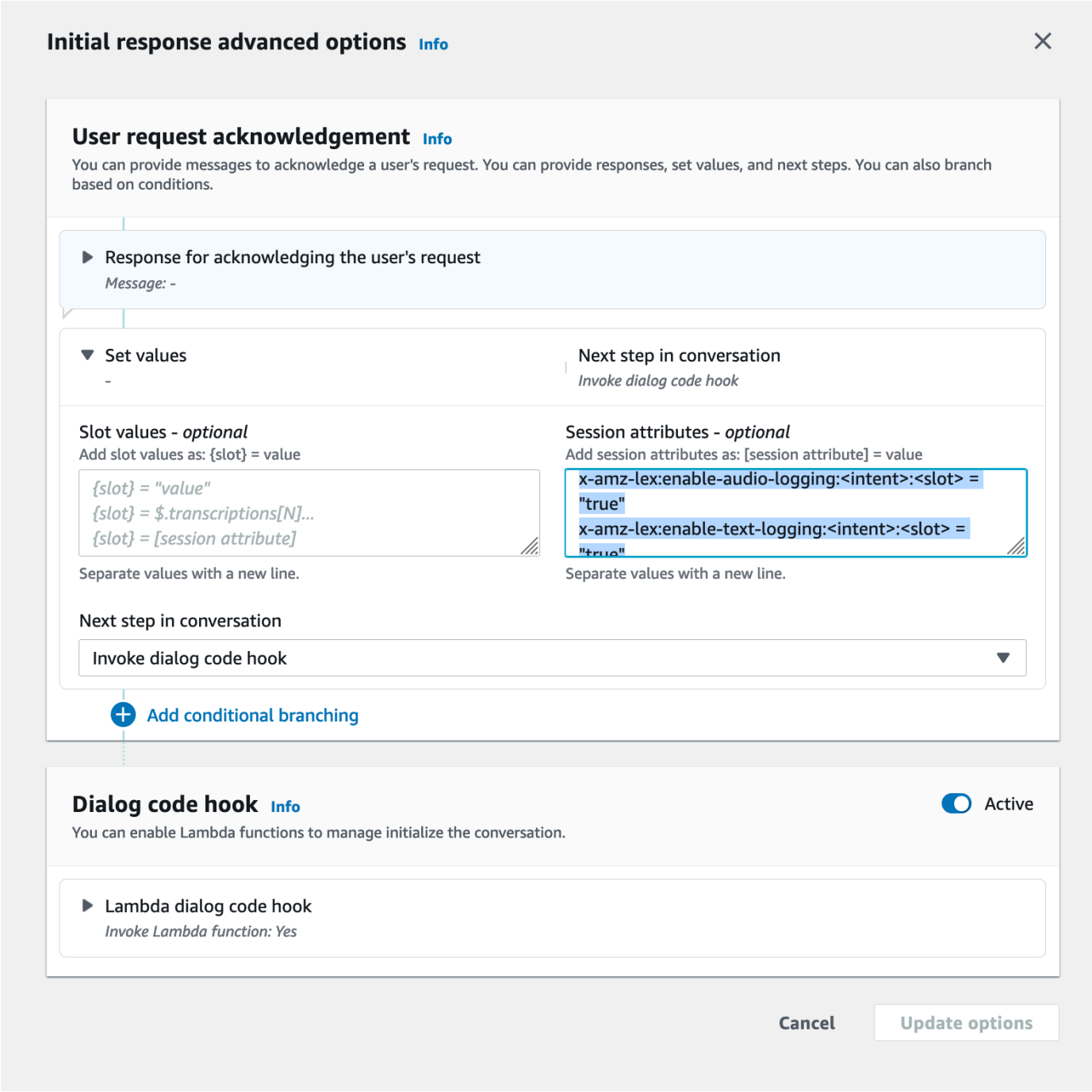


7. Seleccione Guardar en la esquina inferior derecha del panel para guardar la configuración selectiva de captura del registro de conversaciones.

Active la captura selectiva del registro de conversaciones en la consola de Amazon Lex V2:

1. Vaya a Intenciones y seleccione el Nombre de la intención, la Respuesta inicial, la Configuración avanzada, los Valores establecidos y los Atributos de sesión.
2. Defina los siguientes atributos en función de las intenciones y los slots para los que desee habilitar la captura selectiva del registro de conversaciones:

   - `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`*` = "true"`
   - `x-amz-lex:enable-text-logging:`*`intent`*`:`*`slot`*` = "true"`



> **Note**
>
> Se configura `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`*` = "true"` para capturar enunciados que contienen solo un slot específico de la conversación. La acción para registrar un enunciado depende de la valoración de*`intent`* : *`slot`* dentro del

enunciado, en comparación con las expresiones de los atributos de la sesión, y del valor del indicador correspondiente. Para registrar un enunciado, al menos una expresión del atributo de sesión debe permitirlo, con el indicador de activación del registro establecido en `true`. El valor de *`intent`* y `"*"` también *`slot`* puede ser. Si el valor de and/or intención de la ranura es`"*"`, significa que cualquier valor de and/or intención de la ranura `"*"` coincidirá con él. De forma similar a `x-amz-lex:enable-audio-logging`, se utilizará un nuevo atributo de sesión llamado `x-amz-lex:enable-text-logging` para controlar los registros de texto.

3. Seleccione Opciones de actualización y compile el bot para que incluya la configuración actualizada.

> **Note**
>
> Su rol de IAM debe tener permiso de acceso que le permita escribir datos en el bucket de Amazon S3 y utilizar una clave de KMS para cifrar los datos. Lex actualizará su función de IAM con permisos de Lex para acceder al grupo de CloudWatch registros Logs y al bucket de Amazon S3 seleccionado.

Directrices para utilizar la captura selectiva de registros de conversaciones:

Solo puede habilitar la captura selectiva de registros de conversaciones para los registros and/or de audio y texto si ha habilitado los registros de and/or audio de texto en la configuración del registro de conversaciones. Al habilitar la captura selectiva del registro de conversaciones para los registros and/or de audio y texto, se deshabilita el registro a todos los efectos y los intervalos de la conversación. Para generar registros and/or de audio de texto para determinadas intenciones y franjas, debe establecer los atributos de la sesión de captura selectiva del registro de conversaciones de or/and audio y texto para esas intenciones y ranuras en «true».

- Si la captura selectiva del registro de conversaciones está habilitada y no hay ningún atributo de sesión con el prefijo x-amz-lex:enable-audio-logging, el registro se desactivará de forma predeterminada para todos los enunciados. Este escenario también es válido en el caso de x-amz-lex:enable-text-logging.
- Los registros de expresiones se almacenarán exclusivamente para los segmentos de una conversación de and/or audio y texto si al menos una expresión del atributo de sesión lo permite.

- Las configuraciones para la captura selectiva del and/or audio de texto en el registro de conversaciones, tal como se definen en los atributos de la sesión, solo se aplicarán cuando la captura selectiva del registro de conversaciones para and/or audio de texto esté habilitada en la configuración del registro de conversaciones dentro del alias del bot; de lo contrario, no se tendrán en cuenta los atributos de la sesión.
- Cuando se habilita la captura selectiva del registro de conversaciones, los valores de los intervalos SessionState, las interpretaciones y las transcripciones para los que el registro no esté habilitado con los atributos de la sesión se ocultarán en el registro de texto generado.
- La decisión de crear registros de audio y and/or texto se evalúa haciendo coincidir el espacio elegido por el bot con los atributos de la sesión de captura selectiva del registro de conversaciones, excepto en el turno de obtención de intención, en el que el usuario puede proporcionar valores de espacio junto con la obtención de intención. En un turno de búsqueda de intenciones, los slots rellenados en el turno actual se comparan con los atributos de sesión de captura selectiva del registro de conversaciones.
- Los slots que se consideran rellenados se derivan del estado de la sesión al final del turno. Por lo tanto, cualquier alteración que realice el Codehook Lambda de Dialog en los slots del estado de la sesión influirá en el comportamiento de la captura selectiva del registro de conversaciones.
- En un turno de búsqueda de intenciones, si el usuario proporciona varios valores de espacio, el registro de and/or audio de texto solo se generará si los atributos de la text/audio sesión permiten registrar todos los espacios ocupados en este turno.
- El enfoque operativo recomendado consiste en establecer el atributo de la sesión de captura selectiva del registro de conversaciones al principio de la sesión y no modificarlo durante la sesión.
- Si algún slot contiene datos confidenciales, siempre debe habilitar la ofuscación de slots.

Ejemplo de captura selectiva del registro de conversaciones

Este es un ejemplo de un caso de uso empresarial para la captura selectiva de registros de conversaciones.

Caso de uso:

Una empresa de tecnología financiera utiliza un bot de Amazon Lex V2 para respaldar su sistema IVR, que permite a los usuarios realizar pagos de facturas. Para cumplir con los requisitos de conformidad y auditoría, deben conservar las grabaciones de audio con el consentimiento de autorización otorgado por el usuario. Sin embargo, no es posible habilitar los registros de audio generales, ya que los haría incompatibles, ya que no es posible ocultar ranuras sensibles como

CardNumber el CVV y otra información de los registros de audio. En su lugar, pueden habilitar la captura selectiva de registros de conversaciones para los registros de audio y configurar el atributo de sesión para que solo produzca registros de audio para los enunciados que cuenten con el consentimiento de la autorización.

BotAlias Configuración:

- Registros de texto habilitados: verdadero
- Registro selectivo de registros de texto habilitado: falso
- Registros de audio habilitados: verdadero
- Registro selectivo de registros de audio activado: verdadero

Atributos de sesión:

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

Ejemplo de conversación:

- Usuario (entrada de audio): «Quiero pagar mi factura con el número de factura 35XU68».
- Bot: «¿Cuál es el importe adeudado en dólares?»
- Usuario (entrada de audio): «235».
- Bot: «¿Cuál es el número de tu tarjeta de crédito?»
- Usuario (entrada de audio): «9239829722200348».
- Bot: «Está pagando 235 dólares con el número de su tarjeta de crédito que termina en 0348. Diga: «Autorizo a pagar 235 dólares».
- Usuario (entrada de audio): «Autorizo a pagar 235 dólares».
- Bot: «Su factura ha sido pagada».

Salida de registros de conversaciones:

En esta situación, se generarán registros de texto para todos los turnos. Sin embargo, los registros de audio solo se grabarán para un turno concreto cuando se haya obtenido el AuthorizationConsentespacio dentro de la PayBillintención, y no se generará ningún registro de audio para ningún otro turno.

Registro de errores con registros de errores en Lex V2

Se habilitan los registros de errores para almacenar las interacciones de los bots. Puedes usar estos registros de errores para revisar el rendimiento de tu bot y solucionar errores en las conversaciones.

Los registros de errores están configurados para una versión. Cada versión puede tener una configuración diferente para sus registros de errores. Los registros de texto almacenan la entrada de texto en CloudWatch los registros. Puede activar el cifrado de los registros de texto mediante CMK gestionadas por el AWS KMS cliente.

Políticas de IAM para registros de errores

Según el tipo de registro que seleccione, Amazon Lex V2 necesitará permiso para usar Amazon CloudWatch Logs y los depósitos de Amazon Simple Storage Service (S3) para almacenar sus registros. Debe crear funciones y permisos de AWS Identity and Access Management para que Amazon Lex V2 pueda acceder a estos recursos.

Creación de un rol y políticas de IAM para los registros de errores

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

Puede utilizar la consola de IAM, la API de IAM o la AWS Command Line Interface para crear el rol y las políticas. Estas instrucciones utilizan la CLI de AWS para crear el rol y las políticas.

Para crear un rol de IAM para los registros de errores

La función de IAM que utilice para habilitar los registros de conversaciones debe tener el permiso iam:. PassRole La siguiente política debe estar asociada al rol:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/role"
        }
```

```
    ]
}
```

Habilitación de registros de errores en Lex V2

Para activar los registros de errores desde la consola de Amazon Lex V2, haga lo siguiente:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, seleccione el bot que quiere habilitar para los registros de errores.
3. En el menú de la izquierda, elija Versión.
4. En la lista de Versión, elija la versión para la que desea configurar los registros de errores.
5. En la sección de detalles de la versión, elija Activar.
6. Seleccione Guardar para iniciar el registro de conversaciones. Si es necesario, Amazon Lex V2 actualizará su rol de servicio con permisos para acceder al grupo de CloudWatch registros Logs.

Desactivación de registros de error en Lex V2

para desactivar los registros de errores desde la consola de Amazon Lex V2, haga lo siguiente:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, seleccione el bot que quiere habilitar para los registros de errores.
3. En el menú de la izquierda, elija Versión.
4. En la lista de Versión, elija la versión para la que desea configurar los registros de errores.
5. En la sección de detalles de la versión, seleccione Desactivar.
6. Seleccione Guardar para detener el registro de conversaciones.

Monitoreo de métricas operativas en Lex V2

Amazon CloudWatch y AWS CloudTrail son dos servicios de AWS que se integran con Amazon Lex V2 para ayudarlo a monitorear las interacciones de los usuarios con su bot. Utilice estos servicios para registrar las acciones, enviar datos prácticamente en tiempo real y configurar notificaciones y acciones automatizadas cuando se cumplan los criterios.

Temas

- Medición de las métricas operativas con Amazon CloudWatch
- Visualización de eventos con AWS CloudTrail para Lex V2

Medición de las métricas operativas con Amazon CloudWatch

Puede monitorizar Amazon Lex V2 con Amazon Lex V2 CloudWatch, que recopila datos sin procesar y los procesa para convertirlos en métricas legibles prácticamente en tiempo real. Estas estadísticas se mantienen durante 15 meses, de forma que pueda obtener acceso a información histórica y disponer de una mejor perspectiva sobre el desempeño de su aplicación web o servicio. También puede establecer alarmas que vigilen determinados umbrales y enviar notificaciones o realizar acciones cuando se cumplan dichos umbrales. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

El servicio Amazon Lex V2 informa de las siguientes métricas en el espacio de nombres de `AWS/Lex`.

Métrica	Description (Descripción)
`AssistedSlotResolutionModelAccessDeniedErrorCount`	Número de veces que se denegó a Amazon Lex V2 acceso a Amazon Bedrock Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • BotId, BotAliasId LocaleId, Funcionamiento InputMode, ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento, InputMode, ModelType, Modelo Dimensiones válidas de `RecognizeText` : • BotId, BotAliasId, LocaleId, Operación ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento ModelType, Modelo Unidad: recuento

Métrica	Description (Descripción)
`AssistedSlotResolutionModelInvocationCount`	Número de veces que se invocó Amazon Bedrock. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • BotId, BotAliasId LocaleId, Funcionamiento, InputMode, ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento, InputMode, ModelType, Modelo Dimensiones válidas de `RecognizeText` : • BotId, BotAliasId, LocaleId, Operación ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento ModelType, Modelo Unidad: recuento
`AssistedSlotResolutionModelSystemErrorCount`	El número de veces que se produjo un 5xx al llamar a Amazon Bedrock. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • BotId, BotAliasId LocaleId, Funcionamiento, InputMode, Modelo ModelType • BotId, BotVersion LocaleId, Funcionamiento, InputMode, ModelType, Modelo Dimensiones válidas de `RecognizeText` : • BotId, BotAliasId, LocaleId, Operación ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento ModelType, Modelo Unidad: recuento

Métrica	Description (Descripción)
`AssistedSlotResolutionModelThrottlingErrorCount`	Número de veces que Amazon Bedrock limitó Amazon Lex. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • BotId, BotAliasId LocaleId, Funcionamiento, InputMode, ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento, InputMode, ModelType, Modelo Dimensiones válidas de `RecognizeText` : • BotId, BotAliasId, LocaleId, Operación ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento ModelType, Modelo Unidad: recuento
`AssistedSlotResolutionResolvedSlotCount`	Número de veces que Amazon Bedrock devolvió un valor de slot. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • BotId, BotAliasId LocaleId, Funcionamiento, InputMode, ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento, InputMode, ModelType, Modelo Dimensiones válidas de `RecognizeText` : • BotId, BotAliasId, LocaleId, Operación ModelType, Modelo • BotId, BotVersion LocaleId, Funcionamiento ModelType, Modelo Unidad: recuento

Métrica	Description (Descripción)
`KendraIndexAccessError`	El número de veces que Amazon Lex V2 no ha podido acceder a su índice de Amazon Kendra. • Operación, BotId, BotAliasId, LocaleId Unidad: recuento
`KendraLatency`	El tiempo que tarda Amazon Kendra en responder a una solicitud del `AMAZON.KendraSearchIntent` . Dimensiones válidas: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: milisegundos
`KendraSuccess`	El número de veces que Amazon Lex V2 no pudo acceder a su índice de Amazon Kendra. Dimensiones válidas: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: recuento
`KendraSystemErrors`	El número de veces que Amazon Lex V2 no pudo consultar el índice de Amazon Kendra. Dimensiones válidas: • Operación, BotId, BotAliasId, InputMode, LocaleId Unidad: recuento

Métrica	Description (Descripción)
`KendraThrottledEvents`	La cantidad de veces que Amazon Kendra limitó las solicitudes de `AMAZON.KendraSearchIntent` . Dimensiones válidas: • Operación, BotId, BotAliasId, InputMode. LocaleId Unidad: recuento
`RuntimeConcurrency`	El número de conexiones simultáneas en el período especificado. `RuntimeConcurrency` se informa como `StatisticSet` . Dimensiones válidas para las operaciones `RecognizeUtterance` o `StartConversation` : • Operación BotId, BotVersion, InputMode, LocaleId • Operación, BotId, BotAliasId, InputMode, LocaleId Dimensiones válidas para otras operaciones: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: recuento
`RuntimeInvalidLambdaResponses`	El número de AWS Lambda respuestas no válidas en el período especificado. Dimensiones válidas: • Operación, BotId, BotAliasId, InputMode, LocaleId Unidad: recuento

Métrica	Description (Descripción)
`RuntimeLambdaErrors`	El número de errores de tiempo de ejecución de Lambda en el período de tiempo especificado. Dimensiones válidas: • Operación, BotId, BotAliasId, InputMode, LocaleId Unidad: recuento
`RuntimePollyErrors`	Número de respuestas de Amazon Polly no válidas durante el período de tiempo especificado. Dimensiones válidas: • Operación, BotId, BotAliasId, InputMode, LocaleId Unidad: recuento
`RuntimeRequestCount`	El número de solicitudes en tiempo de ejecución durante el período de tiempo especificado. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • Operación, BotId, BotVersion, InputMode, LocaleId • Operación, BotId, BotAliasId, InputMode, LocaleId Dimensiones válidas para otras operaciones: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: recuento

Métrica	Description (Descripción)
`RuntimeRequestLength`	Duración total de una conversación con un bot de Amazon Lex V2. Aplicable únicamente a la StartConversationoperación. Dimensiones válidas: • BotAliasId, BotId, LocaleId, Operación • BotId, BotAliasId, LocaleId, Operación Unidad: milisegundos
`RuntimeSuccessfulRequestLatency` **Important** Esta métrica es `RuntimeSuccessfulRequestLatency` y no `RuntimeSuccessfulRequestLatency` .	La latencia de las solicitudes correctas entre el momento en que se realizó la solicitud y en que se devuelve la respuesta. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • Operación, BotId, BotVersion, InputMode, LocaleId • Operación, BotId, BotAliasId, InputMode, LocaleId Dimensiones válidas para otras operaciones: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: milisegundos

Métrica	Description (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. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation`: • Operación, BotId, BotVersion, InputMode, LocaleId • Operación, BotId, BotAliasId, InputMode, LocaleId Dimensiones válidas para otras operaciones: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: recuento
`RuntimeThrottledEvents`	El número de solicitudes limitadas. Amazon Lex V2 limita un evento 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. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation`: • Operación, BotId, BotVersion, InputMode, LocaleId • Operación, BotId, BotAliasId, InputMode, LocaleId Dimensiones válidas para otras operaciones: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: recuento

Métrica	Description (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. Dimensiones válidas para las operaciones `RecognizeUtterance` y `StartConversation` : • Operación, BotId, BotVersion, InputMode, LocaleId • Operación, BotId, BotAliasId, InputMode, LocaleId Dimensiones válidas para otras operaciones: • Operación, BotId, BotVersion, LocaleId • Operación, BotId, BotAliasId, LocaleId Unidad: recuento

Las siguientes dimensiones se admiten para las métricas de Amazon Lex V2.

Dimensión	Description (Descripción)
`Operation`	El nombre de la operación de Amazon Lex V2 (`RecognizeText` , `RecognizeUtterance` , `StartConversation` , `GetSession` , `PutSession` , `DeleteSession`) que generó la entrada.
`BotId`	Un identificador único alfanumérico para el bot.
`BotAliasId`	Un identificador único alfanumérico para el alias del bot.
`BotVersion`	La versión numérica del bot.
`InputMode`	El tipo de entrada al bot: voz, texto o DTMF.

Dimensión	Description (Descripción)
`LocaleId`	El identificador de la configuración regional del bot, como en-US o fr-CA.
`Model`	Indica el identificador del modelo de lenguaje grande de Amazon Bedrock.
`ModelType`	Indica el tipo de modelo de lenguaje grande invocado desde Amazon Bedrock.

Visualización de eventos con AWS CloudTrail para Lex V2

Amazon Lex V2 está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un servicio de AWS en Amazon Lex V2. CloudTrail captura las llamadas a la API de Amazon Lex V2 como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de Amazon Lex V2 y las llamadas desde el código a las operaciones de la API de Amazon Lex V2. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Amazon Lex V2. Si no configura un registro de seguimiento, puede ver los eventos más recientes en la consola de CloudTrail en el Event history (Historial de eventos). Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon Lex V2, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la Guía AWS CloudTrail del usuario.

Información sobre Amazon Lex V2 en CloudTrail

CloudTrail está activado en su cuenta de AWS al crear la cuenta. Cuando se produce una actividad en Amazon Lex V2, esa actividad se registra en un CloudTrail evento junto con otros eventos del servicio de AWS en el historial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos en su cuenta de AWS, incluidos los eventos de Amazon Lex V2, cree un registro. Un rastro permite CloudTrail entregar archivos de registro a un bucket de 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 Amazon S3 especificado. También puede configurar otros servicios de AWS para analizar y actuar en función de los datos de eventos recopilados en los registros de CloudTrail . Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail servicios e integraciones compatibles
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recepción de archivos de CloudTrail registro de varias regiones y recepción de archivos de CloudTrail registro de varias cuentas

Amazon Lex V2 admite el registro de todas las acciones enumeradas en la API de creación de modelos V2.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de usuario root o de usuario IAM de AWS Identity and Access Management.
- 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 .

Descripción de las entradas de archivos de registro de Amazon Lex V2

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la CreateBotAliasacción.

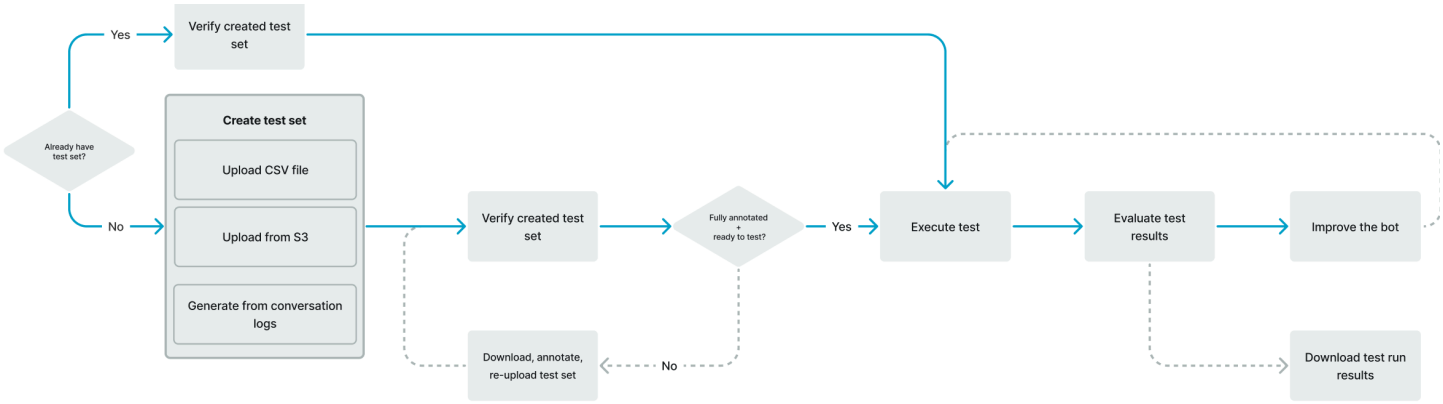
```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ID of caller:temporary credentials",
        "arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ID of caller",
                "arn": "arn:aws:iam::111122223333:role/role name",
                "accountId": "111122223333",
                "userName": "role name"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "creation date"
            }
        }
    },
    "eventTime": "event timestamp",
    "eventSource": "lex.amazonaws.com",
    "eventName": "CreateBotAlias",
    "awsRegion": "Region",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "user agent",
    "requestParameters": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
            }
        },
        "botId": "bot ID",
        "botAliasName": "bot aliase name",
        "botVersion": "1"
    },
    "responseElements": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
```

```
            }
        },
        "botAliasId": "bot alias ID",
        "botAliasName": "bot alias name",
        "botId": "bot ID",
        "botVersion": "1",
        "creationDateTime": creation timestamp
    },
    "requestID": "unique request ID",
    "eventID": "unique event ID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Evaluación del rendimiento de los bots Lex V2 con Test Workbench

Para mejorar el rendimiento del bot, puede evaluarlo a escala. Los resultados de la evaluación de la prueba se muestran en tablas y gráficos sencillos.

Puede utilizar el Test Workbench para crear conjuntos de pruebas de referencia que utilicen los datos de transcripción existentes. Puede probar los bots para evaluar el rendimiento antes de la implementación y ver el desglose de los resultados de las pruebas a escala.



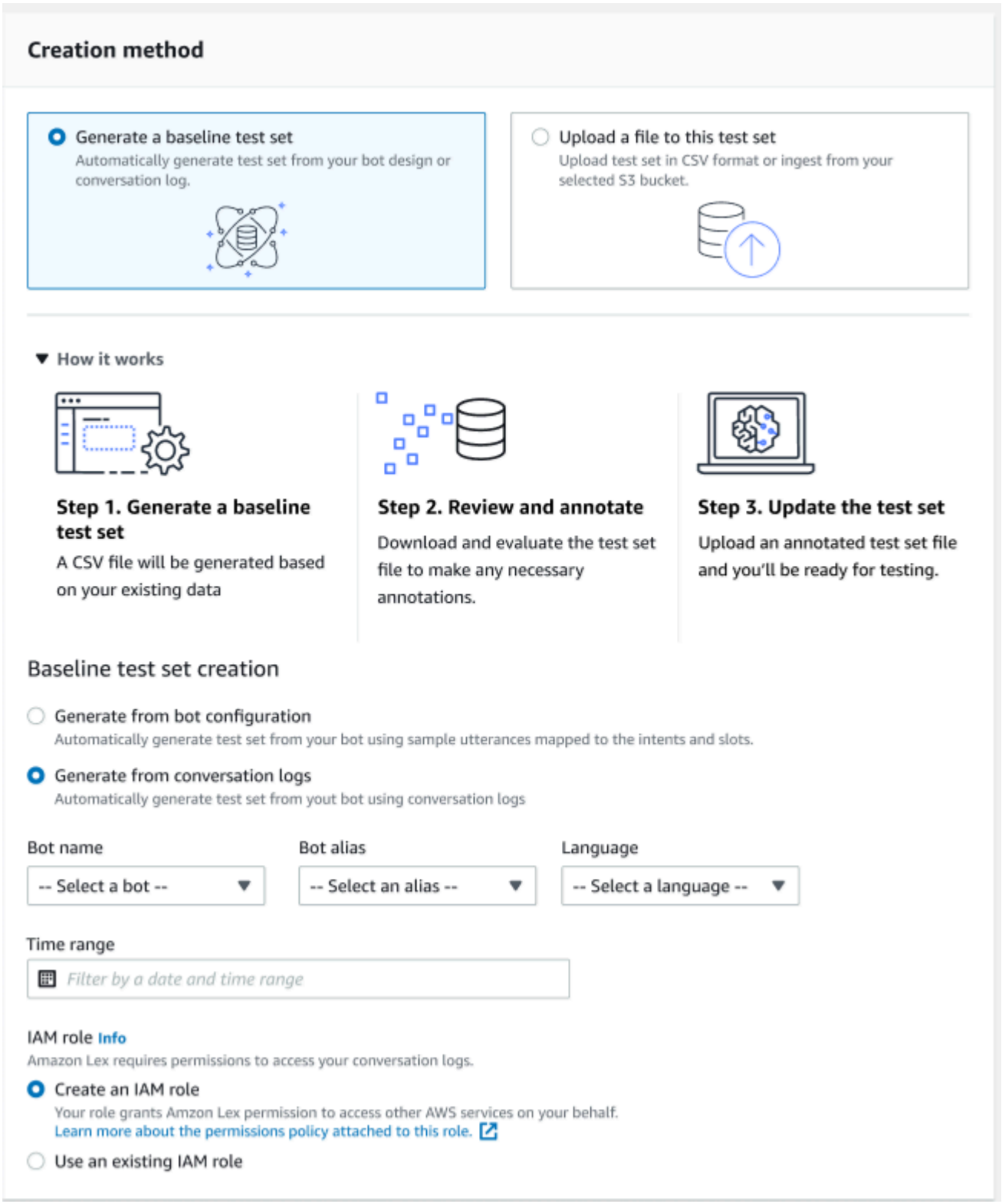
Los usuarios pueden usar el Test Workbench de pruebas para establecer el rendimiento básico de los bots. Esto abarca la intención y el rendimiento de los slots en el caso de los enunciados que se presentan en forma de entradas únicas o conversaciones. Una vez que el conjunto de prueba se haya cargado correctamente, podrá ejecutarlo con sus bots de preproducción o producción existentes. El Test Workbench le ayuda a identificar oportunidades para mejorar la ocupación de los slots y la clasificación por intención.

Temas

- Generación de un conjunto de prueba para Test Workbench
- Administrar conjuntos de prueba
- Ejecutar una prueba
- Cobertura del conjunto de prueba en Test Workbench
- Ver resultados de la prueba
- Detalles de los resultados de la prueba en Test Workbench

Generación de un conjunto de prueba para Test Workbench

Puede crear un conjunto de prueba para evaluar el rendimiento de su bot. Genere un conjunto de prueba subiendo un conjunto de prueba en formato de archivo CSV o generando un conjunto de pruebas a partir de los registros de conversaciones. El conjunto de prueba puede contener entradas de audio o texto.



Si un conjunto de prueba genera errores de validación, elimínelo y sustitúyalo por otra lista de datos del conjunto de prueba, o edite los datos del archivo CSV mediante un programa de edición de hojas de cálculo.

Para crear un conjunto de datos de prueba:

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione Test Workbench en el panel lateral izquierdo.
3. Seleccione Conjuntos de pruebas en las opciones de Test Workbench.
4. Seleccione el botón Crear conjunto de prueba en la consola.
5. En los Detalles, introduzca un nombre del conjunto de prueba y una descripción opcional.
6. Seleccione Generar un conjunto de prueba de referencia.
7. Seleccione Generar a partir de registros de conversaciones.
8. Seleccione el Nombre del bot, el Alias del bot y el Idioma en los menús desplegables.
9. Si va a generar una prueba de referencia a partir de un registro de conversaciones, seleccione Intervalo de tiempo y rol de IAM, si es necesario. Puede crear un rol con los permisos básicos de Amazon Lex V2 o crear un rol existente.
10. Seleccione una modalidad de audio o texto para el conjunto de prueba que va a crear. NOTA: El Test Workbench puede importar hasta 50 000 archivos de texto y hasta 5 horas de audio.
11. Seleccione una ubicación de Amazon S3 para almacenar los resultados de las pruebas y añada una clave KMS opcional para cifrar las transcripciones de salida.
12. Seleccione Crear.

Para cargar un conjunto de prueba existente en un formato de archivo CSV o para actualizar el conjunto de prueba:

1. Seleccione Test Workbench en el panel lateral izquierdo.
2. Seleccione Conjuntos de pruebas en las opciones de Test Workbench.
3. Seleccione Cargar un archivo a este conjunto de prueba en la consola.
4. Seleccione Cargar desde un bucket de Amazon S3 o Cargar desde su ordenador. NOTA: Puede cargar un archivo CSV creado a partir de una plantilla. Haga clic en la plantilla CSV para descargar un archivo zip que contiene las plantillas.
5. Puede Crear un rol con los permisos básicos de Amazon Lex V2 o Utilizar un rol existente para el ARN del rol.
6. Seleccione una modalidad de audio o texto para el conjunto de prueba que va a crear. NOTA: El Test Workbench puede importar hasta 50 000 archivos de texto y hasta 5 horas de audio.

7. Seleccione una ubicación de Amazon S3 para almacenar los resultados de las pruebas y añada una clave KMS opcional para cifrar las transcripciones de salida.
8. Seleccione Crear.

Si la operación se realiza correctamente, el mensaje de confirmación indicará que el conjunto de prueba está listo para la prueba y el estado mostrará Listo para la prueba.

Consejos para crear un conjunto de prueba exitoso

- Puede crear un rol de IAM para el Test Workbench en la consola o puede configurar su rol de IAM paso a paso. Para obtener más información, consulte Crear un rol de IAM para el Test Workbench.
- Antes de ejecutar una prueba, valide el conjunto de prueba y la definición del bot para detectar cualquier incoherencia mediante el botón Validar discrepancia. Si las convenciones de intención y de nomenclatura de los slots utilizadas en el conjunto de prueba son coherentes con las del bot, procede a ejecutar la prueba. Si se identifica alguna anomalía, revise el conjunto de prueba, actualícelo y seleccione Validar discrepancia. Repita esta secuencia de nuevo hasta que no se detecten inconsistencias y, a continuación, ejecute la prueba.
- El Test Workbench puede realizar pruebas con diferentes formatos de valores de slot en la columna Slot de salida esperado. Para cualquier slot integrado, puede elegir el valor proporcionado en la entrada del usuario (por ejemplo, Fecha = mañana) o proporcionar su valor absoluto resuelto (por ejemplo, Fecha = 21-03-2021). Para obtener más información sobre las ranuras integradas y sus valores absolutos, consulte Built-inlas ranuras.
- Para garantizar la coherencia y la legibilidad de las columnas de las ranuras de salida esperadas, siga la convención de SlotValue "SlotName =" (por ejemplo, AppointmentType = limpiar) con un espacio antes y después del signo igual.
- Si el bot incluye ranuras compuestas, en Ranura de salida esperada, defina subranuras con el nombre de la ranura, separadas por un punto (por ejemplo, «Car.Color»). No funcionará ninguna otra sintaxis o puntuación.
- Si el bot incluye ranuras con varios valores, en la ranura de salida esperada proporciona varios valores de ranura separados por una coma («FlowerType = rosas, lirios»). No funcionará ninguna otra sintaxis o puntuación.
- Asegúrese de que el conjunto de prueba se haya creado a partir de registros de conversaciones válidos.
- Slot:slot el valor estará en la misma columna después de las columnas de intención en formato CSV.

- La entrada DTMF de un turno de usuario se interpreta como una transcripción esperada y no incluye una ubicación de Amazon S3.

Creación de un caso de prueba dentro de un conjunto de prueba con Test Workbench

Los resultados de Test Workbench dependen de la definición del bot y de su conjunto de prueba correspondiente. Puede generar un conjunto de prueba con la información de la definición del bot para identificar las áreas que necesitan mejoras. Cree un conjunto de datos de prueba con ejemplos que sospeche (o sepa) que serán difíciles de interpretar correctamente para el bot, teniendo en cuenta el diseño actual del bot y su conocimiento de las conversaciones con sus clientes.

Revise sus intenciones de forma regular en función de lo que aprenda de su bot de producción. Continúe añadiendo y ajustando los enunciados de muestra y los valores de los slots del bot. Considere la posibilidad de mejorar la resolución de los slots mediante las opciones disponibles, como las sugerencias en tiempo de ejecución. El diseño y desarrollo de su bot es un proceso iterativo que es un ciclo continuo.

Estos son algunos consejos más para optimizar su conjunto de prueba:

- Seleccione los casos de uso más comunes con las intenciones y slots más usados en el conjunto de prueba.
- Explore las diferentes formas en que un cliente podría referirse a sus intenciones y slots. Esto puede incluir las entradas del usuario en forma de declaraciones, preguntas y comandos que varían en longitud, desde mínima hasta extensa.
- Incluya las entradas del usuario con un número variado de slots.
- Incluya los sinónimos o abreviaturas más habituales de los valores de los slots personalizados que admite su bot (por ejemplo, «endodoncia», «tratamiento de conductos», «endo»).
- Incluya variaciones de los valores de los slots integrados (por ejemplo, «mañana», «lo antes posible» o «al día siguiente»).
- Examine la solidez del bot en la modalidad hablada recopilando las entradas de los usuarios que puedan malinterpretarse (por ejemplo, «tinta», «tobillo» o «ancla»).

Creación de un conjunto de prueba a partir de un archivo CSV para Test Workbench

Puede crear un conjunto de prueba a partir de la plantilla de archivo CSV proporcionada en la consola de Amazon Lex V2 introduciendo los valores directamente mediante un editor de hojas de cálculo CSV. El conjunto de prueba es un archivo de valores separados por comas (CSV) que consta

de enunciados de un solo usuario y conversaciones de varios turnos grabadas en las siguientes columnas:

- Número de línea: esta columna es un contador incremental que lleva un registro del total de filas rellenadas que se van a probar.
- Número de conversación: esta columna registra el número de turnos de una conversación. Para entradas individuales, esta columna puede dejarse vacía y rellenarse con «-» o "N/A». En el caso de las conversaciones, a cada turno de una conversación se le asignará el mismo número de conversación.
- Fuente: esta columna está configurada como «Usuario» o «Agente». Para entradas individuales, siempre estará configurada como «Usuario».
- Entrada: esta columna incluye el enunciado del usuario o las instrucciones del bot.
- Intención de salida esperada: esta columna captura la intención cumplida en la entrada.
- Slot de salida esperado por intención 1: esta columna captura el primer slot obtenido en la entrada del usuario. El conjunto de prueba debe incluir una columna llamada Slot de salida esperado X para cada slot de la entrada del usuario.

Ejemplo de un conjunto de prueba con entradas individuales:

Número de línea	Número de conversación	origen	Input	Intención de salida esperada	Slot de salida esperado 1	Slot de salida esperado 2
1		Usuario	reserve una cita de limpieza mañana	MakeAppointment	AppointmentType = limpieza	Fecha = mañana
2	N/A	Usuario	reserve una cita de limpieza el 15 de abril	MakeAppointment	AppointmentType = limpieza	Fecha = 4/15 23/3
3	N/A	Usuario	reserve cita para	MakeAppointment	Fecha = uno de diciembre	

Número de línea	Número de conversación	origen	Input	Intención de salida esperada	Slot de salida esperado 1	Slot de salida esperado 2
			el uno de diciembre			
4	N/A	Usuario	reserve una cita de limpieza	MakeAppointment	AppointmentType = limpieza	
1		Usuario	¿Me puede ayudar a reservar una cita?	MakeAppointment		

Ejemplo de un conjunto de prueba con conversaciones

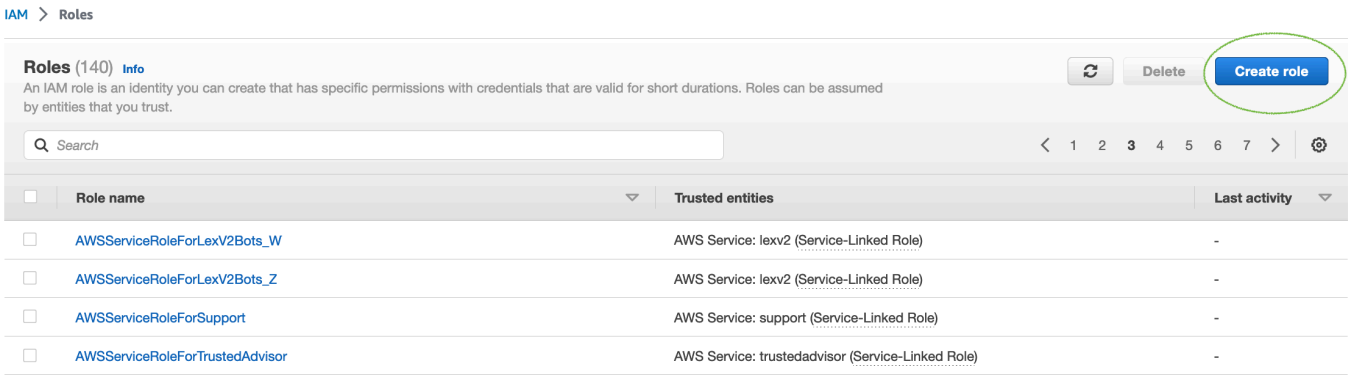
Número de línea	Número de conversación	origen	Input	Intención de salida esperada	Slot de salida esperado 1	Slot de salida esperado 2	Slot de salida esperado 2
1	1	Usuario	concertar una cita	MakeAppointment			
2.	1	Agente	¿Qué tipo de cita desea concertar?	MakeAppointment			
3	1	Usuario	limpieza	MakeAppointment	AppointmentType = limpieza		

Número de línea	Número de conversación	origen	Input	Intención de salida esperada	Slot de salida esperado 1	Slot de salida esperado 2	Slot de salida esperado 2
4	1	Agente	¿Cuándo debo programar su cita?	MakeAppointment			
5	1	Usuario	mañana	MakeAppointment		Fecha = mañana	
6	2	Usuario	reservar una cita para endodoncia hoy	MakeAppointment	AppointmentType = conducto radicular	Fecha = hoy	
7	2	Agente	¿A qué hora debo programar su cita?	MakeAppointment			
8	2	Usuario	once de la mañana	MakeAppointment			Hora = once de la mañana

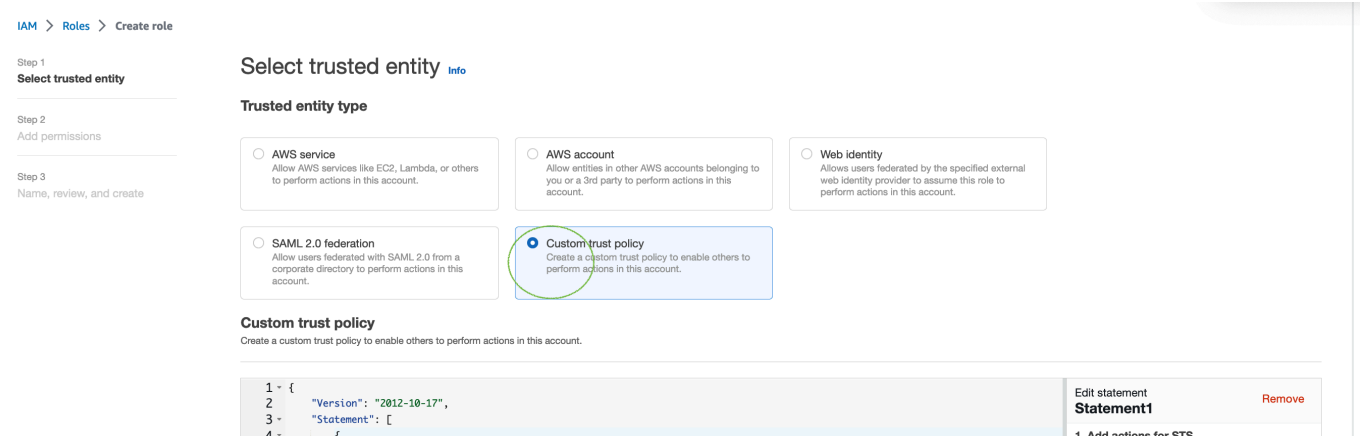
Cree un rol de IAM para el Test Workbench

Crear un rol de IAM para el Test Workbench

1. Siga los pasos que se indican en Crear un usuario de IAM para crear un usuario de IAM que se pueda utilizar para acceder a la consola test-workbench.
2. Seleccione el botón Crear rol.



3. Seleccione la opción Política de confianza personalizada.



4. Introduzca la política de confianza que aparece a continuación y haga clic en Siguiente.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

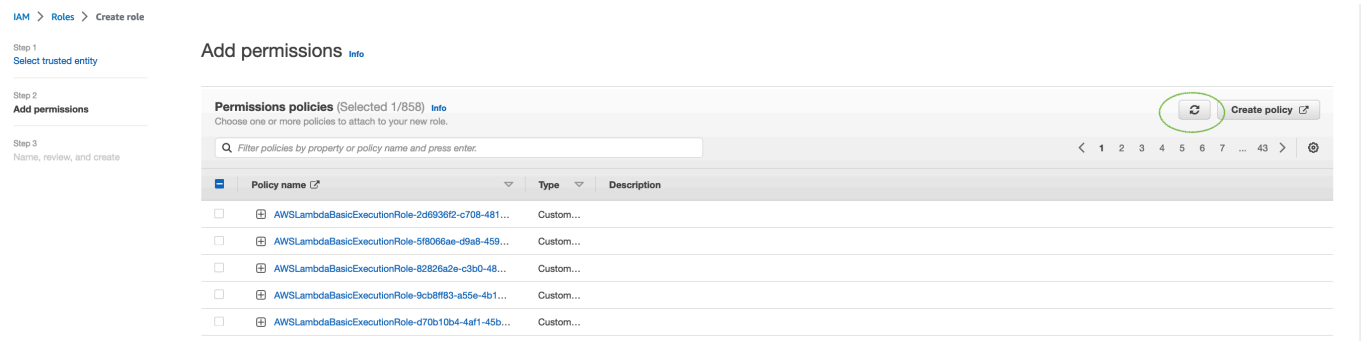
5. Seleccione el botón Crear política.

6. Se abrirá una nueva pestaña en su navegador donde podrá introducir la siguiente política y hacer clic en el botón Siguiente: Etiquetas.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Introduzca el nombre de una política, por ejemplo, «LexTestWorkbenchPolicy» y, a continuación, haga clic en Crear política.
8. Vuelva a la pestaña anterior de su navegador y actualice la lista de políticas haciendo clic en el botón Actualizar, como se muestra a continuación.



9. Busque en la lista de políticas introduciendo el nombre de la política que utilizó en el sexto paso y seleccione la política.
10. Seleccione el botón Siguiente.
11. Introduzca el nombre del rol y, a continuación, haga clic en el botón Crear rol.
12. Seleccione su nuevo rol de IAM cuando se le solicite en la consola de Amazon Lex V2 para Test Workbench.

Cree un rol de IAM para el banco de trabajo de pruebas: funciones avanzadas

Configuración de permisos para el rol de IAM de Test Workbench

En esta sección se muestran varios ejemplos de políticas basadas en la identidad de AWS Identity and Access Management (IAM) para implementar controles de acceso con privilegios mínimos para los permisos de Test Workbench.

1. Política para que Test Workbench lea archivos de audio en S3: esta política permite a Test Workbench leer los archivos de audio que se utilizan en los sets de prueba. La siguiente política debe modificarse en consecuencia para actualizarlos *S3BucketName* y *S3Path* dirigirlos a una ubicación de Amazon S3 de los archivos de audio de un conjunto de prueba.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "TestWorkbenchS3AudioFilesReadOnly",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::S3BucketName/S3Path/*"
      ]
    }
  ]
}
```

2. Política para que Test Workbench lea y escriba conjuntos de pruebas y resultados en un bucket de Amazon S3: esta política permite a Test Workbench almacenar las entradas y los resultados del conjunto de pruebas. La siguiente política debe modificarse para actualizarla *S3BucketName* al bucket de Amazon S3 donde se almacenarán los datos del conjunto de pruebas. Test Workbench almacena estos datos exclusivamente en su bucket de Amazon S3 y no en la infraestructura de Lex Service. Por lo tanto, por este motivo, Test Workbench necesita acceso a su bucket de Amazon S3 para funcionar correctamente.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "TestSetDataUploadWithEncryptionOnly",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::S3BucketName/*/lex_testworkbench/test_set/*",
        "arn:aws:s3:::S3BucketName/*/lex_testworkbench/test_execution/*",
        "arn:aws:s3:::S3BucketName/*/lex_testworkbench/
test_set_discrepancy_report/*"
      ],
      "Condition": {
        "StringEquals": {
          "s3:x-amz-server-side-encryption": "aws:kms"
        }
      }
    },
    {
      "Sid": "TestSetDataGetObject",
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::S3BucketName/*/lex_testworkbench/test_set/*",
        "arn:aws:s3:::S3BucketName/*/lex_testworkbench/test_execution/*",
        "arn:aws:s3:::S3BucketName/*/lex_testworkbench/
test_set_discrepancy_report/*"
      ]
    },
    {
      "Sid": "TestSetListS3Objects",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::S3BucketName"
      ]
    }
  ]
}
```

3. Política para que Test Workbench lea CloudWatch los registros: esta política permite a Test Workbench generar conjuntos de pruebas a partir de los registros de texto de Lex Conversation almacenados en Amazon Logs. CloudWatch La siguiente política debe modificarse para actualizar,,. *Region* *AwsAccountId* *LogGroupName*

4. Política para que Test Workbench llame a Lex Runtime: esta política permite a Test Workbench ejecutar un conjunto de pruebas contra bots de Lex. La siguiente política debe modificarse para actualizar*Region*,,*AwsAccountId*. *BotId* Como Test Workbench puede probar cualquier bot de su entorno de Lex, puede sustituir el recurso por «arn:aws:lex: *Region* ::bot-alias/*» para *AwsAccountId* permitir que Test Workbench acceda a todos los bots de Amazon Lex V2 de una cuenta.

5. (Opcional) Política para que Test Workbench cifre y descifre los datos del conjunto de pruebas: si Test Workbench está configurado para almacenar las entradas y los resultados del conjunto de pruebas en depósitos de Amazon S3 mediante una clave de KMS administrada por el cliente, Test Workbench necesitará permisos de cifrado y descifrado de la clave de KMS. La siguiente

política debe modificarse para actualizarla*Region*, y ¿*KmsKeyId*dónde *KmsKeyId* está el ID de la clave de *AwsAccountId* KMS administrada por el cliente?

6. Política (opcional) para que Test Workbench descifre los archivos de audio: si los archivos de audio se almacenan en el bucket de S3 con una clave KMS administrada por el cliente, Test Workbench necesitará un permiso para descifrar las claves de KMS. Se debe modificar la siguiente política para actualizarla *RegionAwsAccountId*, y *KmsKeyId* ¿dónde *KmsKeyId* está el ID de la clave KMS administrada por el cliente que se utiliza para cifrar los archivos de audio?

Administrar conjuntos de prueba

Puede descargar, actualizar y eliminar conjuntos de prueba desde la ventana del conjunto de prueba. O bien, puede utilizar la lista de conjuntos de pruebas disponibles para editar o anotar manualmente el archivo del conjunto de prueba. A continuación, cárguelo de nuevo para volver a intentar la validación debido a errores u otros problemas de entrada.

Para descargar el archivo del conjunto de prueba desde el registro del conjunto de prueba:

1. Seleccione el nombre del conjunto de prueba en la lista de conjuntos de prueba.
2. En la ventana de registro del conjunto de prueba, seleccione el botón Descargar situado en la parte derecha de la pantalla, en la sección Entradas de prueba.
3. Si hay algún detalle de error de validación en la parte superior de la ventana relacionado con el conjunto de prueba, seleccione el botón Descargar. El archivo se guardará en la carpeta Descargas. Puede corregir los errores de validación del conjunto de prueba a partir de los mensajes de error del archivo CSV del conjunto de prueba. Busque el error identificado en el paso de validación, corrija la línea o elimínela y cargue el archivo para volver a intentar el paso de validación.
4. Si ha descargado correctamente el conjunto de prueba, aparecerá un mensaje en color verde.

Para descargar un conjunto de prueba de la lista de conjuntos de prueba:

1. En la lista de conjuntos de prueba, seleccione el botón de radio situado junto al elemento del conjunto de prueba que desee descargar.
2. En el menú Acción de la parte superior derecha, seleccione Descargar.
3. Si ha descargado correctamente el conjunto de prueba, aparecerá un mensaje en color verde. El archivo se guardará en la carpeta Descargas.

Las columnas del conjunto de pruebas son compatibles con Test Workbench

A continuación, se muestra la lista completa de columnas de conjuntos de pruebas compatibles con el Banco de trabajo de prueba y las instrucciones sobre cómo utilizarlas con Amazon Lex V2.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Line Number (Número de línea)	Texto y audio	Número	No	Se trata de una columna de usuario que Amazon Lex V2 ignora. Su objetivo es ayudar al autor del conjunto de pruebas a ordenar y filtrar las filas del conjunto de pruebas. Se puede utilizar «Número de línea» como nombre de columna alternativo.
Número de conversación	Texto y audio	Número	No	Esta columna te permite juntar filas en una conversación. Se puede usar «Conversación #» como nombre

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				de columna alternativo.
origen	Texto y audio	Enum (“Usuario”, “Agente”)	No	El valor de esta columna indica si la fila es para un usuario o un agente. Se puede utilizar «Participante de la conversación» como nombre de columna alternativo.
Input	Texto	Cadena	No	Esta columna se utiliza para añadir la transcripción del conjunto de pruebas de texto. La entrada de texto debe usarse en las filas de usuarios. La petición del agente debe usarse en las filas Agente.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Transcripción esperada	Audio	Cadena	No	Esta columna se utiliza para añadir la transcripción del conjunto de pruebas de audio. La transcripción esperada del archivo de audio debe usarse en las filas de usuarios con entrada de audio. La entrada DTMF se puede utilizar en filas de usuarios con entrada DTMF. La petición del agente debe usarse en las filas Agente.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Ubicación de audio de S3	Audio	Cadena	No	Esta columna se utiliza para añadir la ubicación del archivo de audio y solo es aplicable a los sets de prueba de audio. La ruta S3 debe usarse en las filas de usuarios con la entrada de audio. Este campo debe dejarse vacío en las filas de usuarios con entrada DTMF. Este campo debe dejarse vacío en las filas Agente.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Etiqueta de contexto de entrada	Texto y audio	Cadena	Sí	Esta columna se utiliza para proporcionar el nombre de un contexto de entrada que se utilizará en la entrada de Lex al ejecutar la fila del conjunto de pruebas. Esto se refiere al contexto de entrada en Cómo establecer un contexto de intención para su bot Lex V2. Tenga en cuenta que Test Workbench solo admite el nombre del contexto. No admite los parámetros de un contexto. Se pueden usar varias columnas denominadas Etiqueta de contexto

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				de entrada 1, Etiqueta de contexto de entrada 2, etc.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Atributo de solicitud	Texto y audio	Cadena	Sí	Esta columna se utiliza para proporcionar un atributo de solicitud que se utilizará como entrada para Lex mientras se ejecuta la fila del conjunto de pruebas. El valor de una columna debe proporcionarse en el formato `<request-attribute-name>=<request-attribute-value>`. Se pueden añadir espacios alrededor del símbolo = para facilitar la lectura. Por ejemplo: request-attribute-foo = esta es una respuesta ficticia request-attribute-foo = 'esta es una «respuesta

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				ficticia» request-attribute-foo = «esta es una 'respuest a ficticia'». Se pueden usar varias columnas con los nombres Atributo de solicitud 1, Atributo de solicitud 2, etc.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Atributos de sesión	Texto y audio	Cadena	Sí	Esta columna se utiliza para proporcionar un atributo de sesión que se utilizará como entrada a Lex mientras se ejecuta la fila del conjunto de pruebas. • El valor de una columna <session-attribute-name>debe proporcionarse <session-attribute-value>con el formato `=`. Se pueden añadir espacios alrededor del símbolo = para facilitar la lectura. • Ejemplos: • session-attribute-foo = esta es una

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				respuesta ficticia • session-attribute-foo = «esta es una «respuesta ficticia» • session-attribute-foo = «esta es una 'respuesta ficticia'» • Se pueden usar varias columnas con nombres como Atributo de sesión 1, Atributo de sesión 2, etc.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
RunTime Sugerencia	Texto y audio	Cadena	Sí	Esta columna se utiliza para proporcionar una pista de tiempo de ejecución para un espacio dentro de una intención que se utilizará como entrada para Lex mientras se ejecuta la fila del conjunto de pruebas. A continuación se muestran algunos ejemplos: • El valor de una columna debe proporcionarse en el formato `<intent-name>. <slot -name>< slot-value>=`. Se pueden añadir espacios alrededor del símbolo = para facilitar la lectura.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)

				• Ejemplos:
				  • IntentNameFoo.slotNameFoo = un valor ficticio
				  • IntentNameFoo.slotNameFoo = 'un «valor ficticio'"
				  • IntentNameFoo.slotNameFoo = «un 'valor ficticio'»
				• Test Workbench no admite ranuras compuestas como sugerencias sobre el tiempo de ejecución.
				• Se pueden usar varias columnas con nombres como «RunTime RunTime Hint

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				1», «Hint 2», etc.
¡Entra en barcaza!	Audio	Booleano	No	Esta columna se utiliza para especificar si Test Workbench debe intervenir al enviar un archivo de audio a Lex Runtime para la fila del conjunto de prueba. • Solo se aplica al conjunto de pruebas de audio para la API de streaming. • Esta columna se ignora al ejecutar un conjunto de pruebas en un modo de API sin transmisi ón.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Intención de salida esperada	Texto y audio	Cadena	No	Esta columna se utiliza para especificar el nombre de una intención esperada en la salida de Lex para la fila del conjunto de pruebas.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Ranura de salida esperada	Texto y audio	Cadena	Sí	Esta columna se utiliza para proporcionar un valor de ranura esperado en la salida de Lex al ejecutar la fila del conjunto de prueba. • < slot-valu e>El valor de una columna debe proporcio narse en el formato `<slot -name>=`. Se pueden añadir espacios alrededor del símbolo = para facilitar la lectura. • Ejemplos de ranuras que no son ranuras de valores múltiples ni ranuras compuestas: • ranura NameFoo

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				= un valor ficticio • slot NameFoo = 'un «valor ficticio"' • slot NameFoo = «un 'valor ficticio'» • Ejemplos de ranuras con varios valores: • ranura NameFoo = valor1, valor2 • ranura NameFoo = valor1, «Objeto de Foo» • ranura NameFoo = valor1, 'valor2' • Ejemplos de ranuras compuestas en las que el nombre de

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				la ranura es «Car» y el nombre de la subranura es «Make»: • Car.Make = Toyota • Car.Make = «Toyota» • Car.Make = 'Toyota' • Se pueden utilizar varias columnas con nombres como: Ranura de salida esperada 1, Ranura de salida esperada 2, etc.

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
Etiqueta de contexto de salida esperada	Texto y audio	Cadena	Sí	Esta columna se usa para especificar el nombre de un contexto de salida esperado en la salida de Lex para la fila del conjunto de prueba. • Esto se refiere al contexto de salida en Cómo establecer un contexto de intención para su bot Lex V2. • Tenga en cuenta que Test Workbench solo admite el nombre del contexto y aún no admite parámetro s dentro del contexto. • Se pueden utilizar varias

Nombre de la columna	tipo de conjunto de pruebas	Tipo de valor	Soporte para múltiples columnas	Description (Descripción)
				columnas con nombres como: Etiqueta de contexto de salida esperada 1, Etiqueta de contexto de salida esperada 2, etc.

Visualización de errores de validación de pruebas en Test Workbench

Puede corregir los conjuntos de prueba que informen de errores de validación. Estos errores de validación se generan cuando un conjunto de prueba no está listo para ser probado. El Test Workbench puede mostrarle qué columnas obligatorias del archivo CSV de entrada del conjunto de prueba no tenían un valor en el formato esperado.

Para ver errores de validación de pruebas:

1. En la lista de conjuntos de prueba, seleccione el nombre del conjunto de prueba que notifique el estado de un Error de validación que desee ver. Los nombres de los conjuntos de prueba son enlaces activos que lo llevan a los detalles relacionados con el conjunto de prueba.
2. El registro del conjunto de prueba muestra detalles de errores de validación en la parte superior de la pantalla. Seleccione Ver detalles para ver el informe sobre los errores de validación.
3. En la ventana del informe de errores, revise el número de línea y el tipo de error para ver dónde se produce el error. Para obtener una lista extensa de errores, puede optar por Descargar el informe de errores.
4. Compare los errores que figuran en el archivo CSV de entrada del conjunto de prueba con el archivo de prueba original para corregir cualquier problema y vuelva a cargar el conjunto de prueba.

En la siguiente tabla se enumeran los mensajes de error de validación de CSV de entrada con escenarios.

Escenario	Mensaje de error	Notas
El tamaño del archivo del conjunto de prueba supera	El tamaño del archivo del conjunto de prueba es superior a 200 MB. Proporcione un archivo más pequeño e intente realizar la solicitud de nuevo.	
El conjunto de prueba supera el número máximo de registros	El archivo de entrada tenía más registros que el número máximo admitido de 200 000.	
Cargue el conjunto de prueba vacío	El conjunto de prueba importado está vacío. Proporcione un conjunto de prueba que no esté vacío e intente realizar la solicitud de nuevo.	
Nombre del encabezado de columna vacío	Fila de encabezados de columna: se encontró un nombre de columna vacío en la columna número 5.	
Nombre de encabezado de columna no reconocido	Fila de encabezados de columna: no se pudo reconocer el nombre de columna «ficticio» en la columna número 2.	
Nombre de encabezado de columna duplicado	Fila de encabezados de columna: se encontraron varias columnas «enlace de audio S3» y «enlace de	

Escenario	Mensaje de error	Notas
	audio S3» que son iguales o equivalentes. Elimine o cambie el nombre de una de esas columnas.	
El nombre de la columna con varios valores ha superado el límite	Fila de encabezados de columna: el número de columnas del «Slot de salida esperado» ha superado el recuento máximo admitido: 6. Elimine algunas columnas de la categoría «Slot de salida esperado» e inténtelo de nuevo.	El número máximo de columnas admitidas para la columna de valores múltiples es 6.
El encabezado de la columna relacionado con el texto o el audio no está presente	No se pudieron encontrar las columnas de las conversaciones de texto o audio. Para las conversaciones de texto, use las columnas {'Entrada de texto'}. Para las conversaciones de audio, use las columnas {'Enlace de audio S3', 'Transcripción esperada'}.	Columnas de audio obligatorias: {'Enlace de audio S3', 'Transcripción esperada'} Columnas de texto obligatorias: {'Entrada de texto'}
Existen encabezados de columna relacionados con el texto y el audio	Se encontraron columnas para conversaciones de texto y audio. Puede usar las columnas {'Entrada de texto'} para las conversaciones de texto o las columnas {'Enlace de audio', 'Transcripción esperada'} para las conversaciones de audio.	Columnas de audio obligatorias: {'Enlace de audio S3', 'Transcripción esperada'} Columnas de texto obligatorias: {'Entrada de texto'}

Escenario	Mensaje de error	Notas
Falta la columna obligatoria	No se encontraron las columnas obligatorias [«Intención de salida esperada»].	Columnas obligatorias: {«Número de línea», «Fuente», «Intención de salida esperada»}
Se encontró un dato en una columna sin encabezado	Se encontraron datos en la columna número 8 de la fila número 6, pero la columna correspondiente no tenía un encabezado de columna.	
No se encontraron datos para las columnas obligatorias	Fila=12: no se encontraron valores para las columnas obligatorias: {«Fuente» , «Intención de salida esperada»}	
Se ha encontrado un ID de conversación duplicada	La conversación número '19' se vio en una conversación anterior en la fila número «39». Asegúrese de que no se haya proporcionado el mismo número de conversación para dos conversaciones. Para ello, asegúrese de que todas las filas de un número de conversación estén agrupadas .	

Escenario	Mensaje de error	Notas
Se ha proporcionado un identificador de conversación no válido	Se encontró un valor no válido «conversación_prueba» en la columna «Número de conversación». El valor de esta columna debe ser numérico o N/A (es decir, no aplicable) debe corresponder a una fila de usuarios.	
Se ha proporcionado un valor no numérico para el número de línea	Se encontró el valor no numérico 'línea_prueba' en la columna 'Número de línea'. Su valor debe ser numérico.	
No se encontró la conversación en la fila de agentes	No se encontró ningún valor para la columna «Número de conversación». Debe proporcionarse para una fila de agentes.	
No se encontró ID de conversación no numérico en la fila de agentes	Se encontró un valor no numérico «conversación_prueba» en la columna «Número de conversación». Su valor debe ser numérico para una fila de agentes.	
Ubicación de S3 no válida	Se ha proporcionado un valor bucket/folder '' no válido. El formato válido es S3://<NombreBucket>/<NombreClave>.	

Escenario	Mensaje de error	Notas
Nombre de bucket de S3 no válido	Se ha proporcionado un nombre de bucket de S3, «bucket_prueba», no válido. Compruebe el nombre del bucket.	
La ubicación del audio de S3 es la carpeta	La ubicación de audio proporcionada «S3://bucket/folder» no es válida. Señala a una carpeta S3.	
Nombre de intención no válido	Había caracteres no válidos en la intención «intención@nombre». Compruebe el nombre de la intención.	Comprobación de expresiones regulares: ^ ([0-9a-zA-Z] [_-]?) +$
Nombre de slot no válido	Había caracteres no válidos en el slot «Slot@Nombre». Compruebe el nombre del slot.	Regexa: ^ ([0-9a-zA-Z] [_-]?) +$No debe empezar ni terminar con un punto (.)
Se proporcionó un valor de slot para el slot principal	Se proporcionaron valores de ranura para la subranura «Address.City» y para la ranura principal «Dirección». Los valores solo deben proporcionarse para el subslot.	El slot principal en CST no debe tener un valor de slot
Carácter no válido en el nombre del contexto	Había caracteres no válidos en el nombre de contexto 'contexto@1 '. Compruebe el nombre del contexto.	Regex: ^ ([] _?) A-Za-z +$

Escenario	Mensaje de error	Notas
Estilo ortográfico de slot no válido	Se ha proporcionado un valor «prueba» no válido. Asegúrese de que estén todos en mayúscula. Los valores válidos son ["Default», "SpellByLetter«," SpellByWord «].	Valores admitidos ["Predeterminado», "SpellByLetter«,"SpellByWord»
El participante o la fuente debe ser un agente o un usuario	Se ha proporcionado un valor «bot» no válido. Los valores válidos son [«Agente, «Usuario»].	Enumeraciones compatibles: «Agente, «Usuario»
El número de línea no debe ser decimal	Se ha proporcionado un valor «10.1» no válido. Debe ser un número válido sin fracciones.	
El número de conversación no debe ser decimal	Se ha proporcionado un valor «10.1» no válido. Debe ser un número válido sin fracciones.	
El número de línea debe estar dentro del rango	Se ha proporcionado un valor «92233720368547758071» no válido. Debe ser mayor o igual que 1 y menor o igual que 9223372036854775807.	
Barge-in la columna solo acepta valores booleanos	Se ha proporcionado un valor «prueba» no válido. Debe ser un valor booleano válido, como «verdadero» o «falso». Alternativamente, se puede usar «sí» y «no».	Valores posibles: «Verdadero», «verdadero», «V», «Sí», «sÍ», «S», «1», «1.0», «Falso», «falso», «F», «No», «no», «N», «0» , «0.0»

Escenario	Mensaje de error	Notas
El slot esperado, el atributo de sesión y el atributo de solicitud deben estar separados por un valor igual a (=)	El valor Name:slotValue «ranura» no tiene «=». Este valor debe proporcionarse como un par clave-valor en el formato '<clave>=<valor>'.	Por ejemplo: NombreSlot = TipoSlot
El slot esperado, el atributo de sesión y el atributo de solicitud deben tener un par clave-valor	'=ValorSlot' no tiene una clave antes de '='. Este valor debe proporcionarse como un par clave-valor en el formato '<clave>=<valor>'.	Por ejemplo: NombreSlot = TipoSlot
La cita al final no es válida	Se encontró una cita incorrect a en «El artículo de Foo». Comienza con el carácter de comilla `«`, pero no termina con el mismo carácter de comilla.	Por ejemplo: `«Foo's item», KFC`
Cita no válida en el centro	Se encontró una cita incorrect a en `"Foo's» Burger, etc.`. Contiene la comilla `»` dentro de su contenido. Los valores que contienen comillas simples deben estar entre comillas dobles y viceversa.	Correcto: Por ejemplo: `«Foo's item», KFC`
Cotizaciones obligatorias	`key = Foo's item` contiene comillas simples o dobles, pero no está entre comillas. Los valores que contienen comillas simples deben estar entre comillas dobles y viceversa.	

Escenario	Mensaje de error	Notas
La clave duplicada se repite en la columna	La clave `clave1` se repitió en dos columnas: `Atributo de sesión 3` y `Atributo de sesión 1`.	
El formato no es válido en la sugerencia de tiempo de ejecución	BookFlight.CarClave no válida `. «`proporcionada para Runtime Hints. En el caso de las sugerencias sobre el tiempo de ejecución , la clave debe estar en formato <intenciónNombre>. <slotNombre>.	Si '.' debe estar presente en medio de la clave, el nombre de la intención y el nombre de la ranura no se pueden extraer de dicha clave. Ejemplos de este tipo de formato incorrect o: "BookFlight«,». BookFligh t.Car«,"BookFlight.Car.»
El nombre de intención no es válido en la clave de sugerenci a de tiempo de ejecución	Se ha encontrado la intención `intención@nombre` no válida en las sugerencias del tiempo de ejecución. Compruebe el nombre de la intención.	Verificación de expresiones regulares: ^ ([0-9a-zA-Z] [_-]?) +$
El nombre del slot no es válido en la clave de sugerencia de tiempo de ejecución	Se ha encontrado un nombre de slot no válido en `Slot@Nombre` para sugerencias en tiempo de ejecución. Compruebe el nombre del slot.	Regexa: ^ ([0-9a-zA-Z] [_-]?) + $No debe empezar ni terminar con un punto (.)

Eliminación de un conjunto de prueba en Test Workbench

Puede eliminar fácilmente un conjunto de prueba de su lista de conjuntos de prueba.

Eliminar un conjunto de prueba:

1. Vaya a la lista de Conjuntos de prueba en el menú de la izquierda lateral para ver la lista de conjuntos de prueba.

2. De la lista de conjuntos de prueba, seleccione el conjunto de prueba que desea eliminar.
3. Vaya al menú desplegable Acciones en la parte superior derecha y seleccione Eliminar.
4. Un mensaje confirma que se ha eliminado el conjunto de prueba.

Editar detalles del conjunto de prueba

Puede editar el nombre y los detalles de un conjunto de prueba en la lista de conjuntos de prueba. El nombre o los detalles se pueden añadir o actualizar más adelante. Sin embargo, tendrá que actualizar su conjunto de prueba antes de ejecutar la prueba con su bot o con los datos de transcripción.

Para editar detalles del conjunto de prueba:

1. Vaya a la lista de conjuntos de prueba en el menú de la izquierda lateral para ver la lista de conjuntos de prueba.
2. De la lista de conjuntos de prueba, seleccione la casilla para el conjunto de prueba que desea editar.
3. Vaya al menú desplegable Acciones en la parte superior derecha y seleccione Editar detalles.
4. Un mensaje confirma que el conjunto de prueba se ha editado correctamente.

Actualizar conjunto de prueba

Puede actualizar, corregir, modificar o eliminar elementos del conjunto de prueba para optimizar los resultados de referencia o para corregir otros errores que puedan haberse producido en el conjunto de prueba

Puede descargar un conjunto de prueba y corregir los errores de validación antes de subir el conjunto de prueba corregido. Ver Errores de validación de pruebas.

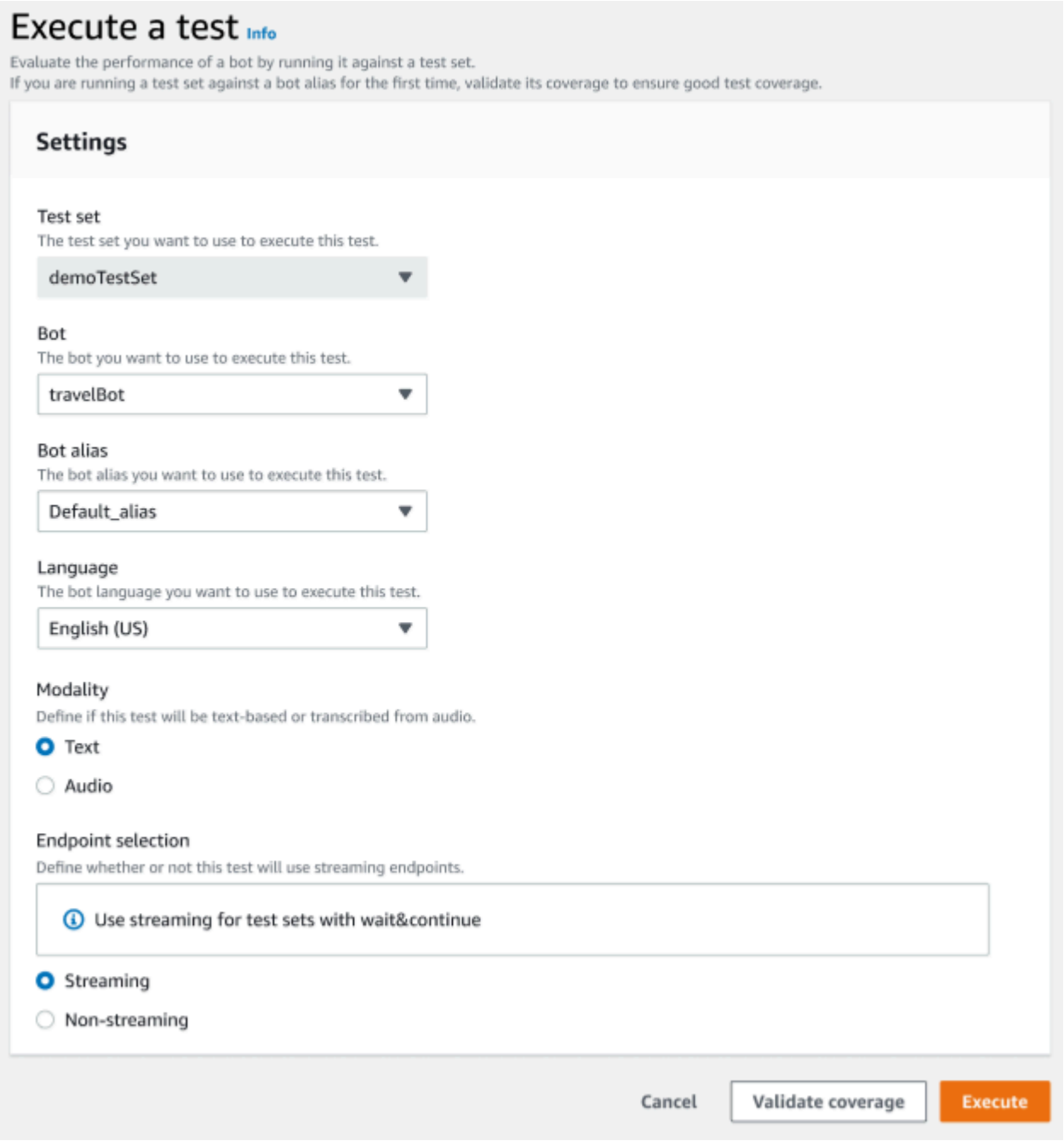
Para actualizar un conjunto de datos de prueba:

1. En el registro del conjunto de prueba, pulse el botón Actualizar conjunto de prueba situado en la parte superior derecha.
2. Seleccione un archivo para cargarlo desde su cuenta de Amazon S3 o cargue un archivo de prueba CSV desde su ordenador. NOTA: Al actualizar un conjunto de prueba, se sobrescribirán los datos existentes.

3. Seleccione el botón Actualizar.
4. Un mensaje confirma que el conjunto de prueba se ha editado correctamente. NOTA: Esta operación puede tardar unos minutos, según la complejidad y el tamaño del conjunto de prueba.
5. Un mensaje confirma que el conjunto de prueba se ha actualizado correctamente y el Estado muestra Listo para la prueba.

Ejecutar una prueba

Para ejecutar un conjunto de prueba, debe elegir el bot adecuado para ejecutar la prueba con el conjunto de prueba. Puede elegir un bot de su cuenta de AWS en el menú desplegable de Test Set. Esta operación pondrá a prueba el bot seleccionado con los datos de prueba validados para generar estadísticas de rendimiento comparándolas con los datos de referencia del conjunto de prueba.



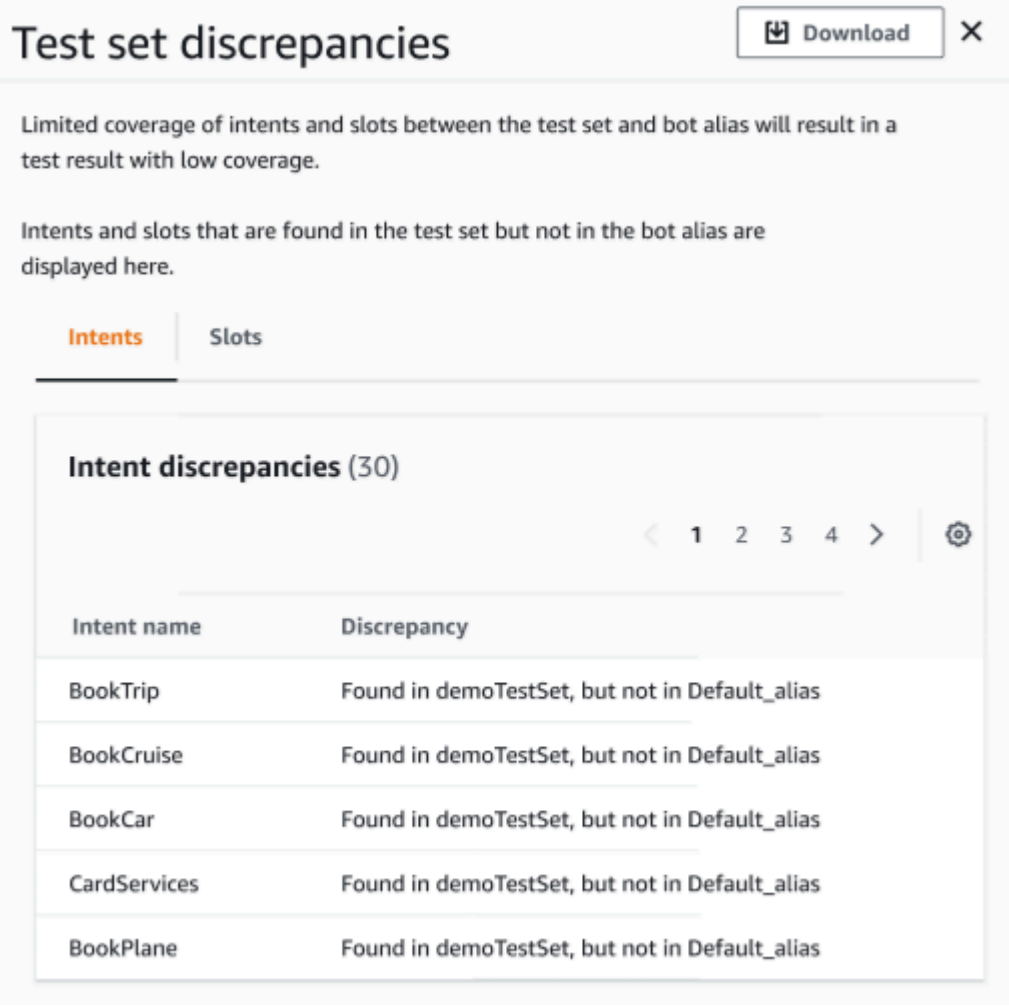
Ejecutar una prueba en el Test Workbench

1. En la página de registro del conjunto de prueba, seleccione Ejecutar prueba.
2. Seleccione el conjunto de prueba que desee utilizar en la prueba.
3. Seleccione el nombre del bot que quiere usar en la prueba en el menú desplegable del bot.
4. Seleccione un alias de bot, si corresponde, en el menú desplegable del alias del bot.
5. En la selección de Idiomas, seleccione una versión en inglés.

6. Seleccione Texto o Audio para el tipo de modalidad.
7. Seleccione su ubicación de Amazon S3. (solo audio)
8. Seleccione su Selección de punto de conexión para su bot. (solo streaming)
9. Seleccione el botón Validar cobertura para confirmar que la prueba está lista para ejecutarse. Si hay algún error en el paso de validación, revise los parámetros anteriores y realice las correcciones necesarias.
10. Seleccione Ejecutar para ejecutar la prueba.
11. Un mensaje confirma que la prueba se ha ejecutado correctamente.

Cobertura del conjunto de prueba en Test Workbench

La cobertura limitada de las intenciones y los slots entre el conjunto de prueba y el bot puede dar como resultado las medidas de rendimiento esperadas. Le recomendamos que revise la cobertura del conjunto de prueba antes de realizar la prueba.



Revisar la cobertura de validación

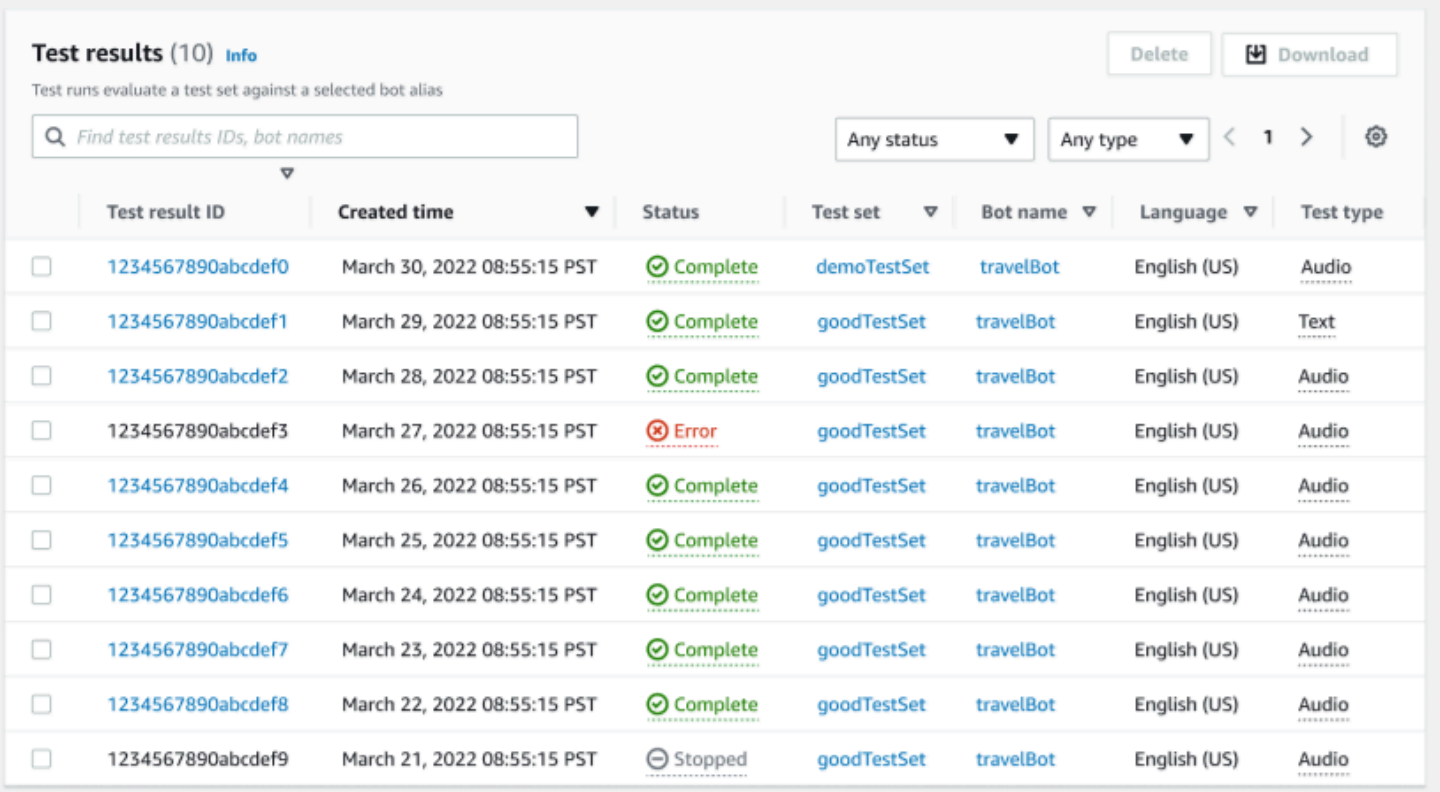
1. En los registros del conjunto de prueba, pulse el botón Validar cobertura.
2. El mensaje indica que está validando la cobertura entre el conjunto de prueba y el bot seleccionado.
3. Una vez completada la operación, el mensaje indica que la Validación de la cobertura se ha realizado correctamente.
4. Pulse el botón Ver detalles en la parte inferior de la ventana.
5. Para ver las discrepancias del conjunto de prueba en cuanto a intenciones y slots, seleccione la pestaña correspondiente a cada una de ellos. Puede descargar estos datos en formato CSV pulsando el botón Descargar.
6. Revise los resultados de la validación de los datos del conjunto de prueba, las intenciones de los bots y los slots. Identifique los problemas y realice cambios en la arquitectura del conjunto de prueba de su bot para mejorar los resultados. Cargue el conjunto de pruebas editado y el bot para ejecutar la prueba una vez que haya realizado los cambios en el archivo CSV. NOTA: La cobertura de validación se aplica al conjunto de pruebas y no al bot. No se tendrán en cuenta las intenciones del bot que no estén presentes en el conjunto de prueba.

Ver resultados de la prueba

Interprete los resultados de las pruebas de Test Workbench para determinar en qué aspectos la conversación entre el bot y el cliente podría estar fallando o requiriendo que el cliente haga varios intentos para cumplir la intención.

Si localiza estos problemas en los resultados de las pruebas, puede optimizar el rendimiento de su bot mejorando el rendimiento de la intención utilizando diferentes datos de entrenamiento o enunciados que sean más coherentes con los valores de transcripción del bot en tiempo real.

Puede obtener una visión detallada de las intenciones y los slots en los que hay discrepancias de rendimiento. Una vez que haya identificado las intenciones o los slots que presentan discrepancias, puede profundizar más y revisar los enunciados y el flujo de la conversación.



Para revisar los resultados de la prueba:

1. Vaya a la lista de conjuntos de prueba del menú lateral de la izquierda y seleccione la opción Resultados de la prueba en el Test Workbench. NOTA: Los resultados de las pruebas indican un Estado completado si se han realizado correctamente.
2. Seleccione el Identificador del resultado de la prueba para los resultados de la prueba que desee revisar.

Detalles de los resultados de la prueba en Test Workbench

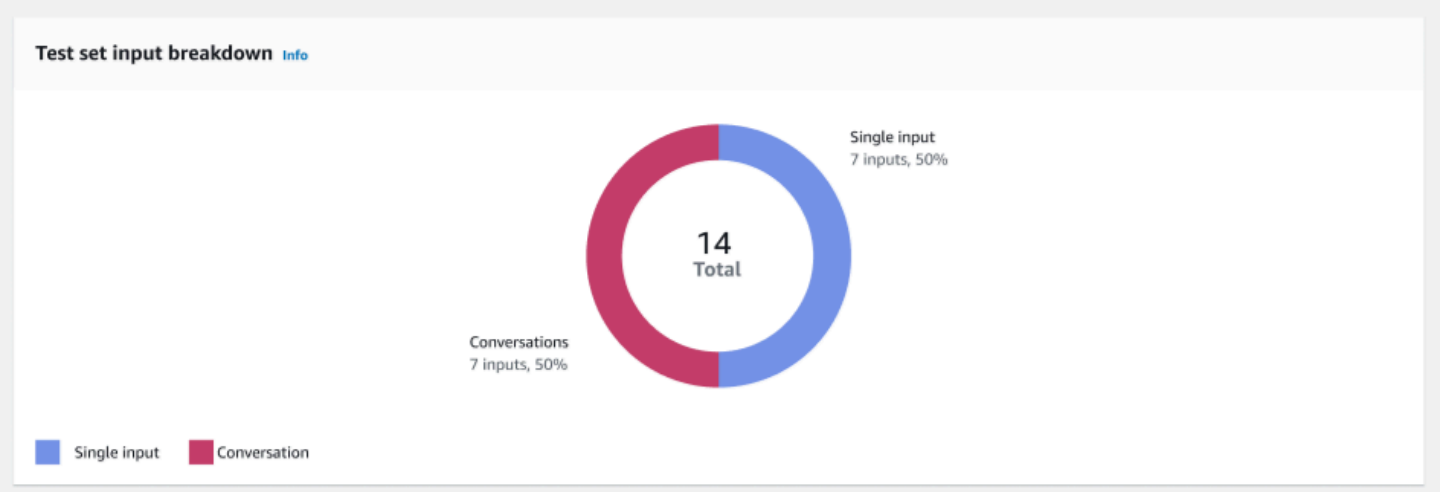
Los resultados de la prueba muestran los detalles del conjunto de prueba, las intenciones utilizadas y los slots usados. También proporciona el desglose general de las entradas del conjunto de prueba, que incluye los resultados generales, los resultados de la conversación, la intención y los resultados de los slots.

Los resultados de las pruebas incluyen toda la información relacionada con las pruebas, como:

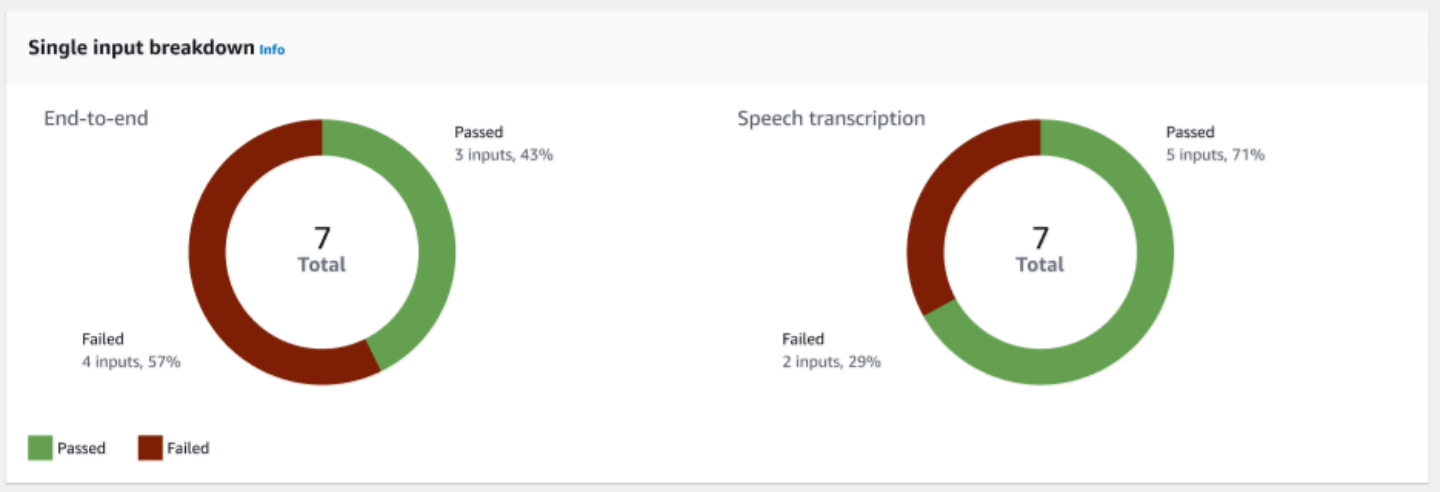
- Metadatos de detalles de prueba
- Resultados generales

- Resultados de la conversación
- Resultados de intención y slot
- Resultados detallados

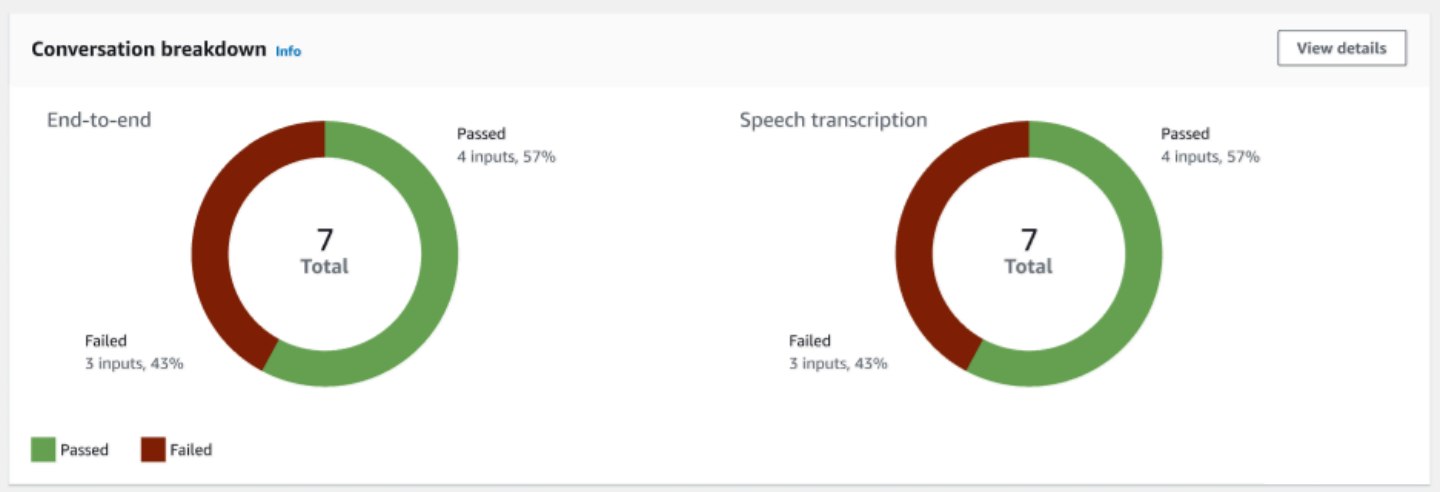
Pestaña de resultados generales:



Desglose de entradas del conjunto de prueba: este gráfico muestra el desglose del número de conversaciones y enunciados de entrada única en el conjunto de prueba.

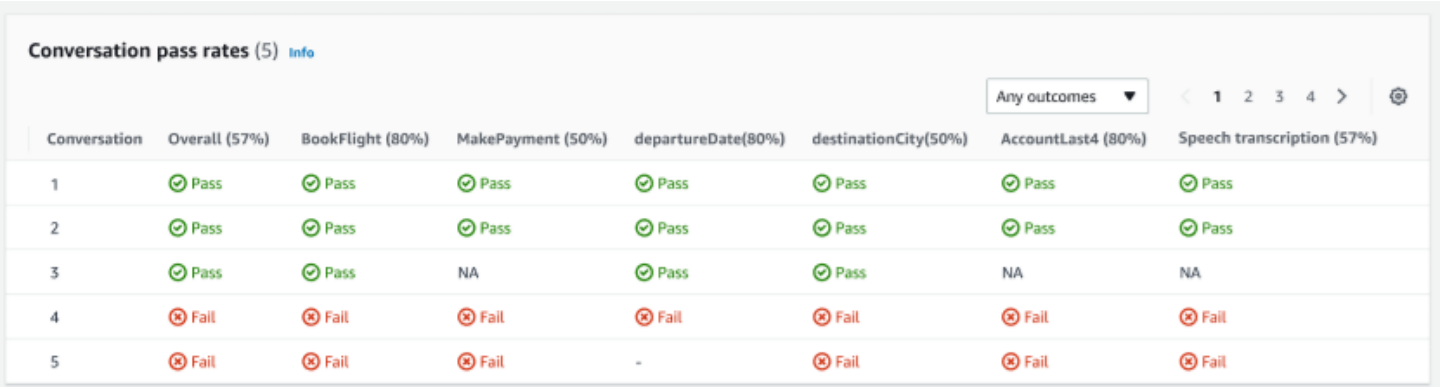


Desglose de entrada única: muestra dos gráficos que incluyen conversaciones de principio a fin y transcripciones de voz. El número de entradas aprobadas y fallidas se indica en cada gráfico. Nota: La tabla de transcripción de voz solo estará visible para el conjunto de prueba de audio.



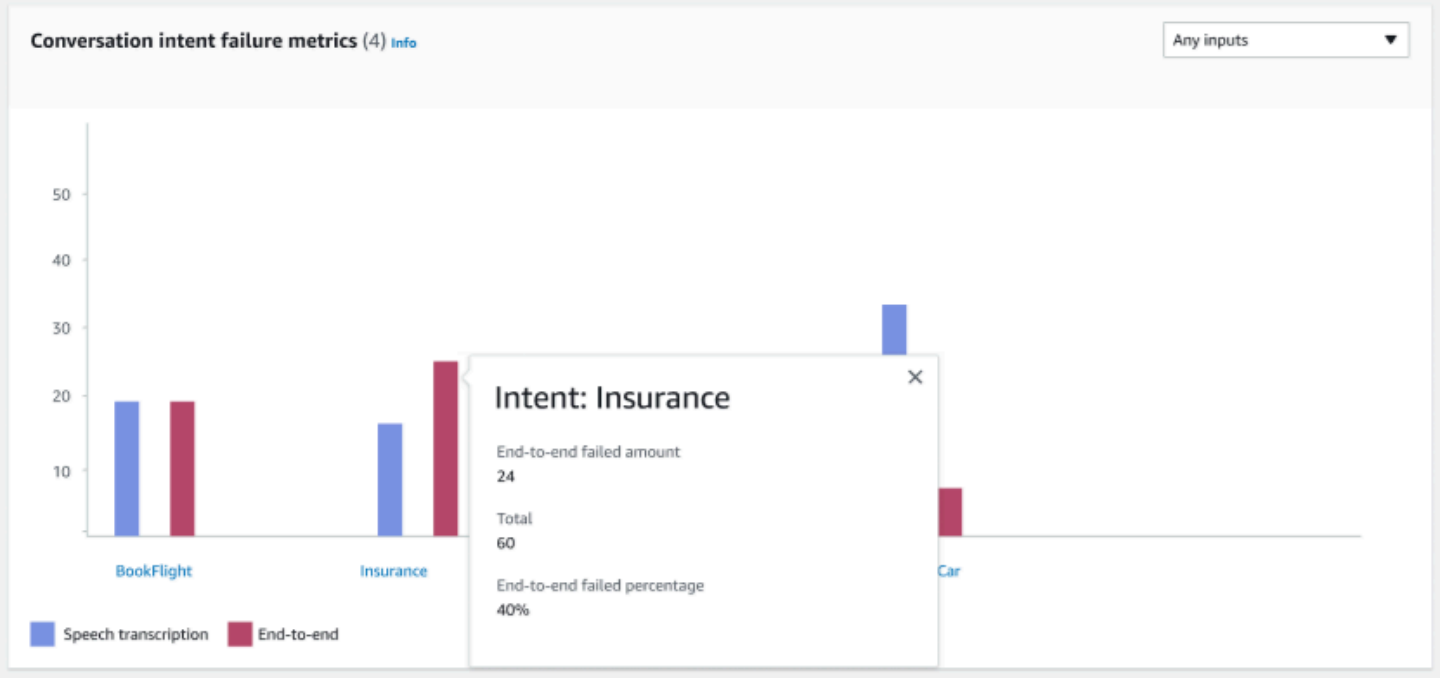
Desglose de conversación: muestra dos gráficos que incluyen conversaciones de principio a fin y transcripciones de voz. El número de entradas aprobadas y fallidas se indica en cada gráfico. Nota: La tabla de transcripción de voz solo estará visible para el conjunto de prueba de audio.

Pestaña de los resultados de la conversación:



Conversation	Overall (57%)	BookFlight (80%)	MakePayment (50%)	departureDate(80%)	destinationCity(50%)	AccountLast4 (80%)	Speech transcription (57%)
1	Pass	Pass	Pass	Pass	Pass	Pass	Pass
2	Pass	Pass	Pass	Pass	Pass	Pass	Pass
3	Pass	Pass	NA	Pass	Pass	NA	NA
4	Fail	Fail	Fail	Fail	Fail	Fail	Fail
5	Fail	Fail	Fail	-	Fail	Fail	Fail

Tasas de aprobación de conversaciones: la tabla de tasas de aprobación de conversaciones se utiliza para ver qué intenciones y slots se utilizan en cada conversación del conjunto de pruebas. Para ver dónde ha fallado la conversación, revise qué intención o slot falló, junto con el porcentaje de aprobaciones de cada intento y slot.

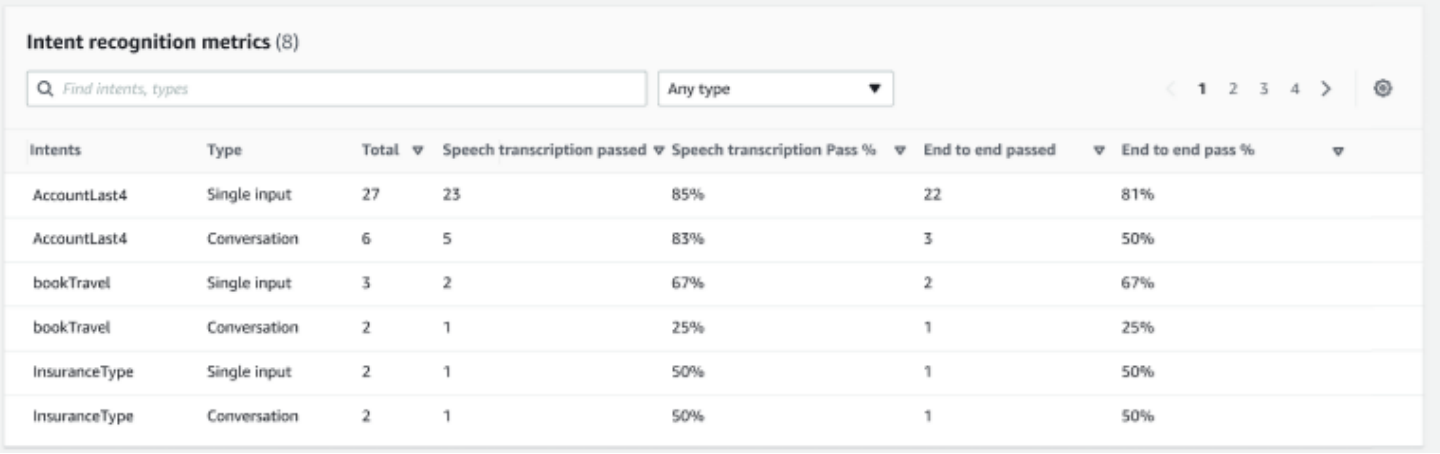


Métricas de fallos en la intención de la conversación: esta métrica muestra las cinco intenciones con peor rendimiento del conjunto de prueba. En este panel se muestra un gráfico con el porcentaje o el número de intenciones que tuvieron éxito o fallaron, en función de los registros de conversaciones o de la transcripción del bot. Una intención exitosa no significa que toda la conversación haya sido un éxito. Estas métricas solo se aplican al valor de las intenciones, independientemente de la intención anterior o posterior.



Métricas de fallos en el slot de la conversación: esta métrica muestra los cinco slots con peor rendimiento del conjunto de prueba. Indicó la tasa de éxito de cada slot de la intención. El gráfico de barras muestra tanto la transcripción del discurso como las conversaciones de principio a fin para cada slot de la intención.

Pestaña de resultados de intención y slot:

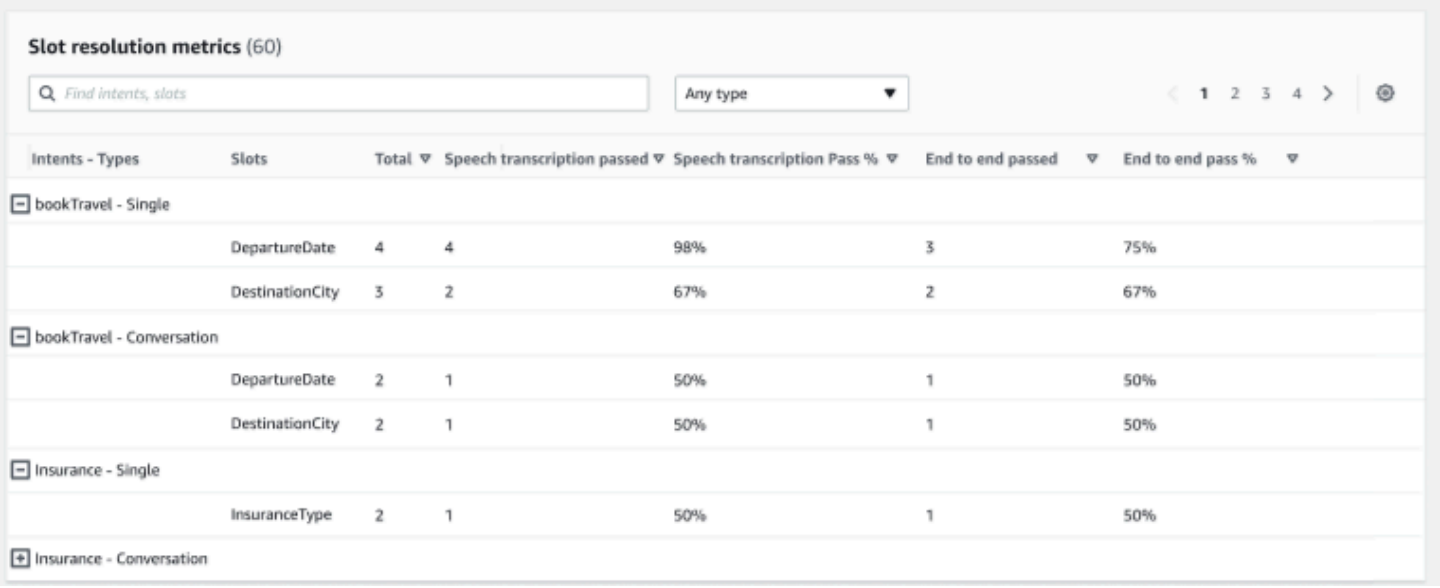


Intent recognition metrics (8)

Find intents, types | Any type | 1 2 3 4

Intents	Type	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
AccountLast4	Single input	27	23	85%	22	81%
AccountLast4	Conversation	6	5	83%	3	50%
bookTravel	Single input	3	2	67%	2	67%
bookTravel	Conversation	2	1	25%	1	25%
InsuranceType	Single input	2	1	50%	1	50%
InsuranceType	Conversation	2	1	50%	1	50%

Métricas de reconocimiento de intenciones: muestra una tabla de cuántas intenciones se reconocieron correctamente. Muestra la tasa de aprobación de la transcripción de voz y de las conversaciones de principio a fin.



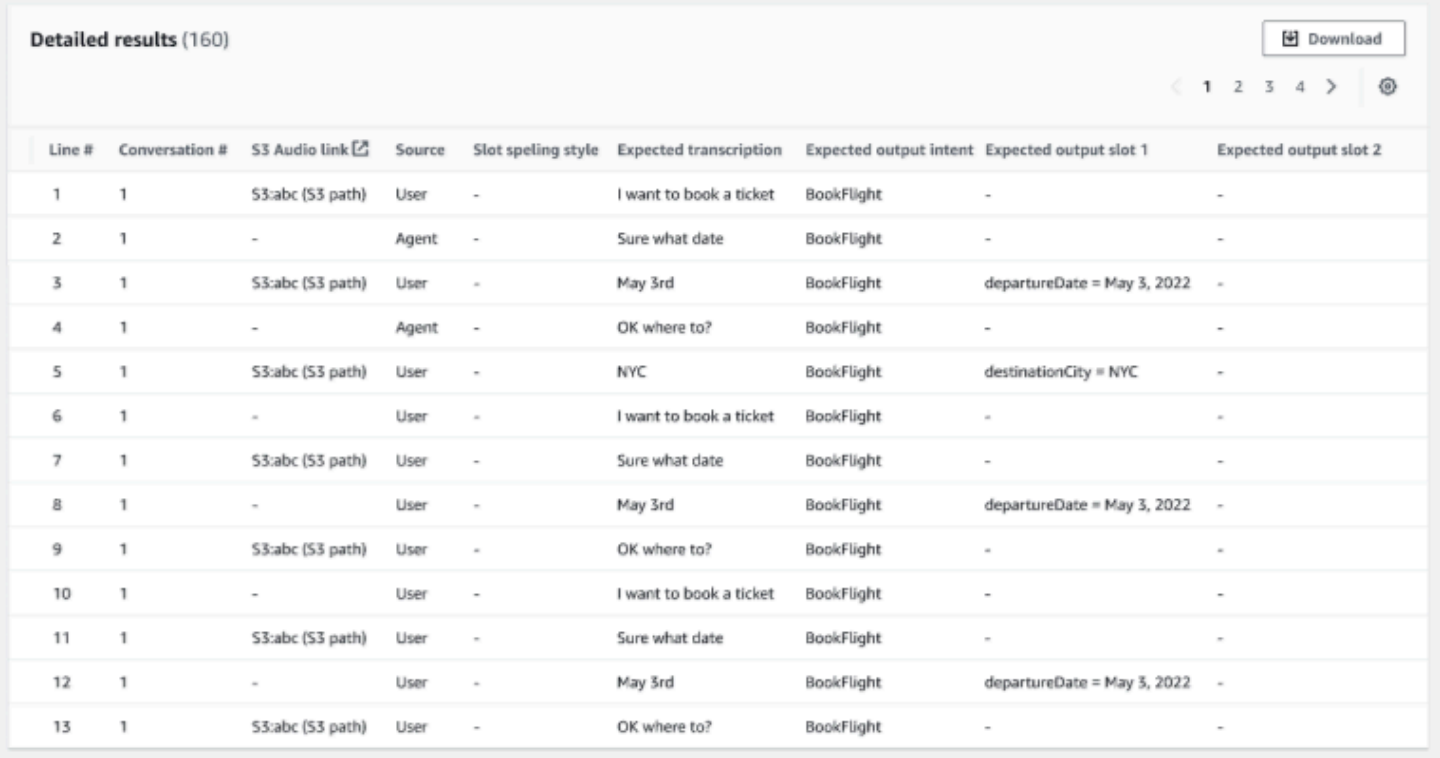
Slot resolution metrics (60)

Find intents, slots | Any type | 1 2 3 4

Intents - Types	Slots	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
bookTravel - Single						
	DepartureDate	4	4	98%	3	75%
	DestinationCity	3	2	67%	2	67%
bookTravel - Conversation						
	DepartureDate	2	1	50%	1	50%
	DestinationCity	2	1	50%	1	50%
Insurance - Single						
	InsuranceType	2	1	50%	1	50%
Insurance - Conversation						

Métricas de resolución de slots: muestra las intenciones y los slots por separado, así como la tasa de éxito y fracaso de cada slot para cada intención utilizada en la conversación o en una sola entrada. Muestra la tasa de aprobación de la transcripción de voz y de las conversaciones de principio a fin.

Pestaña de resultados detallados:



Detailed results (160)

Download

1 2 3 4

Line #	Conversation #	S3 Audio link	Source	Slot speling style	Expected transcription	Expected output intent	Expected output slot 1	Expected output slot 2
1	1	S3:abc (S3 path)	User	-	I want to book a ticket	BookFlight	-	-
2	1	-	Agent	-	Sure what date	BookFlight	-	-
3	1	S3:abc (S3 path)	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
4	1	-	Agent	-	OK where to?	BookFlight	-	-
5	1	S3:abc (S3 path)	User	-	NYC	BookFlight	destinationCity = NYC	-
6	1	-	User	-	I want to book a ticket	BookFlight	-	-
7	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
8	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
9	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-
10	1	-	User	-	I want to book a ticket	BookFlight	-	-
11	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
12	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
13	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-

Resultados detallados: muestra una tabla detallada en el registro de conversaciones con los enunciados del usuario y el agente, así como el resultado esperado y la transcripción prevista para cada slot. Puede descargar este informe pulsando el botón Descargar.

En la siguiente tabla se enumeran los mensajes de error de fallos de resultados con escenarios.

Escenario	Mensaje de error	Action
Discrepancia de intenciones	BookFlight Intención esperada, pero era BookHotel intención.	Sáltese otros turnos de la conversación
Discrepancia en la obtención de slots	Se esperaba que se abriera el slot FechaDeSalida, pero era TipoDeCabina.	Sáltese otros turnos de la conversación
Discrepancia en el valor del slot	Discrepancia entre el valor esperado y el real de los slots.	Continúe con los demás turnos de las conversaciones
Back-to-back falta el aviso del agente	Se esperaba que el bot devolviera un mensaje de	Sáltese otros turnos de la conversación

Escenario	Mensaje de error	Action
	agente en este turno, pero no lo recibió.	
Discrepancia en la transcripción	La transcripción esperada no coincidía con la transcripción real.	Continúe con los demás turnos de las conversaciones
No se ha obtenido el slot opcional	Se espera que aparezca el slot TipoDeCabina en el siguiente turno, sin embargo, la intención actual se cumplió antes de eso.	Sáltese otros turnos de la conversación
Slot no reconocido	El slot esperado FechaDeSalida no se reconoció en este turno.	Sáltese otros turnos de la conversación
Aviso de agente adicional consecutivo	Se esperaba un turno de usuario, pero fue un aviso del agente	Sáltese otros turnos de la conversación

Transmisión de conversaciones a un bot Amazon Lex V2

Puede utilizar la API de transmisión de Amazon Lex V2 para iniciar una transmisión bidireccional entre un bot de Amazon Lex V2 y su aplicación. Al iniciar una transmisión, el bot puede gestionar la conversación entre el bot y el usuario. El bot responde a las entradas del usuario sin necesidad de escribir código para gestionar las respuestas del usuario. El bot puede:

- Gestionar las interrupciones del usuario mientras reproduce un mensaje. Para obtener más información, consulte Cómo permitir que su usuario interrumpa a su bot Amazon Lex V2.
- Esperar a que el usuario introduzca información. Por ejemplo, el bot puede esperar a que el usuario recopile la información de la tarjeta de crédito. Para obtener más información, consulte Cómo permitir que el bot Amazon Lex V2 espere a que el usuario proporcione más información durante una pausa.
- Utilizar la entrada de audio y de doble tono (DTMF) y la entrada de audio en la misma transmisión.
- Gestionar mejor las pausas en las entradas de los usuarios que si gestionara la conversación desde su aplicación.

El bot Amazon Lex V2 no solo responde a los datos enviados desde su aplicación, sino que también envía información sobre el estado de la conversación a su aplicación. Puede usar esta información para cambiar la forma en que la aplicación responde a los clientes.

El bot de Amazon Lex V2 también supervisa la conexión entre el bot y su aplicación. Puede determinar si se ha agotado el tiempo de espera de la conexión.

Para usar la API para iniciar una transmisión a un bot de Amazon Lex V2, consulte Inicio de una transmisión de conversación a un bot de Amazon Lex V2.

Cuando comience a transmitir a un bot de Amazon Lex V2 desde su aplicación, puede configurar el bot para que acepte la entrada de audio o texto del usuario. También puede elegir si el usuario recibe audio o texto en respuesta a su entrada.

Si ha configurado el bot Amazon Lex V2 para que acepte entradas de audio del usuario, no podrá introducir texto. Si ha configurado el bot para que acepte entradas de texto, el usuario solo podrá usar texto escrito para comunicarse con él.

Cuando un bot de Amazon Lex V2 recibe una entrada de audio en streaming, el bot determina cuándo un usuario empieza a hablar y cuándo deja de hablar. Gestiona cualquier pausa o

interrupción por parte del usuario. También puede admitir entradas DTMF (multifrecuencia de doble tono) y entradas de voz en la misma transmisión. Esto ayuda al usuario a interactuar con el bot de forma más natural. Puede enviar mensajes e indicaciones de bienvenida a los usuarios. También puede permitir que los usuarios interrumpan esos mensajes e indicaciones.

Al iniciar una transmisión bidireccional, Amazon Lex V2 utiliza el HTTP/2 protocolo. La aplicación y el bot intercambian datos en una sola transmisión en forma de una serie de eventos. Un evento puede ser uno de los siguientes:

- Entrada de texto, audio o DTMF del usuario.
- Señales de la aplicación al bot de Amazon Lex V2. Estas incluyen una indicación de que se ha completado la reproducción de audio de un mensaje o de que el usuario se ha desconectado de la sesión.

Para obtener más información sobre los eventos, consulte Inicio de una transmisión de conversación a un bot de Amazon Lex V2. Para obtener información sobre cómo agregar un disco, consulte Codificar secuencias de eventos.

Temas

- Inicio de una transmisión de conversación a un bot de Amazon Lex V2
- Codificar secuencias de eventos
- Cómo permitir que su usuario interrumpa a su bot Amazon Lex V2
- Cómo permitir que el bot Amazon Lex V2 espere a que el usuario proporcione más información durante una pausa
- Configuración de las actualizaciones del progreso de cumplimiento para su bot Lex V2
- Configuración de los tiempos de espera para capturar entradas del usuario con un bot Lex V2

Inicio de una transmisión de conversación a un bot de Amazon Lex V2

Utiliza la StartConversationoperación para iniciar una transmisión entre el usuario y el bot Amazon Lex V2 de su aplicación. La solicitud `POST` de la aplicación establece una conexión entre su aplicación y el bot Amazon Lex V2. Esto permite que la aplicación y el bot comiencen a intercambiar información entre sí a través de eventos.

> **Note**
>
> Cuando se utiliza StartConversation, si la duración del audio del enunciado supera el valor configurado`max-length-ms`, Amazon Lex V2 interrumpe el audio con la duración especificada.

La operación `StartConversation` solo es compatible con los siguientes SDK:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para la JavaScript versión 3
- AWS SDK para Ruby V3

El primer evento que su solicitud debe enviar al bot Amazon Lex V2 es un ConfigurationEvent. Este evento incluye información como el formato del tipo de respuesta. Los siguientes son los parámetros que puede usar en un evento de configuración:

- respuesta ContentType: determina si el bot responde a las entradas del usuario con texto o voz.
- Estado de la sesión: información relacionada con la sesión de transmisión con el bot, como la intención predeterminada o el estado del diálogo.
- Mensajes de bienvenida: especifica los mensajes de bienvenida que se reproducen para el usuario al principio de su conversación con un bot. Estos mensajes se reproducen antes de que el usuario introduzca nada. Para activar un mensaje de bienvenida, también debe especificar valores para los parámetros `sessionState` y `dialogAction`.
- Deshabilitar reproducción: determina si el bot debe esperar una señal del cliente antes de empezar a escuchar las entradas de la persona que llama. De forma predeterminada, la reproducción está activada, por lo que el valor de este campo es `false`.
- Solicitar atributos: proporciona información adicional para la solicitud.

Para obtener información sobre cómo especificar los valores de los parámetros anteriores, consulte el tipo de ConfigurationEventdatos de la StartConversationoperación.

Cada transmisión entre un bot y su aplicación solo puede tener un evento de configuración. Una vez que la aplicación haya enviado un evento de configuración, el bot puede recibir comunicaciones adicionales desde la aplicación.

Si especificó que su usuario utiliza el audio para comunicarse con el bot de Amazon Lex V2, su aplicación puede enviar los siguientes eventos al bot durante esa conversación:

- AudioInputEvent— Contiene un fragmento de audio con un tamaño máximo de 320 bytes. La aplicación debe usar varios eventos de entrada de audio para enviar un mensaje desde el servidor al bot. Todos los eventos de entrada de audio de la transmisión deben tener el mismo formato de audio.
- DTMFInputEvent— Envía una entrada DTMF al bot. Cada pulsación de tecla DTMF corresponde a un único evento.
- PlaybackCompletionEvent— Informa al servidor de que se le ha reproducido una respuesta de la entrada del usuario. Debe utilizar un evento de finalización de la reproducción si va a enviar una respuesta de audio al usuario. Si `disablePlayback` de su evento de configuración es `true`, no puede usar esta funcionalidad.
- DisconnectionEvent— Informa al bot de que el usuario se ha desconectado de la conversación.

Si especificó que el usuario utiliza el texto para comunicarse con el bot, su aplicación puede enviar los siguientes eventos al bot durante esa conversación:

- TextInputEvent— Texto que se envía desde tu aplicación al bot. Puede tener 512 caracteres como máximo en un evento de entrada de texto.
- PlaybackCompletionEvent— Informa al servidor de que se le ha reproducido una respuesta de la entrada del usuario. Debe usar este evento si está reproduciendo audio para el usuario. Si `disablePlayback` de su evento de configuración es `true`, no puede usar esta funcionalidad.
- DisconnectionEvent— Informa al bot de que el usuario se ha desconectado de la conversación.

Debe codificar todos los eventos que envíe a un bot de Amazon Lex V2 en el formato correcto. Para obtener más información, consulte Codificar secuencias de eventos.

Cada evento tiene un ID de evento. Para ayudar a solucionar cualquier problema que pueda producirse en la transmisión, asigne un ID de evento único a cada evento de entrada. A continuación, podrá solucionar cualquier fallo de procesamiento con el bot.

Amazon Lex V2 también utiliza marcas de tiempo para cada evento. Puede usar estas marcas de tiempo además del ID del evento para ayudar a solucionar cualquier problema de transmisión de la red.

Durante la conversación entre el usuario y el bot de Amazon Lex V2, el bot puede enviar los siguientes eventos salientes en respuesta al usuario:

- IntentResultEvent— Contiene la intención que Amazon Lex V2 determinó a partir de la declaración del usuario. Cada evento de resultado de intención incluye:
  - Modo de entrada: el tipo de enunciado del usuario. Los valores válidos son `Speech`, `DTMF` o `Text`.
  - interpretaciones: interpretaciones que Amazon Lex V2 determina a partir del enunciado del usuario.
  - Atributos de solicitud: si no ha modificado los atributos de la solicitud mediante una función de Lambda, estos son los mismos atributos que se pasaron al inicio de la conversación.
  - ID de sesión: identificador de sesión utilizado para la conversación.
  - Estado de sesión: el estado de la sesión del usuario con Amazon Lex V2.
- TranscriptEvent— Si el usuario proporciona una entrada a su aplicación, este evento contiene la transcripción de lo que el usuario ha dicho al bot. Su aplicación no recibe un `TranscriptEvent` si no hay ninguna entrada del usuario.

  El valor del evento de transcripción enviado a su aplicación depende de si ha especificado el audio (voz y DMTF) o el texto como modo de conversación:
  - Transcripción de la entrada de voz: si el usuario está hablando con el bot, el evento de transcripción es la transcripción del audio del usuario. Es una transcripción de todo el discurso desde el momento en que el usuario comienza a hablar hasta el momento en que termina de hablar.
  - Transcripción de la entrada DTMF: si el usuario está escribiendo en un teclado, el evento de transcripción contiene todos los dígitos que el usuario presionó en la entrada.
  - Transcripción de la entrada de texto: si el usuario introduce texto, el evento de transcripción contiene todo el texto de la entrada del usuario.
- TextResponseEvent— Contiene la respuesta del bot en formato de texto. De forma predeterminada, se devuelve una respuesta de texto. Si ha configurado Amazon Lex V2 para que devuelva una respuesta de audio, este texto se utilizará para generar una respuesta de audio. Cada evento de respuesta de texto contiene una serie de objetos de mensaje que el bot devuelve al usuario.
- AudioResponseEvent— Contiene la respuesta de audio sintetizada a partir del texto generado en el`TextResponseEvent`. Para recibir eventos de respuesta de audio, debe configurar Amazon Lex V2 para que proporcione una respuesta de audio. Todos los eventos de respuesta de audio

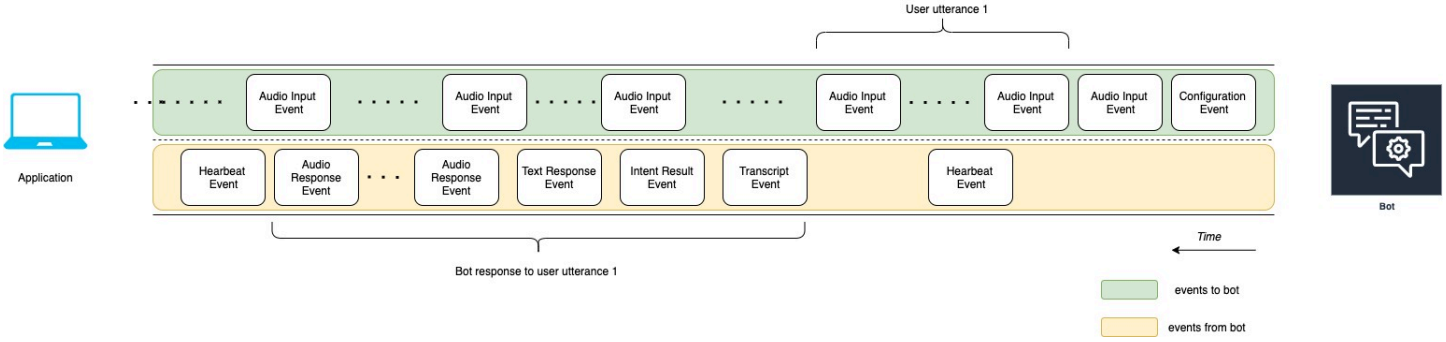
tienen el mismo formato de audio. Cada evento contiene fragmentos de audio de no más de 100 bytes. Amazon Lex V2 envía a su aplicación un fragmento de audio vacío con el campo `bytes` establecido en `null` para indicar el final del evento de respuesta de audio.

- PlaybackInterruptionEvent— Cuando un usuario interrumpe una respuesta que el bot ha enviado a su aplicación, Amazon Lex V2 activa este evento para detener la reproducción de la respuesta.
- HeartbeatEvent— Amazon Lex V2 devuelve este evento periódicamente para evitar que se agote el tiempo de espera de la conexión entre tu aplicación y el bot.

Secuencia temporal de los eventos de una conversación de audio cuando se utiliza un bot de Amazon Lex V2

Los siguientes diagramas muestran una conversación de audio en streaming entre un usuario y un bot de Amazon Lex V2. La aplicación transmite audio al bot de forma continua, y el bot busca las entradas del usuario a partir del audio. En este ejemplo, tanto el usuario como el bot utilizan la voz para comunicarse. Cada diagrama corresponde a un enunciado del usuario y a la respuesta del bot a ese enunciado.

El siguiente diagrama muestra el inicio de una conversación entre la aplicación y el bot. La transmisión comienza en el momento cero (t0).

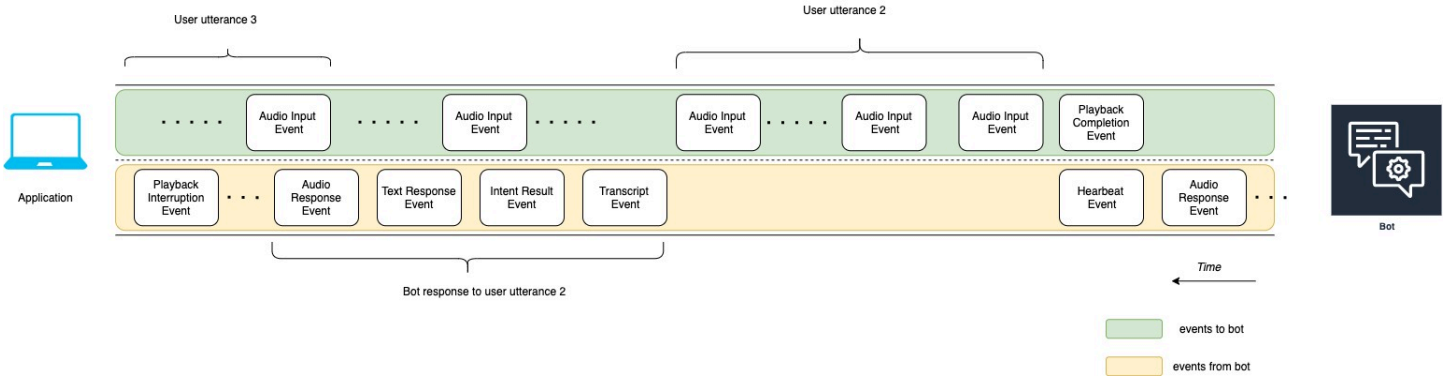


En la siguiente lista se describen los eventos del diagrama anterior.

- t0: La aplicación envía un evento de configuración al bot para iniciar la transmisión.
- t1: La aplicación transmite datos de audio. Estos datos se dividen en una serie de eventos de entrada desde la aplicación.
- t2: Para el enunciado 1 del usuario, el bot detecta un evento de entrada de audio cuando el usuario comienza a hablar.
- t2: Mientras el usuario habla, el bot envía un latido para mantener la conexión. Envía estos eventos de forma intermitente para asegurarse de que la conexión no agota el tiempo de espera.

- t3: El bot detecta el final del enunciado del usuario.
- t4: El bot devuelve a la aplicación un evento de transcripción que contiene una transcripción del discurso del usuario. Este es el principio de la respuesta del bot al enunciado 1 del usuario.
- t5: El bot envía un evento de resultado de intención para indicar la acción que el usuario quiere realizar.
- t6: El bot comienza a proporcionar su respuesta como texto en un evento de respuesta de texto.
- t7: El bot envía una serie de eventos de respuesta de audio a la aplicación para que los reproduzca el usuario.
- t8: El bot envía otro latido del corazón para mantener la conexión de forma intermitente.

El siguiente diagrama es una continuación del diagrama anterior. Muestra a la aplicación enviando un evento de finalización de la reproducción al bot para indicar que ha dejado de reproducir la respuesta de audio para el usuario. La aplicación reproduce la respuesta del Bot al enunciado 1 del usuario. El usuario responde a la respuesta del bot al enunciado 1 del usuario con el enunciado 2 del usuario.

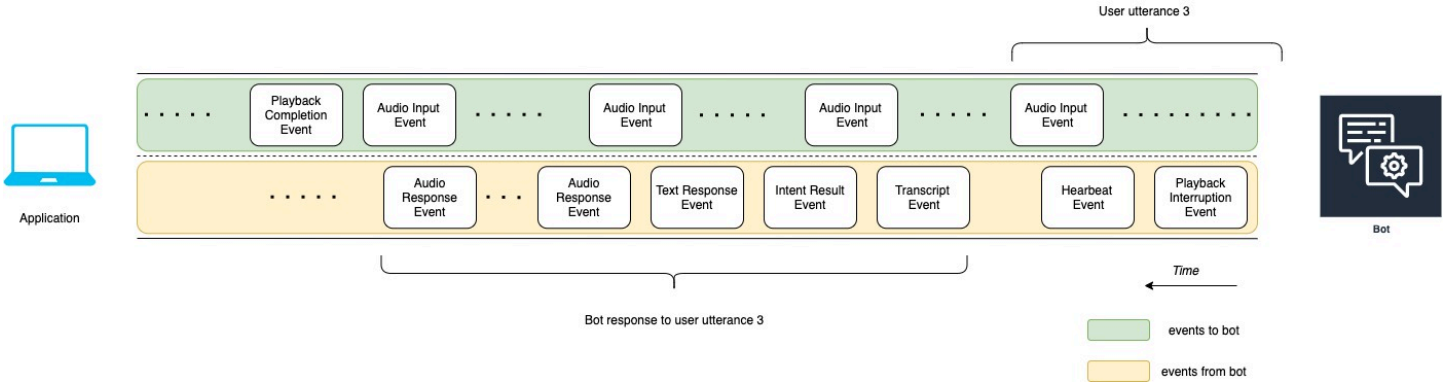


En la siguiente lista se describen los eventos del diagrama anterior:

- t10: La aplicación envía al usuario un evento de finalización de la reproducción para indicar que ha terminado de reproducir el mensaje del bot al usuario.
- t11: La aplicación devuelve la respuesta del usuario al bot como enunciado 2 del usuario.
- t12: En que el bot responda al enunciado 2 del usuario, el bot espera a que el usuario deje de hablar y, a continuación, comienza a dar una respuesta de audio.
- t13: Mientras el bot envía a la aplicación la respuesta del bot al enunciado 2 del usuario, el bot detecta el inicio del enunciado 3 del usuario. El bot detiene la respuesta del bot al enunciado 2 del usuario y envía un evento de interrupción de la reproducción.

- t14: El bot envía un evento de interrupción de la reproducción a la aplicación para indicar que el usuario ha interrumpido el mensaje.

El siguiente diagrama muestra la respuesta del bot al enunciado 3 del usuario y cómo la conversación continúa después de que el bot responde al enunciado del usuario.



Uso de la API para iniciar una conversación en streaming

Al iniciar una transmisión a un bot de Amazon Lex V2, debe realizar las siguientes tareas:

1. Crear una conexión inicial con el servidor.
2. Configurar las credenciales de seguridad y los detalles del bot. Los detalles del bot incluyen si el bot acepta la entrada de DTMF y audio o la entrada de texto.
3. Enviar eventos al servidor. Estos eventos son datos de texto o datos de audio del usuario.
4. Procesar los eventos enviados desde el servidor. En este paso, se determina si la salida del bot se presenta al usuario como texto o voz.

Los siguientes ejemplos de código inicializan una conversación en streaming con un bot de Amazon Lex V2 y su máquina local. Puede modificarlos para que se adapten a sus necesidades.

El siguiente código es un ejemplo de solicitud que se utiliza AWS SDK para Java para iniciar la conexión con un bot y configurar los detalles y las credenciales del bot.

```
package com.lex.streaming.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2AsyncClient;
import software.amazon.awssdk.services.lexruntimev2.model.ConversationMode;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequest;

import java.net.URISyntaxException;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * The following code creates a connection with the Amazon Lex bot and configures the
 bot details and credentials.
 * Prerequisite: To use this example, you must be familiar with the Reactive streams
 programming model.
 * For more information, see
 * https://github.com/reactive-streams/reactive-streams-jvm.
 * This example uses AWS SDK for Java for Amazon Lex V2.
 * <p>
 * The following sample application interacts with an Amazon Lex bot with the streaming
 API. It uses the Audio
 * conversation mode to return audio responses to the user's input.
 * <p>
 * The code in this example accomplishes the following:
 * <p>
 * 1. Configure details about the conversation between the user and the Amazon Lex bot.
 These details include the conversation mode and the specific bot the user is speaking
 with.
 * 2. Create an events publisher that passes the audio events to the Amazon Lex bot
 after you establish the connection. The code we provide in this example tells your
 computer to pick up the audio from
 * your microphone and send that audio data to Amazon Lex.
 * 3. Create a response handler that handles the audio responses from the Amazon Lex
 bot and plays back the audio to you.
 */
public class LexBidirectionalStreamingExample {

    public static void main(String[] args) throws URISyntaxException,
 InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
```

```
        Region region = Region.region_name; // Choose an AWS Region where the Amazon
Lex Streaming API is available.

        AwsCredentialsProvider awsCredentialsProvider = StaticCredentialsProvider
                .create(AwsBasicCredentials.create(accessKey, secretKey));

        // Create a new SDK client. You need to use an asynchronous client.
        System.out.println("step 1: creating a new Lex SDK client");
        LexRuntimeV2AsyncClient lexRuntimeServiceClient =
LexRuntimeV2AsyncClient.builder()
                .region(region)
                .credentialsProvider(awsCredentialsProvider)
                .build();


        // Configure the bot, alias and locale that you'll use to have a conversation.
        System.out.println("step 2: configuring bot details");
        StartConversationRequest.Builder startConversationRequestBuilder =
StartConversationRequest.builder()
                .botId(botId)
                .botAliasId(botAliasId)
                .localeId(localeId);

        // Configure the conversation mode of the bot. By default, the
        // conversation mode is audio.
        System.out.println("step 3: choosing conversation mode");
        startConversationRequestBuilder =
startConversationRequestBuilder.conversationMode(ConversationMode.AUDIO);

        // Assign a unique identifier for the conversation.
        System.out.println("step 4: choosing a unique conversation identifier");
        startConversationRequestBuilder =
startConversationRequestBuilder.sessionId(sessionId);

        // Start the initial request.
        StartConversationRequest startConversationRequest =
startConversationRequestBuilder.build();

        // Create a stream of audio data to the Amazon Lex bot. The stream will start
after the connection is established with the bot.
        EventsPublisher eventsPublisher = new EventsPublisher();

        // Create a class to handle responses from bot. After the server processes the
user data you've streamed, the server responds
```

```
        // on another stream.
        BotResponseHandler botResponseHandler = new
 BotResponseHandler(eventsPublisher);

        // Start a connection and pass in the publisher that streams the audio and
 process the responses from the bot.
        System.out.println("step 5: starting the conversation ...");
        CompletableFuture<Void> conversation =
 lexRuntimeServiceClient.startConversation(
                startConversationRequest,
                eventsPublisher,
                botResponseHandler);

        // Wait until the conversation finishes. The conversation finishes if the
 dialog state reaches the "Closed" state.
        // The client stops the connection. If an exception occurs during the
 conversation, the
        // client sends a disconnection event.
        conversation.whenComplete((result, exception) -> {
            if (exception != null) {
                eventsPublisher.disconnect();
            }
        });

        // The conversation finishes when the dialog state is closed and last prompt
 has been played.
        while (!botResponseHandler.isConversationComplete()) {
            Thread.sleep(100);
        }

        // Randomly sleep for 100 milliseconds to prevent JVM from exiting.
        // You won't need this in your production code because your JVM is
        // likely to always run.
        // When the conversation finishes, the following code block stops publishing
 more data and informs the Amazon Lex bot that there is no more data to send.
        if (botResponseHandler.isConversationComplete()) {
            System.out.println("conversation is complete.");
            eventsPublisher.stop();
        }
    }
}
```

El siguiente código es un ejemplo de solicitud AWS SDK para Java que se utiliza para enviar eventos al bot. El código de este ejemplo usa el micrófono del ordenador para enviar eventos de audio.

```
package com.lex.streaming.sample;

import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

/**
 * You use the Events publisher to send events to the Amazon Lex bot. When you
 establish a connection, the bot uses the
 * subscribe() method and enables the events publisher starts sending events to
 * your computer. The bot uses the "request" method of the subscription to make more
 requests. For more information on the request method, see https://github.com/reactive-
streams/reactive-streams-jvm.
 */
public class EventsPublisher implements Publisher<StartConversationRequestEventStream>
 {

    private AudioEventsSubscription audioEventsSubscription;

    @Override
    public void subscribe(Subscriber<? super StartConversationRequestEventStream>
 subscriber) {
        if (audioEventsSubscription == null) {

            audioEventsSubscription = new AudioEventsSubscription(subscriber);
            subscriber.onSubscribe(audioEventsSubscription);

        } else {
            throw new IllegalStateException("received unexpected subscription
 request");
        }
    }

    public void disconnect() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.disconnect();
        }
    }
```

```
    public void stop() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.stop();
        }
    }

    public void playbackFinished() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.playbackFinished();
        }
    }
}
```

El siguiente código es un ejemplo de solicitud AWS SDK para Java que se utiliza para gestionar respuestas del bot. El código de este ejemplo configura Amazon Lex V2 para que le reproduzca una respuesta de audio.

```
package com.lex.streaming.sample;

import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.lexruntimev2.model.AudioResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DialogActionType;
import software.amazon.awssdk.services.lexruntimev2.model.IntentResultEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackInterruptionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponse;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseEventStream;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseHandler;
import software.amazon.awssdk.services.lexruntimev2.model.TextResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.TranscriptEvent;

import java.io.IOException;
import java.io.UncheckedIOException;
import java.util.concurrent.CompletableFuture;
```

```
/**
 * The following class is responsible for processing events sent from the Amazon Lex
 bot. The bot sends multiple audio events,
 * so the following code concatenates those audio events and uses a publicly available
 Java audio player to play out the message to
 * the user.
 */
public class BotResponseHandler implements StartConversationResponseHandler {

    private final EventsPublisher eventsPublisher;

    private boolean lastBotResponsePlayedBack;
    private boolean isDialogStateClosed;
    private AudioResponse audioResponse;


    public BotResponseHandler(EventsPublisher eventsPublisher) {
        this.eventsPublisher = eventsPublisher;
        this.lastBotResponsePlayedBack = false;// At the start, we have not played back
 last response from bot.
        this.isDialogStateClosed = false; // At the start, the dialog state is open.
    }

    @Override
    public void responseReceived(StartConversationResponse startConversationResponse) {
        System.out.println("successfully established the connection with server.
 request id:" + startConversationResponse.responseMetadata().requestId()); // would
 have 2XX, request id.
    }

    @Override
    public void onEventStream(SdkPublisher<StartConversationResponseEventStream>
 sdkPublisher) {

        sdkPublisher.subscribe(event -> {
            if (event instanceof PlaybackInterruptionEvent) {
                handle((PlaybackInterruptionEvent) event);
            } else if (event instanceof TranscriptEvent) {
                handle((TranscriptEvent) event);
            } else if (event instanceof IntentResultEvent) {
                handle((IntentResultEvent) event);
            } else if (event instanceof TextResponseEvent) {
                handle((TextResponseEvent) event);
```

```
            } else if (event instanceof AudioResponseEvent) {
                handle((AudioResponseEvent) event);
            }
        });
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("got an exception:" + throwable);
    }

    @Override
    public void complete() {
        System.out.println("on complete");
    }

    private void handle(PlaybackInterruptionEvent event) {
        System.out.println("Got a PlaybackInterruptionEvent: " + event);
    }

    private void handle(TranscriptEvent event) {
        System.out.println("Got a TranscriptEvent: " + event);
    }



    private void handle(IntentResultEvent event) {
        System.out.println("Got an IntentResultEvent: " + event);
        isDialogStateClosed =
DialogActionType.CLOSE.equals(event.sessionState().dialogAction().type());
    }

    private void handle(TextResponseEvent event) {
        System.out.println("Got an TextResponseEvent: " + event);
        event.messages().forEach(message -> {
            System.out.println("Message content type:" + message.contentType());
            System.out.println("Message content:" + message.content());
        });
    }

    private void handle(AudioResponseEvent event) {//Synthesize speech
        // System.out.println("Got a AudioResponseEvent: " + event);
        if (audioResponse == null) {
            audioResponse = new AudioResponse();
            //Start an audio player in a different thread.
```

```
            CompletableFuture.runAsync(() -> {
                try {
                    AdvancedPlayer audioPlayer = new AdvancedPlayer(audioResponse);

                    audioPlayer.setPlayBackListener(new PlaybackListener() {
                        @Override
                        public void playbackFinished(PlaybackEvent evt) {
                            super.playbackFinished(evt);

                            // Inform the Amazon Lex bot that the playback has
finished.
                            eventsPublisher.playbackFinished();
                            if (isDialogStateClosed) {
                                lastBotResponsePlayedBack = true;
                            }
                        }
                    });
                    audioPlayer.play();
                } catch (JavaLayerException e) {
                    throw new RuntimeException("got an exception when using audio
player", e);
                }
            });
        }

        if (event.audioChunk() != null) {
            audioResponse.write(event.audioChunk().asByteArray());
        } else {
            // The audio audio prompt has ended when the audio response has no
            // audio bytes.
            try {
                audioResponse.close();
                audioResponse = null;  // Prepare for the next audio prompt.
            } catch (IOException e) {
                throw new UncheckedIOException("got an exception when closing the audio
response", e);
            }
        }
    }

    // The conversation with the Amazon Lex bot is complete when the bot marks the
Dialog as DialogActionType.CLOSE
    // and any prompt playback is finished. For more information, see
    // https://docs.aws.amazon.com/lexv2/latest/dg/API_runtime_DialogAction.html.
```

```
    public boolean isConversationComplete() {
        return isDialogStateClosed && lastBotResponsePlayedBack;
    }

}
```

Para configurar un bot para que responda a los eventos de entrada con audio, primero debe suscribirse a los eventos de audio de Amazon Lex V2 y, a continuación, configurar el bot para que proporcione una respuesta de audio a los eventos de entrada del usuario.

El código siguiente es un AWS SDK para Java ejemplo de suscripción a eventos de audio de Amazon Lex V2.

```
package com.lex.streaming.sample;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lexruntimev2.model.AudioInputEvent;
import software.amazon.awssdk.services.lexruntimev2.model.ConfigurationEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DisconnectionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackCompletionEvent;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
import java.io.IOException;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicLong;
```

```
public class AudioEventsSubscription implements Subscription {
    private static final AudioFormat MIC_FORMAT = new AudioFormat(8000, 16, 1, true,
 false);
    private static final String AUDIO_CONTENT_TYPE = "audio/lpcm; sample-rate=8000;
 sample-size-bits=16; channel-count=1; is-big-endian=false";
    //private static final String RESPONSE_TYPE = "audio/pcm; sample-rate=8000";
    private static final String RESPONSE_TYPE = "audio/mpeg";
    private static final int BYTES_IN_AUDIO_CHUNK = 320;
    private static final AtomicLong eventIdGenerator = new AtomicLong(0);

    private final AudioInputStream audioInputStream;
    private final Subscriber<? super StartConversationRequestEventStream> subscriber;
    private final EventWriter eventWriter;
    private CompletableFuture eventWriterFuture;


    public AudioEventsSubscription(Subscriber<? super
 StartConversationRequestEventStream> subscriber) {
        this.audioInputStream = getMicStream();
        this.subscriber = subscriber;
        this.eventWriter = new EventWriter(subscriber, audioInputStream);
        configureConversation();
    }

    private AudioInputStream getMicStream() {
        try {
            DataLine.Info dataLineInfo = new DataLine.Info(TargetDataLine.class,
 MIC_FORMAT);
            TargetDataLine targetDataLine = (TargetDataLine)
 AudioSystem.getLine(dataLineInfo);

            targetDataLine.open(MIC_FORMAT);
            targetDataLine.start();

            return new AudioInputStream(targetDataLine);
        } catch (LineUnavailableException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void request(long demand) {
        // If a thread to write events has not been started, start it.
        if (eventWriterFuture == null) {
```

```
            eventWriterFuture = CompletableFuture.runAsync(eventWriter);
        }
        eventWriter.addDemand(demand);
    }

    @Override
    public void cancel() {
        subscriber.onError(new RuntimeException("stream was cancelled"));
        try {
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
    }

    public void configureConversation() {
        String eventId = "ConfigurationEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        ConfigurationEvent configurationEvent = StartConversationRequestEventStream
                .configurationEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .responseContentType(RESPONSE_TYPE)
                .build();

        System.out.println("writing config event");
        eventWriter.writeConfigurationEvent(configurationEvent);
    }

    public void disconnect() {

        String eventId = "DisconnectionEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        DisconnectionEvent disconnectionEvent = StartConversationRequestEventStream
                .disconnectionEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .build();

        eventWriter.writeDisconnectEvent(disconnectionEvent);

        try {
```

```
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

    }
    //Notify the subscriber that we've finished.
    public void stop() {
        subscriber.onComplete();
    }

    public void playbackFinished() {
        String eventId = "PlaybackCompletion-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        PlaybackCompletionEvent playbackCompletionEvent =
StartConversationRequestEventStream
                .playbackCompletionEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .build();

        eventWriter.writePlaybackFinishedEvent(playbackCompletionEvent);
    }

    private static class EventWriter implements Runnable {
        private final BlockingQueue<StartConversationRequestEventStream> eventQueue;
        private final AudioInputStream audioInputStream;
        private final AtomicLong demand;
        private final Subscriber subscriber;

        private boolean conversationConfigured;

        public EventWriter(Subscriber subscriber, AudioInputStream audioInputStream) {
            this.eventQueue = new LinkedBlockingQueue<>();

            this.demand = new AtomicLong(0);
            this.subscriber = subscriber;
            this.audioInputStream = audioInputStream;
        }

        public void writeConfigurationEvent(ConfigurationEvent configurationEvent) {
            eventQueue.add(configurationEvent);
        }
```

```
        public void writeDisconnectEvent(DisconnectionEvent disconnectionEvent) {
            eventQueue.add(disconnectionEvent);
        }

        public void writePlaybackFinishedEvent(PlaybackCompletionEvent
playbackCompletionEvent) {
            eventQueue.add(playbackCompletionEvent);
        }

        void addDemand(long l) {
            this.demand.addAndGet(l);
        }

        @Override
        public void run() {
            try {

                while (true) {
                    long currentDemand = demand.get();

                    if (currentDemand > 0) {
                        // Try to read from queue of events.
                        // If nothing is in queue at this point, read the audio events
directly from audio stream.
                        for (long i = 0; i < currentDemand; i++) {

                            if (eventQueue.peek() != null) {
                                subscriber.onNext(eventQueue.take());
                                demand.decrementAndGet();
                            } else {
                                writeAudioEvent();
                            }
                        }
                    }
                }
            } catch (InterruptedException e) {
                throw new RuntimeException("interrupted when reading data to be sent to
server");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
        private void writeAudioEvent() {
            byte[] bytes = new byte[BYTES_IN_AUDIO_CHUNK];

            int numBytesRead = 0;
            try {
                numBytesRead = audioInputStream.read(bytes);
                if (numBytesRead != -1) {
                    byte[] byteArrayCopy = Arrays.copyOf(bytes, numBytesRead);

                    String eventId = "AudioEvent-" +
 String.valueOf(eventIdGenerator.incrementAndGet());

                    AudioInputEvent audioInputEvent =
 StartConversationRequestEventStream
                            .audioInputEventBuilder()

 .audioChunk(SdkBytes.fromByteBuffer(ByteBuffer.wrap(byteArrayCopy)))
                            .contentType(AUDIO_CONTENT_TYPE)
                            .clientTimestampMillis(System.currentTimeMillis())
                            .eventId(eventId).build();

                    //System.out.println("sending audio event:" + audioInputEvent);
                    subscriber.onNext(audioInputEvent);
                    demand.decrementAndGet();
                    //System.out.println("sent audio event:" + audioInputEvent);
                } else {
                    subscriber.onComplete();
                    System.out.println("audio stream has ended");
                }

            } catch (IOException e) {
                System.out.println("got an exception when reading from audio stream");
                System.err.println(e);
                subscriber.onError(e);
            }
        }
    }
}
```

El siguiente AWS SDK para Java ejemplo configura el bot Amazon Lex V2 para que proporcione una respuesta de audio a los eventos de entrada.

```
package com.lex.streaming.sample;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.util.Optional;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

public class AudioResponse extends InputStream{

    // Used to convert byte, which is signed in Java, to positive integer (unsigned)
    private static final int UNSIGNED_BYTE_MASK = 0xFF;
    private static final long POLL_INTERVAL_MS = 10;

    private final LinkedBlockingQueue<Integer> byteQueue = new LinkedBlockingQueue<>();

    private volatile boolean closed;

    @Override
    public int read() throws IOException {
        try {
            Optional<Integer> maybeInt;
            while (true) {
                maybeInt = Optional.ofNullable(this.byteQueue.poll(POLL_INTERVAL_MS,
 TimeUnit.MILLISECONDS));

                // If we get an integer from the queue, return it.
                if (maybeInt.isPresent()) {
                    return maybeInt.get();
                }

                // If the stream is closed and there is nothing queued up, return -1.
                if (this.closed) {
                    return -1;
                }
            }
        } catch (InterruptedException e) {
            throw new IOException(e);
        }
    }
```

```
    /**
     * Writes data into the stream to be offered on future read() calls.
     */
    public void write(byte[] byteArray) {
        // Don't write into the stream if it is already closed.
        if (this.closed) {
            throw new UncheckedIOException(new IOException("Stream already closed when
 attempting to write into it."));
        }

        for (byte b : byteArray) {
            this.byteQueue.add(b & UNSIGNED_BYTE_MASK);
        }
    }

    @Override
    public void close() throws IOException {
        this.closed = true;
        super.close();
    }
}
```

Codificar secuencias de eventos

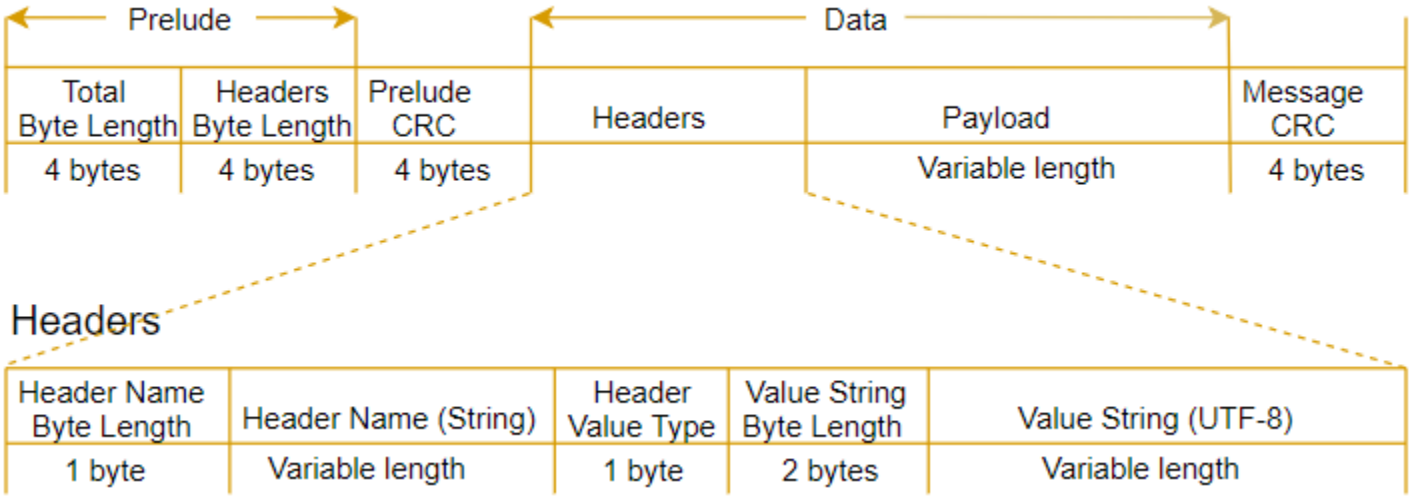
La codificación de secuencias de eventos permite establecer una comunicación bidireccional a través de mensajes entre un cliente y un servidor. Las tramas de datos que se envían al servicio de transmisión de Amazon Lex V2 se codifican en este formato. La respuesta de Amazon Lex V2 también utiliza esta codificación.

Cada mensaje se compone de dos secciones: el preludio y los datos. La sección de preludio contiene la longitud total en bytes del mensaje y la longitud en bytes combinada de todos los encabezados. La sección de datos contiene los encabezados y una carga útil.

Cada sección termina con una suma de comprobación CRC big-endian de enteros de 4 bytes. La suma de verificación CRC del mensaje incluye la sección de preludio y la sección de datos. Amazon Lex V2 utiliza CRC32 (a menudo denominado CRC32 de GZIP) para calcular ambos CRC. Para obtener más información sobre CRC32, consulte GZIP file format specification version 4.3.

La carga total del mensaje, incluido el preludio y las dos sumas de comprobación, es de 16 bytes.

En el siguiente diagrama, se muestran los componentes que conforman un mensaje y un encabezado. Hay varios encabezados en cada mensaje.



Cada mensaje contiene los siguientes componentes:

- Preludio: siempre tiene un tamaño fijo de 8 bytes divididos en dos campos de 4 bytes cada uno.
  - Primeros 4 bytes: longitud total en bytes. Se trata de la longitud en bytes indicada en números enteros big-endian de todo el mensaje, incluido el propio campo de 4 bytes.
  - Segundos 4 bytes: longitud en bytes de los encabezados. Se trata de la longitud en bytes indicada en números enteros big-endian de los encabezados del mensaje, sin incluir el propio campo de longitud de los encabezados.
- CRC del preludio: suma de comprobación CRC de 4 bytes del preludio del mensaje, sin incluir la propia CRC. El preludio tiene una CRC que es independiente de la CRC del mensaje y que garantiza que Amazon Lex V2 puede detectar inmediatamente información dañada sobre la longitud de bytes sin generar errores, como la saturación del búfer.
- Encabezados: metadatos que actúan como comentarios del mensaje; por ejemplo, el tipo de mensaje, el tipo de contenido, etc. Los mensajes tienen varios encabezados. Los encabezados son pares clave-valor en los que la clave es una cadena. UTF-8 Los encabezados pueden aparecer en cualquier orden en la parte del mensaje que les corresponde y cada encabezado solamente puede aparecer una vez. Para conocer los tipos de encabezado obligatorios, consulte las secciones siguientes.
- Carga útil: el contenido de audio o texto que se envía a Amazon Lex.
- CRC del mensaje: suma de comprobación CRC de 4 bytes desde el comienzo del mensaje hasta el inicio de la suma de comprobación. Esto incluye todo el mensaje, excepto la propia CRC.

Cada encabezado contiene los siguientes componentes. Hay varios encabezados en cada trama.

- Longitud en bytes del nombre de encabezado: longitud en bytes del nombre del encabezado.
- Nombre de encabezado: nombre del encabezado que indica el tipo de encabezado. Para ver los valores válidos, consulte las siguientes descripciones de tramas.
- Tipo de valor del encabezado: enumeración que indica el tipo de valor del encabezado.
- Longitud en bytes de cadena de valores: longitud en bytes de la cadena de valores del encabezado.
- Valor del encabezado: el valor de la cadena del encabezado. Los valores válidos de este campo dependen del tipo de encabezado. Para ver los valores válidos, consulte las siguientes descripciones de tramas.

Cómo permitir que su usuario interrumpa a su bot Amazon Lex V2

Al iniciar una transmisión de audio bidireccional entre un bot de Amazon Lex V2 y su aplicación, puede configurar el bot para que escuche las entradas del usuario mientras devuelve un mensaje. Esta entrada se puede configurar como una interrupción. De este modo, el usuario puede interrumpir el mensaje antes de que el bot termine de reproducirlo. Puede usar esta configuración para situaciones en las que es posible que el usuario ya sepa la respuesta a una pregunta, como cuando se le pide que proporcione un código CVV.

Un bot sabe cuándo el usuario interrumpe un mensaje cuando detecta una entrada del usuario antes de que la aplicación pueda devolver un evento `PlaybackCompletion`. Cuando el usuario interrumpe un bot, el bot envía un `PlaybackInterruptionEvent`.

De forma predeterminada, el usuario puede interrumpir cualquier mensaje que el bot esté transmitiendo a su aplicación. Puede cambiar esta configuración en la consola de Amazon Lex V2.

Puede cambiar la forma en que un usuario puede responder a una solicitud editando un slot. Un slot es parte de una intención y es el medio por el cual el usuario le proporciona la información que desea. Cada slot tiene un mensaje para que el usuario le proporcione esa información. Para obtener más información acerca de los slots, consulte Conceptos clave de Amazon Lex V2.

Para cambiar si el usuario puede interrumpir un mensaje (consola)

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en la consola Amazon Lex V2.

2. En Bots, seleccione un bot.
3. En Idioma, seleccione el idioma del bot.
4. Seleccione Ver intenciones.
5. Seleccione la intención .
6. Para los slots, seleccione un slot.
7. En Opciones avanzadas, seleccione Indicaciones de slots.
8. Seleccione Más opciones de mensaje.
9. Seleccionar o deseleccionar Los usuarios pueden interrumpir el mensaje mientras se lee .

Puede probar esta funcionalidad creando un bot con dos slots y especificando que los usuarios no puedan interrumpir un mensaje por un slot. Si interrumpe un mensaje interrumpible, el bot envía un evento de interrupción de la reproducción. Si interrumpe un dispositivo ininterrumpido, el mensaje continúa reproduciéndose.

Cómo permitir que el bot Amazon Lex V2 espere a que el usuario proporcione más información durante una pausa

Al iniciar una transmisión bidireccional desde un bot de Amazon Lex V2 a su aplicación, puede configurar el bot para que espere a que el usuario proporcione información adicional. Hay circunstancias en las que es posible que un usuario no esté preparado para responder a un mensaje. Por ejemplo, es posible que un usuario no esté preparado para proporcionar la información de su tarjeta de crédito porque su cartera está en otra habitación.

Al utilizar el comportamiento Esperar y continuar del bot de Amazon Lex V2, los usuarios pueden decir frases como «espera un segundo» para que el bot espere a que encuentren la información y la proporcionen. Cuando habilita este comportamiento, el bot envía recordatorios periódicos al usuario para que proporcione la información. No devuelve los eventos de la transcripción porque no hay enunciados del usuario que pueda transcribir.

El bot de Amazon Lex V2 gestiona automáticamente una conversación en streaming. No tiene que escribir ningún código adicional para habilitar esta funcionalidad. Cuando el usuario le pide a un bot que espere, el `state` de la `Intent` es `Waiting` y el `type` de `DialogAction` es `ElicitSlot`. Puede utilizar esta información para personalizar la aplicación según sus necesidades. Por ejemplo, puede configurar la aplicación para que reproduzca música cuando el usuario busque su tarjeta de crédito.

Puede activar el comportamiento de esperar y continuar en un slot individual. Para obtener más información acerca de los slots, consulte Conceptos clave de Amazon Lex V2.

Para habilitar esperar y continuar

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en la consola Amazon Lex V2.
2. En Bots, seleccione un bot.
3. En Idioma, seleccione el idioma del bot.
4. Seleccione Ver intenciones.
5. Seleccione la intención.
6. Para los slots, seleccione un slot.
7. En Opciones avanzadas, seleccione Esperar y continuar.
8. En Esperar y continuar, especifique los siguientes campos:

    - Respuesta cuando el usuario quiere que el bot espere: así es como responde el bot cuando el usuario le pide que espere a recibir información adicional.
    - Respuesta si ezl usuario necesita que el bot siga esperando: esta es la respuesta que el bot envía para recordarle al usuario que sigue esperando la información. Puede cambiar la frecuencia con la que el bot se lo recuerda al usuario.
    - Respuesta cuando el usuario quiere continuar: es la respuesta del bot cuando el usuario tiene la información solicitada.

Para cada respuesta del bot, puede dar varias variaciones de la respuesta, y se le presenta una al usuario de forma aleatoria. También puede elegir si el usuario puede interrumpir estas respuestas.

Para probar la funcionalidad de esperar y continuar, configure su bot para que espere la entrada del usuario e inicie una transmisión a un bot de Amazon Lex V2. Para obtener información sobre la transmisión a un bot, consulte Uso de la API para iniciar una conversación en streaming.

Es posible que tenga que desactivar las respuestas de esperar y continuar. Use el conmutador Activar para configurar si se utilizan o no las respuestas de esperar y continuar.



> **Note**
>
> La opción Esperar y continuar está disponible en las siguientes configuraciones regionales: ca-ES, de-AT, de-DE, en-AU, en-GB, en-IN, en-US, en-ZA, es-419, es-ES, es-US, fr-CA, fr-FR, it-IT, ja-JP, ko-KR, pt-BR, pt-PT, zh-CN y zh-HK.

Configuración de las actualizaciones del progreso de cumplimiento para su bot Lex V2

Cuando se llama a la función de Lambda de cumplimiento para una intención, el bot no envía una respuesta hasta que se completa la función. Si la función de Lambda tarda más de unos segundos en completarse, el usuario puede pensar que el bot no responde. Para solucionar este problema, puede configurar su bot para que envíe actualizaciones al usuario mientras se ejecuta la función de Lambda de cumplimiento, de modo que el usuario sepa que el bot sigue trabajando en su solicitud.

Cuando añade actualizaciones de cumplimiento a una intención, el bot responde al inicio de esta y periódicamente mientras esté en curso. Cuando configura la respuesta de inicio, puede especificar un retraso antes de que el bot envíe la respuesta. Con esto, puede respaldar los casos en los que el cumplimiento no finalice con relativa rapidez. Cuando configura una respuesta de actualización, debe especificar la frecuencia con la que desea que se envíen las actualizaciones. También configura un tiempo de espera para limitar el tiempo de ejecución de la función de cumplimiento.

También puede añadir respuestas posteriores al cumplimiento a un bot. Esto permite al bot enviar una respuesta diferente en función de si el cumplimiento se realiza correctamente, si no se cumple o si se agota el tiempo de espera.

Las actualizaciones de cumplimiento solo se utilizan cuando se interactúa con un bot mediante la StartConversationoperación. Puedes usar la actualización posterior a la gestión logística cuando interactúes con el bot mediante las StartConversationoperaciones RecognizeText, y RecognizeUtterance

Actualizaciones de cumplimiento

Las actualizaciones de cumplimiento se envían mientras la función de Lambda cumple una intención. Al activar las actualizaciones de cumplimiento, proporciona una respuesta inicial que se envía al principio del cumplimiento y una respuesta de actualización que se envía periódicamente mientras el cumplimiento está en curso.

Cuando especifica una respuesta de actualización, también especifica un tiempo de espera que determina cuánto tiempo puede funcionar la función de cumplimiento. Puede especificar un tiempo de espera de hasta 15 minutos (900 segundos).

Si desactivas las actualizaciones de gestión logística `active` configurándolas en false en la consola o mediante la UpdateIntentoperación CreateIntento, no se utilizará el tiempo de espera especificado para las actualizaciones de gestión logística y, en su lugar, se utilizará el tiempo de espera predeterminado de 30 segundos.

Si se agota el tiempo de espera de la función de cumplimiento, Amazon Lex V2 realiza una de estas tres acciones:

- La respuesta posterior al cumplimiento está configurada y activa: devuelve la respuesta de tiempo de espera.
- La respuesta posterior al cumplimiento está configurada y no activa: devuelve una excepción.
- La respuesta posterior al cumplimiento no está configurada: devuelve una excepción.

Iniciar respuesta

Amazon Lex V2 devuelve la respuesta de inicio cuando se llama a la función de cumplimiento de Lambda durante una conversación en streaming. Por lo general, indica al usuario que cumplir con la intención lleva algún tiempo y que debe esperar. La respuesta de inicio no se devuelve cuando se utilizan las operaciones `RecognizeText` o `RecognizeUtterance`.

Puede especificar hasta cinco mensajes de respuesta. Amazon Lex V2 elige uno de los mensajes para reproducirlo al usuario.

Puede configurar un retraso entre el momento en que se llama a la función de Lambda y el momento en que se devuelve la respuesta de inicio. La respuesta de inicio no se devuelve si la función de Lambda completa su trabajo antes de que se complete el retraso.

Puedes usar el `active` botón de la consola o de la FulfillmentUpdatesSpecificationestructura para activar y desactivar la respuesta de inicio. Si `active` es falsa, no se reproduce la respuesta de inicio.

Actualizar la respuesta

Amazon Lex devuelve la respuesta de actualización periódicamente durante la conversación en streaming cuando la función de cumplimiento se está ejecutando. La respuesta de actualización no se reproduce cuando se utilizan las operaciones `RecognizeText` o `RecognizeUtterance`.

Puede configurar la frecuencia con la que se reproduce la respuesta de actualización. Por ejemplo, puede reproducir una respuesta de actualización cada 30 segundos mientras se ejecuta la función de cumplimiento para que el usuario sepa que el proceso se está ejecutando y que debe seguir esperando.

Puede especificar hasta cinco mensajes de actualización. Amazon Lex V2 elige uno mensaje para reproducirlo al usuario. El uso de varios mensajes evita que las actualizaciones sean repetitivas.

Si el usuario introduce datos mediante voz, DTMF o texto mientras se ejecuta la función de Lambda de cumplimiento, Amazon Lex V2 devuelve la respuesta de actualización al usuario.

Si la función de Lambda completa su trabajo antes de que finalice el primer período de actualización, no se devuelve la respuesta de actualización.

Puedes usar el `active` botón de la consola o de la FulfillmentUpdatesSpecificationestructura para activar y desactivar la respuesta de actualización. Si `active` es falsa, no se devuelve la respuesta de actualización.

Respuesta posterior al cumplimiento

Amazon Lex V2 devuelve una respuesta posterior al cumplimiento cuando finaliza la función de cumplimiento. La respuesta posterior al cumplimiento se puede utilizar para cumplir cualquier intención, no solo para transmitir conversaciones. La respuesta posterior al cumplimiento permite al usuario saber que la función ha finalizado y el resultado obtenido.

Puedes usar el `active` botón de la consola o de la PostFulfillmentStatusSpecificationestructura para activar y desactivar la respuesta posterior a la gestión logística. Si `active` es falsa, no se reproduce la respuesta.

Hay tres tipos de respuestas posteriores al cumplimiento:

- Éxito: se devuelve cuando la función de Lambda de cumplimiento completa su trabajo correctamente. Si las respuestas posteriores al cumplimiento no están activas. Amazon Lex V2 realiza la siguiente acción configurada.
- Tiempo de espera: se devuelve si la función de Lambda no completa su trabajo antes de que transcurra el período de tiempo de espera configurado. Si las respuestas posteriores al cumplimiento no están activas, Amazon Lex V2 devuelve una excepción.
- Fallo: se devuelve cuando la función de Lambda devuelve el estado `Failed` de la respuesta o cuando Amazon Lex V2 detecta un error mientras se cumple la intención. Si las respuestas posteriores al cumplimiento no están activas, Amazon Lex V2 devuelve una excepción.

Puede especificar hasta cinco mensajes por tipo. Amazon Lex V2 elige uno de los mensajes para reproducirlo al usuario.

A diferencia de las respuestas de inicio y actualización de cumplimiento, las respuestas posteriores al cumplimiento se reproducen tanto para conversaciones en streaming como no en streaming.

También tiene la opción de anular estos mensajes configurando la función de Lambda para que devuelva un mensaje posterior al envío.

> **Note**
>
> Si la intención tiene una respuesta de cierre, se devuelve después de la respuesta posterior al cumplimiento.

Ejemplo posterior al cumplimiento para Lex V2

Para entender mejor la respuesta posterior a la gestión logística, tomemos, como ejemplo, un *BookTrip* bot creado para ayudar a planificar un viaje, con una *BookFlight* intención, configurado con una función Lambda de gestión logística que reserva el vuelo del cliente con una aerolínea. Una vez *BookFlight* que se han obtenido los espacios para, Amazon Lex V2 invoca la función Lambda de gestión logística. Durante este proceso de cumplimiento, puede producirse uno de los tres resultados siguientes:

- Éxito: el vuelo se reservó correctamente.
- Tiempo de espera: el proceso de reserva tarda más que el tiempo de ejecución de cumplimiento Lambda configurado (por ejemplo, si no se puede contactar con la aerolínea dentro del tiempo asignado).
- Fallo: la reserva no se realiza correctamente por otro motivo.

Puede aprovechar la respuesta posterior al cumplimiento para ofrecer una respuesta más significativa a sus clientes en cada una de estas situaciones. Los ejemplos de cada situación son los siguientes:

- Respuesta correcta: «Hemos podido reservar tu billete correctamente y te hemos enviado un correo electrónico de confirmación. No dudes en ponerte en contacto con nosotros utilizando la información de contacto proporcionada en ese correo electrónico si tienes alguna pregunta».

- Respuesta de tiempo de espera agotado: «Debido al intenso tráfico en nuestros sistemas, la reserva de tu billete está tardando más de lo esperado. Tenemos tu solicitud pendiente y te hemos enviado un correo electrónico con el número de referencia correspondiente a esta solicitud. Una vez que reservemos el billete, te enviaremos una confirmación de la reserva. No dudes en ponerte en contacto con nosotros utilizando la información de contacto proporcionada en ese correo electrónico si tienes alguna pregunta».

  > **Note**
  >
  > Si no configura un mensaje de tiempo de espera, Amazon Lex V2 arroja un error 4XX correspondiente al caso de uso.

- Respuesta errónea: «Lamentablemente, no hemos podido reservar tu billete. Le hemos enviado un correo electrónico con detalles sobre el problema que hemos tenido al intentar hacer su reserva".

Configuración de los tiempos de espera para capturar entradas del usuario con un bot Lex V2

La API de transmisión de Amazon Lex V2 permite a un bot detectar automáticamente los enunciados en las entradas de los usuarios. Al crear una intención o un slot, puede configurar algunos aspectos de un enunciado, como la duración máxima del enunciado, el tiempo de espera a la entrada del usuario o el carácter final de la entrada de DTMF. Puede personalizar el comportamiento de un bot para su caso de uso. Por ejemplo, puede limitar el número de dígitos de un número de tarjeta de crédito a 16.

También puede configurar los tiempos de espera mediante atributos de sesión al iniciar una conversación con un bot y sobrescribirlos en su función de Lambda si es necesario.

Las claves de configuración de un atributo utilizan la siguiente sintaxis:

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

`InputType` puede ser **`audio`**, **`dtmf`** o **`text`**.

Puede configurar los ajustes predeterminados para todas las intenciones o slots de un bot especificando **`*`** como intención o nombre de slot. Cualquier configuración específica de intención o slot tiene prioridad sobre la configuración predeterminada.

Amazon Lex V2 proporciona atributos de sesión predefinidos para gestionar el StartConversationfuncionamiento de las operaciones con entradas de texto, voz o DTMF en el bot. Todos los atributos predefinidos se encuentran en el espacio de nombres `x-amz-lex`.

Puede configurar los ajustes predeterminados para todas las intenciones o slots de un bot especificando * como intención o nombre de slot. Cualquier intención o configuración específica de intención o slot tiene prioridad sobre la configuración predeterminada. Use estos patrones para todos los tiempos de espera que aparecen a continuación.

En el caso de un subslot compuesto, puede separarlo por `.`. Por ejemplo:

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

Expression	Escenario
Intento: Slot. SubSlot	Aplicable únicamente a la subranura denominada «SubSlot» dentro de la ranura compuesta denominada «Ranura»
Intención:Slot.*	Aplicable a cualquier subslot dentro del slot compuesto denominado «Slot»
Intención: *. SubSlot	Aplicable únicamente a la ranura secundari a denominada 'SubSlot' dentro de cualquier ranura compuesta
Intención:*.*	Aplicable a cualquier subslot dentro de cualquier slot compuesto

Cómo funciona el comportamiento de interrupción en un bot Lex V2

Puede configurar el comportamiento de interrupción para el bot. Amazon Lex V2 define el atributo.

Permitir interrupción

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

Define si el usuario puede interrumpir el mensaje reproducido por el bot de Amazon Lex V2. Puede desactivarlo de forma selectiva.

Valor predeterminado: Verdadero

Cómo establecer tiempos de espera para la entrada de voz

Puede establecer valores de tiempo de espera para la interacción de voz con su bot mediante los atributos de sesión. Amazon Lex V2 define los atributos. Estos atributos le permiten especificar cuánto tiempo espera Amazon Lex V2 a que un cliente termine de hablar antes de recopilar la voz de entrada.

Todos los atributos predefinidos se encuentran en el espacio de nombres `x-amz-lex:audio`.

Longitud máxima del enunciado

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

Define cuánto tiempo espera Amazon Lex V2 antes de truncar la entrada de voz y devolverla a la aplicación. Puede aumentar el tiempo cuando se espera una gran cantidad de entradas o cuando desea dar a los clientes más tiempo para proporcionar información.

Predeterminado: 55 000 milisegundos (55 segundos). El valor máximo es de 55 000 milisegundos (55 segundos)

Si establece el `max-length-ms` atributo en más de 55 000 milisegundos, el valor predeterminado será 55 000 milisegundos.

Tiempo de espera de voz agotado

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

Cuánto tiempo espera un bot antes de presuponer que el cliente no va a hablar. Puede aumentar el tiempo en situaciones en las el cliente puede requerir más tiempo para buscar o recuperar información antes de hablar. Por ejemplo, es posible que desee dar a los clientes tiempo para sacar su tarjeta de crédito para que puedan escribir el número.

Valor predeterminado: 4000 milisegundos (4 segundos).

Tiempo de espera de silencio

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

Cuánto tiempo espera un bot después de que el cliente deje de hablar para dar por terminado el enunciado. Puede aumentar el tiempo en situaciones en las que se esperan períodos de silencio mientras se proporciona la entrada.

Valor predeterminado: 600 milisegundos (0,6 segundos)

Permitir entrada de audio

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

Puede habilitar este atributo para que el bot acepte las entradas del usuario únicamente mediante la modalidad de audio. El bot no aceptará la entrada de audio si este indicador está establecido en falso. De forma predeterminada, el valor se establece en verdadero.

Valor predeterminado: Verdadero

Tiempos de espera para la entrada de texto

Use el siguiente atributo de sesión para especificar cómo se comporta su bot en el modo de conversación de texto.

Este atributo se encuentra en el espacio de nombres `x-amz-lex:text`.

Umbral de tiempo de espera de inicio

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

Cuánto tiempo espera el bot antes de volver a solicitar al cliente que introduzca un texto. Puede aumentar el tiempo en situaciones en las que desee conceder al cliente más tiempo para buscar o recuperar información antes proporcionar la entrada de texto. Por ejemplo, es posible que desee dar a los clientes más tiempo para encontrar los detalles de su pedido. Como alternativa, puede reducir el límite para avisar a los clientes con mayor antelación.

Valor predeterminado: 30 000 milisegundos (30 segundos).

Cómo establecer la configuración para la entrada de DTMF

Utilice los siguientes atributos de sesión para especificar cómo responde su bot Amazon Lex V2 a la entrada de DTMF cuando utiliza una conversación de audio.

Todos los atributos predefinidos se encuentran en el espacio de nombres `x-amz-lex:dtmf`.

Carácter de eliminación

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

El carácter DTMF que borra los dígitos DTMF acumulados y finaliza inmediatamente la entrada.

Valor predeterminado: *

Carácter final

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

El carácter DTMF que finaliza inmediatamente la entrada. Si el usuario no presiona este carácter, la entrada finaliza después del tiempo de espera de finalización.

Valor predeterminado: #

Tiempo de espera agotado

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

Cuánto tiempo debe esperar el bot desde la última entrada de caracteres de DTMF antes de suponer que la entrada ha finalizado.

Valor predeterminado: 5000 milisegundos (5 segundos).

Número máximo de dígitos DTMF por enunciado

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

El número máximo de dígitos DTMF permitido en un enunciado. Por ejemplo, puede establecer este valor en 16 para limitar el número de caracteres que se pueden introducir para el número de una tarjeta de crédito. Este valor no se puede aumentar.

Predeterminado: 1024 caracteres

Permitir entrada DTMF

Puede configurar el tipo de entrada que el bot puede aceptar mediante los atributos de sesión. Amazon Lex V2 define los atributos.

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

Puede habilitar este atributo para que el bot acepte las entradas del usuario mediante la modalidad de DTMF. El bot no aceptará la entrada de DTMF si este indicador está establecido en falso. De forma predeterminada, el valor se establece en verdadero.

Valor predeterminado: Verdadero

Importación y exportación de bots en Lex V2

Puede exportar una definición de bot, una configuración regional de bot o un vocabulario personalizado y, a continuación, volver a importarlo para crear un nuevo recurso o para sobrescribir un recurso existente en una cuenta de AWS. Por ejemplo, puede exportar un bot de una cuenta de prueba y, a continuación, crear una copia del bot en su cuenta de producción. También puede copiar un bot de una región de AWS a otra región.

Puede cambiar los recursos del recurso exportado antes de importarlo. Por ejemplo, puede exportar un bot y, a continuación, editar el archivo JSON de un slot para añadir o eliminar enunciados que generen valores de slot de un slot específico. Cuando termine de editar la definición, puede importar el archivo modificado.

Temas

- Exportación de bots desde Lex V2
- Importación de bots en Lex V2
- Usar una contraseña al importar o exportar
- Formato JSON para importar y exportar bots en Lex V2

Exportación de bots desde Lex V2

Puede exportar un bot, una configuración regional de un bot o un vocabulario personalizado mediante la consola o la operación `CreatExport`. Usted especifica el recurso que desea exportar y puede proporcionar una contraseña opcional para ayudar a proteger el archivo .zip al iniciar una exportación. Después de descargar el archivo .zip, debe usar la contraseña para acceder al archivo antes de poder usarlo. Para obtener más información, consulte Usar una contraseña al importar o exportar.

Exportar es una operación asíncrona. Una vez iniciada la exportación, puede utilizar la consola o la operación `DescribeExport` para supervisar el progreso de la exportación. Una vez finalizada la exportación, la consola o la operación `DescribeExport` muestran el estado de `COMPLETED` y la consola descarga el archivo .zip de exportación en su navegador. Si utiliza la operación `DescribeExport`, Amazon Lex V2 proporciona una URL de Amazon S3 prefirmada en la que puede descargar los resultados de la exportación. La URL de descarga solo está disponible durante cinco minutos, pero puede obtener una nueva URL si vuelve a llamar a la operación `DescribeExport`.

Puede ver el historial de exportaciones de un recurso con la consola o con la operación `ListExports`. Los resultados muestran las exportaciones junto con su estado actual. Una exportación está disponible en el historial durante siete días.

Al exportar la versión `Draft` de un bot o una configuración regional de un bot, es posible que la definición del archivo JSON tenga un estado incoherente, ya que la versión `Draft` de un bot o una configuración regional de bot se puede cambiar mientras se está realizando la exportación. Si la versión `Draft` se cambia durante la exportación, es posible que los cambios no se incluyan en el archivo de exportación.

Al exportar la configuración regional de un bot, Amazon Lex exporta toda la información que la define, incluida la configuración regional, el vocabulario personalizado, las intenciones, los tipos de slot y los slots.

Al exportar un bot, Amazon Lex exporta todas las configuraciones regionales definidas para el bot, incluidas las intenciones, los tipos de slot y los slots. Los siguientes elementos no se exportan con un bot:

- Alias del bot
- ARN del rol asociado a un bot
- Etiquetas asociadas a los bots y a sus alias
- Enlaces de código Lambda asociados a un alias de bot

El ARN del rol y las etiquetas se ingresan como parámetros de solicitud al importar un bot. Debe crear alias de bots y asignar enlaces de código Lambda después de la importación, si es necesario.

Puede eliminar una exportación y el archivo .zip asociado mediante la consola o la operación `DeleteExport`.

Para ver un ejemplo de cómo exportar un bot mediante la consola, consulte Exportación de un bot Lex V2 (consola).

Permisos de IAM requeridos para exportar bots en Lex V2

Para exportar bots, configuraciones regionales de bots y vocabularios personalizados, el usuario que ejecute la exportación debe tener los siguientes permisos de IAM.

API	• Acciones obligatorias	Recurso
CreateExport	• CreateExport	Bot
UpdateExport	• UpdateExport	Bot
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	Bot
DescribeExportpara vocabularios personalizados	• DescribeExport • DescribeCustomVocabulary	bot
DeleteExport	• DeleteExport	Bot
ListExports	• ListExports	*

Para ver una política de IAM de ejemplo, consulte Permitir a un usuario exportar bots y configuraciones regionales de bots.

Exportación de un bot Lex V2 (consola)

Puede exportar un bot desde la lista de bots, desde la lista de versiones o desde la página de detalles de la versión. Al elegir una versión, Amazon Lex V2 exporta esa versión. En las instrucciones siguientes se parte del supuesto de que se empieza a exportar el bot de la lista de bots, pero cuando se empieza con una versión, los pasos son los mismos.

Exportar un bot utilizando la consola

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. Seleccione el bot para exportar de la lista de bots.
3. En Acción, seleccione Exportar.
4. Seleccione la versión del bot, la plataforma y el formato de exportación.
5. (Opcional) Introduzca una contraseña para el archivo .zip. Proporcionar una contraseña ayuda a proteger el archivo de salida.
6. Seleccione Exportar.

Después de iniciar la exportación, volverá a la lista de bots. Para supervisar el progreso de la exportación, utilice la lista de Historial de importación y exportación. Cuando el estado de la exportación indica Completado, la consola descarga automáticamente el archivo .zip a su ordenador.

Para volver a descargar la exportación, en la import/export lista, seleccione la exportación y, a continuación, elija Descargar. Puede proporcionar una contraseña para el archivo .zip descargado.

Exportar el idioma de un bot

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En la lista de bots, seleccione el bot cuyo idioma desea exportar.
3. En Añadir idiomas, seleccione Ver idiomas.
4. En la lista de Todos los idiomas, seleccione el idioma que desea exportar.
5. En Acción, seleccione Exportar.
6. Seleccione la versión del bot, la plataforma y el formato.
7. (Opcional) Introduzca una contraseña para el archivo .zip. Proporcionar una contraseña ayuda a proteger el archivo de salida.
8. Seleccione Exportar.

Después de iniciar la exportación, volverá a la lista de idiomas. Para supervisar el progreso de la exportación, utilice la lista de Historial de importación y exportación. Cuando el estado de la exportación indica Completado, la consola descarga automáticamente el archivo .zip a su ordenador.

Para volver a descargar la exportación, en la import/export lista, seleccione la exportación y, a continuación, elija Descargar. Puede proporcionar una contraseña para el archivo .zip descargado.

Importación de bots en Lex V2

Para usar la consola para importar un bot, una configuración regional o un vocabulario personalizado previamente exportados, debe proporcionar la ubicación del archivo en su ordenador local y la contraseña opcional para desbloquearlo. Para ver un ejemplo, consulta Importación de un bot Lex V2 (consola).

Cuando utilizas la API, la importación de un recurso es un proceso de tres pasos:

1. Cree una URL de carga mediante la operación `CreateUploadUrl`. No necesita crear una URL de carga para usar la consola.
2. Cargue el archivo .zip que contiene la definición del recurso.
3. Inicie la importación con la operación `StartImport`.

La URL de carga es una URL de Amazon S3 prefirmada con permiso de escritura. La URL estará disponible durante cinco minutos después de generarse. Si protege el archivo .zip con una contraseña, debe proporcionarla al iniciar la importación. Para obtener más información, consulte Usar una contraseña al importar o exportar.

Una importación es un proceso asíncrono. Puede monitorizar el progreso de una importación a través de la consola o de la operación `DescribeImport`.

Al importar un bot o una configuración regional de bots, puede haber conflictos entre los nombres de los recursos del archivo de importación y los nombres de los recursos existentes en Amazon Lex V2. Amazon Lex V2 puede gestionar el conflicto de tres maneras:

- Fallo en caso de conflicto: la importación se detiene y no se importa ningún recurso del archivo .zip de importación.
- Sobrescribir: Amazon Lex V2 importa todos los recursos del archivo .zip de importación y reemplaza cualquier recurso existente por la definición del archivo de importación.
- Añadir: Amazon Lex V2 importa todos los recursos del archivo .zip de importación y añade a cualquier recurso existente la definición del archivo de importación. Esta configuración solo está disponible para la configuración regional del bot.

Puede ver una lista de las importaciones a un recurso mediante la consola o la operación `ListImports`. Las importaciones permanecen en la lista durante siete días. Puede utilizar la consola o la operación `DescribeImport` para ver los detalles de una importación específica.

También puede eliminar una importación y el archivo .zip asociado mediante la consola o la operación `DeleteImport`.

Para ver un ejemplo de cómo importar un bot mediante la consola, consulte Importación de un bot Lex V2 (consola).

Permisos de IAM requeridos para importar

Para importar bots, configuraciones regionales de bots y vocabularios personalizados, el usuario que ejecute la importación debe tener los siguientes permisos de IAM.

API	Acciones obligatorias	Recurso
CreateUploadUrl	• CreateUploadUrl	*
StartImportpara el bot y la configuración regional del bot	• StartImport • objetivo: PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary	1. Para importar un bot nuevo: bot, alias de bot. 2. Para sobrescribir un bot existente: bot. 3. Para importar una nueva configuración regional: bot.

API	Acciones obligatorias	Recurso
	• DeleteLocale • DeleteIntent • DeleteSlot • DeleteSlotType	
StartImportpara vocabularios personalizados	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	bot
DescribeImport	• DescribeImport	Bot
DeleteImport	• DeleteImport	Bot
ListImports	• ListImports	*

Para ver una política de IAM de ejemplo, consulte Permitir a un usuario importar bots y configuraciones regionales de bots.

Importación de un bot Lex V2 (consola)

Importar un bot utilizando la consola

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En Acción, seleccione Importar.
3. En Archivo de entrada, asigne un nombre al bot y, a continuación, seleccione el archivo .zip que contiene los archivos JSON que definen al bot.
4. Si el archivo .zip está protegido con contraseña, introduzca la contraseña del archivo .zip. La protección del archivo mediante contraseña es opcional, pero ayuda a proteger el contenido.
5. Cree o ingrese el rol de IAM que define los permisos para el bot.
6. Indique si el bot está sujeto a la Ley de Protección de la Privacidad en Línea para Niños (COPPA).

7. Proporcione una configuración de tiempo de espera para su bot. Si no proporciona un valor, se utiliza el valor del archivo .zip. Si el archivo .zip no contiene una configuración de tiempo de espera, Amazon Lex V2 utiliza el valor predeterminado de 300 segundos (cinco minutos).
8. (Opcional) Agregue etiquetas para su bot.
9. Seleccione si desea avisar sobre la posibilidad de sobrescribir los bots existentes con el mismo nombre. Si habilita las advertencias, si el bot que va a importar sobrescribe a un bot existente, recibirá una advertencia y el bot no se importará. Si deshabilita las advertencias, el bot importado sustituirá al bot existente con el mismo nombre.
10. Seleccione Importar.

Después de iniciar la importación, volverá a la lista de bots. Para supervisar el progreso de la importación, utilice la lista de Historial de importación y exportación. Cuando el estado de la importación indique Completado, puede elegir el bot de la lista de bots para modificarlo o compilarlo.

Importar el idioma de un bot

1. Inicie sesión en la consola de administración de AWS y abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/casa.
2. En la lista de bots, seleccione el bot al que desea importar un idioma.
3. En Añadir idiomas, seleccione Ver idiomas.
4. En Acción, seleccione Importar.
5. En Archivo de entrada, seleccione el archivo que contiene el idioma que desea importar. Si protegió el archivo .zip, introduzca la contraseña en Contraseña.
6. En Idioma, seleccione el idioma en el que se va a importar. El idioma no tiene que coincidir con el idioma del archivo de importación. Puede copiar las intenciones de un idioma a otro.
7. En Voz, seleccione la voz de Amazon Polly para utilizarla en la interacción por voz, o seleccione Ninguna para un bot de solo texto.
8. En Umbral de puntuación de confianza, introduzca el umbral en el que Amazon Lex V2 inserta la `AMAZON.FallbackIntent`, la `AMAZON.KendraSearchIntent` o ambas al devolver intenciones alternativas.
9. Seleccione si desea avisar sobre la posibilidad de sobrescribir un idioma existente. Si habilita las advertencias, si el idioma que va a importar sobrescribe a un idioma existente, recibirá una advertencia y el idioma no se importará. Si deshabilita las advertencias, el idioma importado sustituirá al idioma existente.

10. Seleccione Importar para iniciar la importación del idioma.

Después de iniciar la importación, volverá a la lista de idiomas. Para supervisar el progreso de la importación, utilice la lista de Historial de importación y exportación. Cuando el estado de la importación indique Completado, puede elegir el idioma de la lista de bots para modificarlo o compilarlo.

Usar una contraseña al importar o exportar

Amazon Lex V2 puede proteger con contraseña los archivos de exportación o leer los archivos de importación protegidos mediante la compresión de archivos .zip estándar. Siempre debe proteger con contraseña sus archivos de importación y exportación.

Amazon Lex V2 envía el archivo de exportación a un bucket de S3 y está disponible con una URL de S3 prefirmada. La URL solo está disponible durante cinco minutos. El archivo está disponible para cualquier persona que tenga acceso a la URL de descarga. Para ayudar a proteger los datos del archivo, proporcione una contraseña al exportar el recurso. Si necesita obtener el archivo después de que caduque la URL, puede utilizar la consola o la operación `DescribeExport` para obtener una nueva URL.

Si pierde la contraseña de un archivo de exportación, puede crear una nueva contraseña para un archivo existente seleccionando Descargar de la tabla del import/export historial o utilizando la `UpdateExport` operación. Si selecciona Descargar en la tabla del historial para una exportación y no proporciona una contraseña, Amazon Lex V2 descargará un archivo .zip desprotegido.

Formato JSON para importar y exportar bots en Lex V2

Los bots, las configuraciones regionales de bots o los vocabularios personalizados se importan y exportan desde Amazon Lex V2 mediante un archivo .zip que contiene estructuras JSON que describen las partes del recurso. Al exportar un recurso, Amazon Lex V2 crea el archivo .zip y lo pone a su disposición mediante una URL prefirmada de Amazon S3. Al importar un recurso, debe crear un archivo .zip que contenga las estructuras JSON y subirlo a una URL prefirmada de S3.

Amazon Lex crea la siguiente estructura de directorios en el archivo .zip al exportar un bot. Al exportar la configuración regional de un bot, solo se exporta la estructura situada debajo de la configuración regional. Al exportar un vocabulario personalizado, solo se exporta la estructura incluida en el vocabulario personalizado.

```
BotName_BotVersion_ExportID_LexJson.zip
            -or-
BotName_BotVersion_LocaleId_ExportId_LEX_JSON.zip
        --> manifest.json
        --> BotName
        ----> Bot.json
        ----> BotLocales
        ------> Locale_A
        --------> BotLocale.json
        --------> Intents
        ----------> Intent_A
        ------------> Intent.json
        ------------> Slots
        --------------> Slot_A
        ----------------> Slot.json
        --------------> Slot_B
        ----------------> Slot.json
        ----------> Intent_B
                        ...
        --------> SlotTypes
        ----------> SlotType_A
        ------------> SlotType.json
        ----------> SlotType_B
                        ...
        --------> CustomVocabulary
        ------------> CustomVocabulary.json

        ------> Locale_B
                        ...
```

Estructura del archivo de manifiesto

El archivo de manifiesto contiene los metadatos del archivo de exportación.

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

Estructura del archivo del bot

El archivo del bot contiene la información de configuración para el bot.

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

Estructura del archivo de configuración regional del bot

El archivo de configuración regional del bot contiene una descripción de la configuración regional o el idioma de un bot. Al exportar un bot, puede haber más de un archivo de configuración regional del bot en el archivo .zip. Al exportar la configuración regional de un bot, solo hay una configuración regional en el archivo .zip.

```
{
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

Estructura del archivo de intenciones

El archivo de intenciones contiene la información de configuración para una intención. Hay un archivo de intenciones en el archivo .zip para cada intención en una configuración regional específica.

A continuación, se muestra un ejemplo de una estructura JSON para la BookCar intención del BookTrip bot de muestra. Para ver un ejemplo completo de la estructura JSON de una intención, consulta la CreateIntentoperación.

```
{
    "name": "BookCar",
    "identifier": "891RWHHICO",
    "description": "Intent to book a car.",
    "parentIntentSignature": null,
    "sampleUtterances": [
        {
            "utterance": "Book a car"
        },
        {
            "utterance": "Reserve a car"
        },
        {
            "utterance": "Make a car reservation"
        }
    ],
    "intentConfirmationSetting": {
        "confirmationPrompt": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have you down for a {CarType} hire in
 {PickUpCity} from {PickUpDate} to {ReturnDate}.  Should I book the reservation?"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "declinationResponse": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
```

```
                            "value": "OK, I have cancelled your reservation in
 progress."
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ]
        }
    },
    "intentClosingSetting": null,
    "inputContexts": null,
    "outputContexts": null,
    "kendraConfiguration": null,
    "dialogCodeHook": null,
    "fulfillmentCodeHook": null,
    "slotPriorities": [
        {
            "slotName": "DriverAge",
            "priority": 4
        },
        {
            "slotName": "PickUpDate",
            "priority": 2
        },
        {
            "slotName": "ReturnDate",
            "priority": 3
        },
        {
            "slotName": "PickUpCity",
            "priority": 1
        },
        {
            "slotName": "CarType",
            "priority": 5
        }
    ]
}
```

Estructura del archivo del slot

El archivo del slot contiene la información de configuración para el slot en una intención. Hay un archivo de slot en el archivo .zip para cada slot definido para una intención en una configuración regional específica.

El siguiente ejemplo es la estructura JSON de un espacio que permite al cliente elegir el tipo de coche que quiere alquilar según la BookCar intención del bot de BookTrip ejemplo. Para ver un ejemplo completo de la estructura JSON de una ranura, consulta la CreateSlotoperación.

```
{
    "name": "CarType",
    "identifier": "KDHJWNGZGC",
    "description": "Type of car being reserved.",
    "multipleValuesSetting": {
        "allowMutlipleValues": false
    },
    "slotTypeName": "CarTypeValues",
    "obfuscationSetting": null,
    "slotConstraint": "Required",
    "defaultValueSpec": null,
    "slotValueElicitationSetting": {
        "promptSpecification": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "What type of car would you like to rent?  Our
 most popular options are economy, midsize, and luxury"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "sampleValueElicitingUtterances": null,
        "waitAndContinueSpecification": null,
    }
```

```
}
```

La estructura JSON de un slot compuesto se muestra en el ejemplo siguiente.

```
{
  "name": "CarType",
  "identifier": "KDHJWNGZGC",
  "description": "Type of car being reserved.",
  "multipleValuesSetting": {
      "allowMutlipleValues": false
  },
  "slotTypeName": "CarTypeValues",
  "obfuscationSetting": null,
  "slotConstraint": "Required",
  "defaultValueSpec": null,
  "slotValueElicitationSetting": {
      "promptSpecification": {
          "messageGroupList": [
              {
                  "message": {
                      "plainTextMessage": {
                          "value": "What type of car would you like to rent?  Our most
 popular options are economy, midsize, and luxury"
                      },
                      "ssmlMessage": null,
                      "customPayload": null,
                      "imageResponseCard": null
                  },
                  "variations": null
              }
          ],
          "maxRetries": 2
      },
      "sampleValueElicitingUtterances": null,
      "waitAndContinueSpecification": null,
  },
  "subSlotSetting": {
    "slotSpecifications": {
      "firstname": {
        "valueElicitationSetting": {
          "promptSpecification": {
            "allowInterrupt": false,
            "messageGroupsList": [
```

```
            {
              "message": {
                "imageResponseCard": null,
                "ssmlMessage": null,
                "customPayload": null,
                "plainTextMessage": {
                  "value": "please provide firstname"
                }
              },
              "variations": null
            }
          ],
          "maxRetries": 2,
          "messageSelectionStrategy": "Random"
        },
        "defaultValueSpecification": null,
        "sampleUtterances": [
          {
            "utterance": "my name is {firstName}"
          }
        ],
        "waitAndContinueSpecification": null
      },
      "slotTypeId": "AMAZON.FirstName"
    },
    "eyeColor": {
      "valueElicitationSetting": {
        "promptSpecification": {
          "allowInterrupt": false,
          "messageGroupsList": [
            {
              "message": {
                "imageResponseCard": null,
                "ssmlMessage": null,
                "customPayload": null,
                "plainTextMessage": {
                  "value": "please provide eye color"
                }
              },
              "variations": null
            }
          ],
          "maxRetries": 2,
          "messageSelectionStrategy": "Random"
```

```
          },
          "defaultValueSpecification": null,
          "sampleUtterances": [
            {
              "utterance": "eye color is {eyeColor}"
            },
            {
              "utterance": "I have eyeColor eyes"
            }
          ],
          "waitAndContinueSpecification": null
        },
        "slotTypeId": "7FEVCB2PQE"
      }
    },
    "expression": "(firstname OR eyeColor)"
  }
}
```

Estructura de tipo de slot

El archivo de tipos de slots contiene la información de configuración de un tipo de slot personalizado utilizado en un idioma o configuración regional. Hay un archivo de tipos de slots en el archivo .zip para cada tipo de slot personalizado en una configuración regional específica.

La siguiente es la estructura JSON para el tipo de ranura, que muestra los tipos de coches disponibles en el bot de BookTrip ejemplo. Para ver un ejemplo completo de la estructura JSON de un tipo de ranura, consulta la CreateSlotTypeoperación.

```
{
    "name": "CarTypeValues",
    "identifier": "T1YUHGD9ZR",
    "description": "Enumeration representing possible types of cars available for
 hire",
    "slotTypeValues": [{
        "synonyms": null,
        "sampleValue": {
            "value": "economy"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
```

```
            "value": "standard"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "midsize"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "full size"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "luxury"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "minivan"
        }
    }],
    "parentSlotTypeSignature": null,
    "valueSelectionSetting": {
        "resolutionStrategy": "TOP_RESOLUTION",
        "advancedRecognitionSetting": {
            "audioRecognitionStrategy": "UseSlotValuesAsCustomVocabulary"
        },
        "regexFilter": null
    }
}
```

La estructura JSON de un tipo de slot compuesto se muestra en el ejemplo siguiente.

```
{
  "name": "CarCompositeType",
  "identifier": "TPA3CC9V",
  "description": null,
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": {
    "regexFilter": null,
```

```
    "resolutionStrategy": "CONCATENATION"
  },
  "compositeSlotTypeSetting": {
    "subSlots": [
      {
        "name": "model",
        "slotTypeId": "MODELTYPEID" # custom slot type Id for model
      },
      {
        "name": "city",
        "slotTypeId": "AMAZON.City"
      },
      {
        "name": "country",
        "slotTypeId": "AMAZON.Country"
      },
      {
        "name": "make",
        "slotTypeId": "MAKETYPEID" # custom slot type Id for make
      }
    ]
  }
}
```

El siguiente es un tipo de slot que utiliza una gramática personalizada para entender los enunciados del cliente. Para obtener más información, consulte Tipo de slot gramatical.

```
{
  "name": "custom_grammar",
  "identifier": "7KEAQIQKPX",
  "description": "Slot type using a custom grammar",
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": null,
  "externalSourceSetting": {
    "grammarSlotTypeSetting": {
      "source": {
        "kmsKeyArn": "arn:aws:kms:Region:123456789012:alias/customer-grxml-key",
        "s3BucketName": "grxml-test",
        "s3ObjectKey": "grxml_files/grammar.grxml"
      }
    }
  }
```

```
}
```

Estructura del archivo de vocabulario personalizado

El archivo de vocabulario personalizado contiene las entradas de un vocabulario personalizado para un solo idioma o configuración regional. Hay un archivo de vocabulario personalizado en el archivo .zip para cada configuración regional que tenga un vocabulario personalizado.

El siguiente es un archivo de vocabulario personalizado para un bot que recibe pedidos de restaurantes. Hay un archivo por configuración regional en el bot.

```
{
    "customVocabularyItems": [
        {
            "weight": 3,
            "phrase": "wafers"
        },
        {
            "weight": null,
            "phrase": "extra large"
        },
        {
            "weight": null,
            "phrase": "cremini mushroom soup"
        },
        {
            "weight": null,
            "phrase": "ramen"
        },
        {
            "weight": null,
            "phrase": "orzo"
        }
    ]
}
```

Recursos de etiquetado en Lex V2

Para ayudarle a administrar sus bots de Amazon Lex V2 y alias de bot, puede asignar metadatos a cada recurso como etiquetas. Una etiqueta es una marca que se asigna a un recurso de AWS. Cada etiqueta consta de una clave y un valor.

Las etiquetas le permiten clasificar los recursos de AWS de diversas maneras, por ejemplo, según su finalidad, propietario o aplicación. Las etiquetas 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 son los mismos. Por ejemplo, puede etiquetar un bot y las funciones de Lambda que utiliza con la misma etiqueta.
- Asignar costes. Las etiquetas se activan en el Administración de facturación y costos de AWS panel de control. AWS utiliza las etiquetas para clasificar los costos y enviarle un informe mensual de asignación de costos. Para Amazon Lex V2, puede asignar costos para cada alias utilizando etiquetas específicas para el alias. Para obtener más información, consulte Uso de etiquetas de asignación en la guía del usuario de Administración de facturación y costos de AWS.
- Controlar el acceso a los recursos. Puede usar etiquetas con Amazon Lex V2 para crear políticas que controlen el acceso a los recursos de Amazon Lex V2. Estas políticas se pueden adjuntar a un rol o usuario de IAM para permitir el control de acceso basado en etiquetas.

Puede trabajar con etiquetas mediante la consola de administración de AWSAWS Command Line Interface, la API Amazon Lex V2 o la API de Amazon Lex.

Cómo utilizar la consola o la API para etiquetar recursos

Si está utilizando la consola de Amazon Lex V2, puede etiquetar recursos al crearlos o agregar las etiquetas más tarde. También puede utilizar la consola para actualizar o quitar etiquetas existentes.

Si utiliza la CLI de AWS o la API Amazon Lex V2, debe utilizar las siguientes operaciones para gestionar las etiquetas de su recurso:

- CreateBot y CreateBotAlias: aplica etiquetas al crear un bot o un alias de bot.
- ListTagsForResource: muestra las etiquetas asociadas a un recurso.
- TagResource: agrega etiquetas a un recurso existente y las modifica.

- UntagResource: elimina las etiquetas de un recurso.

Los siguientes recursos de Amazon Lex V2 admiten el etiquetado:

- Bots: utilice un nombre de recurso de Amazon (ARN) como el siguiente:
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Alias de bot: use un ARN como el siguiente:
  - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

Los `bot-alias-id` valores `bot-id` y son cadenas alfanuméricas en mayúscula de 10 caracteres de longitud.

Restricciones de etiquetas al usar Lex V2

Las siguientes restricciones básicas se aplican a las etiquetas en los recursos de Amazon Lex V2:

- Número máximo de claves: 50 con la consola y 200 con la API
- 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 lo use `aws:` como prefijo para las claves, está reservado para el uso de AWS

Etiquetado de recursos (consola)

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

Agregar una etiqueta al crear un bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione Crear bot.

3. En la sección Configuración avanzada de Configurar ajustes del bot, seleccione Añadir nueva etiqueta. Puede agregar etiquetas al bot y al alias de `TestBotAlias`.
4. Seleccione Siguiente para seguir creando el bot.

Agregar una etiqueta al crear un alias de bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot al que desee agregar al alias del bot.
3. Seleccione Alias y, a continuación, seleccione Crear alias.
4. En Información general, seleccione Añadir nueva etiqueta desde Etiquetas.
5. Seleccione Crear.

Agregar, eliminar o modificar una etiqueta en un bot existente

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot que desea modificar.
3. En el menú izquierdo, seleccione Configuración y, a continuación, Editar.
4. En Etiquetas, realice los cambios que desee.
5. Seleccione Guardar para guardar los cambios en el bot.

Agregar, eliminar o modificar una etiqueta en un bot existente

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Seleccione el bot que desea modificar.
3. En el menú de la izquierda, seleccione Alias y, a continuación, en la lista de alias, seleccione el alias que desee modificar.
4. En Detalles del alias, en Etiquetas, seleccione Modificar etiquetas.
5. En Administrar etiquetas, realice los cambios que desee.
6. Seleccione Guardar para guardar los cambios en el alias.

Seguridad en Amazon Lex V2

La seguridad en la nube de AWS es la mayor prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y de centros de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El modelo de responsabilidad compartida describe estos conceptos como seguridad de la nube y seguridad en la nube:

- Seguridad en la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Third-party los auditores prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de conformidad de AWS Programas de AWS. Para obtener información sobre los programas de conformidad que se aplican a Amazon Lex V2, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le permite comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon Lex V2. En los siguientes temas, se mostrará cómo configurar Amazon Lex V2 para satisfacer sus objetivos de seguridad y conformidad. También puede aprender a utilizar otros servicios de AWS que ayudan a monitorizar y proteger los recursos de Amazon Lex V2.

Temas

- Protección de los datos en Amazon Lex V2
- Administración de identidades y accesos para Amazon Lex V2
- Registrar y monitorear en Amazon Lex V2
- Validación de la conformidad en Amazon Lex V2
- Resiliencia en Amazon Lex V2
- Seguridad de la infraestructura en Amazon Lex V2
- Amazon Lex V2 y puntos de enlace de VPC de interfaz (AWS PrivateLink)

Protección de los datos en Amazon Lex V2

AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los servicios de AWS. AWS mantiene el control sobre los datos alojados en esta infraestructura, incluidos los controles de la configuración de seguridad para la gestión del contenido y los datos personales de los clientes. Los clientes de AWS y los socios de APN, que actúan como controladores de datos o procesadores de datos, son responsables de los datos personales que ponen en la nube de AWS.

Para fines de protección de datos, le recomendamos proteger las credenciales de la cuenta de AWS y configurar cuentas de usuario individuales con AWS Identity and Access Management (IAM), de modo que a cada usuario se le concedan únicamente los permisos necesarios para llevar a cabo su trabajo. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con los recursos de AWS.

- Configure la API y el registro de la actividad de los usuarios con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS junto con todos los controles de seguridad predeterminados dentro de los servicios de AWS.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.

Le recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, números de cuenta de sus clientes, en los campos de formato libre, como el campo Nombre. Esto incluye cuando trabaja con Amazon Lex V2 u otros servicios de AWS mediante la consola, la API, la CLI de AWS o los SDK de AWS. Es posible que cualquier dato que ingrese en Amazon Lex V2 u otros servicios se incluya en los registros de diagnóstico. Cuando proporcione una URL a un servidor externo, no incluya información de credenciales en la URL para validar la solicitud para ese servidor.

Para obtener más información sobre la protección de datos, consulte la entrada de blog sobre modelo de responsabilidad compartida de AWS y RGPD en el blog de seguridad de AWS.

Cifrado en reposo

Amazon Lex V2 cifra los enunciados de los usuarios y otra información que almacena.

Temas

- Enunciados de muestra
- Atributos de sesión
- Atributos de solicitud

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 cifra en reposo, y se utiliza únicamente para compilar el bot y la experiencia del cliente.

Atributos de sesión

Los atributos de sesión contienen información específica de la aplicación que se transfiere entre Amazon Lex V2 y las aplicaciones cliente. Amazon Lex V2 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 V2 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 V2 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 V2 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 identidades y accesos para Amazon Lex V2

AWS Identity and Access Management (IAM) ayuda al administrador a controlar de forma segura el acceso a los recursos de AWS. Servicio de AWS Los administradores de IAM controlan quién se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon Lex V2. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración del acceso con políticas
- Cómo funciona Amazon Lex V2 con IAM
- Identity-based ejemplos de políticas para Amazon Lex V2
- Resource-based ejemplos de políticas para Amazon Lex V2
- Políticas administradas por AWS para Amazon Lex V2
- Roles vinculados a servicios para Amazon Lex V2
- Uso de funciones de servicio para Amazon Lex V2
- Migración a funciones de servicio para Amazon Lex V2
- Solución de problemas de identidad y acceso de Amazon Lex V2

Público

La forma en que utiliza AWS Identity and Access Management (IAM) varía según su función:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de identidad y acceso de Amazon Lex V2).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona Amazon Lex V2 con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Identity-based ejemplos de políticas para Amazon Lex V2).

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir un rol de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte la versión 4 de AWS Signature para ver las solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario raíz

Al crear una Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz, que tiene acceso completo a todos Servicios de AWS los recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de 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 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 empresarial, del proveedor de identidades web o al Directory Service que se accede Servicios de AWS mediante credenciales de una fuente de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, se recomienda AWS IAM Identity Center. 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 con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder a AWS con credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una operación de AWS CLI o API de AWS. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

Puede controlar el acceso a AWS creando políticas y adjuntándolas a las identidades o los recursos de AWS. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Identity-based políticas

Identity-based las políticas son documentos de política de permisos de JSON que se adjuntan a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Identity-based las políticas pueden ser políticas integradas (integradas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Resource-based políticas

Resource-based las políticas son documentos de políticas de JSON que se adjuntan a un recurso. Los ejemplos incluyen 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. Debe especificar una entidad principal en una política basada en recursos.

Resource-based las políticas son políticas en línea que se encuentran en ese servicio. No puede utilizar políticas administradas de AWS de IAM en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos que conceden los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCP): definen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations .
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener 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 obtener información acerca de cómo AWS determina si permitir una solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon Lex V2 con IAM

Antes de utilizar IAM para administrar el acceso a Amazon Lex V2, obtenga información sobre qué características de IAM se encuentran disponibles con Amazon Lex V2.

Características de IAM que puede utilizar con Amazon Lex V2

Característica de IAM	Soporte de Amazon Lex V2
Identity-based políticas	Sí
Resource-based políticas	Sí
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	No
ACL	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	No
Permisos de entidades principales	Sí
Roles de servicio	Sí
Service-linked roles	Parcial

Para obtener una visión general de cómo funcionan Amazon Lex V2 y otros servicios de AWS con la mayoría de las funciones de IAM, consulte los servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Identity-based políticas para Amazon Lex V2

Soporta políticas basadas en la identidad: sí

Identity-based las políticas son documentos de política de permisos de JSON que se pueden adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. 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 la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente 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. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de la política de JSON de IAM en la Guía del usuario de IAM.

Identity-based ejemplos de políticas para Amazon Lex V2

Para ver ejemplos de políticas basadas en identidad de Amazon Lex V2, consulte Identity-based ejemplos de políticas para Amazon Lex V2.

Resource-based políticas de Amazon Lex V2

Soporta políticas basadas en recursos: sí

Resource-based las políticas son documentos de política JSON que se adjuntan a un recurso. Los 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 basada en recursos. Las entidades principales pueden incluir usuarios, roles, usuarios federados o servicios de AWS.

No puede utilizar políticas multicuenta o multiregión con Amazon Lex V2. Si crea una política para un recurso con un ARN entre cuentas o regiones, Amazon Lex V2 devuelve un error.

El servicio Amazon Lex V2 admite políticas basadas en recursos denominadas política de bots y política de alias de bots, que se adjuntan a un bot o a un alias de bot. Estas políticas definen qué entidades principales pueden realizar acciones en el bot o en el alias del bot.

Las acciones solo se pueden usar en recursos específicos. Por ejemplo, la acción `UpdateBot` solo se puede usar en los recursos del bot, la acción `UpdateBotAlias` solo se puede usar en los recursos de alias del bot. Si especifica una acción en una política que no se puede utilizar en el

recurso especificado en la política, Amazon Lex V2 devuelve un error. Para ver la lista de acciones y los recursos con los que se pueden utilizar, consulte la siguiente tabla.

Action	Compatibilidad con las políticas basadas en recursos	Recurso
BuildBotLocale	compatible	BotId
CreateBot	No	
CreateBotAlias	No	
CreateBotChannel [solo permiso]	Soportado	BotId
CreateBotLocale	Soportado	BotId
CreateBotVersion	Soportado	BotId
CreateExport	Soportado	BotId
CreateIntent	Soportado	BotId
CreateResourcePolicy	Soportado	BotId, BotAliasId
CreateSlot	Soportado	BotId
CreateSlotType	compatible	BotId
CreateUploadUrl	No	
DeleteBot	Soportado	BotId, BotAliasId
DeleteBotAlias	compatible	BotAliasId
DeleteBotChannel [solo permiso]	Soportado	BotId
DeleteBotLocale	Soportado	BotId
DeleteBotVersion	Soportado	BotId

Action	Compatibilidad con las políticas basadas en recursos	Recurso
DeleteExport	Soportado	BotId
DeleteImport	Soportado	BotId
DeleteIntent	Soportado	BotId
DeleteResourcePolicy	Soportado	BotId, BotAliasId
DeleteSession	Soportado	BotAliasId
DeleteSlot	Soportado	BotId
DeleteSlotType	Soportado	BotId
DescribeBot	Soportado	BotId
DescribeBotAlias	compatible	BotAliasId
DescribeBotChannel [solo permiso]	Soportado	BotId
DescribeBotLocale	Soportado	BotId
DescribeBotVersion	Soportado	BotId
DescribeExport	Soportado	BotId
DescribeImport	Soportado	BotId
DescribeIntent	Soportado	BotId
DescribeResourcePolicy	Soportado	BotId, BotAliasId
DescribeSlot	Soportado	BotId
DescribeSlotType	Soportado	BotId
GetSession	Soportado	BotAliasId

Action	Compatibilidad con las políticas basadas en recursos	Recurso
ListBotAliases	compatible	BotId
ListBotChannels [solo permiso]	Soportado	BotId
ListBotLocales	compatible	BotId
ListBots	No	
ListBotVersions	compatible	BotId
ListBuiltInIntents	No	
ListBuiltIntSlotTypes	No	
ListExports	No	
ListImports	No	
ListIntents	Soportado	BotId
ListSlots	Soportado	BotId
ListSlotTypes	Soportado	BotId
PutSession	Soportado	BotAliasId
RecognizeText	Soportado	BotAliasId
RecognizeUtterance	Soportado	BotAliasId
StartConversation	Soportado	BotAliasId
StartImport	compatible	BotId, BotAliasId
TagResource	No	
UpdateBot	Soportado	BotId

Action	Compatibilidad con las políticas basadas en recursos	Recurso
UpdateBotAlias	Soportado	BotAliasId
UpdateBotLocale	Soportado	BotId
UpdateBotVersion	Soportado	BotId
UpdateExport	Soportado	BotId
UpdateIntent	Soportado	BotId
UpdateResourcePolicy	Soportado	BotId, BotAliasId
UpdateSlot	Soportado	BotId
UpdateSlotType	compatible	BotId
UntagResource	No	

Para obtener información sobre cómo asociar una política basada en un recurso a un bot o a un alias de un bot, consulte Resource-based ejemplos de políticas para Amazon Lex V2.

Resource-based ejemplos de políticas en Amazon Lex V2

Para ver ejemplos de políticas basadas en recursos de Amazon Lex V2, consulte Resource-based ejemplos de políticas para Amazon Lex V2.

Acciones de políticas para Amazon Lex V2

Soporta acciones políticas: sí

Los administradores pueden usar las políticas JSON de AWS 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. 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 V2, consulte Acciones definidas por Amazon Lex V2 en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Amazon Lex V2 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"
         ]
```

Para ver ejemplos de políticas basadas en identidad de Amazon Lex V2, consulte Identity-based ejemplos de políticas para Amazon Lex V2.

Recursos de políticas para Amazon Lex V2

Apoya los recursos políticos: sí

Los administradores pueden usar las políticas JSON de AWS 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. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de tipos de recursos de Amazon Lex V2 y sus ARN, consulte Tipos de recurso definidos por Amazon Lex V2 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 V2.

Para ver ejemplos de políticas basadas en identidad de Amazon Lex V2, consulte Identity-based ejemplos de políticas para Amazon Lex V2.

Claves de condiciones de políticas para Amazon Lex V2

Admite claves de condición de políticas específicas del servicio: No

Los administradores pueden usar las políticas JSON de AWS 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` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Para ver una lista de las claves de condición de Amazon Lex V2, consulte Claves de condición para Amazon Lex V2 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 V2.

Para ver ejemplos de políticas basadas en identidad de Amazon Lex V2, consulte Identity-based ejemplos de políticas para Amazon Lex V2.

Listas de control de acceso (ACL) de Amazon Lex V2

Soporta 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.

Attribute-based control de acceso (ABAC) con Amazon Lex V2

Compatible con ABAC (etiquetas en las políticas): sí

Attribute-based el control de acceso (ABAC) es una estrategia de autorización que define los permisos en función de unos atributos denominados etiquetas. Puede adjuntar etiquetas a las entidades de IAM y a los recursos de AWS y, a continuación, diseñar políticas de ABAC para permitir las operaciones cuando la etiqueta del principal coincida con la etiqueta del recurso.

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 Definición de permisos con la autorización de 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.

Usar credenciales temporales con Amazon Lex V2

Admite credenciales temporales: no

Las credenciales temporales proporcionan acceso a corto plazo a los recursos de AWS y se crean automáticamente cuando se utiliza la federación o se cambia de rol. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM y Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Cross-service permisos principales para Amazon Lex V2

Admite sesiones de acceso directo (FAS): sí

Las sesiones de acceso directo (FAS) utilizan los permisos de la persona principal que llama y Servicio de AWS, además, los que solicitan Servicio de AWS para realizar solicitudes a los servicios descendentes. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Sesiones de acceso directo.

Roles de servicio para Amazon Lex V2

Soporta funciones 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 Crear 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 V2. Edite los roles de servicio solo cuando Amazon Lex V2 proporcione orientación para hacerlo.

Service-linked funciones para Amazon Lex V2

Soporta funciones vinculadas al servicio: parcial

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 la función de realizar una acción en su nombre. Service-linked las funciones aparecen en su nombre 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 la creación o la administración de funciones vinculadas a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya un servicio **`Yes`** en la Service-linked columna de roles. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Identity-based ejemplos de políticas para Amazon Lex V2

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Amazon Lex V2. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

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 (consola) 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 V2, incluido el formato de los ARN para cada tipo de recurso, consulte Acciones, recursos y claves de condición para Amazon Lex V2 en la Referencia de autorizaciones de servicio.

Temas

- Prácticas recomendadas sobre las políticas
- Usar la consola de Amazon Lex V2
- Permitir a los usuarios añadir funciones a un bot

- Permitir a los usuarios añadir canales a un bot
- Permitir a los usuarios crear y actualizar bots
- Permitir a los usuarios utilizar el Diseñador de chatbots automatizados
- Permita a los usuarios utilizar un AWS KMS clave para cifrar y descifrar archivos
- Permitir a los usuarios eliminar bots
- Permitr a los usuarios mantener una conversación con un bot
- Permitir que un usuario específico administre políticas basadas en recursos
- Permitir a un usuario exportar bots y configuraciones regionales de bots
- Permitir que un usuario exporte un vocabulario personalizado
- Permitir a un usuario importar bots y configuraciones regionales de bots
- Permitir que un usuario importe un vocabulario personalizado
- Permitir a un usuario migrar un bot de Amazon Lex a Amazon Lex V2
- Cómo permitir a los usuarios consultar sus propios permisos
- Permitir que un usuario dibuje el flujo de la conversación con el generador visual de conversaciones de Amazon Lex V2
- Cómo permitir a los usuarios crear y ver réplicas de bots, pero no eliminarlas

Prácticas recomendadas sobre las políticas

Identity-based las políticas determinan si alguien puede crear, acceder o eliminar los recursos de Amazon Lex V2 de su 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 administradas de AWS y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas administradas de AWS 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 los clientes de AWS que sean específicas para sus casos de uso. Para obtener más información, consulte Políticas administradas por AWS o Políticas administradas por AWS para funciones laborales en la Guía del 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 puede utilizar condiciones para conceder el acceso a las acciones de servicio si se utilizan a través de una actividad específica Servicio de AWS, por ejemplo. CloudFormation Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza 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 Validación de políticas con el Analizador de acceso de IAM en la Guía del 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 exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Usar la consola de Amazon Lex V2

Para acceder a la consola de Amazon Lex V2, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de Amazon Lex V2 de su propiedad 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 es necesario que conceda permisos mínimos para la consola a los usuarios que solo realizan llamadas a AWS CLI o a la API de AWS. En su lugar, permita el acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para asegurarse de que los usuarios y los roles puedan seguir utilizando la consola de Amazon Lex V2, los usuarios deben tener acceso a la consola. Para obtener más información sobre la creación de un usuario con acceso a la consola, consulte Crear un usuario de IAM en su cuenta de AWS en la Guía del usuario de IAM.

Permitir a los usuarios añadir funciones a un bot

En este ejemplo, se muestra una política que permite a los usuarios de IAM añadir permisos de Amazon Comprehend, análisis de opiniones y consultas de Amazon Kendra a un bot de Amazon Lex V2.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
lexv2.amazonaws.com/AWSServiceRoleForLexV2Bots*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
lexv2.amazonaws.com/AWSServiceRoleForLexV2Bots*"
        }
    ]
}
```

Permitir a los usuarios añadir canales a un bot

Este ejemplo es una política que permite a los usuarios de IAM añadir un canal de mensajería a un bot. Los usuarios deben contar con esta política antes de poder implementar un bot en una plataforma de mensajería.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        }
    ]
}
```

Permitir a los usuarios crear y actualizar bots

En este ejemplo se muestra un ejemplo de política que permite a los usuarios de IAM crear y actualizar cualquier bot. La política incluye permisos para completar esta acción en la consola o mediante la AWS CLI o la API de AWS.

Permitir a los usuarios utilizar el Diseñador de chatbots automatizados

En este ejemplo se muestra un ejemplo de política que permite a los usuarios de IAM ejecutar el Diseñador de chatbots automatizados.

Permita a los usuarios utilizar un AWS KMS clave para cifrar y descifrar archivos

En este ejemplo se muestra un ejemplo de política que permite a los usuarios de IAM utilizar una clave gestionada por el AWS KMS cliente para cifrar y descifrar datos.

Permitir a los usuarios eliminar bots

En este ejemplo se muestra un ejemplo de política que permite a los usuarios de IAM eliminar cualquier bot. La política incluye permisos para completar esta acción en la consola o mediante la AWS CLI o la API de AWS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:DeleteBot",
                "lex:DeleteBotLocale",
                "lex:DeleteBotAlias",
                "lex:DeleteIntent",
                "lex:DeleteSlot",
                "lex:DeleteSlottype"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123412341234:bot/*",
                "arn:aws:lex:us-east-1:123412341234:bot-alias/*"
            ]
        }
    ]
}
```

Permitr a los usuarios mantener una conversación con un bot

En este ejemplo se muestra un ejemplo de política que permite a los usuarios de IAM tener una conversación con cualquier bot. La política incluye permisos para completar esta acción en la consola o mediante la AWS CLI o la API de AWS.

JSON

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

```
        {
            "Action": [
                "lex:StartConversation",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:GetSession",
                "lex:PutSession",
                "lex:DeleteSession"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:lex:us-east-1:123412341234:bot-alias/*"
        }
    ]
}
```

Permitir que un usuario específico administre políticas basadas en recursos

El siguiente ejemplo concede permiso a un usuario específico para gestionar las políticas basadas en recursos. Permite el acceso desde la consola y la API a las políticas asociadas a los bots y sus alias.

Permitir a un usuario exportar bots y configuraciones regionales de bots

La siguiente política de permisos de IAM permite a los usuarios crear, actualizar y exportar un bot o una configuración regional de bots.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeBot",
                "lex:DescribeBotLocale",
                "lex:ListBotLocales",
                "lex:DescribeIntent",
                "lex:ListIntents",
                "lex:DescribeSlotType",
```

```
                "lex:ListSlotTypes",
                "lex:DescribeSlot",
                "lex:ListSlots",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*"
            ]
        }
    ]
}
```

Permitir que un usuario exporte un vocabulario personalizado

La siguiente política de permisos de IAM permite a los usuarios exportar un vocabulario personalizado desde la configuración regional de un bot.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*"
            ]
        }
    ]
}
```

Permitir a un usuario importar bots y configuraciones regionales de bots

La siguiente política de permisos de IAM permite a los usuarios importar un bot o una configuración regional de bots y comprobar el estado de una importación.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateBot",
                "lex:UpdateBot",
                "lex:DeleteBot",
                "lex:CreateBotLocale",
                "lex:UpdateBotLocale",
                "lex:DeleteBotLocale",
                "lex:CreateIntent",
                "lex:UpdateIntent",
                "lex:DeleteIntent",
                "lex:CreateSlotType",
                "lex:UpdateSlotType",
                "lex:DeleteSlotType",
                "lex:CreateSlot",
                "lex:UpdateSlot",
                "lex:DeleteSlot",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*",
                "arn:aws:lex:us-east-1:123456789012:bot-alias/*"
            ]
        }
    ]
```

```
}
```

Permitir que un usuario importe un vocabulario personalizado

La siguiente política de permisos de IAM permite a los usuarios importar un vocabulario personalizado a la configuración regional de un bot.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*"
            ]
        }
    ]
}
```

Permitir a un usuario migrar un bot de Amazon Lex a Amazon Lex V2

La siguiente política de permisos de IAM permite a un usuario empezar a migrar un bot de Amazon Lex a Amazon Lex V2.

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 llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Permitir que un usuario dibuje el flujo de la conversación con el generador visual de conversaciones de Amazon Lex V2

La siguiente política de permisos de IAM permite a un usuario. dibujar el flujo de la conversación con el generador visual de conversaciones de Amazon Lex V2

Cómo permitir a los usuarios crear y ver réplicas de bots, pero no eliminarlas

Puede adjuntar los siguientes permisos a un rol de IAM para que solo pueda crear y ver réplicas de bots. Al omitir `lex:DeleteBotReplica`, evita que el rol elimine las réplicas de los bots. Para obtener más información, consulte Permisos para replicar bots y administrar réplicas de bots en Lex V2.

Resource-based ejemplos de políticas para Amazon Lex V2

Se asocia una política basada en recursos a un recurso, como un bot o un alias de bot. Con una política basada en recursos puede especificar quién tiene acceso al recurso y qué acciones puede realizar en él. Por ejemplo, puede añadir políticas basadas en recursos que permitan a un usuario modificar un bot específico o utilizar operaciones en tiempo de ejecución en un alias de bot específico.

Cuando utiliza una política basada en recursos, puede permitir que otros servicios de AWS accedan a los recursos de su cuenta. Por ejemplo, puede permitir que Connect Customer acceda a un bot de Amazon Lex V2.

Para obtener más información acerca de cómo crear un bot, consulte Uso de bots Amazon Lex V2.

Temas

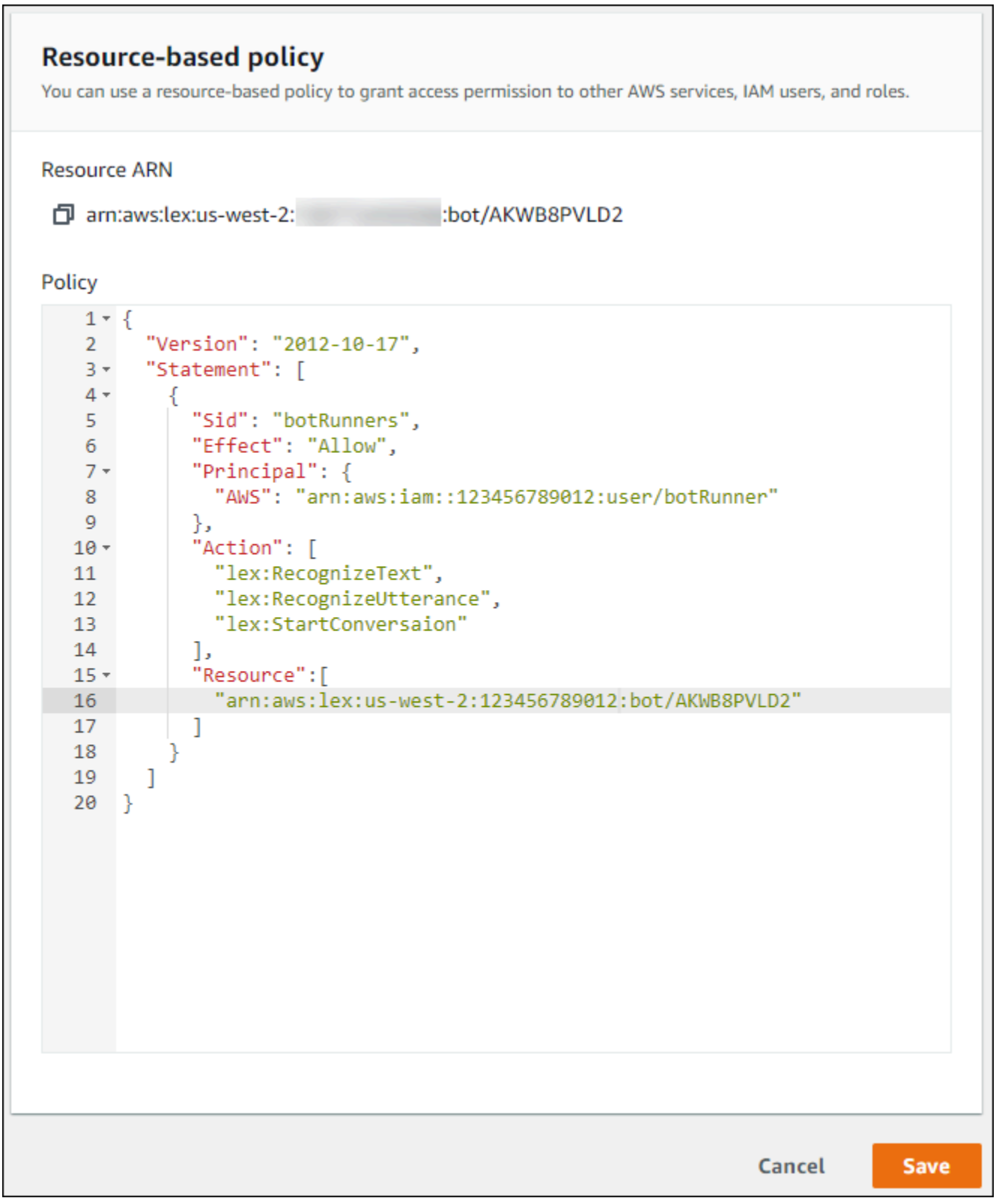
- Utilizar la consola para especificar una política basada en recursos
- Utilizar la API para especificar una política basada en recursos
- Permitir que un rol de IAM actualice un bot y enumere sus alias
- Permitir a los usuarios mantener una conversación con un bot
- Permitir que un servicio de AWS utilice un bot Amazon Lex V2 específico

Utilizar la consola para especificar una política basada en recursos

Puede utilizar la consola Amazon Lex V2 para gestionar las políticas basadas en recursos para sus bots y sus alias. Usted introduce la estructura JSON de una política y la consola la asocia al recurso. Si ya hay una política asociada a un recurso, puede usar la consola para ver y modificar la política.

Al guardar una política en el editor de políticas, la consola comprueba la sintaxis de la política. Si la política contiene errores, como un usuario inexistente o una acción no compatible con el recurso, devuelve un error y no guarda la política.

A continuación, se muestra el editor de políticas basado en recursos para un bot en la consola. El editor de políticas de un alias de bot es similar.


Resource-based policy
You can use a resource-based policy to grant access permission to other AWS services, IAM users, and roles.
Resource ARN
arn:aws:lex:us-west-2: :bot/AKWB8PVLD2
Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "botRunners",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::123456789012:user/botRunner"
},
"Action": [
"lex:RecognizeText",
"lex:RecognizeUtterance",
"lex:StartConversaion"
],
"Resource":[
"arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"
]
}
]
}
Cancel
Save

Abrir el editor de políticas de un bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. De la lista de bots, seleccione el bot de la política que desea editar.
3. En la sección Resource-based de políticas, elija Editar.

Abrir el editor de políticas de un alias de bot

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En la lista de bots, seleccione el bot que contiene el alias que quiere editar.
3. En el menú de la izquierda, seleccione Alias y, a continuación, seleccione el alias que desea editar.
4. En la sección Resource-based de políticas, elija Editar.

Utilizar la API para especificar una política basada en recursos

Puede usar las operaciones de la API para administrar las políticas basadas en recursos para sus bots y alias de bots. Existen operaciones para crear, actualizar y eliminar políticas.

CreateResourcePolicy

Agrega una nueva política de recursos con las instrucciones de política especificadas a un bot o alias de bot.

CreateResourcePolicyStatement

Agrega una nueva declaración de política de recursos a un bot o alias de bot.

DeleteResourcePolicy

Elimina una política de recursos de un bot o un alias de bot.

DeleteResourcePolicyStatement

Elimina una declaración de política de recursos de un bot o un alias de bot.

DescribeResourcePolicy

Obtiene una política de recursos y la revisión de la política.

[UpdateResourcePolicy](#)

Sustituye la política de recursos existente para un bot o un alias de bot por una nueva.

Ejemplos

Java

En el siguiente ejemplo se muestra cómo utilizar las operaciones de política basada en recursos para administrar una política basada en recursos.

```
        /*
         * Create a new policy for the specified bot alias
         * that allows a role to invoke lex:UpdateBotAlias on it.
         * The created policy will have revision id 1.
         */

        CreateResourcePolicyRequest createPolicyRequest =
                CreateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Allow\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.createResourcePolicy(createPolicyRequest);

        /*
         * Overwrite the policy for the specified bot alias with a new policy.
         * Since no expectedRevisionId is provided, this request overwrites the
 current revision.
         * After this update, the revision id for the policy is 2.
         */
        UpdateResourcePolicyRequest updatePolicyRequest =
        UpdateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Deny\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
```

```
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.updateResourcePolicy(updatePolicyRequest);

        /*
         * Creates a statement in an existing policy for the specified bot alias
         * that allows a role to invoke lex:RecognizeText on it.
         * This request expects to update revision 2 of the policy. The request will
 fail
         * if the current revision of the policy is no longer revision 2.
         * After this request, the revision id for this policy will be 3.
         */

        CreateResourcePolicyStatementRequest createStatementRequest =
                CreateResourcePolicyStatementRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .effect("Allow")

 .principal(Principal.builder().arn("arn:aws:iam::123456789012:role/
BotRunner").build())
                        .action("lex:RecognizeText")
                        .statementId("BotRunnerStatement")
                        .expectedRevisionId(2)
                        .build();

        lexmodelsv2Client.createResourcePolicyStatement(createStatementRequest);

        /*
         * Deletes a statement from an existing policy for the specified bot alias
 by statementId.
         * Since no expectedRevisionId is supplied, the request will remove the
 statement from
         * the current revision of the policy for the bot alias.
         * After this request, the revision id for this policy will be 4.
         */
        DeleteResourcePolicyRequest deleteStatementRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .statementId("BotRunnerStatement")
                        .build();
```

```
        lexmodelsv2Client.deleteResourcePolicy(deleteStatementRequest);

        /*
         * Describe the current policy for the specified bot alias
         * It always returns the current revision.
         */
        DescribeResourcePolicyRequest describePolicyRequest =
                DescribeResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .build();

        lexmodelsv2Client.describeResourcePolicy(describePolicyRequest);


        /*
         * Delete the current policy for the specified bot alias
         * This request expects to delete revision 3 of the policy. Since the
 revision id for
         * this policy is already at 4, this request will fail.
         */
        DeleteResourcePolicyRequest deletePolicyRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .expectedRevisionId(3);
                        .build();

        lexmodelsv2Client.deleteResourcePolicy(deletePolicyRequest);
```

Permitir que un rol de IAM actualice un bot y enumere sus alias

El siguiente ejemplo otorga permisos para que un rol de IAM específico llame a las operaciones de la API de creación de modelos de Amazon Lex V2 para modificar un bot existente. El usuario puede enumerar los alias de un bot y actualizarlo, pero no puede eliminar el bot o los alias del bot.

JSON

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

```
        {
            "Sid": "botBuilders",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/BotBuilder"
            },
            "Action": [
                "lex:ListBotAliases",
                "lex:UpdateBot"
            ],
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/MYBOT"
            ]
        }
    ]
}
```

Permitir a los usuarios mantener una conversación con un bot

El siguiente ejemplo concede permiso a un usuario específico para llamar a las operaciones de la API de tiempo de ejecución de Amazon Lex V2 en un único alias de un bot.

Se deniega específicamente al usuario el permiso para actualizar o eliminar el alias del bot.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "botRunners",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation",
                "lex:DeleteSession",
                "lex:GetSession",
                "lex:PutSession"
```

```
            ],
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        },
        {
            "Sid": "botRunners",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:UpdateBotAlias",
                "lex:DeleteBotAlias"
            ],
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        }
    ]
}
```

Permitir que un servicio de AWS utilice un bot Amazon Lex V2 específico

El siguiente ejemplo concede permiso a Connect Customer para AWS Lambda llamar a las operaciones de la API en tiempo de ejecución de Amazon Lex V2.

El bloque de condiciones es obligatorio para las entidades principales de servicio y debe usar las claves de contexto globales `AWS:SourceAccount` y `AWS:SourceArn`.

`AWS:SourceAccount` es el ID de cuenta que llama al bot de Amazon Lex V2.

`AWS:SourceArn`Es el ARN del recurso de la instancia de servicio al cliente de Connect o la función Lambda desde la que se origina la llamada al alias del bot Amazon Lex V2.

Políticas administradas por AWS para Amazon Lex V2

Una política administrada por AWS es una política independiente creada y administrada por AWS. Las políticas administradas de AWS están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y roles.

Tenga en cuenta que es posible que las políticas administradas por AWS no otorguen permisos de privilegios mínimos para sus casos de uso específicos, ya que están disponibles para que los usen todos los clientes de AWS. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puede cambiar los permisos definidos en las políticas gestionadas por AWS. Si AWS actualiza los permisos definidos en una política administrada por AWS, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. Lo más probable es que AWS actualice una política gestionada por AWS cuando Servicio de AWS se lance una nueva o cuando haya nuevas operaciones de API disponibles para los servicios existentes.

Para obtener más información, consulte las políticas administradas por AWS en la Guía del usuario de IAM.

Política gestionada por AWS: AmazonLexReadOnly

Puede adjuntar la política de `AmazonLexReadOnly` a sus 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 de los permisos

Esta política incluye los permisos siguientes:

- `lex`— Read-only acceso a los recursos de Amazon Lex V2 y Amazon Lex en el servicio de creación de modelos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonLexReadOnlyStatement1",
            "Effect": "Allow",
            "Action": [
                "lex:GetBot",
```

```
                "lex:GetBotAlias",
                "lex:GetBotAliases",
                "lex:GetBots",
                "lex:GetBotChannelAssociation",
                "lex:GetBotChannelAssociations",
                "lex:GetBotVersions",
                "lex:GetBuiltinIntent",
                "lex:GetBuiltinIntents",
                "lex:GetBuiltinSlotTypes",
                "lex:GetIntent",
                "lex:GetIntents",
                "lex:GetIntentVersions",
                "lex:GetSlotType",
                "lex:GetSlotTypes",
                "lex:GetSlotTypeVersions",
                "lex:GetUtterancesView",
                "lex:DescribeBot",
                "lex:DescribeBotAlias",
                "lex:DescribeBotChannel",
                "lex:DescribeBotLocale",
                "lex:DescribeBotRecommendation",
                "lex:DescribeBotReplica",
                "lex:DescribeBotVersion",
                "lex:DescribeExport",
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBots",
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotAliasReplicas",
                "lex:ListBotChannels",
                "lex:ListBotRecommendations",
                "lex:ListBotReplicas",
                "lex:ListBotVersions",
                "lex:ListBotVersionReplicas",
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListRecommendedIntents",
```

```
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource",
                "lex:SearchAssociatedTranscripts",
                "lex:ListCustomVocabularyItems"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gestionada por AWS: AmazonLexRunBotsOnly

Puede adjuntar la política de `AmazonLexRunBotsOnly` a sus 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 de los permisos

Esta política incluye los permisos siguientes:

- `lex`— Read-only acceso a todas las acciones en el entorno de ejecución de Amazon Lex V2 y Amazon Lex.

JSON

```
{
    "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 gestionada por AWS: AmazonLexFullAccess

Puede adjuntar la política de `AmazonLexFullAccess` a sus 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 de los permisos

Esta política incluye los permisos siguientes:

- `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 a los directores ver las CloudWatch métricas y alarmas de Amazon.
- `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.

JSON

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

```
        {
            "Sid": "AmazonLexFullAccessStatement1",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lambda:ListAliases",
                "lambda:ListVersionsByFunction",
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement2",
            "Effect": "Allow",
            "Action": [
                "bedrock:ListFoundationModels"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": "arn:aws:bedrock:*::foundation-model/*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement3",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement4",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
```

```
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement5",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement6",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement7",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement8",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "replication.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement9",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement10",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement11",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement12",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement13",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Política gestionada por AWS: AmazonLexReplicationPolicy

No puede asociar `AmazonLexReplicationPolicy` a sus entidades IAM. Esta política está asociada a un rol vinculado a servicios que permite a Amazon Lex V2 realizar acciones en su nombre. Para obtener más información, consulte Roles vinculados a servicios para Amazon Lex V2.

Esta política otorga permisos administrativos que permiten a Amazon Lex V2 replicar los recursos de AWS en todas las regiones en su nombre. Puede asociar esta política para que un rol pueda replicar fácilmente los recursos, incluidos los bots, las configuraciones regionales, las versiones, los alias, las intenciones, los tipos de ranuras, las ranuras y los vocabularios personalizados.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `lex`: permite a las entidades principales replicar recursos en otras regiones.
- `iam`: permite a las entidades principales transferir roles de IAM. Esto es necesario para que Amazon Lex V2 tenga permisos para replicar recursos en otras regiones.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
 {
  "Sid": "ReplicationPolicyStatement1",
  "Effect": "Allow",
  "Action": [
   "lex:BuildBotLocale",
   "lex:ListBotLocales",
   "lex:CreateBotAlias",
   "lex:UpdateBotAlias",
   "lex:DeleteBotAlias",
   "lex:DescribeBotAlias",
   "lex:CreateBotVersion",
   "lex:DeleteBotVersion",
   "lex:DescribeBotVersion",
   "lex:CreateExport",
   "lex:DescribeBot",
   "lex:UpdateExport",
   "lex:DescribeExport",
   "lex:DescribeBotLocale",
   "lex:DescribeIntent",
   "lex:ListIntents",
   "lex:DescribeSlotType",
   "lex:ListSlotTypes",
   "lex:DescribeSlot",
   "lex:ListSlots",
   "lex:DescribeCustomVocabulary",
   "lex:StartImport",
```

```
    "lex:DescribeImport",
    "lex:CreateBot",
    "lex:UpdateBot",
    "lex:DeleteBot",
    "lex:CreateBotLocale",
    "lex:UpdateBotLocale",
    "lex:DeleteBotLocale",
    "lex:CreateIntent",
    "lex:UpdateIntent",
    "lex:DeleteIntent",
    "lex:CreateSlotType",
    "lex:UpdateSlotType",
    "lex:DeleteSlotType",
    "lex:CreateSlot",
    "lex:UpdateSlot",
    "lex:DeleteSlot",
    "lex:CreateCustomVocabulary",
    "lex:UpdateCustomVocabulary",
    "lex:DeleteCustomVocabulary",
    "lex:DeleteBotChannel",
    "lex:ListTagsForResource",
    "lex:TagResource",
    "lex:UntagResource",
    "lex:CreateResourcePolicy",
    "lex:DeleteResourcePolicy",
    "lex:DescribeResourcePolicy",
    "lex:UpdateResourcePolicy"
   ],
   "Resource": [
    "arn:aws:lex:*:*:bot/*",
    "arn:aws:lex:*:*:bot-alias/*"
   ]
  },
  {
   "Sid": "ReplicationPolicyStatement2",
   "Effect": "Allow",
   "Action": [
    "lex:CreateUploadUrl",
    "lex:ListBots"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ReplicationPolicyStatement3",
```

```
  "Effect": "Allow",
  "Action": [
   "iam:PassRole"
  ],
  "Resource": "*",
  "Condition": {
   "StringEquals": {
    "iam:PassedToService": "lexv2.amazonaws.com"
   }
  }
 }
    ]
}
```

Política gestionada por AWS: AmazonLexV2BedrockAgentPolicy

Política de Amazon Lex V2 para agentes de Amazon Bedrock

Respuesta

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2TrustPolicy",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Política gestionada por AWS: AmazonLexV2BedrockKnowledgeBasePolicy

Política de Amazon Lex V2 para las bases de conocimiento de Amazon Bedrock

Respuesta

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2TrustPolicy",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
     "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Política gestionada por AWS: AmazonLexV2BedrockAgentPolicyInternal

Política interna de Amazon Lex V2 para los agentes de Amazon Bedrock

Respuesta

JSON

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

```
      "Effect": "Allow",
      "Sid": "LexV2InternalTrustPolicy",
      "Principal": {
        "Service": "lexv2.aws.internal"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Política gestionada por AWS: AmazonLexV2BedrockKnowledgeBasePolicyInternal

Política interna de Amazon Lex V2 para las bases de conocimiento de Amazon Bedrock

Respuesta

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "LexV2InternalTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.aws.internal"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Amazon Lex V2 actualiza las políticas gestionadas por AWS

Consulte los detalles sobre las actualizaciones de las políticas gestionadas por AWS para Amazon Lex V2 desde que este servicio comenzó a realizar el 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 V2 de Amazon Lex V2.

Cambio	Descripción	Fecha
AmazonLexReplicationPolicy: política actualizada	Amazon Lex V2 actualizó la política para permitir la replicación de etiquetas y ResourceBasedPolicy.	24 de junio de 2025
AmazonLexV2BedrockKnowledgeBasePolicyInternal: política nueva	Amazon Lex V2 agregó una nueva política para permitir la replicación de los recursos de la base de conocimiento de Amazon Bedrock.	30 de agosto de 2024
AmazonLexV2BedrockAgentPolicyInternal: política nueva	Amazon Lex V2 agregó una nueva política para permitir la replicación de los recursos de los agentes de Amazon Bedrock.	30 de agosto de 2024
AmazonLexV2BedrockKnowledgeBasePolicy: política nueva	Amazon Lex V2 agregó una nueva política para permitir la replicación de los recursos de la base de conocimiento de Amazon Bedrock.	30 de agosto de 2024

Cambio	Descripción	Fecha
AmazonLexV2BedrockAgentPolicy: política nueva	Amazon Lex V2 agregó una nueva política para permitir la replicación de los recursos de los agentes de Amazon Bedrock.	30 de agosto de 2024
AmazonLexReadOnly: actualización de una política actual	Amazon Lex V2 agregó nuevos permisos para permitir acceso de solo lectura a réplicas de los recursos de bots.	10 de mayo de 2024
AmazonLexFullAccess: actualización de una política actual	Amazon Lex V2 agregó nuevos permisos para permitir la replicación de los recursos de bots a otras regiones.	16 de abril de 2024
AmazonLexFullAccess: actualización de una política actual	Amazon Lex V2 agregó nuevos permisos para permitir la replicación de los recursos de bots a otras regiones.	31 de enero de 2024
AmazonLexReplicationPolicy: política nueva	Amazon Lex V2 agregó una nueva política para permitir la replicación de los recursos de bots en otras regiones.	31 de enero de 2024
AmazonLexReadOnly: actualización de una política actual	Amazon Lex V2 agregó nuevos permisos para permitir acceso de solo lectura a la lista de elementos de vocabulario personalizados.	29 de noviembre de 2022

Cambio	Descripción	Fecha
AmazonLexFullAccess: actualización de una política actual	Amazon Lex V2 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 actual	Amazon Lex V2 agregó nuevos permisos para permitir el acceso de solo lectura a las operaciones del Diseñador de chatbots automatizados de Amazon Lex V2.	1 de diciembre de 2021
AmazonLexFullAccess: actualización de una política actual	Amazon Lex V2 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 actual	Amazon Lex V2 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
AmazonLexRunBotsOnly: actualización de una política actual	Amazon Lex V2 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

Cambio	Descripción	Fecha
Amazon Lex V2 ha comenzado a hacer un seguimiento de los cambios	Amazon Lex V2 comenzó a realizar un seguimiento de los cambios en sus políticas gestionadas por AWS.	18 de agosto de 2021

Roles vinculados a servicios para Amazon Lex V2

Amazon Lex V2 utiliza funciones vinculadas al servicio AWS Identity and Access Management (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a Amazon Lex V2. Service-linked Amazon Lex V2 predefine los roles 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 ECS porque ya no tendrá que agregar manualmente los permisos requeridos. Amazon Lex V2 define los permisos de sus roles vinculados a servicios y, a menos que esté definido de otra manera, solo Amazon Lex V2 puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda asociar a ninguna otra entidad de IAM.

Para obtener información sobre otros servicios que admiten funciones vinculadas a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que tengan la palabra Sí en la Service-Linked columna Rol. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

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 los permisos de Service-Linked rol en la Guía del usuario de IAM.

Solo puede eliminar un rol vinculado a servicios después de eliminar los recursos relacionados. De esta forma, se protegen los recursos de Amazon Lex V2, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Temas

- Crear un rol vinculado a un servicio para Amazon Lex V2
- Editar un rol vinculado a un servicio para Amazon Lex V2
- Eliminar un rol vinculado a un servicio para Amazon Lex V2

- Service-linked permisos de rol para Amazon Lex V2
- Regiones compatibles para los roles vinculados a servicios de Amazon Lex V2

Crear un rol vinculado a un servicio para Amazon Lex V2

No necesita crear manualmente un rol vinculado a un servicio, ya que Amazon Lex V2 crea el rol vinculado al servicio automáticamente cuando lleva a cabo la acción correspondiente (consulte Service-linked permisos de rol para Amazon Lex V2 para obtener más información) en la consola de administración de AWS, la CLI de AWS o la API de AWS.

Si elimina este rol vinculado al servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para crear un nuevo rol en su cuenta.

Editar un rol vinculado a un servicio para Amazon Lex V2

Amazon Lex V2 no le permite editar roles vinculados a servicios. Después de crear un rol vinculado a un servicio, no puede cambiarle el nombre, ya que varias entidades pueden hacer referencia a él. Sin embargo, puede editar la descripción de un rol con IAM. Para obtener más información, consulte Edición de un Service-Linked rol en la Guía del usuario de IAM.

Eliminar un rol vinculado a un servicio para Amazon Lex V2

Si ya no necesita usar una característica o servicio que requieran un rol vinculado a un servicio, le recomendamos que elimine dicho rol. Así no tendrá una entidad no utilizada que no se supervise ni mantenga de forma activa. Sin embargo, debe limpiar los recursos de su rol vinculado al servicio antes de eliminarlo manualmente.

> **Note**
>
> Si el servicio de Amazon Lex V2 utiliza el rol al intentar eliminar los recursos, se podría producir un error en la eliminación. En tal caso, espere unos minutos e intente de nuevo la operación.

Si quiere consultar los pasos para eliminar recursos de roles específicas vinculadas a servicios en Amazon Lex V2, consulte la sección específica del rol en Service-linked permisos de rol para Amazon Lex V2.

Para eliminar manualmente un rol vinculado a un servicio con IAM

Tras eliminar los recursos relacionados con un rol vinculado a un servicio, utilice la consola de IAM, la CLI de AWS o la API de AWS para eliminar el rol. Para obtener más información, consulte Eliminar un Service-Linked rol en la Guía del usuario de IAM.

Service-linked permisos de rol para Amazon Lex V2

Amazon Lex V2 utiliza roles vinculados a servicios con los siguientes prefijos.

Temas

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

El rol AWSServiceRoleForLexV2Bots _ otorga permisos para conectar el bot a otros servicios necesarios. Este rol incluye una política de confianza que permite que el servicio lexv2.amazonaws.com la asuma e incluye permisos para llevar a cabo las siguientes acciones.

- Utilice Amazon Polly para sintetizar la voz en todos los recursos de Amazon Lex V2 compatibles con la acción.
- Si un bot está configurado para utilizar el análisis de opiniones de Amazon Comprehend, detecte las opiniones en todos los recursos de Amazon Lex V2 compatibles con la acción.
- Si un bot está configurado para almacenar registros de audio en un bucket de S3, coloque el objeto en un bucket específico.
- Si un bot está configurado para almacenar registros de audio y texto, cree un flujo de registro y coloque los registros en un grupo de registro específico.
- Si un bot está configurado para usar una AWS KMS clave para cifrar datos, genera una clave de datos específica.
- Si un bot está configurado para usar la intención `KendraSearchIntent`, solicite el acceso a un índice de Amazon Kendra específico.

Para crear el rol

Amazon Lex V2 crea un nuevo rol AWSServiceRoleForLexV2Bots _ con un sufijo aleatorio en su cuenta cada vez que crea un bot. Amazon Lex V2 modifica el rol al agregar capacidades adicionales

a un bot. Por ejemplo, si agrega el análisis de opiniones de Amazon Comprehend a un bot, Amazon Lex V2 agrega el permiso para la acción `lex:DetectSentiment` al rol de servicio.

Para eliminar el rol

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. En el panel de navegación izquierdo, seleccione Bots y elija el bot con el rol vinculado a servicios que desee eliminar.
3. Seleccione cualquier versión del bot.
4. El rol de tiempo de ejecución de los permisos de IAM se encuentra en los Detalles de la versión.
5. Regrese a la página Bots y elija el botón de opción situado junto al bot para eliminarlo.
6. Seleccione Acciones y, a continuación, elija Eliminar.
7. Siga los pasos que encontrará en Deleting a service-linked role para eliminar el rol de IAM.

AWSServiceRoleForLexV2Channels_

La función AWSServiceRoleForLexV2Channels _ permite enumerar los bots de una cuenta y llamar a las API de conversación de un bot. Este rol incluye una política de confianza para permitir que el servicio channels.lexv2.amazonaws.com asuma el rol. Si un bot está configurado para usar un canal para comunicarse con un servicio de mensajería, la política de permisos AWSServiceRoleForLexV2Channels _ role permite a Amazon Lex V2 realizar las siguientes acciones.

- Enumerar los permisos de todos los bots de una cuenta.
- Reconocer el texto, obtener la sesión y asignar los permisos de sesión a un alias de bot específico.

Para crear el rol

Cuando crea una integración de canales para implementar un bot en una plataforma de mensajería, Amazon Lex V2 crea un nuevo rol vinculado a un servicio en su cuenta para cada canal con un sufijo aleatorio.

Para eliminar el rol

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.

2. En el panel de navegación izquierdo, seleccione Bots.
3. Seleccione un bot.
4. En el panel de navegación izquierdo, elija Integraciones de canales en Implementaciones.
5. Seleccione un canal con el rol vinculado a un servicio que desee eliminar.
6. El rol de tiempo de ejecución de permisos de IAM se encuentra en la Configuración general.
7. Seleccione Eliminar y, a continuación, vuelva a seleccionar Eliminar para eliminar el canal.
8. Siga los pasos que encontrará en Deleting a service-linked role para eliminar el rol de IAM.

AWSServiceRoleForLexV2Replication

El AWSServiceRoleForLexV2Replication rol permite replicar bots en una segunda región. Este rol incluye una política de confianza para permitir que el servicio replication.lexv2.amazonaws.com asuma el rol y también incluye la política administrada por AmazonLexReplicationPolicyAWS, que permite permisos para las siguientes acciones.

- Transfiera los roles de IAM del bot al robot de réplica para volver a duplicar los permisos correspondientes para el robot de réplica.
- Cree y administre bots y recursos de bots (versiones, alias, intenciones, ranuras, vocabularios personalizados, etc.) en otras regiones.

Para crear el rol

Cuando habilita la resiliencia global para un bot, Amazon Lex V2 crea el rol AWSServiceRoleForLexV2Replication vinculado al servicio en su cuenta. Asegúrese de tener los permisos correctos para conceder al servicio Amazon Lex V2 los permisos necesarios para crear el rol vinculado al servicio.

Para eliminar los recursos de Amazon Lex V2 utilizados por AWSServiceRoleForLexV2Replication para que pueda eliminar el rol

1. Inicie sesión en la consola de administración de AWS y abra la consola de Amazon Lex en https://console.aws.amazon.com/lex/.
2. Elija un bot para el que esté habilitada la resiliencia global.
3. Seleccione Resiliencia global en Implementación.
4. Seleccione Deshabilitar Resiliencia global.

5. Repita el proceso para todos los bots que tengan habilitada la resiliencia global.
6. Siga los pasos que encontrará en Deleting a service-linked role para eliminar el rol de IAM.

Regiones compatibles para los roles vinculados a servicios de Amazon Lex V2

Amazon Lex V2 admite el uso de roles vinculados a servicios en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte Regiones y puntos de enlace de AWS.

Uso de funciones de servicio para Amazon Lex V2

Amazon Lex V2 utiliza un rol de servicio de IAM para acceder a otros servicios de AWS en su nombre durante el tiempo de ejecución. El rol de servicio proporciona los permisos necesarios para crear y ejecutar el bot. Un rol de servicio es un rol de IAM estándar gestionado por el cliente en tu cuenta. Respeta las políticas de control de servicios (SCP) que configura en AWS Organizations y le proporciona un control total sobre los permisos que utiliza Amazon Lex V2.

AmazonLexServiceRole-

La función AmazonLexServiceRole - otorga permisos para conectar el bot a otros servicios necesarios. Este rol incluye una política de confianza que permite que el servicio lexv2.amazonaws.com la asuma e incluye permisos para llevar a cabo las siguientes acciones.

- Si un bot está configurado para utilizar la conversión de texto a voz, utilice Amazon Polly para sintetizar la voz en todos los recursos de Amazon Lex V2 compatibles con la acción.
- Si un bot está configurado para utilizar el análisis de opiniones de Amazon Comprehend, detecte las opiniones en todos los recursos de Amazon Lex V2 compatibles con la acción.
- Si un bot está configurado para almacenar registros de audio en un bucket de S3, coloque el objeto en un bucket específico.
- Si un bot está configurado para almacenar registros de audio y texto, cree un flujo de registro y coloque los registros en un grupo de registro específico.
- Si un bot está configurado para usar una AWS KMS clave para cifrar datos, genere una clave de datos específica.
- Si un bot está configurado para utilizar las funciones de IA generativa de Amazon Bedrock, invoque los modelos, los perfiles de inferencia y las bases de conocimiento de Amazon Bedrock especificados.

Creación del rol

Amazon Lex V2 crea un nuevo AmazonLexServiceRole rol con un sufijo aleatorio en su cuenta cuando crea un bot y decide crear un nuevo rol. Amazon Lex V2 modifica el rol al agregar capacidades adicionales a un bot. Por ejemplo, si añade el análisis de opiniones de Amazon Comprehend a un bot, Amazon Lex V2 añade el permiso para la `comprehend:DetectSentiment` acción al rol de servicio.

También puede elegir un rol de IAM existente que haya creado y configurado usted mismo, siempre que confíe en el principal de servicio de Amazon Lex V2 (`lexv2.amazonaws.com`).

Editar o eliminar el rol

Dado que un rol de servicio es un rol de IAM estándar administrado por el cliente, usted lo administra de la misma manera que cualquier otro rol de IAM. Utilice la consola de IAM, la CLI de AWS o la API de IAM. Puede revisar la política de confianza del rol y las políticas de permisos adjuntas en cualquier momento en la consola de IAM.

> **Important**
>
> Antes de eliminar un rol de servicio, asegúrate de que ningún bot lo siga utilizando. Al eliminar un rol del que depende un bot, este pierde los permisos que necesita para ejecutarse.

Regiones compatibles con las funciones de servicio de Amazon Lex V2

Amazon Lex V2 admite el uso de funciones de servicio en todas las regiones en las que el servicio esté disponible. Para obtener más información, consulte Regiones y puntos de enlace de AWS.

Migración a funciones de servicio para Amazon Lex V2

Amazon Lex V2 está pasando de las funciones AWS-managed vinculadas a servicios (SLR) a las funciones de servicio gestionadas por el cliente para los permisos de ejecución de los bots. En este tema se explica por qué Amazon Lex V2 está realizando este cambio, cómo migrar los bots existentes y el cronograma de la transición.

Por qué Amazon Lex V2 pasa a ocupar funciones de servicio

Un rol vinculado a un servicio es un rol de IAM que está vinculado directamente a Amazon Lex V2 y que es administrado por AWS. Service-linked los roles no respetan las políticas de control de servicios (SCP) que usted configura en AWS Organizations.

Un rol de servicio es un rol de IAM que usted posee y administra en su propia cuenta. Como un rol de servicio es un rol estándar administrado por el cliente, respeta las políticas de control de servicios (SCP) que usted configura en su organización. Esto le da un control total sobre los permisos que Amazon Lex V2 utiliza en su nombre. Pasar a funciones de servicio alinea Amazon Lex V2 con las mejores prácticas de IAM y mejora su postura de seguridad y conformidad.

Migre un bot existente a un rol de servicio

Puede migrar un bot existente que utilice una función vinculada a un servicio a una función de servicio gestionada por el cliente directamente desde la consola Amazon Lex V2.

> **Note**
>
> Amazon Lex V2 planea realizar la migración de un solo bot mediante el botón Migrar a un rol de servicio disponible antes del 1 de septiembre de 2026. Cuando el botón esté disponible, complete los siguientes pasos para migrar un bot.

1. Abra la consola Amazon Lex V2 en https://console.aws.amazon.com/lexv2/home.
2. En el panel de navegación izquierdo, selecciona Bots y, a continuación, elige el bot que deseas migrar.
3. En la página de la versión preliminar del bot, si el bot usa un rol vinculado a un servicio, aparece un banner que indica que el bot está usando un rol vinculado al servicio. Selecciona Migrar a rol de servicio.
4. Amazon Lex V2 crea una nueva función de servicio gestionada por el cliente, adjunta los permisos que requiere el bot y actualiza el bot para que utilice la nueva función.
5. Cuando se complete la migración, el banner del rol vinculado al servicio y el botón Migrar al rol de servicio ya no aparecerán para el bot, lo que confirma que el bot ahora usa un rol de servicio.

> **Note**
>
> El botón Migrar a un rol de servicio solo aparece en el caso de un bot que utilice un rol vinculado a un servicio. La migración no modifica ni elimina tu rol original vinculado al servicio. Si la migración no se completa, tu bot seguirá utilizando su función vinculada al servicio existente y seguirá funcionando a pleno rendimiento.

Cronograma de migración

La transición de las funciones vinculadas al servicio a las funciones de servicio sigue el cronograma que se describe en la siguiente lista.

- Nuevos bots: los nuevos bots creados en la consola ahora tienen un rol de servicio de forma predeterminada. Recomendamos crear nuevos bots con un rol de servicio.
- Migración de un solo bot: el botón Migrar a un rol de servicio con un solo clic aparece solo para los bots que usan un rol vinculado a un servicio. Con un solo clic, puedes migrar un bot a un rol de servicio. Amazon Lex V2 planea que esté disponible antes del 1 de septiembre de 2026.
- Migración de varios bots: un asistente enumera sus bots para que pueda seleccionar varios bots de rol vinculados a un servicio y migrarlos a roles de servicio a la vez. Amazon Lex V2 planea que esté disponible antes del 15 de septiembre de 2026.
- Fin del soporte para las funciones vinculadas al servicio: el soporte para las funciones vinculadas al servicio finaliza a finales de 2026.

Solución de problemas de identidad y acceso de Amazon Lex V2

Utilice la siguiente información para diagnosticar y solucionar los problemas habituales que pueden surgir cuando se trabaja con Amazon Lex V2 e IAM.

Temas

- No tengo autorización para realizar una acción en Amazon Lex V2
- No estoy autorizado a realizar tareas como: PassRole
- Soy administrador y deseo permitir que otros accedan a Amazon Lex V2
- Conceder acceso programático a un usuario
- Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de Amazon Lex V2

No tengo autorización para realizar una acción en Amazon Lex V2

Si la Consola de administración de AWS le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

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, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *`my-example-widget`* mediante la acción `lex:`*`GetWidget`*.

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 V2.

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 la función 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 V2. 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 de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Soy administrador y deseo permitir que otros accedan a Amazon Lex V2

Para permitir que otros accedan a Amazon Lex V2, debe conceder permiso a las personas o aplicaciones que necesitan acceso. Si usa AWS IAM Identity Center para administrar las personas y las aplicaciones, debe asignar conjuntos de permisos a los usuarios o grupos para definir su nivel de acceso. Los conjuntos de permisos crean políticas de IAM y las asignan a los roles de IAM asociados a la persona o aplicación de forma automática. Para obtener más información, consulte la sección Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .

Si no utiliza IAM Identity Center, debe crear entidades de IAM (usuarios o roles) para las personas o aplicaciones que necesitan acceso. A continuación, debe asociar una política a la entidad que les conceda los permisos correctos en Amazon Lex V2. Una vez concedidos los permisos, proporcione las credenciales al usuario o al desarrollador de la aplicación. Utilizarán esas credenciales para acceder a AWS. Para obtener más información sobre la creación de usuarios, grupos, políticas y permisos de IAM, consulte Identidades de IAM y Políticas y permisos en IAM en la Guía del usuario de IAM.

Conceder acceso programático a un usuario

Para obtener información sobre cómo obtener el identificador de clave de acceso y la clave de acceso secreta, consulte Cómo entender y obtener sus credenciales de AWS en la Referencia general de AWS.

Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de Amazon Lex V2

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se 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 obtener más información, consulte lo siguiente:

- Para saber si Amazon Lex V2 admite estas características, consulte Cómo funciona Amazon Lex V2 con IAM.
- Para obtener información sobre cómo proporcionar acceso a sus recursos a través de Cuentas de AWS los suyos, consulte Proporcionar acceso a un usuario de IAM en otro de su Cuenta de AWS propiedad 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 sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registrar y monitorear en Amazon Lex V2

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon Lex V2 y de las demás soluciones de AWS. AWS ofrece las siguientes herramientas de monitorización para vigilar a Amazon Lex V2, informar cuando algo no funciona y realizar acciones automáticas cuando proceda:

- Amazon CloudWatch supervisa los recursos de AWS y las aplicaciones que ejecuta en AWS en tiempo real. Puede recopilar métricas y realizar un seguimiento de las métricas, crear paneles personalizados y definir alarmas que le advierten o que toman medidas cuando una métrica determinada alcanza el umbral que se especifique. Por ejemplo, puede CloudWatch hacer un seguimiento del uso de la CPU u otras métricas de sus instancias de Amazon EC2 y lanzar automáticamente nuevas instancias cuando sea necesario. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.
- AWS CloudTrailcaptura las llamadas a la API y los eventos relacionados realizados por su cuenta de AWS o en su nombre y entrega los archivos de registro a un bucket de Amazon S3 que especifique. También pueden identificar qué usuarios y cuentas llamaron a AWS, la dirección IP de origen desde la que se realizaron las llamadas y el momento en que se efectuaron. Para obtener más información, consulte la Guía del usuario de AWS CloudTrail.

Validación de la conformidad en Amazon Lex V2

Third-party los auditores evalúan la seguridad y la conformidad de Amazon Lex V2 como parte de varios programas de conformidad de AWS. Amazon Lex V2 es un servicio compatible con HIPAA Es compatible con PCI, SOC e ISO.

Nuestra nueva experiencia de registro en AWS no está diseñada para cargas de trabajo reguladas. Si utiliza nuestra nueva experiencia de registro en AWS, pero quiere utilizar AWS para cargas de trabajo reguladas, puede suscribirse a AWS (avanzado) o activar funciones avanzadas para su entorno de AWS.

Para saber si alguno Servicio de AWS está dentro del ámbito de aplicación de programas de conformidad específicos, consulte Servicios de AWS Alcance por programa de conformidad Servicios de AWS y elija el programa de conformidad que le interese. Para obtener información general, consulte Programas de conformidad de AWS Programas de .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. Para obtener más información sobre su responsabilidad de conformidad al usarlo Servicios de AWS, consulte la documentación de seguridad de AWS.

Resiliencia en Amazon Lex V2

La infraestructura global de AWS está conformada por regiones y zonas de disponibilidad de AWS. Las regiones de AWS proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre las zonas de disponibilidad y las regiones de AWS, consulte Infraestructura global de AWS.

Además de la infraestructura global de AWS, Amazon Lex V2 ofrece varias funciones para ayudarlo a satisfacer sus necesidades de respaldo y resiliencia de datos.

> **Note**
>
> Para obtener más información sobre la resiliencia global de Amazon Lex V2, que le permite crear un bot replicado en una segunda región en pares predeterminados, consulte Resiliencia global.

Seguridad de la infraestructura en Amazon Lex V2

Como servicio gestionado, Amazon Lex V2 está protegido por los procedimientos de seguridad de red global de AWS que se describen en el documento técnico Amazon Web Services: Overview of Security Processes.

Utiliza las llamadas a la API publicadas por AWS para acceder a Amazon Lex V2 a través de la red. Los clientes deben ser compatibles con la seguridad de la capa de transporte (TLS) 1.0 o una versión posterior. Recomendamos TLS 1.2 o una versión posterior. Los clientes también deben admitir conjuntos de cifrado con perfecto secreto directo (PFS), como Ephemeral (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman 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 IAM. También puedes utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Amazon Lex V2 y puntos de enlace de VPC de interfaz (AWS PrivateLink)

Puede establecer una conexión privada entre la VPC y Amazon Lex V2 mediante la creación de un punto de conexión de VPC de interfaz. Los puntos de enlace de la interfaz funcionan con una tecnología que le permite acceder de forma privada a las API de Amazon Lex V2 sin una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión AWS Direct Connect. AWS PrivateLink Las instancias de la VPC no necesitan direcciones IP públicas para comunicarse con las API de Amazon Lex V2. El tráfico entre la VPC y Amazon Lex V2 no sale de la red de Amazon.

Cada punto de conexión de la interfaz está representado por una o más interfaces de red elásticas en las subredes.

Para obtener más información, consulte Interface VPC Endpoints (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Consideraciones sobre los puntos de conexión de Amazon Lex V2

Antes de configurar un punto de conexión de VPC tipo interfaz para Amazon Lex V2, asegúrese de revisar las Propiedades y limitaciones de puntos de conexión de interfaz en la Guía del usuario de Amazon VPC.

Amazon Lex V2 admite realizar llamadas a todas sus acciones de la API desde su VPC.

Crear un punto de conexión de VPC de interfaz para Amazon Lex V2

Puede crear un punto de enlace de VPC para el servicio Amazon Lex V2 mediante la consola de Amazon VPC o la (AWS Command Line Interface AWS CLI). Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto de conexión de VPC para Amazon Lex V2 mediante el siguiente nombre de servicio:

- com.amazonaws. *region*.models-v2-lex
- com.amazonaws. *region*.runtime-v2-lex

Si habilita DNS privado para el punto de conexión, puede realizar solicitudes a la API para Amazon Lex V2 usando su nombre de DNS predeterminado para la región, por ejemplo .

Para más información, consulte Acceso a un servicio a través de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Crear una política de puntos de conexión de VPC para Amazon Lex V2

Puede asociar una política de puntos de conexión con su punto de conexión de VPC que controla el acceso a Amazon Lex V2. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para obtener más información, consulte Control del acceso a los servicios con puntos de conexión de VPC en la Guía del usuario de Amazon VPC.

Ejemplo: política de punto de conexión de VPC para acciones de Amazon Lex V2

A continuación, se muestra un ejemplo de una política de punto de conexión para Amazon Lex V2. Cuando se asocia a un punto de conexión, esta política concede acceso a las acciones de Amazon Lex V2 enumeradas para todos las entidades principales de todos los recursos.

```
{
   "Statement":[
```

```
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lex:RecognizeText",
            "lex:RecognizeUtterance",
            "lex:StartConversation",
            "lex:DeleteSession",
            "lex:GetSession",
            "lex:DeleteSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Ejemplos de código para Amazon Lex V2 utilizando AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar Amazon Lex V2 con un kit de desarrollo de AWS software (SDK).

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon Lex V2 con un AWS SDK. En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código

- Escenarios para el uso de Amazon Lex V2 AWS SDK
  - Cree un chatbot Amazon Lex V2 para atraer a los visitantes de su sitio web

Escenarios para el uso de Amazon Lex V2 AWS SDK

Los siguientes ejemplos de código muestran cómo implementar escenarios comunes en Amazon Lex V2 con AWS SDK. Estos escenarios muestran cómo realizar tareas específicas llamando a varias funciones de Amazon Lex V2 o combinándolas con otras Servicios de AWS. En cada escenario se incluye un enlace al código fuente completo, con instrucciones de configuración y ejecución del código.

Los escenarios requieren un nivel intermedio de experiencia para entender las acciones de servicio en su contexto.

Ejemplos

- Cree un chatbot Amazon Lex V2 para atraer a los visitantes de su sitio web

Cree un chatbot Amazon Lex V2 para atraer a los visitantes de su sitio web

En los siguientes ejemplos de código se muestra cómo crear un chatbot para atraer a visitantes a su sitio web.

Java

SDK para Java 2.x

Muestra cómo usar la API Amazon Lex V2 para crear un Chatbot dentro de una aplicación web para atraer a los visitantes de su sitio web.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon Lex V2
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Muestra cómo usar la API Amazon Lex V2 para crear un Chatbot dentro de una aplicación web para atraer a los visitantes de su sitio web.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo sobre cómo crear un chatbot Amazon Lex V2 en la guía para AWS SDK para JavaScript desarrolladores.

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon Lex V2
- Amazon Translate

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Amazon Lex V2 con un AWS SDK. En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Directrices y prácticas recomendadas

Consulte las siguientes pautas y prácticas recomendadas para optimizar el comportamiento y las interacciones de su bot con los clientes.

Firma de solicitudes

Todas las solicitudes en tiempo de ejecución y de desarrollo de modelo de Amazon Lex V2 incluidas en la Referencia de API utilizan Signature Version 4 para autenticar las solicitudes. Para obtener más información sobre las solicitudes de autenticación, consulte Proceso de firma de Signature Version 4 en la Referencia general de AWS.

Protección de la información confidencial

Las operaciones de la API en tiempo de ejecución RecognizeText y RecognizeUtterance utilizan un identificador de sesión como 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 usa principalmente para identificar de forma única una conversación con un bot.

Capturar valores de slot a partir de los enunciados de los usuarios

Amazon Lex V2 utiliza los valores de enumeración facilitados en una definición de tipo de slot 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 con el tipo personalizado `ZodiacSign` que tiene 12 valores de enumeración: de `Aries` a `Pisces`. Ahora supongamos que el usuario dice «Dime la predicción para la Tierra»:

- Amazon Lex V2 deduce que «tierra» es un valor SignoDelZodiaco cuando realiza una de las siguientes acciones:
  - Configurar el campo `valueSelectionStrategy` en `ORIGINAL_VALUE` mediante la operación CreateSlotType
  - Seleccionar Expandir valores en la consola

- Amazon Lex V2 no reconoce el valor «tierra» si limita el reconocimiento a los valores que ha definido para el tipo de slot mediante una de las siguientes acciones:
  - Configurar el campo `valueSelectionStrategy` en `TOP_RESOLUTION` mediante la operación `CreateSlotType`
  - Seleccionar Restringir a los valores y sinónimos de los slots en la consola

Al definir sinónimos para los valores de los slots, se reconoce que son iguales a los valores de los slots. Sin embargo, se devuelve el valor del slot en lugar del sinónimo.

Dado que Amazon Lex V2 transfiere este valor a la aplicación de su cliente o a la función de Lambda, debe comprobar que los valores de los slots son valores válidos antes de utilizarlos en su actividad de cumplimiento.

Cuando Amazon Lex V2 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 slot. 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`.

Acrónimos en los valores de slots

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)

Slots integrados para fecha y hora

Los tipos de slot integrados AMAZON.Date y AMAZON.Time capturan fechas y horas relativas y absolutas. Las fechas y horas relativas se resuelven en el momento y la fecha en que Amazon Lex V2 recibe la solicitud y en la región en la que la procesa.

Para el tipo de slot integrado `AMAZON.Time`, si el usuario no especifica si una hora es antes o después del mediodía, esa hora es ambigua. En ese caso, Amazon Lex V2 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 p. m.» o «a las 6 de la tarde».

Evitar la ambigüedad en los datos de entrenamiento de su bot

Si proporciona datos de aprendizaje confusos en su bot, se reducirá la capacidad de Amazon Lex V2 para comprender la entrada del usuario. Supongamos que su bot tiene dos intenciones (`OrderPizza` y `OrderDrink`) e incluye «Quiero pedir» como ejemplo de enunciado. Cuando compila su bot, Amazon Lex V2 no puede asignar este enunciado a una intención específica. Como resultado, cuando un usuario dice este enunciado en tiempo de ejecución, Amazon Lex v2 no puede elegir una intención con un alto nivel de confianza.

Si tiene dos intenciones con el mismo enunciado de ejemplo, utilice los contextos de entrada para ayudar a Amazon Lex V2 a distinguir entre las dos intenciones en tiempo de ejecución. Para obtener más información, consulte Configurar contexto de intención.

Usar el alias TSTALIASID

- El alias TSTALIASID de su bot se refiere a la versión preliminar y solo debería utilizarse para la realización de pruebas manuales. Amazon Lex limita el número de solicitudes de tiempo de ejecución que puede realizar al alias TSTALIASID del bot.
- Al actualizar la versión preliminar del bot, Amazon Lex finaliza cualquier conversación en curso de cualquier aplicación cliente que utilice el alias TSTALIASID del bot. Por lo general, no debería usar el alias TSTALIASID de un bot en producción porque la versión preliminar se puede actualizar. En su lugar, debe publicar una versión y un alias y usarlas.
- Al actualizar un alias, Amazon Lex tarda unos minutos en aplicar los cambios. Si modifica la versión preliminar del bot, el cambio se aplica inmediatamente al alias TSTALIASID.

Cuotas

Las cuotas de servicio, también denominadas límites, son la cantidad máxima de recursos de servicio permitida para su cuenta de AWS. Para obtener más información, consulte las cuotas de servicio de AWS en la referencia general de AWS.

Algunas cuotas de servicio se pueden ajustar o aumentar. Consulte la columna Ajustable de las siguientes tablas para ver si se puede ajustar una cuota y consulte la Self-servicecolumna para comprobar si puede solicitar un ajuste de cuota a través de la consola de cuotas de servicio. Póngase en contacto con nosotros Soporte para aumentar una cuota que sea ajustable, pero no mediante el autoservicio. 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.

> **Note**
>
> Los límites de caracteres se calculan como el número de Unidades de código Unicode. En la mayoría de los casos, un carácter Unicode equivale a una unidad de código Unicode. Algunos caracteres especiales pueden tener más de una unidad y los recuentos pueden variar según las distintas codificaciones. Para obtener más información sobre cómo calcular la longitud de una cadena, consulte esta documentación.

Build-time cuotas

Al crear un bot, se aplican las siguientes cuotas máximas.

Description (Descripción)	Predeterminado	Ajustable	Self-service
Bots por cuenta de AWS	100	Sí	Sí
Asociaciones de canales de bots por cuenta de AWS	5 000	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Compilaciones de configuración regional paralela por cuenta de AWS	5	Sí	No
Bots por red de bots	5	No	N/A
Bots por red de bots	25	No	N/A
Versiones por bot	100	No	N/A
Intenciones por configuración regional en cada bot	• 1000 en en-AU, en-GB y en-US • 250 en todas las demás configuraciones regionales	Sí	No
Slots por configuración regional en cada bot	• 4000 en en-AU, en-GB y en-US • 2000 en todas las demás configuraciones regionales	No	N/A
Tipos de slots personalizados para cada configuración regional del bot	• 250 en en-AU, en-GB y en-US • 100 en todas las demás configuraciones regionales	No	N/A
Valores de tipos de slot personalizados y sinónimos por configuración regional en cada bot	50 000	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Número total de caracteres en los ejemplos de enunciados por configuración regional de cada bot	• 2 000 000 en en-AU, en-GB y en-US • 200 000 en todas las demás configuraciones regionales	No	N/A
Asociaciones de canales por alias de bot	10	No	N/A
Slots por intención	100	No	N/A
Ejemplos de enunciados por intención	1500	Sí	Sí
Caracteres por ejemplo de enunciado	500	No	N/A
Longitud de la respuesta del texto	4.000	No	N/A
Ejemplos de enunciados por slot	10	Sí	Sí
Caracteres por ejemplo de enunciado de slot	500	No	N/A
Indicaciones por slot	30	No	N/A
Valores y sinónimos por tipo de slot personalizado	10 000	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Caracteres por valor de tipo de slot personalizado	500	No	N/A
Caracteres del nombre de una asociación de canales	100	No	N/A
Número de trabajos de análisis simultáneos del Diseñador de chatbots automatizados en todos los bots de su cuenta por región	10	No	N/A
Tamaño del archivo XML de tipo de slot gramatical personalizado	100 KB	No	N/A

Cuotas de tiempo de ejecución

Las siguientes cuotas máximas se aplican en tiempo de ejecución.

Description (Descripción)	Predeterminado	Ajustable	Self-service
Introduzca el tamaño del texto para y [Recognize](#) [TextRecognize](#) [Utterance](#)	1024 caracteres	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Longitud de entrada de voz para operación `RecognizeUtterance`	55 segundos	Sí	No
Tamaño de encabezados de `RecognizeUtterance`	16 KB	No	N/A
Tamaño de encabezados combinados de solicitud y sesión para `RecognizeUtterance`	12 KB	No	N/A
Número máximo de conversaciones simultáneas en modo texto para `RecognizeText` `RecognizeUtterance`, o para `StartConversation` TestBotAlias	2	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Número máximo de conversaciones simultáneas en modo texto para `RecognizeText`, `RecognizeUtterance` o `StartConversation` para otros alias	50	Sí	No
Número máximo de conversaciones simultáneas en modo voz para `RecognizeUtterance` TestBotAlias	2	No	N/A
Número máximo de conversaciones simultáneas en modo voz para `RecognizeUtterance` para otros alias	125	Sí	No
Número máximo de conversaciones simultáneas en modo de voz para `StartConversation` TestBotAlias	2	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Número máximo de conversaciones simultáneas en modo voz para `StartConversation` para otros alias.	400	Sí	No
Número máximo de operaciones de administración de sesiones simultáneas (`PutSession` , `GetSession` , o`DeleteSession`) al utilizar el TestBotAlias	2	No	N/A
Número máximo de operaciones de administración de sesiones simultáneas (`PutSession` , `GetSession` o `DeleteSession`) al usar otros alias	50	Sí	No
El tamaño máximo de entrada a una función de Lambda	12 KB	No	N/A
El tamaño máximo de salida de una función de Lambda	50 KB	No	N/A

Description (Descripción)	Predeterminado	Ajustable	Self-service
Tamaño máximo de los atributos de sesión en la salida de la función de Lambda (después de la codificación en base 64)	12 KB	No	N/A
El tamaño máximo de salida de una función de Lambda	30 segundos	Sí	No
Duración máxima de una sola conversación	15 minutos	No	No
Duración máxima del audio	55 segundos	No	No

Guía de migración de Amazon Lex de V1 a V2

La consola Amazon Lex V2 y APIs facilitan la creación y la administración de bots. Utilice esta guía para obtener información sobre las mejoras en la API de Amazon Lex V2 a medida que migra los bots.

Para migrar un bot, utilice la consola o la API de Amazon Lex. Para obtener más información, consulte Migrar un bot en la Guía para desarrolladores de Amazon Lex.

Descripción de Amazon Lex V2

Se pueden agregar varios idiomas a un bot para que pueda administrarlos como un único recurso. Una arquitectura de información simplificada le permite administrar de manera eficiente las versiones de sus bots. Funciones como el «flujo de conversación», el almacenamiento parcial de la configuración del bot y la carga masiva de enunciados le ofrecen más flexibilidad.

Varios idiomas en un bot

Puede añadir varios idiomas con la API de Amazon Lex V2. Puede añadir, modificar y compilar cada idioma de forma independiente. Los recursos, como los tipos de slots, se basan en el nivel de idioma. Puede cambiar rápidamente de un idioma a otro para comparar y afinar las conversaciones. Puede usar un panel de control de la consola para revisar los enunciados de todos los idiomas y agilizar el análisis y las iteraciones. El operador de un bot puede gestionar los permisos y las operaciones de registro para todos los idiomas con una sola configuración de bot. Debe proporcionar un idioma como parámetro de tiempo de ejecución para conversar con un bot de Amazon Lex V2. Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.

Arquitectura de la información simplificada

La API Amazon Lex V2 sigue una arquitectura de información (IA) simplificada con tipos de intención y franjas horarias que dependen de un idioma. La versión se realiza a nivel de bot para que los recursos, como las intenciones y los tipos de slots, no se versionen de forma individual. De forma predeterminada, se crea un bot con una versión Preliminar que es mutable y se usa para probar los cambios. Puede crear instantáneas numeradas a partir de la versión preliminar. Puede elegir los idiomas que desee incluir en una versión. Todos los recursos del bot (idiomas, intenciones, tipos de slot) se archivan como parte de la creación de una versión del bot. Para obtener más información, consulte Versiones.

Mejorar la productividad de los creadores

Dispone de herramientas y funciones adicionales de productividad para los creadores que le proporcionan más flexibilidad y control del proceso de diseño de sus bots.

Guardar parte de la configuración

La API de Amazon Lex V2 le permite guardar cambios parciales durante el desarrollo. Por ejemplo, puede guardar un slot que haga referencia a un tipo de slot eliminado. Esta flexibilidad le permite guardar su trabajo y volver a él más adelante. Puede resolver estos cambios antes de compilar el bot. En Amazon Lex V2, el guardado parcial se puede aplicar a slots, versiones y alias.

Cambio de nombre de los recursos

Con Amazon Lex V2, puede cambiar el nombre de un recurso una vez creado. Use un nombre de recurso para asociar metadatos fáciles de usar a cada recurso. La API de Amazon Lex V2 asigna a cada recurso un identificador de recurso único de 10 caracteres. Todos los recursos tienen un nombre de recurso. Puede etiquetar los siguientes recursos:

- bot
- Intención
- Tipo de ranura
- Ranura
- Alias

Puede usar el recurso IDs para leer y modificar sus recursos. Si utiliza la API Amazon Lex V2 AWS Command Line Interface o la API Amazon Lex V2 para trabajar con Amazon Lex V2, necesitará recursos IDs para algunos comandos.

Administración simplificada de las funciones de Lambda

En la API de Amazon Lex V2, se define una función de Lambda por idioma en lugar de una función para cada intención. La función de Lambda está configurada en el alias del idioma y se utiliza tanto para el cuadro de diálogo como para el enlace de códigos de cumplimiento. Aún puede activar o desactivar el diálogo y los enlaces de código de cumplimiento de forma independiente para cada intención. Para obtener más información, consulte Integración de una AWS Lambda función en su bot Amazon Lex V2.

Configuración granular

La API Amazon Lex V2 traslada el umbral de la puntuación de confianza en la clasificación de voz e intención del bot al ámbito del lenguaje. El indicador de análisis de opiniones pasa del ámbito del bot al ámbito del alias. El tiempo de espera de la sesión y la configuración de privacidad en el ámbito del bot y los registros de conversaciones en el ámbito de los alias permanecen inalterados.

Intención alternativa predeterminada

La API de Amazon Lex V2 añade una intención alternativa predeterminada al crear un idioma. Úsela para configurar la gestión de errores para su bot en lugar de utilizar indicaciones específicas sobre la gestión de errores.

Actualización optimizada de variables de sesión

Con la API Amazon Lex V2, puede actualizar el estado de la sesión directamente con RecognizeUtterancelas operaciones RecognizeTexty sin depender de la sesión APIs.

Creación de recursos de Amazon Lex V2 con AWS CloudFormation

Amazon Lex V2 está integrado con AWS CloudFormation un servicio que le ayuda a modelar y configurar sus recursos de AWS para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Debe crear una plantilla que describa todos los recursos de AWS que desee (como los chatbots de Amazon Lex V2) y CloudFormation aprovisione y configure esos recursos por usted.

Cuando la utilice CloudFormation, podrá reutilizar la plantilla para configurar los recursos de Amazon Lex V2 de forma coherente y repetida. Describa sus recursos una vez y, a continuación, aprovisione los mismos recursos una y otra vez en varias Cuentas de AWS regiones.

Amazon Lex V2 y CloudFormation plantillas

Para aprovisionar y configurar los recursos de Amazon Lex V2 y los servicios relacionados, debe entender las CloudFormation plantillas. Las plantillas son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas de CloudFormation . Si no está familiarizado con JSON o YAML, puede usar CloudFormation Designer como ayuda para comenzar con CloudFormation las plantillas. Para obtener más información, consulta ¿Qué es CloudFormation Designer? en la Guía AWS CloudFormation del usuario.

Amazon Lex V2 admite la creación de los siguientes tipos de recursos en CloudFormation:

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

Para obtener más información, incluidos ejemplos de plantillas JSON y YAML para estos recursos, consulte la referencia del tipo de recurso de Amazon Lex V2 en la Guía del usuario de AWS CloudFormation .

Obtenga más información sobre CloudFormation

Para obtener más información CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- AWS CloudFormation guía del usuario
- CloudFormation Referencia de la API
- AWS CloudFormation guía de usuario de la interfaz de línea de comandos

Historial de documentos de Amazon Lex V2

- Última actualización de la documentación: 27 de octubre de 2025

En la siguiente tabla se describen los cambios importantes en cada versión de Amazon Lex V2. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

Cambio	Descripción	Fecha
Actualización de funciones	Esperar y continuar ya está disponible en más configuraciones regionales: ca-ES, de-AT, de-DE, es-419, es-ES, es-US, fr-CA, fr-FR, it-IT, ja-JP, ko-KR, pt-BR, pt-PT, zh-CN y zh-HK. Para obtener más información, consulte Cómo permitir que el bot de Amazon Lex V2 espere a que el usuario proporcione más información durante una pausa.	27 de octubre de 2025
Actualización de funciones	Los tipos de ranura integrada AMAZON.Currency y AMAZON.Confirmation ahora admiten todas las configuraciones regionales. Para obtener más información, consulte Tipos de ranuras integrados.	18 de agosto de 2025
Actualización de la documentación	Se ha actualizado la documentación de la característica NLU asistida con ejemplos y aclaraciones	4 de agosto de 2025

	mejorados. Para obtener más información, consulte NLU asistida.	
Nueva característica	Puedes usar AMAZON. BedrockAgentIntent intención integrada de conectar su bot Amazon Lex V2 con los agentes y bases de conocimiento de Amazon Bedrock. Para obtener más información, consulte AMAZON. BedrockAgentIntent intención incorporada.	3 de julio de 2025
Actualización de la política gestionada por AWS	Amazon Lex V2 actualizó la política para permitir la replicación de etiquetas y, ResourceBasedPolicy cuando utilice la resiliencia global, consulte AmazonLexReplicationPolicy.	24 de junio de 2025
Nueva característica	Ahora puede utilizar la NLU asistida para mejorar la clasificación de intenciones y la resolución de las ranuras en Amazon Lex V2. Esta función utiliza modelos de lenguaje extensos (LLMs) para mejorar la precisión y, al mismo tiempo, mantenerse dentro de los espacios e intenciones configurados por el bot. Para obtener más información, consulte NLU asistida.	11 de junio de 2025

Actualización de la función (2025-06-03)	Puede usar 17 idiomas nuevos con los vocabularios personalizados en Amazon Lex V2. Para obtener más información, consulte Vocabularios personalizados.	3 de junio de 2025
Nueva característica (1/12/2024)	Puede utilizar la intención QinConnect integrada para conectar su bot Amazon Lex V2 con Amazon Connect. Para obtener más información, consulta la intención QinConnect integrada.	1 de diciembre de 2024
Nueva característica (1/10/2024)	Puede usar la ranura integrada AMAZON.Currency para configuraciones regionales de EE. UU. y España en Amazon Lex V2. Para obtener más información, consulte AMAZON.Currency.	1 de octubre de 2024
Actualización de la función (26 de julio de 2020)	La intención integrada QnA para IA generativa ahora es compatible con la Base de conocimiento y las barreras de protección de Bedrock en Amazon Lex V2. Para obtener más información, consulta Amazon.QN. AIntent	26 de julio de 2024

Actualización de la característica (10/7/2024)	La ranura integrada QnA para IA generativa ahora es compatible con los modelos Anthropic Claude 3 Haiku y Anthropic Claude 3 Sonnet en Amazon Lex V2. Para obtener más información, consulte Optimize bot creation and performance with generative AI.	10 de julio de 2024
Actualización de la política gestionada por AWS (2024-05-10)	Amazon Lex V2 agregó nuevos permisos a la política AmazonLex ReadOnlyadministrada para permitir el acceso de lectura a los recursos de bots que se han replicado en otras regiones.	10 de mayo de 2024
Actualización de la política gestionada por AWS (15 de abril de 2020)	Amazon Lex V2 agregó nuevos permisos a la política AmazonLexFullAccess administrada para permitir la actualización de los recursos de bots replicados en otras regiones.	15 de abril de 2024
Ampliación de las regiones	Amazon Lex V2 ya está disponible en AWS GovCloud (EE. UU. Oeste) (us-gov-west-1).	22 de marzo de 2024

Actualización de la política gestionada por AWS (2024-03-07)	Amazon Lex V2 agregó nuevos permisos a la política AmazonLexReplicationPolicyadministrada para permitir la actualización de los recursos de bots replicados en otras regiones.	7 de marzo de 2024
Nueva función	Puede utilizar la función fn.LENGTH() para determinar la longitud de un valor de cadena en Amazon Lex V2. Para obtener más información, consulte Conditional branching - Functions.	4 de marzo de 2024
Actualización de la característica (28/2/2024)	La ranura QnA integrada para capacidades de IA generativa de Amazon Lex V2 ahora es GA. Para obtener más información, consulte Optimize bot creation and performance with generative AI.	28 de febrero de 2024
Actualización de la política gestionada por AWS (2024-02-08)	Amazon Lex V2 agregó nuevos permisos a la política AmazonLexFullAccesssadministrada para permitir la replicación de los recursos de bots en otras regiones.	8 de febrero de 2024

Nueva política administrada	Amazon Lex V2 agregó una política administrada que proporciona permisos para replicar recursos de bots en otras regiones. Para obtener más información, consulte AmazonLexReplicationPolicy.	8 de febrero de 2024
Nueva función (2024-02-08)	Puede utilizar la resiliencia global para replicar su bot en una segunda región de AWS en Amazon Lex V2. Para obtener más información, consulte Resiliencia global.	8 de febrero de 2024
Nueva característica (29/11/2023)	Ahora puede aprovechar las capacidades de la IA generativa de Amazon Lex V2. Para obtener más información, consulte Optimize bot creation and performance with generative AI.	29 de noviembre de 2023
Nueva característica (8/11/2023)	Amazon Lex V2 ahora puede utilizar el registro selectivo para capturar el and/or audio del texto a nivel de intención o de ranura. Para obtener más información, consulte Registro selectivo.	8 de noviembre de 2023

Nueva característica (17/8/2023)	Amazon Lex V2 ahora puede usar una ranura integrada para determinar las respuestas Yes/No/Maybe/Don no conocidas mediante`AMAZON.Confirmation` . Para obtener más información, consulte Tipos de slots integrados.	17 de agosto de 2023
Nueva característica (18/7/2023)	Puede ver las métricas de rendimiento de las intenciones y los slots, además de otras métricas de conversación, mediante el panel de análisis. Para obtener más información, consulte Análisis.	18 de julio de 2023
Nueva función (2023-06-06)	Puede mejorar la precisión y el éxito de su bot con el Test Workbench. Para obtener más información, consulte Test Workbench.	6 de junio de 2023
Nueva función (23-02-23)	Ahora puede crear bots a partir de una plantilla para algunos sectores empresariales populares. Para obtener más información, consulte Plantillas de ISM	23 de febrero de 2023
Nueva función (2023-02-09)	Ahora puede combinar varios bots en una red de bots para crear una experiencia de cliente integrada. Para obtener más información, consulte Red de bots.	9 de febrero de 2023

Nueva característica (6/12/2022)	Amazon Lex V2 ahora es compatible con los idiomas árabe del Golfo (Emiratos Árabes Unidos), cantonés (Hong Kong), finés (Finlandia), noruego (Noruega), polaco (Polonia) y sueco (Suecia). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	6 de diciembre de 2022
Actualizado a la política gestionada por AWS	Amazon Lex V2 agregó nuevos permisos a la política AmazonLexReadOnly administrada para permitir la visualización de elementos de vocabulario personalizados.	29 de noviembre de 2022
Nueva característica (7/11/2022)	Amazon Lex V2 puede mostrar una representación alternativa de una frase o palabra mediante la consola o APIs para personalizar la conversión de voz a texto. Para obtener más información, consulte Crear un vocabulario personalizado para el reconocimiento de voz.	7 de noviembre de 2022

Nueva característica (28/10/2022)	Amazon Lex V2 puede añadir un atributo de peso a un elemento del artículo que representará el grado en el que se potencie la frase durante el reconocimiento de voz. Para obtener más información, consulte Ponderaciones gramaticales.	28 de octubre de 2022
Nueva característica (19/10/2022) (19/10/2022) - 2	Amazon Lex V2 puede mostrar una representación alternativa de una frase o palabra en la conversión de voz a texto final. Para obtener más información, consulte Crear un vocabulario personalizado para el reconocimiento de voz.	19 de octubre de 2022
Nueva característica (19/10/2022) (19/10/2022)	Amazon Lex V2 se puede utilizar para capturar entradas de formato libre del usuario final compuestas por palabras o caracteres mediante `AMAZON.FreeFormInput` . Para obtener más información, consulte Tipos de slots integrados.	19 de octubre de 2022

Nueva característica (14/10/2022)	Amazon Lex V2 ahora es compatible con las configuraciones regionales hindi (India) y neerlandés (Países Bajos). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	14 de octubre de 2022
Nueva función (2022-09-22)	Se ha actualizado la forma en que Amazon Lex V2 gestiona las entradas de los usuarios. Ahora puede seleccionar si Amazon Lex V2 acepta entradas de texto, audio o DTMF en cualquier punto del flujo de la conversación. Para obtener información acerca de los Atributos de configuración, consulte .	22 de septiembre de 2022
Nueva característica (14/9/2022)	Se ha actualizado la forma en que Amazon Lex V2 gestiona los flujos de conversaciones. El generador visual de conversaciones es un generador de conversaciones de arrastrar y soltar para diseñar y visualizar fácilmente las rutas de conversación. Para obtener más información, consulte el Generador de conversaciones visual.	14 de septiembre de 2022

Nueva característica (9/9/2022)	Se ha actualizado la forma en que Amazon Lex V2 crea slots complejos. Ahora puede crear slots secundarios complejos dentro de los slots para gestionar las intenciones en un diseño de conversación complejo. Para obtener más información, consulte Crear slots compuestos.	9 de septiembre de 2022
Nueva característica (17/8/2022)	Se ha actualizado la forma en que Amazon Lex V2 gestiona el flujo de las rutas de conversación con los usuarios. Ahora puede crear rutas de conversación complejas solicitando el siguiente paso de la conversación. Para obtener más información, consulte Crear rutas de conversación.	17 de agosto de 2022
Nueva función (2022-07-05)	Se ha actualizado la forma en que Amazon Lex V2 gestiona el flujo de las conversaciones con los usuarios. Ahora puede crear conversaciones complejas solicitando las indicaciones. Para obtener más información, consulte Configuración de indicaciones.	5 de julio de 2022

Nueva función (2022-05-03)	Se ha actualizado la forma en que Amazon Lex V2 gestiona el flujo de las conversaciones con los usuarios. Ahora puede crear conversaciones complejas usando condiciones. Para obtener más información, consulte Comprender los nuevos flujos de conversación.	3 de mayo de 2022
Nueva función (2022-03-22)	Se agregaron ejemplos de gramáticas industriales para el tipo de slot gramatical integrado. Para obtener más información, consulte Gramáticas industriales.	22 de marzo de 2022
Nueva característica (18/3/2022)	Se agregó documentación sobre la integración de Amazon Lex V2 con el Amazon Chime SDK. Para obtener más información, consulte Amazon Chime SDK.	18 de marzo de 2022
Nueva característica (27/1/2022)	Amazon Lex V2 ahora proporciona puntuaciones de confianza para las transcripciones de voz. Utilice la puntuación para ayudar a determinar la respuesta correcta del usuario. Para obtener más información, consulte Uso de puntuaciones de confianza en la transcripción de voz.	27 de enero de 2022

Nueva característica (13/1/2022)	Ahora puede añadir sugerencias contextuales y dinámicas a los slots para mejorar la precisión de su bot. Para obtener más información, consulte Usar sugerencias para mejorar la precisión.	13 de enero de 2022
Nueva característica (12/1/2022)	Amazon Lex V2 añade compatibilidad con vocabularios personalizados para mejorar el reconocimiento de voz en las entradas de audio. Para obtener más información, consulte Crear un vocabulario personalizado para mejorar el reconocimiento de voz.	12 de enero de 2022
Nueva característica (7/1/2022)	Amazon Lex V2 ahora es compatibleAWS PrivateLink. Para obtener más información, consulte Puntos de enlace de VPC (AWS). PrivateLink	7 de enero de 2022
Nueva característica (3/1/2022)	Amazon Lex V2 ahora es compatible con la configuración regional catalana (España). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	3 de enero de 2022

Nueva característica (20/12/2021) (20/12/2021) - 2	AWS CloudFormationahora es compatible con Amazon Lex V2. Para obtener más información, consulte Recursos de AWS CloudFormation.	20 de diciembre de 2021
Nueva característica (20/12/2021) (20/12/2021)	Ahora puede crear tipos de slots con su propia gramática personalizada. Para más información, consulte Usar un tipo de slot con gramática personalizada.	20 de diciembre de 2021
Nueva característica (16/12/2021)	Amazon Lex V2 ahora es compatible con las configuraciones regionales en portugués (Brasil), portugués (Portugal) y mandarín (PRC). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	16 de diciembre de 2021
Nueva característica (1/12/2021)	Amazon Lex V2 ahora ofrece una vista previa del Diseñador de chatbots automatizados para ayudarle a empezar a crear un chatbot a partir de las transcripciones del centro de contacto. Para obtener más información, consulte Usar el Diseñador de chatbots automatizados (versión preliminar).	1 de diciembre de 2021

Nueva característica (19/11/2021) (19/11/2021) - 2	Ahora puede usar las voces neuronales de conversión de texto a voz (NTTS) de Amazon Polly para conversaciones de audio con sus usuarios. Para obtener más información, consulte Voces en Amazon Polly.	19 de noviembre de 2021
Nueva característica (19/11/2021) (19/11/2021)	Ahora puede usar spell-by-letter spell-by-word estilos para introducir valores de ranura que Amazon Lex V2 tiene dificultades para entender. Para obtener más información, consulte Usar estilos de deletreo para capturar los valores de los slots.	19 de noviembre de 2021
Nueva característica (9/11/2021)	Amazon Lex V2 ahora es compatible con la configuración regional en inglés (Sudáfrica). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	9 de noviembre de 2021
Nueva característica (5/11/2021)	Amazon Lex V2 ahora es compatible con la configuración regional alemana (Austria). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	5 de noviembre de 2021

Nueva característica (7/10/2021)	Ahora puede proporcionar a los usuarios mensajes de actualización que se reproducen al inicio de una función de cumplimiento y periódicamente mientras se ejecuta la función. También puede crear mensajes que informen al usuario del estado del cumplimiento cuando la función esté completa. Para obtener más información, consulte Configuración de actualizaciones del cumplimiento.	7 de octubre de 2021
Expansión de la región (22/9/2021)	Amazon Lex V2 ya está disponible en las África (Ciudad del Cabo) (af-south-1) y Asia Pacífico (Seúl) (ap-northeast-2).	22 de septiembre de 2021
Nueva característica (22/9/2021)	Ahora puede ver las estadísticas de los enunciados que sus usuarios envían a su bot. Para obtener más información, consulte Visualización de estadísticas de enunciados.	22 de septiembre de 2021
Nueva característica (9/9/2021)	Amazon Lex V2 ahora es compatible con la configuración regional coreana (Corea). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	9 de septiembre de 2021

Nueva característica (27/7/2021)	Amazon Lex V2 ahora incluye un tipo de slot integrado para códigos postales del Reino Unido. Para obtener más información, consulte AMAZON. UKPostalCódigo.	27 de julio de 2021
Nueva característica (15/7/2021)	Amazon Lex V2 ahora es compatible con la configuración regional en inglés (India). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	15 de julio de 2021
Nueva característica (13/7/2021)	Amazon Lex V2 ahora proporciona una herramienta para migrar un bot de Amazon Lex V1 a la API de Amazon Lex V2. Para obtener más información, consulte Migrar un bot en la Guía para desarrolladores de Amazon Lex.	13 de julio de 2021

Nueva característica (26/6/2021)	Ahora puede configurar su bot de Amazon Lex V2 para que vuelva a activar las ranuras que ya se han rellenado estableciendo los valores de las ranuras en valores null y configurando el flujo de conversación para que se repita. Esta característica permite mantener conversaciones más dinámicas basadas en las respuestas de los clientes. Para obtener más información, consulte Obtención de ranuras de nuevo.	26 de junio de 2021
Nueva documentación	Amazon Lex V2 ahora proporciona directrices exhaustivas y prácticas recomendadas para el desarrollo de bots para ayudarlo a crear experiencias de conversación más eficaces. Para obtener más información, consulte Directrices para desarrolladores de bots Amazon Lex V2.	26 de junio de 2021
Nueva característica (15/6/2021)	Amazon Lex V2 ahora le permite aceptar varios valores para un solo slot en inglés (EE. UU.). Para obtener más información, consulte Usar varios valores en un slot.	15 de junio de 2021

Nueva característica (11/6/2021)	Ahora puede compilar bots más grandes para el idioma inglés. Para obtener más información, consulte Cuotas.	11 de junio de 2021
Nueva característica (20/5/2021)	Use las políticas basadas en recursos de Amazon Lex V2 para administrar el acceso a sus bots y alias de bots. Para obtener más información, consulte Políticas basadas en recursos de Amazon Lex V2.	20 de mayo de 2021
Nueva característica (18/5/2021)	Amazon Lex V2 ahora le permite importar y exportar bots y configuraciones regionales de bots. Puede usar esta función para copiar bots y configuraciones regionales de bots entre cuentas y regiones de AWS. Para obtener más información, consulte Importar y exportar.	18 de mayo de 2021
Expansión de la región (17/5/2021)	Amazon Lex V2 ya está disponible en Canadá (centro) (ca-central-1).	17 de mayo de 2021
Nueva característica (1/4/2021)	Amazon Lex V2 ahora es compatible con la configuración regional japonesa (Japón). Para obtener más información, consulte Idiomas y configuraciones regionales compatibles con Amazon Lex V2.	1 de abril de 2021

Nueva característica (12/3/2021)	Amazon Lex V2 ahora admite tres nuevos tipos de slots integrados: `AMAZON.City` , `AMAZON.Country` y `AMAZON.State` .	12 de marzo de 2021
Nueva guía	Esta es la primera versión de la Guía del usuario de Amazon Lex V2.	21 de enero de 2021

Referencia de la API

La Referencia de la API ahora es un documento aparte.

AWSGlosario de

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia de Glosario de AWS.

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.