



V2-Entwicklerhandbuch

Amazon Lex



Amazon Lex: V2-Entwicklerhandbuch

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon Lex V2?	1
Bezahlen für	3
Sind Sie ein Erstbenutzer von Amazon Lex V2?	3
Arbeitet mit AWS SDKs	3
Verwenden von Amazon Lex V2 mit einem AWS-SDK	4
Neueste Funktionen	7
AMAZON. BedrockAgentIntent für die Verbindung mit Bedrock Agents und Knowledge Bases	7
Unterstützte NLU für eine verbesserte Absichtsklassifizierung und Slot-Auflösung	7
Unterstützung für benutzerdefiniertes Vokabular für 17 weitere Sprachen	7
QinConnect integrierte Absicht für die Amazon Connect Connect-Integration	8
Integrierter Steckplatz für Amazon.Currency für US-amerikanische und spanische Sprachversionen	8
Verbesserte QnA-Absicht mit der Bedrock Knowledge Base und Guardrails	8
Support für Anthropic Claude 3-Modelle	8
Globale Resilienz für die Bot-Replikation in verschiedenen Regionen	9
Regionaler Support für AWS GovCloud (USA West)	9
Generative KI-Funktionen für Amazon Lex V2	9
Integrierter Steckplatz von Amazon.Confirmation für unbekanntes Yes/No/Maybe/Don	
Begriffsverständnis.	10
Messung der Geschäftsleistung mit Analytics	10
Bewertung der Bot-Leistung mit Test Workbench	10
Vertikale spezifische Bot-Vorlagen	10
Netzwerk von Bots	11
Visueller Konversationsgenerator	11
Steckplatztyp aus Verbundwerkstoff	11
Bedingte Verzweigung	11
Automatisierter Chatbot-Designer	12
Hinweise zur Laufzeit	12
Benutzerdefinierter Wortschatz	12
Typ des Grammatik-Slots	12
Kernkonzepte von Amazon Lex V2	14
Unterstützte Sprachen	16
Unterstützte Sprachen und Gebietsschemas	16

Sprachen und Gebietsschemas, die von den Funktionen von Amazon Lex V2 unterstützt werden	18
Regionen	20
Erste Schritte	21
Schritt 1: Einrichten eines -Kontos	21
Melden Sie sich an für AWS	22
Erstellen eines IAM-Benutzers	22
Erteilen programmgesteuerten Zugriffs	23
Nächster Schritt	25
Schritt 2: Erste Schritte (Konsole)	25
Übung 1: Einen Bot anhand eines Beispiels erstellen	26
Übung 2: Überprüfen Sie den Gesprächsablauf	28
Arbeiten mit Amazon Lex V2-Bots	40
Änderungen an Konversationsabläufen in Amazon Lex V2	41
Verschiedene Möglichkeiten, einen Bot mit Amazon Lex V2 zu erstellen	43
Verwenden der Konsole	44
Bot-Vorlagen verwenden	44
Amazon Lex V2-Bots mit dem Automated Chatbot Designer erstellen	47
Hinzufügen einer neuen Sprache zu einem Amazon Lex V2-Bot	57
Absichten hinzufügen	58
Konfiguration von Eingabeaufforderungen in einer bestimmten Reihenfolge	60
Beispiele für Äußerungen	61
Absicht-Struktur	63
Gesprächspfade erstellen	86
Verwenden von Visual Conversation Builder	104
Integrierte Absichten	115
Slot-Typen hinzufügen	148
Integrierte Steckplatztypen	149
Benutzerdefinierter Slot-Typ	166
Grammatik-Slot-Typ	169
Steckplatztyp aus Verbundwerkstoff	318
Einen Bot testen	326
Optimieren Sie Lex V2-Bots mit generativer KI	331
Beschreibender Bot-Builder aus Ihrer Beschreibung	333
Beispiele für einen deskriptiven Bot-Builder	338
Für NLD sind Berechtigungen erforderlich	340

Verwenden Sie die Generierung von Äußerungen, um Musteräußerungen zur Absichtserkennung zu generieren	341
Verwendung der unterstützten Slot-Auflösung für Slot-Werte	343
Beispiele für unterstützte Schlitzauflösung	344
In generativen KI-Konfigurationen aktivieren	348
Aktivieren Sie die unterstützte Steckplatzauflösung für Ihren Steckplatz	349
Genehmigungen für die unterstützte Slot-Auflösung	352
Verwendung BedrockAgentIntent eines Bedrock Agents in Amazon Lex V2	352
In generativen KI-Konfigurationen aktivieren	354
Aktivieren Sie Bedrock Agent Intent für Ihren Bot	354
Berechtigungen für Bedrock Agent Intent	355
Beispielanforderung	357
Verbessern Sie die Klassifizierung von Absichten und die Auflösung der Zeitnischen	358
Deaktivieren von Assisted NLU	360
Amazon.qn Alintent	360
Berechtigungen	362
Erstellen Sie ein Netzwerk von Bots für Ihre Lex V2-Bots	365
Erstellen Sie ein Netzwerk von Bots für Ihre Lex V2-Bots	366
Verwalte dein Netzwerk von Bots für deine Lex V2 Bots	367
Versionen Ihres Bot-Netzwerks für Lex V2	368
Aliase für Ihr Netzwerk von Bots für Lex V2	368
Kanalintegrationen für Ihr Lex V2-Bot-Netzwerk	369
Bereitstellung von Bots von Lex V2 für Ihre Produktionsumgebung	370
Versionierung und Aliase mit Ihrem Lex V2-Bot	370
Versionen	370
Aliase für deinen Lex V2-Bot	372
Integration mit einer Java-Anwendung	374
Verwenden Sie Global Resiliency, um Bots in anderen Regionen einzusetzen	378
Berechtigungen	380
Bereitstellung globaler Resilienz mit Ihrem Lex V2-Bot	381
Integration mit Messaging-Plattformen	385
Integration mit Facebook	386
Integrieren mit Slack	389
Integration mit Twilio SMS	393
Integration mit Kontaktzentren	395
Amazon Chime SDK	396

Amazon Connect	398
Genesys Cloud	399
Bot-Konversationen verstehen	400
Konversationskontext mit Ihren Lex V2-Bots	401
Den Absichtskontext für Ihren Lex V2-Bot festlegen	402
Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot	404
Sitzungsattribute für Ihren Lex V2-Bot festlegen	405
Einstellung von Anforderungsattributen für Ihren Lex V2-Bot	407
Einstellung des Sitzungs-Timeouts	410
Austausch von Informationen zwischen Absichten mit Ihrem Lex V2-Bot	410
Einstellen komplexer Attribute in Ihrem Lex V2-Bot	411
Bot-Sitzungen verstehen	413
Eine neue Sitzung starten	415
Absicht wechseln	415
Wiederaufnahme einer vorherigen Absicht	415
Slot-Werte werden validiert	416
AWS Lambda Funktionen integrieren	417
AWS Lambda Eingabeereignisformat für Lex V2	417
AWS Lambda Antwortformat für Lex V2	425
Erforderliche Felder in der Antwort	428
Gemeinsame Strukturen	431
Absicht	431
Slots	432
Status der Sitzung	436
Eine Lambda-Funktion erstellen und an einen Bot-Alias anhängen	440
Hängen Sie über die Konsole eine AWS Lambda Funktion an einen Bot an	443
Hängen Sie mithilfe von API-Operationen eine AWS Lambda Funktion an einen Bot an	445
Debugging einer Lambda-Funktion	452
Anpassen von Bot-Interaktionen mit Benutzern in Lex V2	453
Analysieren Sie die Stimmung der Benutzer in der Konversation	453
Verwendung von Konfidenzwerten zur Verbesserung der Konversationsgenauigkeit	454
Verwendung von Intent-Konfidenzwerten zur Verbesserung der Absichtsauswahl mit Lex V2	455
Verwenden Sie Konfidenzwerte für die Sprachtranskription, um die Konversation mit Ihrem Lex V2-Bot zu verbessern	458
Anpassen von Sprachtranskriptionen für die Verwendung mit Ihrem Lex V2-Bot	468

Verbesserung der Spracherkennung mit einem benutzerdefinierten Wortschatz	469
Verbesserung der Erkennung von Slot-Werten mit Laufzeithinweisen in der Konversation ...	478
Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation	482
Überwachung der Bot-Leistung in Lex V2	490
Messung der Geschäftsleistung mit Analytics	490
Die wichtigsten Definitionen	491
Filtern von Ergebnissen	493
Übersicht	494
Konversations-Dashboard	499
Leistungs-Dashboard	505
Für Analysen verwenden APIs	509
Verwaltung von Zugriffsberechtigungen für Analysen	516
Konversationsprotokolle für Ihre Lex V2-Bots aktivieren	517
Protokollierung von Konversationen mit Konversationsprotokollen in Lex V2	518
Verdecken von Slot-Werten in Konversationsprotokollen von Lex V2	537
Selektive Erfassung von Konversationsprotokollen in Lex V2	538
Fehler mit Fehlerprotokollen in Lex V2 protokollieren	547
IAM-Richtlinien für Fehlerprotokolle	547
Fehlerprotokolle in Lex V2 aktivieren	548
Fehlerprotokolle in Lex V2 deaktivieren	548
Überwachung der Betriebskennzahlen in Lex V2	549
Messung betrieblicher Kennzahlen mit CloudWatch für Lex V2	549
Ereignisse anzeigen mit CloudTrail	559
Bewertung der Leistung von Lex V2-Bots mit der Test Workbench	562
Generieren Sie ein Testset für Test Workbench	563
Testsätze verwalten	579
Führen Sie einen Test aus	608
Abdeckung der Testsätze in Test Workbench	610
Testergebnisse anzeigen	612
Details zu den Testergebnissen in Test Workbench	613
Konversationen auf Ihren Lex V2-Bot streamen	620
Starten eines Konversationsstreams zu einem Amazon Lex V2-Bot	622
Zeitliche Abfolge von Ereignissen für eine Audiokonversation bei Verwendung eines Amazon	
Lex V2-Bots	625
Eine Streaming-Konversation starten	628
Ereignis-Stream-Kodierung	644

Damit Ihr Bot unterbrochen werden kann	646
Wartet darauf, dass der Benutzer zusätzliche Informationen bereitstellt	647
Updates zum Fulfillment-Fortschritt für Ihren Lex V2-Bot konfigurieren	649
Aktualisierungen bei der Auftragsabwicklung	650
Antwort nach dem Versand	651
Timeouts für Benutzereingaben	653
So funktioniert das Interrupt-Verhalten in einem Lex V2-Bot	655
Stellen Sie die Timeouts für die Spracheingabe ein	655
Timeouts für die Texteingabe	657
Stellen Sie die Konfiguration für die DTMF-Eingabe ein	657
Bots in Lex V2 importieren und exportieren	659
Bots aus Lex V2 exportieren	659
Für den Export von Bots in Lex V2 sind IAM-Berechtigungen erforderlich	661
Einen Lex V2-Bot exportieren (Konsole)	662
Bots in Lex V2 importieren	663
Für den Import sind IAM-Berechtigungen erforderlich	664
Einen Lex V2-Bot importieren (Konsole)	665
Beim Import oder Export ein Passwort verwenden	667
JSON-Format für den Import und Export von Bots in Lex V2	668
Struktur der Manifestdatei	669
Struktur der Bot-Datei	669
Struktur der Bot-Locale-Datei	669
Struktur der beabsichtigten Datei	670
Struktur der Slot-Datei	672
Dateistruktur vom Typ Slot	675
Struktur der benutzerdefinierten Vokabeldatei	678
Ressourcen in Lex V2 kennzeichnen	680
Markieren Sie Ihre Ressourcen mit der Konsole oder API	680
Tag-Einschränkungen bei der Verwendung von Lex V2	681
Ressourcen taggen (Konsole)	681
Sicherheit	684
Datenschutz	685
Verschlüsselung im Ruhezustand	686
Verschlüsselung während der Übertragung	687
Identity and Access Management	687
Zielgruppe	687

Authentifizierung mit Identitäten	688
Verwalten des Zugriffs mit Richtlinien	692
So funktioniert Amazon Lex V2 mit IAM	695
Beispiele für identitätsbasierte Richtlinien	706
Beispiele für eine ressourcenbasierte Richtlinie	722
Von AWS verwaltete Richtlinien	731
Verwenden von serviceverknüpften Rollen	752
Fehlerbehebung	757
Protokollierung und Überwachung	762
Compliance-Validierung	762
Ausfallsicherheit	764
Sicherheit der Infrastruktur	764
VPC-Endpunkte (AWS PrivateLink)	765
Überlegungen zu Amazon Lex V2 VPC-Endpunkten	765
Erstellen eines Schnittstellen-VPC-Endpunkts für Amazon Lex V2	765
Erstellen einer VPC-Endpunktrichtlinie für Amazon Lex V2	766
Codebeispiele	768
Szenarien	768
Einen Amazon Lex Lex-Chatbot erstellen	768
Leitlinien und Bewährte Methoden	770
Kontingente	773
Kontingente für die Erstellung	773
Laufzeitkontingente	776
Migrationshandbuch	780
Überblick über Amazon Lex V2	780
Mehrere Sprachen in einem Bot	780
Vereinfachte Informationsarchitektur	780
Verbesserte Produktivität des Builders	781
AWS CloudFormation Ressourcen	783
Amazon Lex V2 und AWS CloudFormation Vorlagen	783
Erfahren Sie mehr über AWS CloudFormation	784
Dokumentverlauf	785
API-Referenz	804
AWS Glossar	805
.....	dccvvi

Was ist Amazon Lex V2?

Amazon Lex V2 ist ein AWS-Service zum Erstellen von Konversationsschnittstellen für Anwendungen, die Sprache und Text verwenden. Amazon Lex V2 bietet die umfassende Funktionalität und Flexibilität von Natural Language Understanding (NLU) und automatischer Spracherkennung (ASR), sodass Sie äußerst ansprechende Benutzererlebnisse mit lebensechten Konversationsinteraktionen schaffen und neue Produktkategorien erstellen können.

Amazon Lex V2 ermöglicht es jedem Entwickler, schnell Konversationsbots zu erstellen. Bei Amazon Lex V2 ist keine Deep-Learning-Expertise erforderlich. Um einen Bot zu erstellen, geben Sie den grundlegenden Konversationsablauf in der Amazon Lex V2-Konsole an. Amazon Lex V2 verwaltet den Dialog und passt die Antworten in der Konversation dynamisch an. Mithilfe der Konsole können Sie den Text- oder Stimme-Chatbot erstellen, testen und veröffentlichen. Sie können dann die umgangssprachlichen Schnittstellen zu Bots auf mobilen Geräten, Webanwendungen und Chat-Plattformen (z. B. Facebook Messenger) hinzufügen.

Amazon Lex V2 bietet Integration mit AWS Lambda vielen anderen Services auf der AWS-Plattform, einschließlich Amazon Connect, Amazon Comprehend und Amazon Kendra, und Sie können diese integrieren. Die Integration mit Lambda bietet Bots Zugriff auf vorgefertigte serverlose Unternehmenskonnektoren, um eine Verbindung zu Daten in SaaS-Anwendungen wie Salesforce herzustellen.

Für Bots, die nach dem 17. August 2022 erstellt wurden, können Sie bedingte Verzweigungen verwenden, um den Konversationsfluss mit Ihrem Bot zu steuern. Mit bedingter Verzweigung können Sie komplexe Konversationen erstellen, ohne Lambda-Code schreiben zu müssen.

Amazon Lex V2 bietet die folgenden Vorteile:

- **Einfachheit** — Amazon Lex V2 führt Sie durch die Verwendung der Konsole, um in wenigen Minuten Ihren eigenen Bot zu erstellen. Sie geben einige Beispielsätze an, und Amazon Lex V2 erstellt ein vollständiges Modell für natürliche Sprache, mit dem der Bot mithilfe von Sprache und Text interagieren kann, um Fragen zu stellen, Antworten zu erhalten und anspruchsvolle Aufgaben zu erledigen.
- **Demokratisierte Deep-Learning-Technologien** — Amazon Lex V2 bietet ASR- und NLU-Technologien zur Erstellung eines Speech Language Understanding (SLU) -Systems. Über SLU

verarbeitet Amazon Lex V2 Sprach- und Texteingaben in natürlicher Sprache, versteht die Absicht hinter der Eingabe und erfüllt die Benutzerabsicht, indem die entsprechende Geschäftsfunktion aufgerufen wird.

Spracherkennung und das Verstehen natürlicher Sprache gehören zu den schwierigsten Problemen, die es in der Informatik zu lösen gilt. Deshalb müssen ausgefeilte Deep-Learning-Algorithmen mit riesigen Datenmengen und Infrastrukturen trainiert werden. Amazon Lex V2 macht Deep-Learning-Technologien für alle Entwickler zugänglich. Amazon Lex V2-Bots konvertieren eingehende Sprache in Text und verstehen die Absicht des Benutzers, um eine intelligente Antwort zu generieren, sodass Sie sich darauf konzentrieren können, Ihre Bots mit Mehrwert für Ihre Kunden zu entwickeln und völlig neue Produktkategorien zu definieren, die durch Konversationsschnittstellen ermöglicht werden.

- **Reibungslose Bereitstellung und Skalierung** — Mit Amazon Lex V2 können Sie Ihre Bots direkt von der Amazon Lex V2-Konsole aus erstellen, testen und bereitstellen. Mit Amazon Lex V2 können Sie Ihre Sprach- oder Text-Bots zur Verwendung auf Mobilgeräten, Web-Apps und Chat-Diensten (z. B. Facebook Messenger) veröffentlichen. Amazon Lex V2 skaliert automatisch. Sie müssen sich keine Gedanken über die Bereitstellung von Hardware und die Verwaltung der Infrastruktur machen, um Ihr Bot-Erlebnis zu optimieren.
- **Integrierte Integration mit der AWS-Plattform** — Amazon Lex V2 funktioniert nativ mit anderen AWS-Services wie AWS Lambda Amazon CloudWatch. Sie profitieren von den Stärken der AWS-Plattform in Bezug auf Sicherheit, Überwachung, Benutzerauthentifizierung, Geschäftslogik, Speicher und Entwicklung von mobilen Anwendungen.
- **Wirtschaftlichkeit** — Bei Amazon Lex V2 fallen keine Vorabkosten oder Mindestgebühren an. Es werden Ihnen nur die Text- oder Sprachanforderungen berechnet, die getätigt wurden. Die pay-as-you-go Preisgestaltung und die niedrigen Kosten pro Anfrage machen den Service zu einer kostengünstigen Möglichkeit, Konversationsschnittstellen zu erstellen. Mit dem kostenlosen Kontingent für Amazon Lex V2 können Sie Amazon Lex V2 ganz einfach und ohne Anfangsinvestitionen testen.

Bezahlen für

Amazon Lex V2 berechnet Ihnen nur die von Ihnen gestellten Text- oder Sprachanfragen. Dieses Modell bietet Ihnen einen Service mit variablen Kosten, der mit Ihrem Unternehmen mitwachsen kann und Ihnen gleichzeitig die Kostenvorteile der AWS-Infrastruktur bietet. Weitere Informationen finden Sie unter [– Preise](#).

Wenn Sie sich für registrieren AWS, wird Ihr AWS Konto automatisch für alle Dienste in AWS angemeldet, einschließlich. Es werden jedoch nur die Services in Rechnung gestellt, die Sie tatsächlich nutzen. Wenn Sie ein neuer -Kunde sind, können Sie kostenlos mit beginnen. Weitere Informationen finden Sie unter [Kostenloses Kontingent für AWS](#).

Um Ihre Rechnung anzuzeigen, navigieren Sie zu Abrechnungs- und Kostenverwaltungs-Dashboard in der [AWS Fakturierung und Kostenmanagement -Konsole](#). Weitere Informationen zur AWS-Konto Abrechnung finden Sie im [AWS Billing Benutzerhandbuch](#). Wenn Sie Fragen zur AWS Abrechnung und haben AWS-Konten, wenden Sie sich an den [AWS Support](#).

Sind Sie ein Erstbenutzer von Amazon Lex V2?

Wenn Sie Amazon Lex V2 zum ersten Mal verwenden, empfehlen wir Ihnen, die folgenden Abschnitte der Reihe nach zu lesen:

1. [Kernkonzepte von Amazon Lex V2](#)— In diesem Abschnitt werden Amazon Lex V2 und die Funktionen vorgestellt, mit denen Sie einen Chatbot erstellen.
2. [Erste Schritte mit Amazon Lex V2](#)— In diesem Abschnitt richten Sie Ihr Konto ein und testen Amazon Lex V2.
3. [API-Referenz](#) — Dieser Abschnitt enthält Einzelheiten zu API-Vorgängen.

Amazon Lex V2 mit einem AWS SDK verwenden

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele

SDK-Dokumentation	Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK für Go	AWS SDK für Go Codebeispiele
AWS SDK für Java	AWS SDK für Java Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Codebeispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Codebeispiele
AWS SDK für .NET	AWS SDK für .NET Codebeispiele
AWS SDK für PHP	AWS SDK für PHP Codebeispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Codebeispiele
AWS SDK für Ruby	AWS SDK für Ruby Codebeispiele
AWS SDK for Rust	AWS SDK for Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK for Swift	AWS SDK for Swift Codebeispiele

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Feedback geben auswählen.

Verwenden von Amazon Lex V2 mit einem AWS-SDK

Neben der Konsolenoberfläche bietet Amazon Lex V2 umfassenden programmatischen Zugriff über AWS SDKs. Auf diese Weise können Sie Konversations-KI-Funktionen direkt in Ihre Anwendungen integrieren, Bot-Management-Aufgaben automatisieren und skalierbare Lösungen entwickeln.

Wenn Sie ein AWS-SDK mit Amazon Lex V2 verwenden, können Sie:

- Automatisieren Sie die Erstellung und Verwaltung von Bots — Programmgesteuertes Erstellen, Aktualisieren und Bereitstellen von Bots ohne manuelle Interaktion mit der Konsole
- Integration in bestehende Anwendungen — Fügen Sie Konversationsschnittstellen zu Webanwendungen, mobilen Apps und Unternehmenssystemen hinzu
- Skalieren Sie den Bot-Betrieb — Verwalten Sie mehrere Bots, Intents und Slot-Typen effizient mithilfe von Code
- Implementieren Sie benutzerdefinierte Workflows — Erstellen Sie ausgefeilte Konversationsabläufe, die sich in Ihre Geschäftslogik integrieren

Die folgende Tabelle zeigt die AWS SDKs , die Amazon Lex V2-Operationen unterstützen. Wählen Sie das SDK aus, das zu Ihrer Entwicklungsumgebung passt, und folgen Sie den bereitgestellten Links, um mit der Installation und Implementierung zu beginnen.

Programmiersprache	AWS SDK	Ressourcen für den Einstieg
Java	AWS SDK for Java 2.x	Entwicklerhandbuch API Reference
Python	AWS SDK für Python (Boto3)	Erste Schritte API Reference
JavaScript/ Node.js	AWS-SDK für JavaScript Version 3	Entwicklerhandbuch API Reference
.NET/C#	AWS SDK für .NET	Entwicklerhandbuch API Reference
Go	AWS-SDK SDK for Go v2	Entwicklerhandbuch API Reference
Ruby	AWS SDK für Ruby	Entwicklerhandbuch

Programmi ersprache	AWS SDK	Ressourcen für den Einstieg
		API Reference
PHP	AWS SDK für PHP	Entwicklerhandbuch API Reference

Um mit einem beliebigen SDK zu beginnen:

1. Installieren Sie das SDK für Ihre bevorzugte Programmiersprache mithilfe der Installationsanweisungen im Entwicklerhandbuch
2. Konfigurieren Sie Ihre AWS-Anmeldeinformationen und Regionseinstellungen
3. Richten Sie die erforderlichen IAM-Berechtigungen für Amazon Lex V2-Operationen ein
4. Informationen zu den spezifischen Vorgängen, die Sie benötigen, finden Sie in der API-Referenzdokumentation
5. Beginnen Sie mit grundlegenden Vorgängen wie dem Erstellen eines Bots oder dem Auflisten vorhandener Ressourcen

Ausführliche Beispiele und step-by-step Anleitungen zur programmgesteuerten Erstellung von Bots finden Sie unter den Links zur SDK-Dokumentation in der obigen Tabelle.

Neuste Funktionen

Dieses Thema enthält Informationen zu den neuesten Funktionen, die Amazon Lex V2 bietet:

AMAZON. BedrockAgentIntent für die Verbindung mit Bedrock Agents und Knowledge Bases

Amazon Lex V2 bietet jetzt den AMAZON. BedrockAgentIntent integrierte Absicht, Ihren Bot nahtlos mit Amazon Bedrock Agents und Knowledge Bases zu verbinden, um erweiterte Konversations-KI-Funktionen zu erhalten.

- [Dokumentation](#)

Unterstützte NLU für eine verbesserte Absichtsklassifizierung und Slot-Auflösung

Amazon Lex V2 bietet jetzt Assisted NLU, um die Absichtsklassifizierung und Slot-Auflösung mithilfe von Large Language Models (LLMs) zu verbessern und gleichzeitig die konfigurierten Absichten und Slots Ihres Bots einzuhalten.

- [Dokumentation](#)

Unterstützung für benutzerdefiniertes Vokabular für 17 weitere Sprachen

Amazon Lex V2 unterstützt jetzt benutzerdefinierte Vokabulare in 17 weiteren Sprachen, um die Genauigkeit der Spracherkennung zu verbessern.

- [Dokumentation](#)

QinConnect integrierte Absicht für die Amazon Connect Connect-Integration

Amazon Lex V2 bietet jetzt die QinConnect integrierte Möglichkeit, Ihren Bot nahtlos mit Amazon Connect zu verbinden, um das Contact-Center-Erlebnis zu verbessern.

- [Dokumentation](#)

Integrierter Steckplatz für Amazon.Currency für US-amerikanische und spanische Sprachversionen

Amazon Lex V2 bietet jetzt den integrierten Slot-Typ Amazon.Currency für Gebietsschemas in den USA und Spanien, um Währungswerte in Konversationen zu erfassen.

- [Dokumentation](#)

Verbesserte QnA-Absicht mit der Bedrock Knowledge Base und Guardrails

Die in QnA integrierte Absicht für generative KI unterstützt jetzt Bedrock Knowledge Base und Guardrails für sicherere und genauere Antworten.

- [Dokumentation](#)

Support für Anthropic Claude 3-Modelle

Die integrierte QnA-Funktion für generative KI unterstützt jetzt die Modelle Anthropic Claude 3 Haiku und Anthropic Claude 3 Sonnet für erweiterte Konversationsmöglichkeiten.

- [Dokumentation](#)

Globale Resilienz für die Bot-Replikation in verschiedenen Regionen

Amazon Lex V2 bietet jetzt globale Ausfallsicherheit, um Ihren Bot in einer zweiten AWS-Region zu replizieren und so die Verfügbarkeit und Notfallwiederherstellung zu verbessern.

- [Dokumentation](#)

Regionaler Support für AWS GovCloud (USA West)

Amazon Lex V2 ist jetzt in AWS GovCloud (USA West) verfügbar.

- [Amazon Lex Lex-Endpunkte und Kontingente](#)

Generative KI-Funktionen für Amazon Lex V2

Mit Amazon Lex V2 können Sie jetzt die generativen KI-Funktionen von Amazon Bedrock für Ihren Bot nutzen.

- Beschreibender Bot-BUILDER
 - [Was ist ein neuer Beitrag](#)
 - [Dokumentation](#)
- Unterstützte Steckplatzauflösung
 - [Was ist ein neuer Beitrag](#)
 - [Dokumentation](#)
- Generierung von Äußerungen
 - [Was ist ein neuer Beitrag](#)
 - [Dokumentation](#)
- AMAZON.QnAIntent(Häufig gestellte Fragen zu Konversationen)
 - [Was ist ein neuer Beitrag](#)
 - [Dokumentation](#)
- [AWS Blogbeitrag Machine Learning](#)

Integrierter Steckplatz von Amazon.Confirmation für unbekanntes Yes/No/Maybe/Don Begriffsverständnis.

Amazon Lex V2 bietet jetzt einen AMAZON.Confirmation integrierten Steckplatz, um die Genauigkeit von Slot-Bestätigungen und Antworten zu verbessern, die Yes/No/Maybe/Don nicht bekannt sind.

- [Dokumentation](#)

Messung der Geschäftsleistung mit Analytics

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, die Leistung von Intents und Slots im Analytics-Dashboard einzusehen.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)

Bewertung der Bot-Leistung mit Test Workbench

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, Testsätze zu erstellen und auszuführen, um die Bot-Leistung zu messen und Bot-Metriken zu verbessern.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)
- [AWS Blogbeitrag Machine Learning](#)

Vertikale spezifische Bot-Vorlagen

Amazon Lex V2 bietet Benutzern jetzt vorgefertigte Bot-Vorlagen mit ready-to-use Konversationsabläufen sowie Trainingsdaten und Dialogaufforderungen, sowohl für Sprach- als auch für Chat-Modalitäten.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)

Netzwerk von Bots

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, mehrere Bots in einem einzigen Netzwerk zu kombinieren und Anfragen auf der Grundlage von Benutzereingaben an den entsprechenden Bot weiterzuleiten.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)

Visueller Konversationsgenerator

Amazon Lex V2 bietet jetzt einen Drag-and-Drop-Konversationsgenerator, mit dem Sie Konversationspfade einfach entwerfen und visualisieren können, indem Sie Absichten in einer reichhaltigen visuellen Umgebung verwenden.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)
- [AWS Blogbeitrag Machine Learning](#)

Steckplatztyp aus Verbundwerkstoff

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, mehrere Steckplätze mithilfe logischer Ausdrücke zu einem zusammengesetzten Steckplatz zu kombinieren.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)

Bedingte Verzweigung

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, Bedingungen zu schreiben, um den Weg, den Kunden in einer Konversation mit Ihrem Bot einschlagen, besser kontrollieren zu können.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)

Automatisierter Chatbot-Designer

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, automatisch einen Chatbot aus Gesprächsprotokollen zu entwerfen. Lesen Sie die Anwendungsbeispiele.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)
- [AWS Blogbeitrag Machine Learning](#)
- [Amazon Lex — Seite mit automatisiertem Chatbot-Designer](#)

Hinweise zur Laufzeit

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, Laufzeithinweise zu konfigurieren, um die Erkennung von Phrasen zu verbessern und die Abfrage von Slot-Werten zu verbessern.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)

Benutzerdefinierter Wortschatz

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, ein benutzerdefiniertes Vokabular, eine Liste von Phrasen, die Eigennamen oder domänenspezifische Wörter enthalten können, zu erstellen, die Amazon Lex V2 in der Audioeingabe erkennt.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)
- [AWS Blogbeitrag Machine Learning](#)

Typ des Grammatik-Slots

Amazon Lex V2 bietet Benutzern jetzt die Möglichkeit, Grammatiken im XML-Format gemäß der Speech Recognition Grammar Specification (SRGS) zu verfassen, um Informationen in einer Konversation zu sammeln.

- [Was ist ein neuer Beitrag](#)
- [Dokumentation](#)
- [AWS-Blogbeitrag Machine Learning](#)

Kernkonzepte von Amazon Lex V2

Amazon Lex V2 ermöglicht es Ihnen, Chat-Anwendungen (Bots) zu erstellen, um Informationen von Benutzern zu erhalten, um eine Aufgabe zu erledigen. Sie können beispielsweise einen Bot erstellen, um Blumen zu bestellen oder ein Hotelzimmer zu buchen. Im Folgenden sind die typischen Schritte für die Arbeit mit Amazon Lex V2 aufgeführt:

1. Erstellen Sie einen Bot und fügen Sie eine oder mehrere Sprachen hinzu. Konfigurieren Sie den Bot so, dass er das Ziel des Benutzers versteht, mit dem Benutzer Gespräche führt, um Informationen zu erhalten, und die Absicht des Benutzers erfüllt.
2. Testen Sie den Bot. Sie können den Testfenster-Client verwenden, der von der Amazon Lex V2-Konsole bereitgestellt wird.
3. Veröffentlichen Sie eine Version und erstellen Sie einen Alias.
4. Stellen Sie den Bot bereit. Sie können den Bot auf Ihren eigenen Anwendungen oder Messaging-Plattformen wie Facebook Messenger oder Slack bereitstellen

Bevor Sie beginnen, sollten Sie sich mit den folgenden Kernkonzepten und der Terminologie von Amazon Lex V2 vertraut machen:

- Bot — Ein Bot führt automatisierte Aufgaben wie die Bestellung einer Pizza, die Buchung eines Hotels, die Bestellung von Blumen usw. aus. Ein Amazon Lex V2-Bot basiert auf Funktionen zur automatischen Spracherkennung (ASR) und zum Verstehen natürlicher Sprache (NLU).

Amazon Lex V2-Bots können Benutzereingaben verstehen, die mit Text oder Sprache bereitgestellt werden, und natürliche Sprache kommunizieren.

- Sprache — Ein Amazon Lex V2-Bot kann sich in einer oder mehreren Sprachen unterhalten. Jede Sprache ist unabhängig von den anderen. Sie können Amazon Lex V2 so konfigurieren, dass es sich mit einem Benutzer mit einheimischen Wörtern und Ausdrücken unterhält. Weitere Informationen finden Sie unter [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#).
- Absicht — Eine Absicht steht für eine Aktion, die der Benutzer ausführen möchte. Sie erstellen einen Bot, um eine oder mehrere Absichten zu unterstützen. Sie könnten beispielsweise eine Absicht erstellen, bei der Pizzen und Getränke bestellt werden. Für jede Absicht geben Sie die folgenden erforderlichen Informationen ein:
 - Name der Absicht — Ein beschreibender Name für die Absicht. Beispiel, **OrderPizza**.

- **Beispieläußerungen** — Wie ein Benutzer die Absicht vermitteln könnte. Ein Benutzer könnte beispielsweise sagen: „Kann ich eine Pizza bestellen“ oder „Ich möchte eine Pizza bestellen“.
- **So erfüllen Sie die Absicht** — Wie Sie die Absicht erfüllen möchten, nachdem der Benutzer die erforderlichen Informationen bereitgestellt hat. Wir empfehlen, dass Sie eine Lambda-Funktion erstellen, um die Absicht zu erfüllen.

Sie können die Absicht optional so konfigurieren, dass Amazon Lex V2 die Informationen zur erforderlichen Erfüllung an die Client-Anwendung zurücksendet.

Neben benutzerdefinierten Intents bietet Amazon Lex V2 integrierte Intents, mit denen Sie Ihren Bot schnell einrichten können. Weitere Informationen finden Sie unter [Integrierte Absichten](#).

Amazon Lex beinhaltet immer eine Fallback-Absicht für jeden Bot. Die Fallback-Absicht wird immer dann verwendet, wenn Amazon Lex die Absicht des Benutzers nicht ableiten kann. Weitere Informationen finden Sie unter [AMAZON.FallbackIntent](#).

- **Slot** — Eine Absicht kann null oder mehr Slots oder Parameter erfordern. Sie fügen Slots als Teil der Konfiguration einer Absicht hinzu. Zur Laufzeit fordert Amazon Lex V2 den Benutzer zur Eingabe bestimmter Slot-Werte auf. Der Benutzer muss Werte für alle erforderlichen Slots angeben, bevor Amazon Lex V2 die Absicht erfüllen kann.

Für die `OrderPizza` Absicht sind beispielsweise Felder wie Größe, Art der Kruste und Anzahl der Pizzen erforderlich. Für jeden Steckplatz geben Sie den Slot-Typ und eine oder mehrere Eingabeaufforderungen an, die Amazon Lex V2 an den Client sendet, um Werte vom Benutzer abzurufen. Ein Benutzer kann mit einem Slot-Wert antworten, der zusätzliche Wörter wie „große Pizza bitte“ oder „Bleiben wir bei klein“ enthält. Amazon Lex V2 versteht den Slot-Wert immer noch.

- **Steckplatztyp** — Jeder Steckplatz hat einen Typ. Sie können Ihren eigenen Slot-Typ erstellen oder integrierte Slot-Typen verwenden. Beispiel: Sie möchten die folgenden Slot-Typen für die Absicht `OrderPizza` erstellen und verwenden:
 - **Größe** - Mit Aufzählungswerten `Small`, `Medium`, und `Large`.
 - **Kruste** - Mit Aufzählungswerten `Thick` und `Thin`.

Amazon Lex V2 bietet auch integrierte Steckplatztypen. `AMAZON.Number` ist beispielsweise ein integrierter Slot-Typ, den Sie für die Anzahl bestellter Pizzas verwenden können. Weitere Informationen finden Sie unter [Integrierte Absichten](#).

- **Version** — Eine Version ist eine nummerierte Momentaufnahme Ihrer Arbeit, die Sie zur Verwendung in verschiedenen Teilen Ihres Workflows veröffentlichen können, z. B. in der Entwicklung, Betabereitstellung und Produktion. Sobald Sie eine Version erstellt haben, können

Sie einen Bot so verwenden, wie er zum Zeitpunkt der Erstellung der Version vorhanden war. Nachdem Sie eine Version erstellt haben, bleibt sie unverändert, während Sie weiter an Ihrer Anwendung arbeiten.

- **Alias** — Ein Alias ist ein Verweis auf eine bestimmte Version eines Bots. Mit einem Alias können Sie die Version aktualisieren, die Ihre Client-Anwendungen verwenden. Beispielsweise können Sie einen Alias auf Version 1 Ihres Bot zeigen lassen. Wenn Sie bereit sind, den Bot zu aktualisieren, veröffentlichen Sie Version 2 und ändern den Alias so, dass er auf die neue Version zeigt. Da Ihre Anwendungen den Alias anstelle einer bestimmten Version verwenden, erhalten alle Ihre Clients die neuen Funktionen, ohne dafür aktualisiert werden zu müssen.

Eine Liste der AWS Regionen, in denen Amazon Lex V2 verfügbar ist, finden Sie unter [Amazon Lex V2-Endpunkte und Kontingente](#) in der Amazon Web Services General Reference.

Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas

Amazon Lex V2 unterstützt eine Vielzahl von Sprachen und Gebietsschemas. In diesem Thema sind die unterstützten Sprachen und die Funktionen aufgeführt, die diese Sprachen unterstützen.

Unterstützte Sprachen und Gebietsschemas

Amazon Lex V2 unterstützt die folgenden Sprachen und Gebietsschemas.

Code	Sprache und Gebietsschema
ar_AE	Golf-Arabisch (Vereinigte Arabische Emirate) *
ca_ES	Katalanisch (Spanien)
de_AT	Deutsch (Österreich)
de_DE	Deutsch (Deutschland)
de_DE	Englisch (Australien)
de_DE	Englisch (UK)
de_DE	Englisch (Indien)
de_DE	Englisch (USA)

Code	Sprache und Gebietsschema
de_ZA	Englisch (Südafrika)
es_419	Spanisch (Lateinamerika)
es_ES	Spanisch (Spanien)
es_US	Spanisch (USA)
fi_FI	Finnisch (Finnland)
fr_CA	Französisch (Kanada)
fr_FR	Französisch (Frankreich)
Hi_IN	Hindi (Indien) **
es_IT	Italienisch (Italien)
ja_JP	Japanisch (Japan)
ko_KR	Koreanisch (Korea)
nl_NL	Niederländisch (Niederlande)
Nein_nein	Norwegisch (Norwegen)
pl_PL	Polnisch (Polen)
pt_BR	Portugiesisch (Brasilien)
pt_PT	Portugiesisch (Portugal)
sv_SE	Schwedisch (Schweden)
zh_CN	Mandarin (VR China)
zh_HK	Kantonesisch (Hongkong)

*Arabisch

Amazon Lex V2 ist die Variante des Arabischen, auf der Amazon Lex V2 trainiert wurde, ist Golf-Arabisch. Denken Sie daran, wenn Sie Beispieläußerungen für Ihren Bot bereitstellen. Beachten Sie, dass die arabische Schrift von rechts nach links geschrieben wird.

****Hindi**

Amazon Lex V2 ist in der Lage, Hindi-Endbenutzer zu bedienen, die frei zwischen Hindi und Englisch wechseln können. Wenn Sie planen, einen Bot zu entwickeln, der diesen Sprachwechsel unterstützt, empfehlen wir die folgenden bewährten Methoden:

- Schreiben Sie in der Bot-Definition englische Wörter in lateinischer Schrift.
- Bei mindestens 50% deiner Beispieläußerungen sollte es sich um einen Sprachwechsel innerhalb desselben Satzes handeln. Verwenden Sie in diesen Äußerungen die Devanagari-Schrift für Hindi-Wörter und die lateinische Schrift für englische Wörter (z. B. „Ticketbuch“). „).
- Wenn Sie erwarten, dass Benutzer mit dem Bot Hindi-Wörter in lateinischer Schrift oder englische Wörter in Devanagari-Schrift kommunizieren, sollten Sie Beispiele für Hindi-Wörter in lateinischer Schrift (z. B. „mujhe ek ticket book karni hai“) und englische Wörter in Devanagari-Schrift (z. B. „“) einbeziehen in Ihren Beispieläußerungen.
- Wenn Sie erwarten, dass Benutzer mit dem Bot mithilfe von Sätzen kommunizieren, die vollständig auf Hindi oder vollständig auf Englisch verfasst sind, sollten Sie Beispieläußerungen hinzufügen, die vollständig in einer Sprache verfasst sind (z. B. „Ich möchte ein Ticket buchen“).

Sprachen und Gebietsschemas, die von den Funktionen von Amazon Lex V2 unterstützt werden

In der folgenden Tabelle sind die Funktionen von Amazon Lex V2 aufgeführt, die auf bestimmte Sprachen und Gebietsschemas beschränkt sind. Alle anderen Funktionen von Amazon Lex V2 werden in allen Sprachen und Regionen unterstützt.

Funktion	Unterstützte Sprachen und Gebietsschemas
AMAZON.AlphaNumeric	Alle Sprachen und Gebietsschemas außer Koreanisch (ko_KR)
AMAZON.KendraSearchIntent	Englisch (US) (en_US)

Funktion	Unterstützte Sprachen und Gebietsschemas
Verbesserung der Spracherkennung mit einem benutzerdefinierten Wortschatz	Englisch (UK) (en_GB) Englisch (US) (en_US)
Automatisierter Chatbot-Designer	Englisch (US) (en_US)
Verfügbarkeit in Regionen	Die folgenden Sprachen und Gebietsschemas sind in den Regionen Asien-Pazifik (Singapur) (ap-southeast-1) und Afrika (Kapstadt) (ap-south-1) nicht verfügbar: • Golfarabisch (Vereinigte Arabische Emirate) (ar_AE) • Katalanisch (Spanien) (ca_ES) • Finnisch (Finnland) (fi_FI) • Hindi (Indien) (hi_IN) • Niederländisch (Niederlande) (nl_NL) • Norwegisch (Norwegen) (no_NO) • Polnisch (pl_PL) • Portugiesisch (Brasilien) (pt_BR) • Portugiesisch (Portugal) (pt_PT) • Schwedisch (sv_SE) • Mandarin (VR China) (zh_CN) • Kantonesisch (Hongkong) (zh_HK)
Den Absichtskontext für Ihren Lex V2-Bot festlegen	Englisch (US) (en_US)
Grammatik-Slot-Typ	Englisch (Australien) (en_AU) Englisch (UK) (en_GB) Englisch (US) (en_US)
Verwendung mehrerer Werte in einem Slot	Englisch (US) (en_US)

Funktion	Unterstützte Sprachen und Gebietsschemas
Verbesserung der Erkennung von Slot-Werten mit Laufzeithinweisen in der Konversation	Englisch (UK) (en_GB) Englisch (US) (en_US)
Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation	Englisch (Australien) (en_AU) Englisch (UK) (en_GB) Englisch (US) (en_US)
Verwendung von Konfidenzwerten zur Verbesserung der Konversationsgenauigkeit	Englisch (UK) (en_GB) Englisch (US) (en_US)

Regionen

Eine Liste der AWS Regionen, in denen Amazon Lex V2 verfügbar ist, finden Sie unter [AWS-Regionen und -Endpunkte](#) in der Allgemeine AWS-Referenz.

Erste Schritte mit Amazon Lex V2

Wenn Sie Amazon Lex V2 noch nicht kennen, empfehlen wir Ihnen, [Kernkonzepte von Amazon Lex V2](#) zuerst zu lesen.

Bevor Sie Amazon Lex V2 verwenden können, müssen Sie ein AWS-Konto erstellen und eine AWS-Konto-ID erhalten. Sie erstellen ein Benutzerkonto und eine IAM-Rolle, die Ihnen die für die Nutzung erforderlichen Berechtigungen gibt

Nachdem Sie Ihre Konten erstellt haben, können Sie sie mit der AWS Konsole oder der API verwenden.

Amazon Lex V2 bietet API-Operationen, die Sie in Ihre vorhandenen Anwendungen integrieren können. Eine Liste der unterstützten Operationen finden Sie in der [API-Referenz](#). Sie können alle der folgenden Optionen verwenden:

- **AWS-SDK** — Wenn SDKs Sie das verwenden, werden Ihre Anfragen an Amazon Lex V2 automatisch mit den von Ihnen angegebenen Anmeldeinformationen signiert und authentifiziert. Wir empfehlen, dass Sie ein SDK verwenden, um Ihre Anwendung zu erstellen.
- **AWS CLI** — Sie können das verwenden AWS CLI , um auf jede Amazon Lex V2-Funktion zuzugreifen, ohne Code schreiben zu müssen.

Themen

- [Schritt 1: Richten Sie ein AWS Konto ein und erstellen Sie einen Administratorbenutzer](#)
- [Schritt 2: Erste Schritte \(Konsole\)](#)

Schritt 1: Richten Sie ein AWS Konto ein und erstellen Sie einen Administratorbenutzer

Bevor Sie Amazon Lex V2 zum ersten Mal verwenden, führen Sie die folgenden Aufgaben aus:

1. [Melden Sie sich an für AWS](#)
2. [Erstellen eines IAM-Benutzers](#)

Melden Sie sich an für AWS

Wenn Sie bereits ein AWS Konto haben, überspringen Sie diese Aufgabe.

Wenn Sie sich für Amazon Web Services (AWS) registrieren, wird Ihr AWS Konto automatisch für alle Dienste in registriert AWS, einschließlich Amazon Lex V2. Berechnet werden Ihnen aber nur die Services, die Sie nutzen.

Mit Amazon Lex V2 zahlen Sie nur für die Ressourcen, die Sie nutzen. Wenn Sie ein neuer AWS Kunde sind, können Sie kostenlos mit Amazon Lex V2 beginnen. Weitere Informationen finden Sie unter [AWS – kostenloses Nutzungskontingent](#).

Wenn Sie bereits ein AWS Konto haben, fahren Sie mit der nächsten Aufgabe fort. Wenn Sie noch kein AWS Konto haben, gehen Sie wie folgt vor, um eines zu erstellen.

Um ein AWS Konto zu erstellen

1. Öffnen Sie [https://portal.aws.amazon.com/billing/die Anmeldung](https://portal.aws.amazon.com/billing/die-Anmeldung).
2. Folgen Sie den Online-Anweisungen.

Ein Teil des Anmeldevorgangs umfasst den Empfang eines Telefonanrufs oder einer Textnachricht und die Eingabe eines Bestätigungscode auf der Telefontastatur.

Wenn Sie sich für eine anmelden AWS-Konto, wird eine Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um [Aufgaben auszuführen, die Root-Benutzerzugriff erfordern](#).

Notieren Sie sich Ihre AWS Konto-ID, da Sie sie für die nächste Aufgabe benötigen.

Erstellen eines IAM-Benutzers

Dienste in AWS, wie Amazon Lex V2, erfordern, dass Sie beim Zugriff auf sie Anmeldeinformationen angeben, damit der Service feststellen kann, ob Sie über Berechtigungen für den Zugriff auf die Ressourcen verfügen, die diesem Service gehören.

Erstellen Sie ein IAM-Benutzerkonto, um auf Ihr Konto für Amazon Lex V2 zuzugreifen:

- Verwenden Sie AWS Identity and Access Management (IAM), um einen IAM-Benutzer zu erstellen

- Fügen Sie den Benutzer einer IAM-Gruppe mit Administratorrechten hinzu
- Erteilen Sie dem IAM-Benutzer, den Sie erstellt haben, Administratorrechte.

Sie können dann AWS mit einer speziellen URL und den Anmeldeinformationen des IAM-Benutzers darauf zugreifen.

Für die Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie einen Benutzer namens (`adminuser`) mit Administratorrechten haben. Befolgen Sie die Schritte zum Einrichten des `adminuser` in Ihrem Konto.

Erstellen eines Administrator-Benutzers und Anmelden in der Konsole

1. Erstellen Sie einen Administratorbenutzer mit dem Namen `adminuser` in Ihrem AWS Konto. Anweisungen finden Sie im IAM-Benutzerhandbuch unter [Erstellen Ihres ersten IAM-Benutzers und Ihrer ersten Administratorgruppe](#).
2. Als Benutzer können Sie sich AWS Management Console mit einer speziellen URL bei der anmelden. Weitere Informationen finden Sie unter [Wie sich Benutzer in Ihrem Konto anmelden](#) im IAM-Benutzerhandbuch.

Weitere Informationen zu IAM finden Sie unter:

- [AWS Identity and Access Management \(IAM\)](#)
- [Erste Schritte](#)
- [IAM Benutzerhandbuch](#)

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb des Interagierens möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldinformationen im AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Nächster Schritt

[Schritt 2: Erste Schritte \(Konsole\)](#)

Schritt 2: Erste Schritte (Konsole)

Der einfachste Weg, die Verwendung von Amazon Lex V2 zu erlernen, ist die Verwendung der Konsole. Damit Sie starten können, haben wir die folgenden Übungen erstellt. Für alle wird die Konsole benötigt:

- Übung 1 — Erstellen Sie einen Amazon Lex V2-Bot mithilfe eines Blueprints, eines vordefinierten Bots, der die gesamte erforderliche Bot-Konfiguration bereitstellt. Sie müssen nur ein Minimum an Arbeit aufwenden, um das end-to-end Setup zu testen.
- Übung 2 — Überprüfen Sie die JSON-Strukturen, die zwischen Ihrer Client-Anwendung und einem Amazon Lex V2-Bot gesendet wurden.

Themen

- [Übung 1: Einen Bot anhand eines Beispiels erstellen](#)

- [Übung 2: Überprüfen Sie den Gesprächsablauf](#)

Übung 1: Einen Bot anhand eines Beispiels erstellen

In dieser Übung erstellen Sie Ihren ersten Amazon Lex V2-Bot und testen ihn in der Amazon Lex V2-Konsole. Für diese Übung verwenden Sie das OrderFlowersBeispiel.

Überblick über das Beispiel

Sie verwenden das OrderFlowersBeispiel, um einen Amazon Lex V2-Bot zu erstellen. Weitere Informationen zur Struktur eines Bots finden Sie unter [Kernkonzepte von Amazon Lex V2](#).

- Absicht — OrderFlowers
- Slot-Typen - Ein benutzerdefinierter Slot-Typ namens FlowerTypes mit Aufzählungswerten: roseslilies, und tulips.
- Slots: Die Absicht erfordert die folgenden Informationen (d. h. Slots), bevor der Bot die Absicht erfüllen kann.
 - PickupTime(AMAZON.TIME integrierter Typ)
 - FlowerType(FlowerTypesbenutzerdefinierter Typ)
 - PickupDate (AMAZON.DATE integrierter Typ)
- Äußerung: Die folgenden Beispieläußerungen zeigen die Absicht des Benutzers an:
 - "Ich möchte Blumen abholen."
 - "Ich möchte einige Blumen bestellen."
- fordert - Nachdem der Bot die Absicht identifiziert, verwendet er die folgenden Anweisungen zum Ausfüllen der Slots:
 - Anforderung für den FlowerType Slot - "Was für Blumen möchten Sie bestellen?"
 - Aufforderung zur Eingabe des PickupDate Automaten — „An welchem Tag soll der {FlowerType} abgeholt werden?“
 - Aufforderung zur Eingabe des PickupTime Automaten — „Um wie viel Uhr soll der {FlowerType} abgeholt werden?“
 - Bestätigungserklärung — „Okay, Ihr {FlowerType} wird bis {PickupTime} am {PickupDate} zur Abholung bereit sein. Ist das OK?“

So erstellen Sie einen Amazon Lex V2-Bot (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie Create bot aus.
3. Wählen Sie für die Methode Erstellung die Option Mit einem Beispiel beginnen aus.
4. Wählen Sie im Abschnitt Beispiel-Bots OrderFlowers aus der Liste aus.
5. Geben Sie dem Bot im Abschnitt Bot-Konfiguration einen Namen und eine optionale Beschreibung. Der Name muss in Ihrem Konto eindeutig sein.
6. Wählen Sie im Abschnitt Berechtigungen die Option Neue Rolle mit grundlegenden Amazon Lex-Berechtigungen erstellen aus. Dadurch wird eine AWS Identity and Access Management (IAM-) Rolle mit den Berechtigungen erstellt, die Amazon Lex V2 benötigt, um Ihren Bot auszuführen.
7. Treffen Sie im Abschnitt Gesetz zum Schutz der Privatsphäre von Kindern im Internet (COPPA) die entsprechende Wahl.
8. Behalten Sie in den Abschnitten Sitzungs-Timeout und Erweiterte Einstellungen die Standardeinstellungen bei.
9. Wählen Sie Weiter aus. Amazon Lex V2 erstellt Ihren Bot.

Nachdem Sie Ihren Bot erstellt haben, müssen Sie eine oder mehrere Sprachen hinzufügen, die der Bot unterstützt. Eine Sprache enthält die Absichten, Slot-Typen und Slots, die der Bot verwendet, um mit Benutzern zu kommunizieren.

Um einem Bot eine Sprache hinzuzufügen

1. Wählen Sie im Abschnitt Sprache eine unterstützte Sprache aus und fügen Sie eine Beschreibung hinzu.
2. Behalten Sie die Standardwerte für die Felder Sprachinteraktion und Intent Classification Confidence Score bei.
3. Wählen Sie Fertig, um die Sprache zum Bot hinzuzufügen.

Nachdem Sie „Fertig“ ausgewählt haben, öffnet die Konsole den Intent-Editor. Sie können den Intent-Editor verwenden, um die vom Bot verwendeten Absichten zu untersuchen. Wenn Sie mit der Untersuchung des Bots fertig sind, können Sie ihn testen.

Um den OrderFlowers Bot zu testen

1. Wählen Sie oben auf der Seite Build aus. Warte, bis der Bot gebaut hat.
2. Wenn der Build abgeschlossen ist, wählen Sie Test, um das Testfenster zu öffnen.
3. Testen Sie den Bot. Beginnen Sie das Gespräch mit einer der Beispielaussagen, z. B. „Ich würde gerne Blumen pflücken“.

Nächste Schritte

Nachdem Sie Ihren ersten Bot mithilfe einer Vorlage erstellt haben, können Sie die Konsole verwenden, um Ihren eigenen Bot zu erstellen. Anweisungen zum Erstellen eines benutzerdefinierten Bots und weitere Informationen zum Erstellen von Bots finden Sie unter [Arbeiten mit Amazon Lex V2-Bots](#).

Übung 2: Überprüfen Sie den Gesprächsablauf

In dieser Übung überprüfen Sie die JSON-Strukturen, die zwischen Ihrer Client-Anwendung und dem Amazon Lex V2-Bot, in dem Sie erstellt haben, gesendet [Übung 1: Einen Bot anhand eines Beispiels erstellen](#) werden. Die Konversation verwendet den [RecognizeText](#) Vorgang, um die JSON-Strukturen zu generieren. Das [RecognizeUtterance](#) gibt dieselben Informationen zurück wie HTTP-Header in der Antwort.

Die JSON-Strukturen werden bei jeder Runde der Konversation unterteilt. Ein Turn ist eine Anfrage von der Client-Anwendung und eine Antwort vom Bot.

Turn 1

Während der ersten Runde der Konversation initiiert die Client-Anwendung die Konversation mit Ihrem Bot. Sowohl der URI als auch der Hauptteil der Anfrage enthalten Informationen über die Anfrage.

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

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

- Der URI identifiziert den Bot, mit dem die Client-Anwendung kommuniziert. Er enthält auch eine von der Client-Anwendung generierte Sitzungs-ID, die eine bestimmte Konversation zwischen einem Benutzer und dem Bot identifiziert.
- Der Hauptteil der Anfrage enthält den Text, den der Benutzer in die Client-Anwendung eingegeben hat. In diesem Fall wird nur der Text gesendet, Ihre Anwendung kann jedoch zusätzliche Informationen wie Anforderungsattribute oder den Sitzungsstatus senden. Weitere Informationen finden Sie unter dem Vorgang [RecognizeText](#).

Von aus text erkennt Amazon Lex V2 die Absicht des Benutzers, Blumen zu bestellen. Amazon Lex V2 wählt einen der Intent-Slots (FlowerType) und eine der Prompts für den Slot aus und sendet dann die folgende Antwort an die Client-Anwendung. Der Client zeigt dem Benutzer die Antwort an.

```
{
  "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"
  }
}
```

Runde 2

In Runde 2 antwortet der Benutzer auf die Aufforderung des Amazon Lex V2-Bots in Runde 1 mit einem Wert, der den `FlowerType` Slot füllt.

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

In der Antwort für Runde 2 wird angezeigt, dass der `FlowerType` Slot gefüllt ist, und Sie werden aufgefordert, den nächsten Slot-Wert zu ermitteln.

```
{
  "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"
}
```

Dreh dich um

In Runde 3 antwortet der Benutzer auf die Aufforderung des Amazon Lex V2-Bots in Runde 2 mit einem Wert, der den PickupDate Slot füllt.

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

Bei der Antwort für Runde 3 werden sowohl die FlowerType Slots als auch die PickupDate Slots gefüllt und Sie werden aufgefordert, den letzten Slot-Wert ausfindig zu machen.

```
{
  "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": {}
}
```

Runde 4

In Runde 4 gibt der Benutzer den endgültigen Slot-Wert für die Absicht an, also den Zeitpunkt, zu dem die Blumen abgeholt werden.

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

In der Antwort sendet Amazon Lex V2 eine Bestätigungsaufforderung an den Benutzer, um zu bestätigen, dass die Bestellung korrekt ist. Das `dialogAction` ist auf eingestellt `ConfirmIntent` und das `confirmationState` ist `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"
}
```

Biegen Sie 5 ab

In der letzten Runde reagiert der Benutzer mit der Bestätigungsaufforderung.

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

In der Antwort, die Amazon Lex V2 sendet, wird angezeigt, dass die Absicht erfüllt wurde, indem die Optionen `confirmationState` An `Confirmed` und `dialogAction` Bis geschlossen gesetzt wurden. Alle Slot-Werte sind für die Client-Anwendung verfügbar.

```
{
  "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"
}
```

Arbeiten mit Amazon Lex V2-Bots

Sie erstellen einen Amazon Lex V2-Bot, der mit Ihren Benutzern interagiert und Informationen zur Ausführung einer Aufgabe abrufen. Sie können beispielsweise einen Bot erstellen, der die Informationen sammelt, die für die Bestellung eines Blumenstraußes oder die Buchung eines Hotelzimmers erforderlich sind.

Um einen Bot zu erstellen, benötigen Sie die folgenden Informationen:

1. Die Sprache, die der Bot verwendet, um mit dem Kunden zu interagieren. Sie können eine oder mehrere Sprachen auswählen. Jede Sprache enthält unabhängige Absichten, Slots und Slot-Typen.
2. Die Absichten oder Ziele, bei deren Erfüllung der Bot dem Benutzer hilft. Ein Bot kann eine oder mehrere Absichten enthalten, z. B. die Bestellung von Blumen oder die Buchung eines Hotels und eines Mietwagens. Sie müssen entscheiden, welche Aussagen oder Äußerungen der Benutzer macht, um die Absicht zu initiieren.
3. Die Informationen oder Slots, die Sie vom Benutzer einholen müssen, um eine Absicht zu erfüllen. Möglicherweise benötigen Sie vom Benutzer die Art der Blumen oder das Startdatum einer Hotelreservierung. Sie müssen eine oder mehrere Eingabeaufforderungen definieren, die Amazon Lex V2 verwendet, um den Slot-Wert vom Benutzer abzurufen.
4. Der Typ der Steckplätze, die Sie vom Benutzer benötigen. Möglicherweise müssen Sie einen benutzerdefinierten Slot-Typ erstellen, z. B. eine Liste mit Blumen, die ein Benutzer bestellen kann, oder Sie können einen integrierten Slot-Typ verwenden, z. B. den AMAZON.Date Slot-Typ für das Startdatum einer Reservierung verwenden.
5. Der Ablauf der Benutzerinteraktion innerhalb und zwischen Absichten. Sie können den Konversationsablauf so konfigurieren, dass die Interaktion zwischen dem Benutzer und dem Bot definiert wird, sobald die Absicht aufgerufen wird. Sie können eine Lambda-Funktion erstellen, um die Absicht zu validieren und zu erfüllen.

Themen

- [Änderungen an Konversationsabläufen in Amazon Lex V2](#)
- [Verschiedene Möglichkeiten, einen Bot mit Amazon Lex V2 zu erstellen](#)
- [Hinzufügen einer neuen Sprache zu einem Amazon Lex V2-Bot](#)
- [Absichten hinzufügen](#)

- [Slot-Typen hinzufügen](#)
- [Einen Bot mit der Konsole testen](#)

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Änderungen an Konversationsabläufen in Amazon Lex V2

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt.

Vor der Änderung verwaltete Amazon Lex V2 die Konversation, indem es Slots auf der Grundlage ihrer Absichtsprioritäten auslotete. Sie könnten dieses Verhalten dynamisch ändern und den Konversationspfad basierend auf Benutzereingaben ändern, indem Sie die Lambda-Funktion verwenden `DialogAction`. Dies könnte erreicht werden, indem der aktuelle Status der Konversation verfolgt und anhand des Sitzungsstatus programmgesteuert entschieden wird, was als Nächstes zu tun ist.

Mit dieser Änderung können Sie Konversationspfade und bedingte Verzweigungen mit der Amazon Lex V2-Konsole oder APIs ohne Verwendung einer Lambda-Funktion erstellen. Amazon Lex V2 verfolgt den Status der Konversation und steuert anhand der bei der Erstellung des Bots definierten Bedingungen, was als Nächstes zu tun ist. Auf diese Weise können Sie problemlos komplexe Konversationen erstellen, während Sie Ihren Bot entwerfen.

Diese Änderungen geben Ihnen die vollständige Kontrolle über die Konversation mit Ihrem Kunden. Sie müssen jedoch keinen Pfad definieren. Wenn Sie keinen Konversationspfad angeben, erstellt Amazon Lex V2 einen Standardpfad, der auf der Priorität der Slots in Ihrer Absicht basiert. Sie können weiterhin Lambda-Funktionen verwenden, um Konversationspfade dynamisch zu definieren.

In einem solchen Szenario wird die Konversation auf der Grundlage des in der Lambda-Funktion konfigurierten Sitzungsstatus wieder aufgenommen.

Dieses Update bietet Folgendes:

- Eine neue Konsolenerfahrung für die Erstellung von Bots mit komplexen Konversationsabläufen.
- Aktualisierungen der bestehenden Version APIs für die Erstellung von Bots zur Unterstützung der neuen Konversationsabläufe.
- Eine erste Antwort zum Senden einer Nachricht bei einem Absichtsaufwurf.
- Neue Antworten für Slot-Elicitation, Lambda-Aufruf als Dialog-Code-Hook und Bestätigung.
- Möglichkeit, bei jeder Konversation die nächsten Schritte festzulegen.
- Bewertung der Bedingungen für die Gestaltung mehrerer Konversationspfade.
- Einstellung von Slot-Werten und Sitzungsattributen zu einem beliebigen Zeitpunkt während der Konversation.

Beachten Sie Folgendes für ältere Bots:

- Bots, die vor dem 17. August 2022 erstellt wurden, verwenden weiterhin den alten Mechanismus zur Verwaltung von Konversationsabläufen. Bots, die nach diesem Datum erstellt wurden, verwenden die neue Art der Verwaltung des Konversationsflusses.
- Neue Bots, die nach dem 17. August 2022 über Importe erstellt wurden, verwenden das neue Konversationsflussmanagement. Bei Importen vorhandener Bots wird weiterhin die alte Art des Konversationsmanagements verwendet.
- Um das neue Konversationsflussmanagement für einen Bot zu aktivieren, der vor dem 17. August 2022 erstellt wurde, exportieren Sie den Bot und importieren Sie den Bot dann mit einem neuen Bot-Namen. Der aus dem Import neu erstellte Bot verwendet das neue Konversationsflussmanagement.

Beachten Sie Folgendes für neue Bots, die nach dem 17. August 2022 erstellt wurden:

- Amazon Lex V2 folgt dem definierten Konversationsablauf genau so, wie er entworfen wurde, um das gewünschte Erlebnis zu bieten. Sie sollten alle Flow-Zweige konfigurieren, um Standard-Konversationspfade während der Laufzeit zu vermeiden.
- Konversationsschritte, die auf einen Code-Hook folgen, sollten vollständig konfiguriert sein, da unvollständige Schritte zum Ausfall des Bots führen können. Wir empfehlen Ihnen, vor dem 17.

August 2022 erstellte Bots zu validieren, da für diese Bots keine automatische Überprüfung der Konversationsschritte nach einem Code-Hook erfolgt.

Verschiedene Möglichkeiten, einen Bot mit Amazon Lex V2 zu erstellen

Sie können einen Bot mit Amazon Lex V2 auf folgende Weise erstellen:

1. Verwenden Sie die Amazon Lex V2-Konsole, um mithilfe einer Website-Oberfläche einen Bot zu erstellen. Weitere Informationen finden Sie unter [Einen Bot mit der Amazon Lex V2-Konsole erstellen](#).
2. Verwenden Sie den Descriptive Bot Builder, um mithilfe der generativen KI-Funktionen von Amazon Bedrock einen Bot zu erstellen. Weitere Informationen finden Sie unter [Verwenden Sie eine Beschreibung, um einen Bot in Lex V2 mit dem beschreibenden Bot-BUILDER zu erstellen](#).
3. Verwenden Sie Bot-Vorlagen, um einen vorkonfigurierten Bot zu erstellen, der gängigen Geschäftsanwendungsfällen entspricht. Weitere Informationen finden Sie unter [Amazon Lex V2-Bots mithilfe von Vorlagen erstellen](#).
4. Verwenden Sie ein [AWS SDK](#), um mithilfe von API-Operationen einen Bot zu erstellen.
5. Verwenden Sie den automatisierten Chatbot-Designer, um mithilfe vorhandener Chatprotokolle zwischen Agenten und Kunden einen Bot zu erstellen. Weitere Informationen finden Sie unter [Amazon Lex V2-Bots mit dem Automated Chatbot Designer erstellen](#).
6. Importieren Sie eine bestehende Bot-Definition. Weitere Informationen finden Sie unter [Bots in Lex V2 importieren](#).
7. Wird verwendet AWS CloudFormation , um einen Bot zu erstellen. Weitere Informationen finden Sie unter [Erstellen von Amazon Lex V2-Ressourcen mit AWS CloudFormation](#).

Themen

- [Einen Bot mit der Amazon Lex V2-Konsole erstellen](#)
- [Amazon Lex V2-Bots mithilfe von Vorlagen erstellen](#)
- [Amazon Lex V2-Bots mit dem Automated Chatbot Designer erstellen](#)

Einen Bot mit der Amazon Lex V2-Konsole erstellen

Sie erstellen einen Amazon Lex V2-Bot, der mit Ihren Benutzern interagiert und Informationen zur Ausführung einer Aufgabe abrufen. Sie können beispielsweise einen Bot erstellen, der die Informationen sammelt, die Sie benötigen, um Blumen zu bestellen oder ein Hotelzimmer zu buchen. Um einen Bot mithilfe der AWS Konsole zu erstellen, definieren Sie zunächst den Namen, die Beschreibung und einige grundlegende Informationen.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie Create bot aus.
3. Wählen Sie im Abschnitt Erstellungsmethode die Option Traditionell und dann Leeren Bot erstellen aus.
4. Geben Sie dem Bot im Abschnitt Bot-Konfiguration einen Namen und optional eine Beschreibung.
5. Wählen Sie im Abschnitt IAM-Berechtigungen eine AWS Identity and Access Management (IAM-) Rolle aus, die Amazon Lex V2-Berechtigungen für den Zugriff auf andere AWS Dienste wie Amazon gewährt. CloudWatch Sie können Amazon Lex V2 die Rolle erstellen lassen, oder Sie können eine vorhandene Rolle mit CloudWatch Berechtigungen auswählen.
6. Wählen Sie im Abschnitt Gesetz zum Schutz der Privatsphäre von Kindern im Internet (COPPA) die entsprechende Antwort aus.
7. Wählen Sie im Abschnitt Timeout für Leerlaufsitzen die Dauer aus, für die Amazon Lex V2 eine Sitzung mit einem Benutzer geöffnet hält. Amazon Lex V2 verwaltet Sitzungsvariablen für die Dauer der Sitzung, sodass Ihr Bot eine Konversation mit denselben Variablen fortsetzen kann.
8. Fügen Sie im Abschnitt Erweiterte Einstellungen (optional) Tags hinzu, mit denen Sie den Bot identifizieren, den Zugriff kontrollieren und Ressourcen überwachen können.
9. Wählen Sie Weiter, um den Bot zu erstellen, und fahren Sie mit dem Hinzufügen einer Sprache fort.

Amazon Lex V2-Bots mithilfe von Vorlagen erstellen

Amazon Lex V2 bietet vorgefertigte Lösungen, um Erlebnisse in großem Maßstab zu schaffen und die digitale Interaktion zu fördern. Die vorgefertigten Bot-Vorlagen automatisieren und standardisieren Kundenerlebnisse. Die Bot-Vorlagen bieten ready-to-use Gesprächsabläufe sowie Trainingsdaten

und Dialogaufforderungen, sowohl für Sprach- als auch für Chat-Modalitäten. Sie können die Bereitstellung von Bot-Lösungen beschleunigen und gleichzeitig die Ressourcen optimieren, sodass Sie sich auf Kundenbeziehungen konzentrieren können.

Sie können vorgefertigte Bots erstellen, die auf Ihren geschäftlichen Anwendungsfall zugeschnitten sind. Sie können die AWS CloudFormation Konsole verwenden, um die vorgefertigten Optionen für die zugehörigen Dienste wie Amazon S3, Amazon Connect und DynamoDB auszuwählen.

Derzeit unterstützt Amazon Lex V2 die folgenden Geschäftsbereiche:

- Finanzdienstleistungen
- Bestellungen im Einzelhandel
- Autoversicherung
- Telekommunikation
- Dienstleistungen von Fluggesellschaften
- Zusätzliche Vorlagen werden in future verfügbar sein.

Sie können mit der bereitgestellten Vorlage für Geschäftslösungen einen Bot erstellen und ihn an Ihre Geschäftsanforderungen anpassen.

 Note

Die Vorlagen erstellen Ressourcen außerhalb von Amazon Lex V2 durch AWS CloudFormation Stapel. Der Stack muss möglicherweise in anderen Konsolen wie Lambda und DynamoDB geändert werden.

Voraussetzungen für die Erstellung und Bereitstellung der Bot-Vorlage:

- Ein AWS Konto
- Zugriff auf die folgenden AWS Dienste:
 - Amazon Lex V2 zum Erstellen von Bots
 - Lambda für die Login-Funktionen für Unternehmen
 - DynamoDB zum Erstellen der Tabellen
 - IAM-Zugriff zum Erstellen von Richtlinien und Rollen
 - AWS CloudFormation um den Stack auszuführen

- IAM-Zugriff und geheime Schlüsselanmeldedaten
- Amazon Connect Connect-Instanz (optional)

 Note

Bei der Nutzung verschiedener AWS Dienste fallen für jeden Dienst entsprechende Nutzungskosten an.

So erstellen Sie einen Bot aus Amazon Lex V2-Vorlagen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie im linken Navigationsbereich Bot-Vorlagen aus.
3. Wählen Sie aus, welche Branche Sie für Ihre Bot-Vorlage verwenden möchten. HINWEIS: Derzeit sind 5 Bot-Vorlagen verfügbar. Weitere werden bald folgen.
4. Wählen Sie für die Vorlage, die Sie verwenden möchten, die Option Erstellen aus. Es öffnet sich eine Registerkarte AWS CloudFormation, in der Sie die Parameter für den AWS CloudFormation Stack bearbeiten können. Alle Optionen für die von Ihnen gewählte Vorlage sind bereits abgeschlossen. Sie können auch mehr darüber erfahren, wie die Bot-Vorlage funktioniert, indem Sie Weitere Informationen auswählen.
5. AWS CloudFormation Erstellt in der AWS CloudFormation Konsole eine Standardkonfiguration für jeden der Werte für die Vorlage, die Sie ausgewählt haben. Sie können auch Ihren eigenen Stacknamen, Ihre eigenen AWS CloudFormation Parameter, Ihre Amazon DynamoDB-Tabelle und (optional) Amazon Connect Connect-Parameter auswählen.
6. Wählen Sie unten im Fenster die Option Stack erstellen aus.
7. AWS CloudFormation verarbeitet die Anfrage mehrere Minuten lang im Hintergrund, um Ihren neuen Bot zu konfigurieren. HINWEIS: Der Prozess erstellt automatisch Ressourcen für eine DynamoDB-Tabelle, einen Amazon Connect Connect-Kontaktablauf und eine Amazon Connect Connect-Instance. Sie können den Fortschritt in der AWS CloudFormation Konsole verfolgen und dann zurück zur Amazon Lex V2-Konsole navigieren, sobald die CloudFormation Stack-Erstellung abgeschlossen ist.
8. Wenn die Erstellung erfolgreich war, wird eine Meldung angezeigt und Sie können „Gehe zur Bot-Liste“ auswählen, um zur Seite „Bots“ zu gelangen, auf der Sie Ihren neuen Bot finden, der für Ihre Tests und Verwendung bereit ist.

Konfiguration Ihrer Bot-Vorlage

Lambda-Funktionen — Die Bot-Vorlage erstellt automatisch die benötigten Lambda-Funktionen für Ihre Bereitstellung. Wenn mehrere Bots Teil der Vorlagenlösung sind, werden mehrere Lambda-Funktionen in den AWS CloudFormation Parametern aufgeführt. Wenn Sie über bestehende Lambda-Funktionen verfügen, die Sie mit Ihrem Bot bereitstellen möchten, können Sie den Namen Ihrer benutzerdefinierten Lambda-Funktion eingeben.

Amazon DynamoDB — Die Bot-Vorlage erstellt automatisch die DynamoDB-Tabelle, die zum Laden Ihrer Beispielrichtliniendaten benötigt wird. Sie können auch den Namen Ihrer benutzerdefinierten DynamoDB-Tabelle eingeben. Ihre benutzerdefinierte DynamoDB-Tabelle sollte genauso formatiert sein wie die Standardtabelle, die durch die Bot-Vorlagenbereitstellung erstellt wurde.

Amazon Connect — Sie können Ihre Amazon Connect Connect-Instance so konfigurieren, dass sie mit Ihrer neuen Bot-Vorlage funktioniert, indem ConnectInstance Sie den ARN und einen eindeutigen Code eingeben ContactFlowName. Mit Amazon Connect können Sie Ihren Bot mit einem IVR-System von end-to-end testen.

Fehlerbehebung bei Ihrer Bot-Vorlage

- Vergewissern Sie sich, dass Sie über die erforderlichen Berechtigungen verfügen, um die Vorlage zu erstellen, die Sie auswählen. Benutzer benötigen CloudFormation: CreateStack Berechtigungen sowie Berechtigungen für die AWS Ressourcen, die in der Vorlage aufgeführt sind. Eine Liste der Ressourcen, für die Benutzerberechtigungen erforderlich sind, befindet sich unten auf der Seite Vorlage erstellen.
- Wenn Ihre Bot-Vorlage nicht erstellt werden kann, enthält das rote Banner in der Amazon Lex V2-Konsole einen Link zu dem AWS CloudFormation Stack, der für die Erstellung der Vorlage verantwortlich ist. In der AWS CloudFormation Konsole können Sie auf der Registerkarte Ereignisse den spezifischen Fehler sehen, der zum Fehlschlagen der Vorlage geführt hat. Nachdem Sie den AWS CloudFormation Fehler überprüft haben, finden Sie CloudFormation weitere Informationen unter [Problembehandlung](#).
- Bot-Vorlagen funktionieren nur mit den Beispieldaten. Sie müssen die DynamoDB-Tabelle mit Ihren Daten füllen, damit die Vorlagen mit Ihren benutzerdefinierten Daten funktionieren.

Amazon Lex V2-Bots mit dem Automated Chatbot Designer erstellen

Der Automated Chatbot Designer hilft Ihnen dabei, Bots aus vorhandenen Konversationsprotokollen zu entwerfen. Er analysiert die Transkripte und schlägt ein erstes Design mit Absichten und Slot-

Typen vor. Sie können das Bot-Design wiederholen, Eingabeaufforderungen hinzufügen, den Bot erstellen, testen und bereitstellen.

Nachdem Sie mithilfe der Amazon Lex V2-Konsole oder API einen neuen Bot erstellt oder Ihrem Bot eine Sprache hinzugefügt haben, können Sie Transkripte von Konversationen zwischen zwei Parteien hochladen. Der automatisierte Chatbot-Designer analysiert die Transkripte und bestimmt die Absichten und Slot-Typen für den Bot. Es kennzeichnet auch die Konversationen, die die Erstellung einer bestimmten Absicht oder eines bestimmten Slot-Typs beeinflusst haben, für Ihre Überprüfung.

Sie verwenden die Amazon Lex V2-Konsole oder die API, um Gesprächsprotokolle zu analysieren und Absichten und Slot-Typen für einen Bot vorzuschlagen.

 Note

Sie können nur Transkripte in englischer Sprache (USA) verwenden.

Sie können die vorgeschlagenen Absichten und Slot-Typen überprüfen, nachdem der Chatbot-Designer die Analyse abgeschlossen hat. Nachdem Sie eine vorgeschlagene Absicht oder einen Slot-Typ hinzugefügt haben, können Sie ihn mithilfe der Konsole oder der API ändern oder aus dem Bot-Design löschen.

Der automatisierte Chatbot-Designer unterstützt Gesprächsprotokolldateien mithilfe des Contact Lens for Amazon Connect Connect-Schemas. Wenn Sie eine andere Contact-Center-Anwendung verwenden, müssen Sie die Gesprächsprotokolle in das vom Chatbot-Designer verwendete Format umwandeln. Weitere Informationen finden Sie unter [Format des Eingabeprotokolls](#).

Um den automatisierten Chatbot-Designer verwenden zu können, müssen Sie der IAM-Rolle, die den Designer ausführt, Zugriff gewähren. Die spezifische IAM-Richtlinie finden Sie unter [Erlauben Sie Benutzern, den Automated Chatbot Designer zu verwenden](#). Damit Amazon Lex V2 Ausgabedaten mit einem optionalen AWS KMS Schlüssel verschlüsseln kann, müssen Sie den Schlüssel mit der unter angegebenen Richtlinie aktualisieren. [Erlaubt Benutzern die Verwendung eines AWS KMS Schlüssels zum Verschlüsseln und Entschlüsseln von Dateien](#)

 Note

Wenn Sie eine verwenden KMS key, müssen Sie unabhängig von der verwendeten IAM Rolle eine KMS key Richtlinie angeben.

Themen

- [Gesprächsprotokolle importieren](#)
- [Absichten und Slot-Typen erstellen](#)
- [Format des Eingabeprotokolls](#)
- [Format des Ausgabeprotokolls](#)

Gesprächsprotokolle importieren

Das Importieren von Gesprächsprotokollen erfolgt in drei Schritten:

1. Bereiten Sie die Transkripte für den Import vor, indem Sie sie in das richtige Format konvertieren. Wenn Sie Contact Lens für Amazon Connect verwenden, haben die Transkripte bereits das richtige Format.
2. Laden Sie die Transkripte in einen Amazon S3 S3-Bucket hoch. Wenn Sie Contact Lens verwenden, befinden sich Ihre Transkripte bereits in einem S3-Bucket.
3. Analysieren Sie die Transkripte mithilfe der Amazon Lex V2-Konsole oder API-Operationen. Die Zeit, die für den Abschluss der Schulung benötigt wird, hängt vom Umfang der Transkripte und der Komplexität der Konversation ab. In der Regel werden jede Minute 500 Zeilen von Transkripten analysiert.

Jeder dieser Schritte wird in den folgenden Abschnitten beschrieben.

Transkripte aus Contact Lens für Amazon Connect importieren

Der automatisierte Chatbot-Designer von Amazon Lex V2 ist mit Transkriptdateien von Contact Lens kompatibel. Um Kontaktlinsen-Transkriptdateien verwenden zu können, müssen Sie Contact Lens einschalten und den Speicherort der Ausgabedateien notieren.

Um Transkripte aus Contact Lens zu exportieren

1. Aktivieren Sie Contact Lens in Ihrer Amazon Connect Connect-Instanz. Anweisungen finden Sie unter [Kontaktlinsen für Amazon Connect aktivieren](#) im Amazon Connect Connect-Administratorhandbuch.
2. Notieren Sie sich den Speicherort des S3-Buckets, den Amazon Connect für Ihre Instance verwendet. Um den Standort zu sehen, öffnen Sie die Datenspeicherseite in der Amazon Connect Connect-Konsole. Anweisungen finden Sie unter [Instance-Einstellungen aktualisieren](#) im Amazon Connect Connect-Administratorhandbuch.

Nachdem Sie Kontaktlinsen aktiviert und den Speicherort Ihrer Transkriptdateien notiert haben, finden Sie unter Anweisungen [Analysieren Sie Ihre Transkripte mit der Amazon Lex V2-Konsole](#) zum Importieren und Analysieren Ihrer Transkripte.

Bereiten Sie die Transkripte vor

Bereiten Sie Ihre Transkripte vor, indem Sie Transkriptdateien erstellen.

- Erstellen Sie pro Konversation eine Protokolldatei, in der die Interaktionen zwischen den Parteien aufgeführt sind. Jede Interaktion in der Konversation kann sich über mehrere Zeilen erstrecken. Sie können sowohl redigierte als auch nicht redigierte Versionen der Konversation bereitstellen.
- Die Datei muss das unter angegebene JSON-Format haben. [Format des Eingabeprotokolls](#)
- Sie müssen mindestens 1.000 Konversationsrunden angeben. Um Ihre Absichten und Slot-Typen besser erkennen zu können, sollten Sie etwa 10.000 oder mehr Konversationsrunden einplanen. Der automatisierte Chatbot-Designer verarbeitet nur die ersten 700.000 Runden.
- Es gibt keine Begrenzung für die Anzahl der Transkriptdateien, die Sie hochladen können, und es gibt auch keine Dateigrößenbeschränkung.

Wenn Sie die Transkripte, die Sie importieren, nach Datum filtern möchten, müssen sich die Dateien in der folgenden Verzeichnisstruktur befinden:

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

Die Transkriptdatei muss das Datum im Format "yyyy-mm-dd" irgendwo im Dateinamen enthalten.

Um Transkripte aus anderen Contact-Center-Anwendungen zu exportieren

1. Verwenden Sie die Tools Ihrer Contact-Center-Anwendung, um Konversationen zu exportieren. Die Konversation muss mindestens die unter angegebenen Informationen enthalten [Format des Eingabeprotokolls](#).
2. Wandeln Sie die von Ihrer Contact-Center-Anwendung erstellten Transkripte in [Format des Eingabeprotokolls](#) das unter beschriebene Format um. Sie sind für die Durchführung der Transformation verantwortlich.

Wir stellen drei Skripte für die Erstellung von Transkripten zur Verfügung. Diese sind:

- Ein Skript zum Kombinieren von Kontaktlins-Transkripten mit Amazon Lex V2-Konversationsprotokollen. Kontaktlinsen-Transkripte enthalten keine Teile von Amazon Connect Connect-Konversationen, die mit Amazon Lex V2-Bots interagieren. Das Skript erfordert, dass Konversationsprotokolle für Amazon Lex V2 aktiviert sind und dass entsprechende Berechtigungen zum Abfragen von CloudWatch Konversationsprotokollprotokollen und Contact Lens S3-Buckets erforderlich sind.
- Ein Skript zur Umwandlung von Amazon Transcribe Transcribe-Anrufanalysen in das Amazon Lex V2-Eingabeformat.
- Ein Skript zur Umwandlung von Amazon Connect Connect-Chat-Transkripten in das Amazon Lex V2-Eingabeformat.

Sie können die Skripts aus diesem GitHub Repository herunterladen: <https://github.com/aws-samples/amazon-lex-bot-recommendation-integration>.

Laden Sie Ihre Transkripte in einen S3-Bucket hoch

Wenn Sie Contact Lens verwenden, sind Ihre Transkriptdateien bereits in einem S3-Bucket enthalten. Den Speicherort und die Dateinamen Ihrer Transkriptdateien finden Sie unter [Beispiel für Kontaktlinsen-Ausgabedateien](#) im Amazon Connect Connect-Administratorhandbuch.

Wenn Sie eine andere Contact-Center-Anwendung verwenden und keinen S3-Bucket für Ihre Transkriptdateien eingerichtet haben, gehen Sie wie folgt vor. Andernfalls, wenn Sie über einen vorhandenen S3-Bucket verfügen, gehen Sie nach der Anmeldung bei der Amazon S3 S3-Konsole wie folgt vor, und beginnen Sie mit Schritt 5.

So laden Sie die Dateien zu einem S3-Bucket: hoch

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Wählen Sie Create Bucket (Bucket erstellen) aus.
3. Geben Sie dem Bucket einen Namen und wählen Sie eine Region aus. Die Region muss dieselbe sein, die Sie für Amazon Lex V2 verwenden. Stellen Sie die anderen Optionen nach Bedarf für Ihren Anwendungsfall ein.
4. Wählen Sie Create Bucket (Bucket erstellen) aus.

5. Wählen Sie aus der Liste der Buckets einen vorhandenen Bucket oder den Bucket, den Sie gerade erstellt haben
6. Klicken Sie auf Upload.
7. Fügen Sie die Transkriptdateien hinzu, die Sie hochladen möchten.
8. Klicken Sie auf Upload.

Analysieren Sie Ihre Transkripte mit der Amazon Lex V2-Konsole

Sie können automatisiertes Bot-Design nur in einer leeren Sprache verwenden. Sie können einem vorhandenen Bot eine neue Sprache hinzufügen oder einen neuen Bot erstellen.

Um eine neue Sprache in einem neuen Bot zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie Bot erstellen
3. Wählen Sie Start with Automated Chatbot Designer. Füllen Sie die Informationen aus, um Ihren neuen Bot zu erstellen.
4. Wählen Sie Weiter
5. Geben Sie unter Sprache zum Bot hinzufügen die Informationen für die Sprache ein.
6. Wählen Sie im Abschnitt Speicherort der Transkriptdatei auf S3 den S3-Bucket aus, der Ihre Transkriptdateien und gegebenenfalls den lokalen Pfad zu den Dateien enthält.
7. Sie können optional Folgendes wählen:
 - Ein AWS KMS Schlüssel zum Verschlüsseln der Transkriptdaten während der Verarbeitung. Wenn Sie keinen Schlüssel auswählen, wird ein AWS KMS Serviceschlüssel verwendet.
 - Um die Transkripte nach einem bestimmten Datumsbereich zu filtern. Wenn Sie die Transkripte filtern möchten, müssen sie sich in der richtigen Ordnerstruktur befinden. Weitere Informationen finden Sie unter [Bereiten Sie die Transkripte vor](#).
8. Wählen Sie Erledigt aus.

Warten Sie, bis Amazon Lex V2 das Transkript verarbeitet hat. Sie erhalten eine Abschlussmeldung, wenn die Analyse abgeschlossen ist.

Wie beenden Sie die Analyse Ihres Transkripts

Falls Sie die Analyse der von Ihnen hochgeladenen Transkripte beenden müssen, können Sie einen laufenden `BotRecommendation` Job beenden, der `BotRecommendationStatus` den Status `Bearbeitung` hat. Sie können auf die Schaltfläche `Verarbeitung beenden` klicken, die sich auf dem Banner befindet, nachdem Sie einen Job von der Konsole aus eingereicht haben, oder indem Sie das CLI SDK für die `StopBotRecommendation` API verwenden. Weitere Informationen finden Sie unter [StopBotRecommendation](#)

Nach dem `StopBotRecommendation` Aufrufen von `BotRecommendationStatus` ist der interne Modus auf eingestellt `Stopping` und es fallen keine Gebühren an. Um sicherzustellen, dass der Job beendet wurde, können Sie die `DescribeBotRecommendation` API aufrufen und überprüfen, ob der Vorgang abgeschlossen `BotRecommendationStatus` ist `Stopped`. Dies dauert normalerweise 3-4 Minuten.

Die Verarbeitung nach dem Aufruf der `StopBotRecommendation` API wird Ihnen nicht in Rechnung gestellt.

Absichten und Slot-Typen erstellen

Nachdem der Chatbot-Designer Intents und Slot-Typen erstellt hat, wählen Sie die Intents und Slot-Typen aus, die Sie Ihrem Bot hinzufügen möchten. Sie können die Details der einzelnen Absichten und Slot-Typen überprüfen, um zu entscheiden, welche Empfehlungen für Ihren Anwendungsfall am relevantesten sind.

Du kannst auf den Namen einer empfohlenen Absicht klicken, um dir die vom Chatbot-Designer vorgeschlagenen Beispieläußerungen und Slots anzusehen. Wenn du „Zugeordnete Abschriften anzeigen“ auswählst, kannst du auch durch die Konversationen blättern, die du bereitgestellt hast. Diese Transkripte beeinflussen die Empfehlung des Chatbot-Designers zu dieser Absicht. Wenn Sie auf eine Beispieläußerung klicken, können Sie die Hauptkonversation und die entsprechende Dialogform überprüfen, die diese bestimmte Äußerung beeinflusst hat.

Sie können auf den Namen eines bestimmten Slot-Typs klicken, um die empfohlenen Slot-Werte zu sehen. Wenn Sie die Option `Zugeordnete Transkripte anzeigen` auswählen, können Sie die Konversationen überprüfen, die diesen Slot-Typ beeinflusst haben. Dabei ist die Agenten-Aufforderung, die für den Slot-Typ auslöst, hervorgehoben. Wenn Sie auf einen bestimmten Wert für den Slot-Typ klicken, können Sie die Hauptkonversation und den entsprechenden Dialog überprüfen, der diesen Wert beeinflusst hat.

Um Absichten und Slot-Typ zu überprüfen und hinzuzufügen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Bot aus, mit dem Sie arbeiten möchten.
3. Wählen Sie Sprachen anzeigen.
4. Wählen Sie aus der Liste der Sprachen die Sprache aus, mit der Sie arbeiten möchten.
5. Wählen Sie unter Konversationsstruktur die Option Überprüfen aus.
6. Wählen Sie in der Liste der Absichten und Slot-Typen diejenigen aus, die Sie dem Bot hinzufügen möchten. Du kannst eine Absicht oder einen Slot-Typ wählen, um Details und die zugehörigen Transkripte zu sehen.

Die Absichten werden danach sortiert, ob Amazon Lex V2 sicher ist, dass die Absicht mit den verarbeiteten Transkripten verknüpft ist.

Format des Eingabeprotokolls

Im Folgenden finden Sie das Eingabedateiformat zum Generieren von Absichten und Slot-Typen für Ihren Bot. Die Eingabedatei muss diese Felder enthalten. Andere Felder werden ignoriert.

Das Eingabeformat ist mit dem Ausgabeformat von Contact Lens für Amazon Connect kompatibel. Wenn Sie Contact Lens verwenden, müssen Sie Ihre Transkriptdateien nicht ändern. Weitere Informationen finden Sie unter [Beispiel für Kontaktlinsen-Ausgabedateien](#). Wenn Sie eine andere Contact-Center-Anwendung verwenden, müssen Sie Ihre Transkriptdatei in dieses Format umwandeln.

```
{
  "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"
      }
    ]
  }
}

```

Die folgenden Felder müssen in der Eingabedatei vorhanden sein:

- Teilnehmer Identifiziert die Teilnehmer der Konversation und die Rolle, die sie spielen.
- Version Die Version des Eingabedateiformats. Immer „1.1.0“.
- ContentMetadata Gibt an, ob Sie vertrauliche Informationen aus dem Protokoll entfernt haben. Stellen Sie das Output Feld auf „Raw“ ein, wenn das Transkript vertrauliche Informationen enthält.
- CustomerMetadata Eine eindeutige Kennung für die Konversation.
- Transkript Der Text der Konversation zwischen den Gesprächspartnern. Jede Runde der Konversation wird mit einer eindeutigen Kennung identifiziert.

Format des Ausgabeprotokolls

Das Ausgabe-Transkriptformat entspricht fast dem Eingabe-Transkriptformat. Es enthält jedoch auch einige Kundenmetadaten und ein Feld mit Segmenten, die den Vorschlag von Absichten und Slot-Typen beeinflusst haben. Sie können das Ausgabetranskript von der Überprüfungsseite in der Konsole oder mithilfe der Amazon Lex V2-API herunterladen. Weitere Informationen finden Sie unter [Format des Eingabeprotokolls](#).

```

{
  "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**— Dem CustomerMetadata Feld werden zwei Felder hinzugefügt, der Name der Eingabedatei, die die Konversation enthält, und das Eingabeformat, das immer „Lex“ ist.
- **InfluencingSegments**— Identifiziert die Segmente der Konversation, die den Vorschlag einer Absicht oder eines Slot-Typs beeinflusst haben. Die ID der Absicht oder des Slot-Typs identifiziert die spezifische Absicht, die von der Konversation beeinflusst wurde.

Hinzufügen einer neuen Sprache zu einem Amazon Lex V2-Bot

Sie fügen Ihrem Bot eine oder mehrere Sprachen und Gebietsschemas hinzu, damit er mit Benutzern in deren Sprache kommunizieren kann. Sie definieren die Intentionen, Slots und Slot-Typen für jede Sprache separat, sodass die Äußerungen, Aufforderungen und Slot-Werte sprachspezifisch sind.

Ihr Bot muss mindestens eine Sprache enthalten.

Um deinem Bot eine Sprache hinzuzufügen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie im Bereich Bots den Bot aus, dem Sie eine Sprache hinzufügen möchten.
3. Klicken Sie im Abschnitt Sprachen hinzufügen auf Sprachen anzeigen.
4. Klicken Sie im Abschnitt Alle Sprachen auf Sprache hinzufügen.
5. Wählen Sie im Abschnitt Neue Sprache hinzufügen die Option Sprache von Grund auf hinzufügen aus.
6. Wählen Sie im Abschnitt Sprachdetails die Sprache aus, die Sie hinzufügen möchten.
7. Wenn Ihr Bot Sprachinteraktion unterstützt, wählen Sie im Bereich Sprache die Amazon Polly-Stimme aus, die Amazon Lex V2 für die Kommunikation mit dem Benutzer verwendet. Wenn Ihr Bot Sprache nicht unterstützt, wählen Sie Keine.

8. Legen Sie im Abschnitt Schwellenwert für die Vertrauensbewertung der Klassifizierung den Wert fest, anhand dessen Amazon Lex V2 bestimmt, ob eine Absicht richtig ist. Sie können diesen Wert anpassen, nachdem Sie Ihren Bot getestet haben.
9. Wählen Sie Hinzufügen aus.

Absichten hinzufügen

Absichten sind die Ziele, die Ihre Nutzer erreichen möchten, wie zum Beispiel Blumen zu bestellen oder ein Hotel zu buchen. Ihr Bot muss mindestens eine Absicht haben.

Standardmäßig enthalten alle Bots eine einzige integrierte Absicht, die Fallback-Absicht. Diese Absicht wird verwendet, wenn Amazon Lex V2 keine andere Absicht erkennt. Wenn ein Benutzer beispielsweise zu einer Hotelbuchungsabsicht „Ich möchte Blumen bestellen“ sagt, wird die alternative Absicht ausgelöst.

Um eine Absicht hinzuzufügen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Bot aus, dem Sie die Absicht hinzufügen möchten, und wählen Sie dann unter Sprachen hinzufügen die Option Sprachen anzeigen aus.
3. Wählen Sie die Sprache aus, zu der Sie die Absicht hinzufügen möchten, und wählen Sie dann Absichten aus.
4. Wählen Sie „Absicht hinzufügen“, geben Sie Ihrer Absicht einen Namen und wählen Sie dann „Hinzufügen“.
5. Fügen Sie im Absichtseditor die Details Ihrer Absicht hinzu.
 - Konversationsablauf — Verwenden Sie das Konversationsflussdiagramm, um zu sehen, wie ein Dialog mit Ihrem Bot aussehen könnte. Sie können verschiedene Abschnitte der Konversation auswählen, um zu diesem Abschnitt des Absichtseditors zu springen.
 - Einzelheiten zur Absicht — Geben Sie der Absicht einen Namen und eine Beschreibung, um den Zweck der Absicht leichter zu erkennen. Sie können auch die eindeutige Kennung sehen, die Amazon Lex V2 der Absicht zugewiesen hat.
 - Kontexte — Legen Sie die Eingabe- und Ausgabekontexte für die Absicht fest. Ein Kontext ist eine Zustandsvariable, die einer Absicht zugeordnet ist. Ein Ausgabekontext wird festgelegt,

wenn eine Absicht erfüllt ist. Eine Absicht mit einem Eingabekontext kann nur erkannt werden, wenn der Kontext aktiv ist. Eine Absicht ohne Eingabekontexte kann immer erkannt werden.

- **Beispielaussagen** — Sie sollten 10 oder mehr Ausdrücke angeben, von denen Sie erwarten, dass Ihre Benutzer sie verwenden, um eine Absicht auszulösen. Amazon Lex V2 verallgemeinert anhand dieser Ausdrücke, um zu erkennen, dass der Benutzer die Absicht initiieren möchte.
- **Erste Antwort** — Die erste Nachricht, die an den Benutzer gesendet wird, nachdem die Absicht aufgerufen wurde. Sie können Antworten geben, Werte initialisieren und den nächsten Schritt definieren, den Amazon Lex V2 unternimmt, um dem Benutzer zu Beginn der Absicht zu antworten.
- **Slots** — Definieren Sie die Slots oder Parameter, die zur Erfüllung der Absicht erforderlich sind. Jeder Slot hat einen Typ, der die Werte definiert, die in den Slot eingegeben werden können. Sie können aus Ihren benutzerdefinierten Slot-Typen wählen, oder Sie können einen integrierten Slot-Typ wählen.
- **Bestätigung** — Diese Aufforderungen und Antworten werden verwendet, um die Erfüllung der Absicht zu bestätigen oder abzulehnen. In der Bestätigungsaufforderung wird der Benutzer aufgefordert, die Slot-Werte zu überprüfen. Zum Beispiel: „Ich habe für Freitag ein Hotelzimmer gebucht. Ist das richtig?“ Die Ablehnungsantwort wird an den Benutzer gesendet, wenn er die Bestätigung ablehnt. Sie können Antworten geben, Werte festlegen und den nächsten Schritt definieren, den Amazon Lex V2 entsprechend einer Bestätigungs- oder Ablehnungsantwort des Benutzers unternimmt.
- **Erfüllung** — Antwort, die dem Benutzer während des Versandvorgangs gesendet wird. Sie können festlegen, dass der Versandstatus zu Beginn der Auftragsabwicklung und in regelmäßigen Abständen während der Ausführung aktualisiert wird. Zum Beispiel „Ich ändere Ihr Passwort, das kann ein paar Minuten dauern“ und „Ich arbeite noch an Ihrer Anfrage“. Fulfillment-Updates werden nur für Streaming-Konversationen verwendet. Sie können auch eine Erfolgsmeldung nach dem Versand, eine Fehlernachricht und eine Timeout-Meldung einrichten. Sie können Nachrichten nach der Auftragsabwicklung sowohl für Streaming-Konversationen als auch für normale Konversationen senden. Wenn der Versand erfolgreich ist, kannst du beispielsweise „Ich habe dein Passwort geändert“ senden. Wenn der Versand nicht erfolgreich ist, können Sie eine Antwort mit weiteren Informationen senden, z. B. „Ich konnte Ihr Passwort nicht ändern. Wenden Sie sich an den Helpdesk, um Unterstützung zu erhalten.“ Wenn der Versand länger dauert als der konfigurierte Timeout-Zeitraum, können Sie dem Benutzer eine Nachricht senden, z. B. „Unsere Server sind derzeit sehr ausgelastet. Versuchen Sie es später noch einmal mit Ihrer Anfrage.“ Sie können Antworten geben, Werte

festlegen und den nächsten Schritt definieren, den Amazon Lex V2 unternimmt, um dem Benutzer zu antworten.

- **Abschlussantworten** — Antwort, die an den Benutzer gesendet wird, nachdem die Absicht erfüllt wurde und alle anderen Nachrichten abgespielt wurden. Zum Beispiel ein Dankeschön für die Buchung eines Hotelzimmers. Oder es kann den Benutzer auffordern, eine andere Absicht zu starten, wie zum Beispiel: „Danke, dass Sie ein Zimmer gebucht haben. Möchten Sie einen Mietwagen buchen?“ Sie können Antworten geben und die nächsten Folgeaktionen konfigurieren, nachdem Sie die Absicht erfüllt und mit der abschließenden Antwort geantwortet haben.
- **Code-Hooks** — Geben Sie an, ob Sie eine AWS Lambda Funktion verwenden, um die Absicht zu initialisieren und Benutzereingaben zu validieren. Sie geben die Lambda-Funktion in dem Alias an, mit dem Sie den Bot ausführen.

6. Wählen Sie Absicht speichern, um die Absicht zu speichern.

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Konfiguration von Eingabeaufforderungen in einer bestimmten Reihenfolge

Sie können den Bot so konfigurieren, dass er Nachrichten in einer vordefinierten Reihenfolge abspielt, indem Sie das Kontrollkästchen Nachrichten der Reihe nach abspielen aktivieren. Andernfalls spielt der Bot die Nachricht und die Variationen in zufälliger Reihenfolge ab.

Durch geordnete Eingabeaufforderungen können die Nachricht und die Varianten einer Nachrichtengruppe in der Reihenfolge der Wiederholungen abgespielt werden. Sie können eine Nachricht alternativ umformulieren, wenn der Benutzer eine ungültige Antwort auf die Aufforderung gegeben hat, oder um die Absicht zu bestätigen. In jedem Slot können bis zu zwei Varianten der

ursprünglichen Nachricht gespeichert werden. Sie können wählen, ob Sie die Nachrichten der Reihe nach oder nach dem Zufallsprinzip abspielen möchten.

Ordered Prompt unterstützt alle vier Nachrichtentypen: Text, benutzerdefinierte Payload-Antwort, SSML und Kartengruppe. Die Antworten werden innerhalb derselben Nachrichtengruppe angeordnet. Verschiedene Nachrichtengruppen sind unabhängig.

Themen

- [Beispiele für Äußerungen](#)
- [Absicht-Struktur](#)
- [Gesprächspfade erstellen](#)
- [Verwenden von Visual Conversation Builder](#)
- [Integrierte Absichten](#)

Beispiele für Äußerungen

Sie erstellen Beispieläußerungen, bei denen es sich um Varianten von Ausdrücken handelt, von denen Sie erwarten, dass Benutzer sie verwenden, um eine Absicht zu wecken. Für eine **BookFlight** Absicht könnten Sie beispielsweise Äußerungen wie die folgenden einbeziehen:

1. Ich möchte einen Flug buchen
2. hilf mir, einen Flug zu bekommen.
3. Flugtickets, bitte!
4. Flug von *{DepartureCity}* nach *{DestinationCity}*

Sie sollten 10 oder mehr Beispieläußerungen angeben. Geben Sie Stichproben an, die eine Vielzahl von Satzstrukturen und Wörtern repräsentieren, die Benutzer möglicherweise aussprechen. Denken Sie auch an unvollständige Sätze, wie in den Beispielen 3 und 4 oben. Sie können auch Slots verwenden, die Sie für die Absicht in einer Beispieläußerung definiert haben, indem Sie den Slot-Namen in geschweifte Klammern setzen, wie in *{DepartureCity}* in Beispiel 4. Wenn Sie Slot-Namen in eine Beispieläußerung aufnehmen, füllt Amazon Lex V2 die Slots der Absicht mit den Werten, die der Benutzer in der Äußerung angibt.

Eine Vielzahl von Beispieläußerungen hilft Amazon Lex V2 bei der Generalisierung, sodass effektiv erkannt wird, dass der Benutzer die Absicht initiieren möchte.

Sie können Beispieläußerungen im Absichtseditor, im Visual Conversation Builder oder mit den [CreateIntent](#) API-Operationen oder hinzufügen. [UpdateIntent](#) Sie können auch automatisch Beispieläußerungen generieren, indem Sie die generativen KI-Funktionen von Amazon Bedrock nutzen. Weitere Informationen finden Sie unter [Verwenden Sie die Generierung von Äußerungen, um Musteräußerungen zur Absichtserkennung zu generieren](#).

Verwenden Sie den Intent-Editor oder den Visual Conversation Builder

1. Navigieren Sie im Intent-Editor zum Abschnitt Beispieläußerungen. Suchen Sie im Visual Conversation Builder im Startblock nach dem Abschnitt Beispieläußerungen.
2. Geben Sie in das Feld mit dem transparenten Text **I want to book a flight** eine Beispieläußerung ein. Wählen Sie Äußerung hinzufügen aus, um die Äußerung hinzuzufügen.
3. Sehen Sie sich die Beispieläußerungen an, die Sie entweder im Vorschaumodus oder im Nur-Text-Modus hinzugefügt haben. Im Nur-Text-Modus ist jede Zeile eine separate Äußerung. Bewegen Sie im Vorschaumodus den Mauszeiger über eine Äußerung, um die folgenden Optionen einzublenden:
 - Wählen Sie das Textfeld aus, um die Äußerung zu bearbeiten.
 - Wählen Sie die X-Schaltfläche rechts neben dem Textfeld, um die Äußerung zu löschen.
 - Ziehen Sie die Schaltfläche links neben dem Textfeld, um die Reihenfolge der Beispieläußerungen zu ändern.
4. Verwenden Sie die Suchleiste oben, um Ihre Beispieläußerungen zu durchsuchen, und das Dropdownmenü daneben, um nach der Reihenfolge zu sortieren, in der Sie die Äußerungen hinzugefügt haben, oder in alphabetischer Reihenfolge.

Verwenden Sie eine API-Operation

1. Erstellen Sie eine neue Absicht mit der [CreateIntent](#) Operation oder aktualisieren Sie eine bestehende Absicht mit der [UpdateIntent](#) Operation.
2. Die API-Anfrage enthält ein `sampleUtterances` Feld, das einem Array von [SampleUtterance](#) Objekten zugeordnet ist.
3. Fügen Sie für jede Beispieläußerung, die Sie hinzufügen möchten, ein `SampleUtterance` Objekt an das Array an. Fügen Sie die Beispieläußerung als Wert des Felds hinzu. `utterance`
4. Senden Sie eine Anfrage, um Beispieläußerungen zu bearbeiten und zu löschen. `UpdateIntent` Die Liste der Äußerungen, die Sie in dem `sampleUtterances` Feld angeben, ersetzt die vorhandenen Äußerungen.

Important

Jedes Feld, das Sie in der UpdateIntent Anfrage leer lassen, führt dazu, dass bestehende Konfigurationen in der Absicht gelöscht werden. Verwenden Sie den [DescribeIntent](#) Vorgang, um die Bot-Konfiguration zurückzugeben und alle Konfigurationen, die nicht gelöscht werden sollen, in die UpdateIntent Anfrage zu kopieren.

Absicht-Struktur

Absichten sind die Ziele, die Ihre Benutzer erreichen möchten, z. B. Blumen bestellen oder ein Hotel buchen. Ihr Bot muss mindestens eine Absicht haben. Eine Absicht besteht aus den folgenden Komponenten

- **Erste Antwort** — Die erste Nachricht, die an den Benutzer gesendet wird, nachdem die Absicht aufgerufen wurde. Sie können Antworten festlegen, Werte initialisieren und den nächsten Schritt definieren, den Ihr Bot unternimmt, um dem Benutzer zu Beginn der Absicht zu antworten.
- **Slots** — Die Parameter, die zur Erfüllung einer Absicht erforderlich sind. Jeder Slot hat einen Typ, der die Werte definiert, die in den Slot eingegeben werden können. Sie können aus Ihren benutzerdefinierten Slot-Typen wählen oder Sie können einen integrierten Slot-Typ wählen.
- **Bestätigung** — Nachdem die Konversation mit dem Benutzer abgeschlossen ist und die Slot-Werte für die Absicht gefüllt sind, können Sie eine Bestätigungsaufforderung einrichten, um den Benutzer zu fragen, ob die Slot-Werte korrekt sind.
- **Erfüllung** — Die Antwort, die während der Ausführung an einen Benutzer gesendet wird. Sie können zu Beginn des Fulfillments Updates zum Status des Fulfillments einrichten und während der Ausführung weiterhin regelmäßig Updates senden. Sie können auch eine Erfolgsmeldung nach dem Versand, eine Fehlermeldung und eine Timeout-Meldung einrichten.
- **Abschlussantwort** — Die abschließende Antwort, die an den Benutzer gesendet wird, nachdem seine Absicht erfüllt wurde. Sie können die abschließende Antwort so einstellen, dass die Konversation beendet wird, oder Sie können sie so einstellen, dass der Benutzer weiß, dass er mit einer anderen Absicht fortfahren kann.

Themen

- [Erste Antwort](#)

- [Slots](#)
- [Bestätigung](#)
- [Bereitstellung](#)
- [Antwort schließen](#)

Erste Antwort

Die erste Antwort wird an den Benutzer gesendet, nachdem Amazon Lex V2 die Absicht ermittelt hat und bevor es beginnt, Slot-Werte zu ermitteln. Sie können diese Antwort verwenden, um den Benutzer über die erkannte Absicht zu informieren und ihn auf die Informationen vorzubereiten, die Sie zur Erfüllung der Absicht sammeln.

Wenn Sie beispielsweise beabsichtigen, einen Wartungstermin für ein Auto zu vereinbaren, könnte die erste Reaktion wie folgt lauten:

Ich kann Ihnen helfen, einen Termin zu vereinbaren. Sie müssen die Marke, das Modell und das Jahr Ihres Autos angeben.

Eine erste Antwortnachricht ist nicht erforderlich. Wenn Sie keine angeben, folgt Amazon Lex V2 weiterhin dem nächsten Schritt der ersten Antwort.

In der ersten Antwort können Sie die folgenden Optionen konfigurieren:

- Nächsten Schritt konfigurieren — Sie können den nächsten Schritt in der Konversation angeben, z. B. zu einer bestimmten Dialogaktion springen, einen bestimmten Slot auslösen oder zu einer anderen Absicht springen. Weitere Informationen finden Sie unter [Konfigurieren Sie die nächsten Schritte in der Konversation](#).
- Werte festlegen — Sie können Werte für Slots und Sitzungsattribute festlegen. Weitere Informationen finden Sie unter [Legen Sie Werte während der Konversation fest](#).
- Bedingte Verzweigung hinzufügen — Sie können Bedingungen anwenden, nachdem Sie die erste Antwort abgespielt haben. Wenn eine Bedingung als wahr bewertet wird, werden die von Ihnen definierten Aktionen ausgeführt. Weitere Informationen finden Sie unter [Fügen Sie Bedingungen zu Konversationen in Filialen hinzu](#).
- Dialogcode-Hook ausführen — Sie können einen Lambda-Code-Hook definieren, um Daten zu initialisieren und Geschäftslogik auszuführen. Weitere Informationen finden Sie unter [Rufen Sie](#)

[den Code-Hook des Dialogs auf](#). Wenn die Option zum Ausführen der Lambda-Funktion für den Intent aktiviert ist, wird der Dialogcode-Hook standardmäßig ausgeführt. Sie können den Dialogcode-Hook deaktivieren, indem Sie auf die Schaltfläche Aktiv klicken.

In Ermangelung einer Bedingung oder eines ausdrücklichen nächsten Schritts wechselt Amazon Lex V2 in Prioritätsreihenfolge zum nächsten Slot.

User request acknowledgement Info

You can provide messages to acknowledge a user's request. You can provide responses, set values, and next steps. You can also branch based on conditions.

▼ Response for acknowledging the user's request

Message: -

Message - optional

Okay, I can help you with that

► Variations - optional

More response options

Add custom payloads, SSML, and card groups.

► Set values

-

Next step in conversation

Execute dialog code hook

+

 Add conditional branching

Dialog code hook Info

You can enable Lambda functions to manage initialize the conversation.

► Lambda dialog code hook

Invoke Lambda for user request validation: Yes

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen

mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Slots

Slots sind Werte, die vom Benutzer bereitgestellt werden, um die Absicht zu erfüllen. Es gibt zwei Arten von Steckplätzen:

- **Eingebauter Steckplatztyp** — Sie können integrierte Steckplatztypen verwenden, um Standardwerte wie Nummer, Name und Stadt zu erfassen. Eine Liste der unterstützten integrierten Steckplatztypen finden Sie unter [Integrierte Steckplatztypen](#).
- **Benutzerdefinierter Slot-Typ** — Sie können benutzerdefinierte Slot-Typen verwenden, um benutzerdefinierte Werte zu erfassen, die für die jeweilige Absicht spezifisch sind. Sie können beispielsweise einen benutzerdefinierten Slot-Typ verwenden, um den Kontotyp „Girokonto“ oder „Sparkonto“ zu erfassen. Weitere Informationen finden Sie unter [Benutzerdefinierter Slot-Typ](#).

Um einen Slot in einer Absicht zu definieren, müssen Sie Folgendes konfigurieren:

- **Slot-Info** — Dieses Feld enthält einen Namen und eine optionale Beschreibung für den Slot. Sie können zum Beispiel den Slot-Namen als „AccountNumber“ angeben, um Kontonummern zu erfassen. Wenn der Slot als Teil des Konversationsablaufs zur Erfüllung der Absicht erforderlich ist, muss er als erforderlich gekennzeichnet werden.
- **Slot-Typ** — Ein Slot-Typ definiert die Liste von Werten, die ein Slot akzeptieren kann. Sie können einen benutzerdefinierten Slot-Typ erstellen oder einen vordefinierten Slot-Typ verwenden.
- **Slot-Prompt** — Ein Slot-Prompt ist eine Frage, die dem Benutzer gestellt wird, um Informationen zu sammeln. Sie können die Anzahl der Wiederholungen, die zum Sammeln von Informationen verwendet werden, und die Variation der Eingabeaufforderung, die für jede Wiederholung verwendet wird, konfigurieren. Sie können auch nach jedem erneuten Versuch einen Lambda-Funktionsaufruf aktivieren, um die erfasste Eingabe zu verarbeiten und zu versuchen, zu einer gültigen Eingabe aufzulösen.
- **Warten und fortfahren (optional)** — Wenn Sie dieses Verhalten aktivieren, können Benutzer Formulierungen wie „Moment mal“ sagen, sodass der Bot darauf wartet, dass sie die Informationen

finden und bereitstellen. Dies ist nur für Streaming-Konversationen aktiviert. Weitere Informationen finden Sie unter [Den Amazon Lex V2-Bot so aktivieren, dass er während einer Pause darauf wartet, dass der Benutzer weitere Informationen bereitstellt](#).

- Antworten auf Slot-Capture — Sie können je nach Ergebnis der Erfassung des Slot-Werts anhand von Benutzereingaben eine Erfolgs- und eine Fehlerantwort konfigurieren.
- Bedingte Verzweigung — Sie können Bedingungen anwenden, nachdem Sie die erste Antwort abgespielt haben. Wenn eine Bedingung als wahr bewertet wird, werden die von Ihnen definierten Aktionen ausgeführt. Weitere Informationen finden Sie unter [Fügen Sie Bedingungen zu Konversationen in Filialen hinzu](#).
- Dialog-Code-Hook — Sie können auch einen Lambda-Code-Hook verwenden, um die Slot-Werte zu validieren und Geschäftslogik auszuführen. Weitere Informationen finden Sie unter [Rufen Sie den Code-Hook des Dialogs auf](#).
- Benutzereingabetyp — Sie können den Eingabetyp so konfigurieren, dass der Bot eine bestimmte Modalität akzeptiert. Standardmäßig werden sowohl Audio- als auch DTMF-Modalitäten akzeptiert. Sie können ihn wahlweise auf Nur Audio oder Nur DTMF einstellen.
- Timeouts und Längen für Audioeingänge — Sie können Audio-Timeouts wie Sprach-Timeout und Timeout für Stille konfigurieren. Sie können auch die maximale Audiolänge festlegen.
- Timeout, Zeichen und Längen bei der DTMF-Eingabe — Sie können das DTMF-Timeout zusammen mit dem Löschrzeichen und dem Endzeichen festlegen. Sie können auch die maximale DTMF-Länge festlegen.
- Textlänge — Sie können die maximale Länge für die Textmodalität festlegen.

Nachdem die Slot-Eingabeaufforderung abgespielt wurde, gibt der Benutzer den Slot-Wert als Eingabe ein. Wenn Amazon Lex V2 einen vom Benutzer angegebenen Slot-Wert nicht versteht, versucht es erneut, den Slot auszulösen, bis es einen Wert versteht oder bis er die maximale Anzahl von Wiederholungen überschreitet, die Sie für den Slot konfiguriert haben. Mithilfe der erweiterten Einstellungen für Wiederholungsversuche können Sie die Timeouts konfigurieren, die Art der Eingabe einschränken und die Unterbrechung für die erste Aufforderung und Wiederholungsversuche aktivieren oder deaktivieren. Nach jedem Versuch, die Eingabe zu erfassen, kann Amazon Lex V2 die für den Bot konfigurierte Lambda-Funktion mit einem Aufruf-Label aufrufen, das für Wiederholungsversuche bereitgestellt wird. Sie können die Lambda-Funktion beispielsweise verwenden, um Ihre Geschäftslogik anzuwenden und zu versuchen, sie in einen gültigen Wert aufzulösen. Diese Lambda-Funktion kann in den erweiterten Optionen für Slot-Prompts aktiviert werden.

Slot prompts [Info](#)

Prompts to elicit the slot.

► Bot elicits information

Message: What is your account number?

Sie können Antworten definieren, die der Bot an den Benutzer senden soll, sobald der Slot-Wert eingegeben wurde oder wenn die maximale Anzahl von Wiederholungen überschritten wurde. Beispielsweise können Sie bei einem Bot zur Serviceplanung für ein Auto eine Nachricht an den Benutzer senden, wenn die Fahrzeugidentifikationsnummer (VIN) eingegeben wird:

Vielen Dank für die Angabe der Fahrgestellnummer Ihres Fahrzeugs. Ich werde jetzt einen Termin vereinbaren.

Sie können zwei Antworten erstellen:

- Erfolgsantwort — gesendet, wenn Amazon Lex V2 einen Slot-Wert versteht.
- Fehlerantwort — wird gesendet, wenn Amazon Lex V2 nach der maximalen Anzahl von Wiederholungen einen Slot-Wert vom Benutzer nicht verstehen kann.

Sie können Werte festlegen, die nächsten Schritte konfigurieren und Bedingungen anwenden, die jeder Antwort entsprechen, um den Gesprächsablauf zu gestalten.

In Ermangelung einer Bedingung oder eines ausdrücklichen nächsten Schritts wechselt Amazon Lex V2 in Prioritätsreihenfolge zum nächsten Slot.

Slot capture: success response [Info](#)

You can provide responses, set values, and next steps. You can also branch based on conditions.

► Response when user provides slot value

Message: -

► Set values

-

Next step in conversation

Elicit a slot

[+ Add conditional branching](#)

Slot capture: failure response [Info](#)

You can provide responses, set values, and next steps. You can also branch based on conditions.

► Response when slot value isn't understood

Message: -

► Set values

-

Next step in conversation

Switch to intent: *FallbackIntent*

[+ Add conditional branching](#)

Sie können eine Lambda-Funktion verwenden, um einen Slot-Wert, den ein Benutzer eingegeben hat, zu überprüfen und zu bestimmen, was die nächste Aktion sein soll. Sie können beispielsweise die Validierungsfunktion verwenden, um sicherzustellen, dass der eingegebene Wert in den richtigen Bereich fällt oder dass er korrekt formatiert ist. Um die Lambda-Funktion zu aktivieren, aktivieren Sie das Kontrollkästchen Lambda-Funktion aufrufen und klicken Sie im Abschnitt Dialog-Code-Hook auf die Schaltfläche Aktiv. Sie können ein Aufruf-Label für den Dialog-Code-Hook angeben. Dieses Aufrufetikett kann in der Lambda-Funktion verwendet werden, um die Geschäftslogik zu schreiben, die der Slot-Auslösung entspricht.

Dialog code hook [Info](#) Active

You can enable Lambda functions to validate user input.

▼ **Lambda dialog code hook**
Invoke Lambda for user request validation: Yes

☒ **Invoke Lambda for user request validation**

Advanced options

Configure dialog code hook success, failure and timeout responses.

Slots, die für die Absicht nicht erforderlich sind, sind nicht Teil des Hauptkonversationsablaufs. Wenn eine Benutzeräußerung jedoch einen Wert enthält, den Ihr Bot als einem optionalen Slot entsprechend identifiziert, kann er den Slot mit diesem Wert füllen. Wenn Sie beispielsweise einen Business Intelligence-Bot so konfigurieren, dass er über einen optionalen City Slot und die Benutzeräußerung verfügt **What is the sales for April in San Diego?**, füllt der Bot den optionalen Slot mit **San Diego**. Sie können die Geschäftslogik so konfigurieren, dass sie den optionalen Slot-Wert verwendet, falls vorhanden.

Slots, die für die Absicht nicht erforderlich sind, können mit den nächsten Schritten nicht ermittelt werden. Diese Schritte können nur während der Absichtsermittlung (wie im vorherigen Beispiel) ausgefüllt werden oder können durch Einstellen des Dialogstatus in der Lambda-Funktion ausgelöst werden. Wenn der Slot mithilfe der Lambda-Funktion ausgelöst wird, müssen Sie die Lambda-Funktion verwenden, um nach Abschluss der Slot-Abfrage über den nächsten Schritt in der Konversation zu entscheiden. Um die Unterstützung für den nächsten Schritt bei der Erstellung des Bots zu aktivieren, müssen Sie den Slot als für die Absicht erforderlich markieren.

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

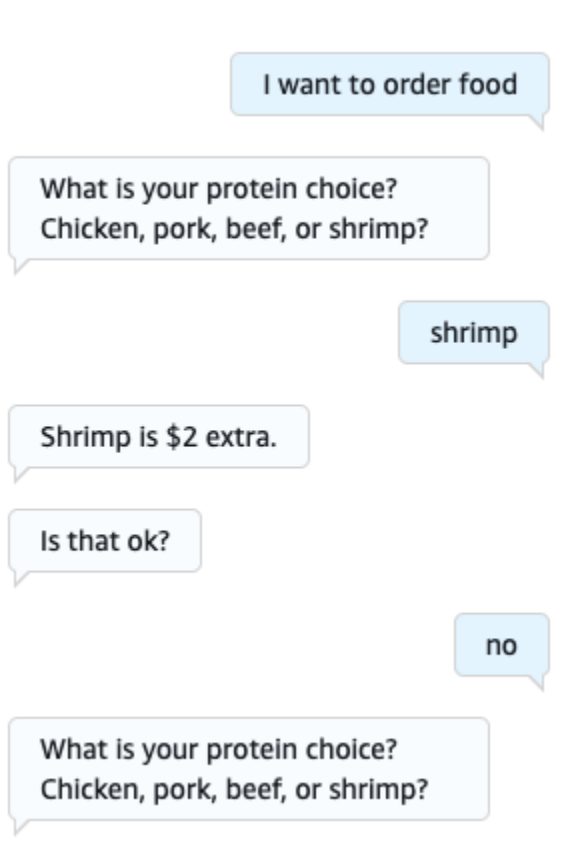
In den folgenden Themen wird beschrieben, wie Sie einen Bot so konfigurieren, dass er einen bereits gefüllten Slot-Wert erneut auslöst, und wie Sie einen Slot erstellen, der aus mehreren Werten besteht:

Themen

- [Wiedererweckende Slots](#)
- [Verwendung mehrerer Werte in einem Slot](#)

Wiedererweckende Slots

Sie können Ihren Bot so konfigurieren, dass er einen bereits gefüllten Slot erneut auslöst, indem Sie diesen Slot-Wert auf setzen **null** und im nächsten Schritt in der Konversation festlegen, dass dieser Slot erneut abgerufen wird. Möglicherweise möchten Sie einen Slot erneut abrufen, nachdem Ihr Kunde eine Bestätigung des Slot-Zugangs auf der Grundlage zusätzlicher Informationen abgelehnt hat, wie in der folgenden Konversation:



Sie können eine Schleife von der Bestätigungsantwort zurück konfigurieren, um den Slot entweder mit dem Intent-Editor oder dem erneut abzurufen. [Verwenden von Visual Conversation Builder](#)

 Note

Sie können zu einem beliebigen Zeitpunkt in der Konversation eine Schleife zurückschalten, um einen Slot erneut auszulösen, vorausgesetzt, Sie haben diesen Slot-Wert zuvor auf festgelegt. **null**

Reproduzieren Sie das obige Beispiel mit dem Intent-Editor

1. Wählen Sie im Bereich Bestätigung des Absichtseditors den Rechtspfeil neben Eingabeaufforderungen aus, um die Absicht zu bestätigen, den Bereich zu erweitern.
2. Wählen Sie unten Erweiterte Optionen aus.
3. Wählen Sie im Abschnitt Antwort ablehnen den Rechtspfeil neben Werte festlegen aus, um den Bereich zu erweitern. Füllen Sie diesen Abschnitt wie in der Abbildung unten mit den folgenden Schritten aus:
 - a. Stellen Sie den Slot-Wert ein, auf den Sie erneut abrufen möchten. **null** In diesem Beispiel wollen wir den Slot erneut ermitteln, also geben **{Meat} = null** wir ihn im **Meat** Abschnitt Slot-Werte ein.
 - b. Wählen Sie im Drop-down-Menü unter Nächster Gesprächsschritt die Option Slot auswählen aus.
 - c. Ein Slot-Bereich wird angezeigt. Wählen Sie im Drop-down-Menü darunter den Slot aus, den Sie erneut abrufen möchten.
 - d. Wählen Sie Aktualisierungsoptionen aus, um Ihre Änderungen zu bestätigen.

Decline response [Info](#)

When the user declines an intent, these are the responses Amazon Lex uses.

► Bot confirms cancellation

Message: -

▼ Set values

`{Meat} = null`

Next step in conversation

Elicit a slot

Slot values - *optional*

Add slot values as: `{slot} = "value"`

`{Meat} = null`

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

Elicit a slot ▼

Slot

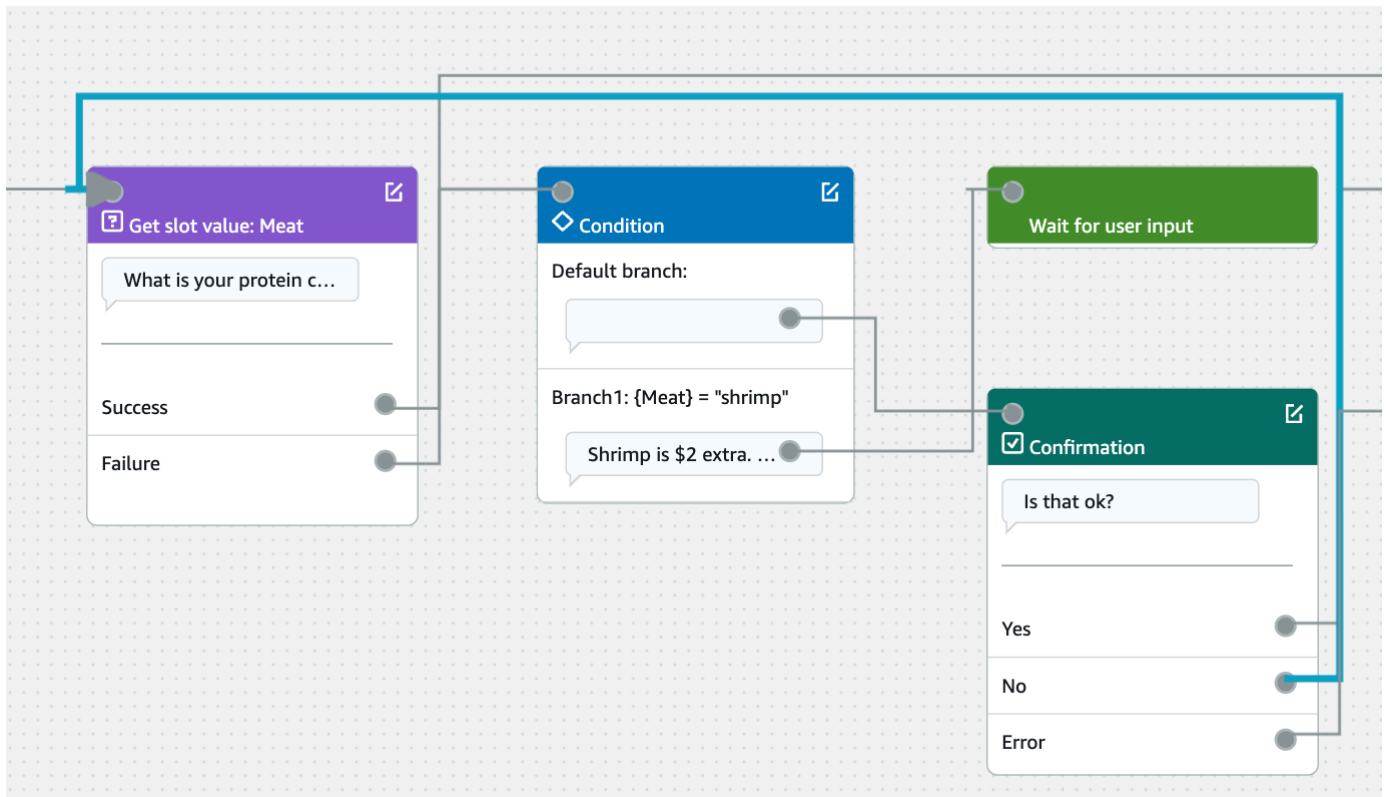
Meat ▼

☐ Skip elicitation prompt

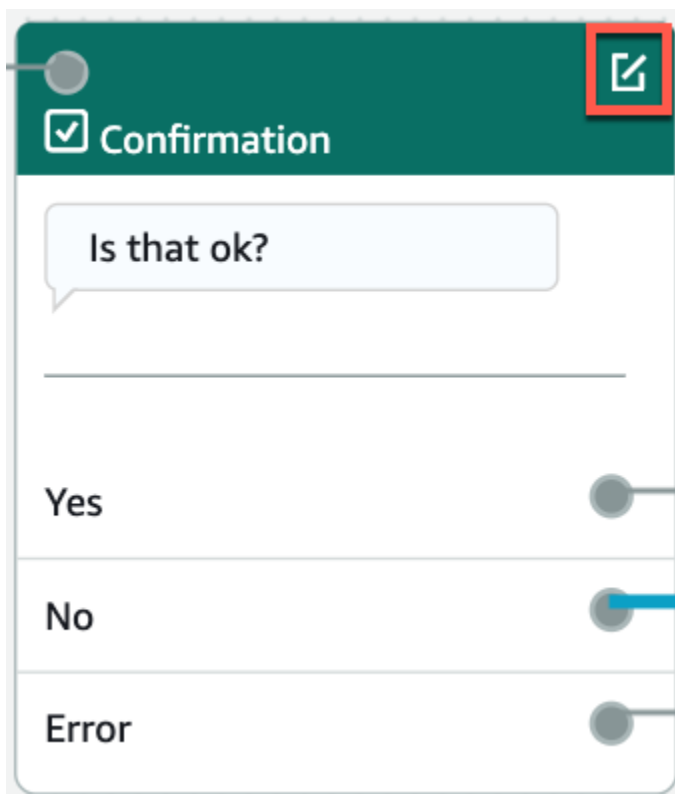
[+ Add conditional branching](#)

Reproduzieren des obigen Beispiels mit dem Visual Conversation Builder

1. Stellen Sie eine Verbindung vom No-Port des Bestätigungsblocks zum Eingangsport des Blocks Get slot value: Meat her.



2. Wählen Sie das Bearbeiten-Symbol in der oberen rechten Ecke des Bestätigungsblocks aus.



3. Wählen Sie im Abschnitt Antwort ablehnen das Zahnradsymbol neben der Bot-Antwort aus.

Confirmation [Info](#) Active ×

Confirmation prompt
Message to ask user to confirm this intent.
 

Confirmation response: Yes - optional [Info](#)
Bot response when user confirms this intent.
 

Decline response: No - optional [Info](#)
Bot response when user declines.
 

Failure response: Error - optional [Info](#)
Bot response when user response failed to be captured.
 

4. Fügen Sie im Bereich Werte festlegen „{Meat} = null“ in das Feld Slot-Werte ein.

< **Decline response** Info ×

▼ **Response advanced settings**

☒ **Users can interrupt the response when it is being read**

This functionality is available only in streaming conversations.

► **Define response**

▼ **Set values**

Slot values - *optional*

Add slot values as: {slot} = "value"

`{Meat} = null`

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.

5. Wählen Sie Save Intent aus.

Verwendung mehrerer Werte in einem Slot

Note

Felder mit mehreren Werten werden nur in englischer Sprache (USA) unterstützt.

In einigen Fällen möchten Sie möglicherweise mehrere Werte für einen einzelnen Slot erfassen. Ein Bot, der Pizza bestellt, könnte beispielsweise eine Absicht mit der folgenden Äußerung haben:

I want a pizza with {toppings}

Die Absicht geht davon aus, dass der {toppings} Slot eine Liste der Beläge enthält, die der Kunde auf seiner Pizza haben möchte, zum Beispiel „Peperoni und Ananas“.

Um einen Slot so zu konfigurieren, dass er mehrere Werte erfasst, setzen Sie das `allowMultipleValues` Feld auf dem Slot auf `true`. Sie können das Feld mit der Konsole oder mit der [UpdateSlot](#) Operation [CreateSlot](#) oder festlegen.

Sie können nur Steckplätze mit benutzerdefinierten Slot-Typen als Slots mit mehreren Werten markieren.

Für einen Slot mit mehreren Werten gibt Amazon Lex V2 in der Antwort auf die [RecognizeUtterance](#) Operation [RecognizeText](#) oder eine Liste von Slot-Werten zurück. Im Folgenden finden Sie die Slot-Informationen, die für die Äußerung „Ich möchte eine Pizza mit Peperoni und Ananas“ vom Bot zurückgegeben wurden. `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"
    ]
  }
}
```

Slots mit mehreren Werten geben immer eine Liste von Werten zurück. Wenn die Äußerung nur einen Wert enthält, enthält die Liste der zurückgegebenen Werte nur eine Antwort.

Amazon Lex V2 erkennt mehrere Werte, die durch Leerzeichen, Kommas (,) und die Konjunktion „und“ getrennt sind. Slots mit mehreren Werten funktionieren sowohl bei Text- als auch bei Spracheingabe.

Sie können mehrwertige Slots in Eingabeaufforderungen verwenden. Sie können die Bestätigungsaufforderung für eine Absicht beispielsweise wie folgt einrichten

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

Wenn Amazon Lex V2 die Aufforderung an den Benutzer sendet, wird die Meldung „Möchten Sie, dass ich Ihre Pizza mit Peperoni und Ananas bestelle?“ gesendet.

Slots mit mehreren Werten unterstützen einzelne Standardwerte. Wenn mehrere Standardwerte angegeben werden, füllt Amazon Lex V2 den Steckplatz nur mit dem ersten verfügbaren Wert. Weitere Informationen finden Sie unter [Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot](#).

Sie können Slot-Obfuscation verwenden, um die Werte eines Slots mit mehreren Werten in Konversationsprotokollen zu maskieren. Wenn Sie Slot-Werte verschleiern, wird der Wert jedes Slot-Werts durch den Namen des Slots ersetzt. Weitere Informationen finden Sie unter [Verdecken von Slot-Werten in Konversationsprotokollen von Lex V2](#).

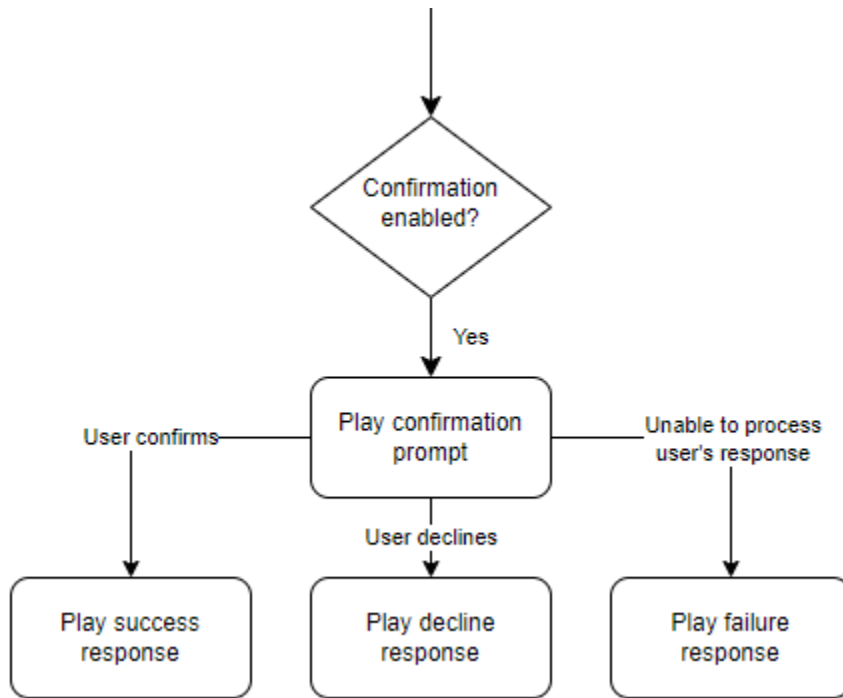
Bestätigung

Nachdem die Konversation mit dem Benutzer abgeschlossen ist und die Slot-Werte für die Absicht gefüllt sind, können Sie eine Bestätigungsaufforderung konfigurieren, um den Benutzer zu fragen, ob die Slot-Werte korrekt sind. Beispielsweise könnte ein Bot, der Wartungstermine für Autos vereinbart, den Benutzer zu folgenden Informationen auffordern:

Ich habe den Service für Ihren Honda Civic 2017 für den 25. März um 15:00 Uhr geplant. Ist das in Ordnung?

Sie können 3 Arten von Antworten auf die Bestätigungsaufforderung definieren:

- **Bestätigungsantwort** — Diese Antwort wird an den Benutzer gesendet, wenn der Benutzer die Absicht bestätigt. Zum Beispiel, nachdem der Benutzer auf die Aufforderung „Möchten Sie die Bestellung aufgeben?“ mit „Ja“ geantwortet hat
- **Antwort ablehnen** — Diese Antwort wird an den Benutzer gesendet, wenn der Benutzer die Absicht ablehnt. Zum Beispiel, nachdem der Benutzer auf die Frage „Möchten Sie die Bestellung aufgeben?“ mit „Nein“ geantwortet hat
- **Fehlerhafte Antwort** — Diese Antwort wird an den Benutzer gesendet, wenn die Bestätigungsaufforderung nicht bearbeitet werden kann. Zum Beispiel, wenn die Antwort des Benutzers nicht verstanden werden konnte oder nicht mit Ja oder Nein beantwortet werden konnte.



Wenn Sie keine Bestätigungsaufforderung angeben, geht Amazon Lex V2 zum Erfüllungsschritt oder zur abschließenden Antwort über.

Sie können Werte festlegen, die nächsten Schritte konfigurieren und Bedingungen anwenden, die jeder Antwort entsprechen, um den Gesprächsablauf zu gestalten. In Ermangelung einer Bedingung oder eines ausdrücklichen nächsten Schritts geht Amazon Lex V2 zum Erfüllungsschritt über.

Sie können auch den Dialog-Code-Hook aktivieren, um die in der Absicht erfassten Informationen zu überprüfen, bevor Sie sie zum Versand senden. Um einen Code-Hook zu verwenden, aktivieren Sie den Dialog-Code-Hook in den erweiterten Optionen der Bestätigungsaufforderung. Konfigurieren Sie außerdem den nächsten Schritt des vorherigen Status so, dass der Dialog-Code-Hook ausgeführt wird. Weitere Informationen finden Sie unter [Rufen Sie den Code-Hook des Dialogs auf](#).

Note

Wenn Sie einen Code-Hook verwenden, um den Bestätigungsschritt zur Laufzeit auszulösen, müssen Sie den Bestätigungsschritt zur Buildzeit als Aktiv markieren.

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

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Verwenden einer Lambda-Funktion zur Validierung einer Absicht.

Sie können einen Lambda-Code-Hook definieren, um die Absicht zu überprüfen, bevor Sie sie zur Erfüllung senden. Um einen Code-Hook zu verwenden, aktivieren Sie den Dialog-Code-Hook in den erweiterten Optionen der Bestätigungsaufforderung.

Wenn Sie einen Code-Hook verwenden, können Sie die Aktionen definieren, die Amazon Lex V2 nach der Ausführung des Code-Hooks ausführt. Sie können drei Arten von Antworten erstellen:

- Erfolgsantwort — Wird an den Benutzer gesendet, wenn der Code-Hook erfolgreich abgeschlossen wurde.
- Fehlerantwort — Wird an den Benutzer gesendet, wenn der Code-Hook nicht erfolgreich ausgeführt wird oder wenn der Code-Hook `Failure` in der Antwort zurückkehrt.
- Timeout-Antwort — Wird an den Benutzer gesendet, wenn der Code-Hook nicht innerhalb des konfigurierten Timeout-Zeitraums abgeschlossen wird.

Bereitstellung

Nachdem der Benutzer alle Slot-Werte für die Absicht angegeben hat, erfüllt Amazon Lex V2 die Anfrage des Benutzers. Sie können die folgenden Optionen für den Versand konfigurieren.

- Fulfillment-Code-Hook — Sie können diese Option verwenden, um den Fulfillment-Lambda-Aufruf zu steuern. Wenn die Option deaktiviert ist, ist die Erfüllung erfolgreich, ohne dass die Lambda-Funktion aufgerufen wird.
- Fulfillment-Updates — Sie können Fulfillment-Updates für Lambda-Funktionen aktivieren, deren Abschluss mehr als ein paar Sekunden dauert, sodass der Benutzer weiß, dass der Vorgang

gerade läuft. Weitere Informationen finden Sie unter [Updates zum Fulfillment-Fortschritt für Ihren Lex V2-Bot konfigurieren](#). Diese Funktion ist nur für Streaming-Konversationen verfügbar.

- **Fulfillment-Antworten** — Sie können eine Erfolgsantwort, eine Fehlerantwort und eine Timeout-Antwort konfigurieren. Die entsprechende Antwort wird basierend auf dem Status des Fulfillment-Lambda-Aufrufs an den Benutzer zurückgegeben.

Es gibt drei mögliche Fulfillment-Antworten:

- **Erfolgsantwort** — Eine Nachricht, die gesendet wird, wenn der Fulfillment-Lambda erfolgreich abgeschlossen wurde.
- **Fehlerantwort** — Eine Nachricht, die gesendet wird, wenn die Erfüllung fehlgeschlagen ist oder Lambda aus irgendeinem Grund nicht abgeschlossen werden kann.
- **Timeout-Antwort** — Eine Nachricht, die gesendet wird, wenn die Fulfillment-Lambda-Funktion nicht innerhalb des konfigurierten Timeouts beendet wird.

Sie können Werte festlegen, die nächsten Schritte konfigurieren und Bedingungen anwenden, die jeder Antwort entsprechen, um den Gesprächsablauf zu gestalten. In Ermangelung einer Bedingung oder eines ausdrücklichen nächsten Schritts geht Amazon Lex V2 zur abschließenden Antwort über.

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

Absicht-Struktur

84

 Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Antwort schließen

Die abschließende Antwort wird an Ihren Benutzer gesendet, nachdem seine Absicht erfüllt wurde. Sie können die abschließende Antwort verwenden, um die Konversation zu beenden, oder Sie können sie verwenden, um den Benutzer darüber zu informieren, dass er mit einer anderen Absicht fortfahren kann. In einem Reisebuchungs-Bot können Sie beispielsweise die abschließende Antwort für die Absicht „Hotelzimmer buchen“ wie folgt festlegen:

In Ordnung, ich habe dein Hotelzimmer gebucht. Kann ich dir noch mit etwas anderem weiterhelfen?

Sie können Werte festlegen, die nächsten Schritte konfigurieren und nach der abschließenden Antwort Bedingungen anwenden, um den Konversationspfad zu entwerfen. In Ermangelung einer Bedingung oder eines ausdrücklichen nächsten Schritts beendet Amazon Lex V2 die Konversation.

Wenn Sie keine abschließende Antwort geben oder wenn keine der Bedingungen als wahr bewertet wird, beendet Amazon Lex V2 die Konversation mit Ihrem Bot.

Closing response [Info](#) Active

You can define the response when closing the intent.

▶ Response sent to the user after the intent is fulfilled
Message: -

▶ Set values
-

Next step in conversation
End conversation

+ Add conditional branching

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

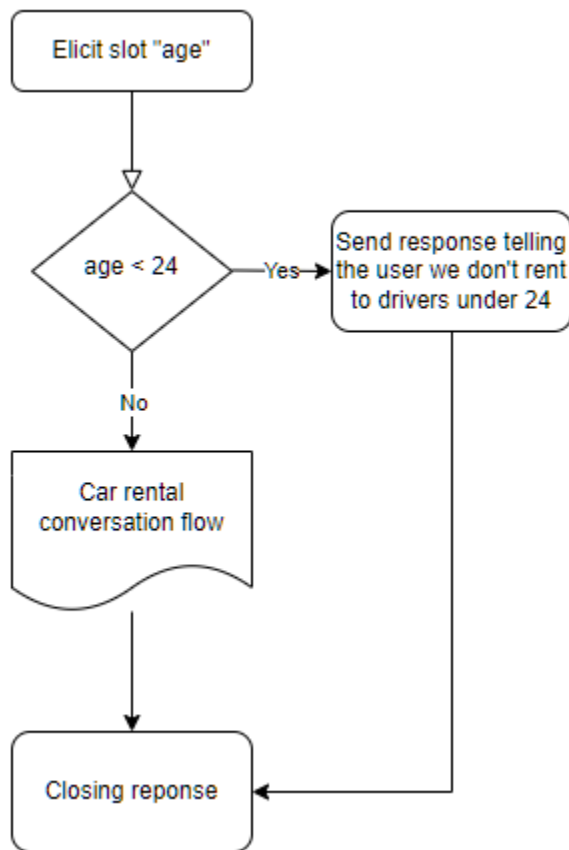
Gesprächspfade erstellen

In der Regel verwaltet Amazon Lex V2 den Gesprächsfluss mit Ihren Benutzern. Bei einfachen Bots kann der Standardablauf ausreichen, um Ihren Benutzern ein gutes Erlebnis zu bieten. Bei komplexeren Bots möchten Sie jedoch möglicherweise die Kontrolle über die Konversation übernehmen und den Ablauf auf komplexere Pfade lenken.

In einem Bot, der Mietwagen bucht, vermieten Sie beispielsweise möglicherweise nicht an jüngere Fahrer. In diesem Fall können Sie eine Bedingung erstellen, die überprüft, ob ein Fahrer ein bestimmtes Alter noch nicht erreicht hat, und falls ja, direkt zur abschließenden Antwort übergehen.

Gesprächspfade erstellen

86



Um solche Interaktionen zu entwerfen, können Sie an jedem Punkt der Konversation den nächsten Schritt konfigurieren, Bedingungen auswerten, Werte festlegen und Code-Hooks aufrufen.

Conditional Branching hilft Ihnen dabei, durch komplexe Interaktionen Pfade für Ihre Benutzer zu erstellen. Sie können eine bedingte Verzweigung an jedem Punkt verwenden, an dem Sie die Kontrolle über die Konversation an Ihren Bot übergeben. Sie können beispielsweise eine Bedingung erstellen, bevor der Bot den ersten Slot-Wert auslöst, Sie können eine Bedingung zwischen den einzelnen Slot-Werten erstellen oder Sie können eine Bedingung erstellen, bevor der Bot die Konversation beendet. Eine Liste der Orte, an denen Sie Bedingungen hinzufügen können, finden Sie unter [Absichten hinzufügen](#)

Wenn Sie einen Bot erstellen, erstellt Amazon Lex V2 einen Standardpfad durch die Konversation, der auf der Prioritätsreihenfolge der Slots basiert. Um den Konversationspfad anzupassen, können Sie den nächsten Schritt zu einem beliebigen Zeitpunkt in der Konversation ändern. Weitere Informationen finden Sie unter [Konfigurieren Sie die nächsten Schritte in der Konversation](#).

Um alternative Pfade auf der Grundlage von Bedingungen zu erstellen, können Sie an einem beliebigen Punkt in der Konversation eine bedingte Verzweigung verwenden. Sie können beispielsweise eine Bedingung erstellen, bevor der Bot den ersten Slot-Wert auslöst. Sie können

eine Bedingung zwischen dem Abrufen der einzelnen Slot-Werte erstellen oder Sie können eine Bedingung erstellen, bevor der Bot die Konversation beendet. Eine Liste der Orte, an denen Sie Bedingungen hinzufügen können, finden Sie unter [Fügen Sie Bedingungen zu Konversationen in Filialen hinzu](#)

Sie können Bedingungen auf der Grundlage von Slot-Werten, Sitzungsattributen, dem Eingabemodus und dem Eingabeprotokoll oder einer Antwort von Amazon Kendra festlegen.

Sie können an jedem Punkt der Konversation die Werte für Slot- und Sitzungsattribute festlegen. Weitere Informationen finden Sie unter [Legen Sie Werte während der Konversation fest](#).

Sie können die nächste Aktion auch auf den Dialogcode-Hook setzen, um eine Lambda-Funktion auszuführen. Weitere Informationen finden Sie unter [Rufen Sie den Code-Hook des Dialogs auf](#).

Die folgende Abbildung zeigt die Erstellung eines Pfads für einen Slot in der Konsole. In diesem Beispiel ruft Amazon Lex V2 den Slot „age“ hervor. Wenn der Wert des Slots kleiner als 24 ist, springt Amazon Lex V2 zur abschließenden Antwort, andernfalls folgt es dem Standardpfad.

Conditional branching Info

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

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Konfigurieren Sie die nächsten Schritte in der Konversation

Sie können in jeder Phase der Konversation einen nächsten Schritt konfigurieren, um Konversationen zu gestalten. In der Regel konfiguriert Amazon Lex V2 automatisch die standardmäßigen nächsten Schritte für jede Phase der Konversation gemäß der folgenden Reihenfolge.

Erste Antwort → Slot-Auswahl → Bestätigung (falls aktiv) → Versand (falls aktiv) → Abschlussantwort (falls aktiv) → Konversation beenden

Sie können die standardmäßigen nächsten Schritte ändern und die Konversation auf der Grundlage der erwarteten Benutzererfahrung gestalten. Die folgenden nächsten Schritte können in jeder Phase der Konversation konfiguriert werden:

Springe zu

- Erste Antwort — Die Konversation wird am Anfang der Absicht neu gestartet. Sie können bei der Konfiguration dieses nächsten Schritts wählen, ob Sie die erste Antwort überspringen möchten
- Einen Slot ausloten — Sie können einen beliebigen Slot in der Absicht ermitteln.
- Bedingungen auswerten — Sie können die Bedingungen bewerten und die Konversation in jeder Phase der Konversation abzweigen.
- Dialog-Code-Hook aufrufen — Sie können die Geschäftslogik bei jedem Schritt aufrufen.
- Absicht bestätigen — Der Benutzer wird aufgefordert, die Absicht zu bestätigen.
- Absicht erfüllen — Die Erfüllung der Absicht beginnt als nächster Schritt.
- Abschlussantwort — Die abschließende Antwort wird an den Benutzer zurückgesendet.

Wechseln Sie zu

- **Absicht** — Sie können zu einer anderen Absicht wechseln und die Konversation für diese Absicht fortsetzen. Sie können optional die erste Antwort auf die Absicht während des Übergangs überspringen.
- **Absicht: bestimmter Slot** — Sie können direkt einen bestimmten Slot in einem anderen Intent aufrufen, wenn Sie in der aktuellen Absicht bereits einige Slot-Werte erfasst haben.

Auf Benutzereingaben warten — Der Bot wartet darauf, dass der Benutzer Eingaben macht, um eine neue Absicht zu erkennen. Sie können Eingabeaufforderungen wie „Kann ich Ihnen noch mit etwas anderem weiterhelfen?“ konfigurieren, bevor Sie diesen nächsten Schritt festlegen. Der Bot wird sich im `ElicitIntent` Dialogstatus befinden.

Konversation beenden — Die Konversation mit dem Bot ist geschlossen.

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Legen Sie Werte während der Konversation fest

Amazon Lex V2 bietet die Möglichkeit, in jedem Schritt der Konversation Slot-Werte und Sitzungsattributwerte festzulegen. Sie können diese Werte dann während der Konversation verwenden, um Bedingungen zu bewerten, oder sie bei der Erfüllung von Absichten verwenden.

Sie können Slot-Werte für die aktuelle Absicht festlegen. Wenn der nächste Schritt in der Konversation darin besteht, eine andere Absicht aufzurufen, können Sie Slot-Werte für die neue Absicht festlegen.

Wenn der zugewiesene Slot nicht gefüllt ist oder der JSON-Pfad nicht analysiert werden kann, wird das Attribut auf gesetzt. `null`

Verwenden Sie die folgende Syntax, wenn Sie Slot-Werte und Sitzungsattribute verwenden:

- Slot-Werte — umgeben Sie den Slot-Namen mit geschweiften Klammern („{}“). Für Slot-Werte in der aktuellen Absicht müssen Sie nur den Slot-Namen verwenden. Beispiel, {slot}. Wenn Sie in der nächsten Absicht einen Wert festlegen, müssen Sie sowohl den Namen der Absicht als auch den Slot-Namen verwenden, um den Slot zu identifizieren. Beispiel, {intent.slot}.

Beispiele:

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

Der Wert eines Slots kann einer der folgenden sein:

- eine konstante Zeichenfolge
- ein JSON-Pfad, der auf den Transkriptionsblock in der Amazon Lex Lex-Antwort verweist (für en-US und en-GB)
- ein Sitzungsattribut

Beispiele:

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

Note

Slot-Werte können auch auf gesetzt werden `null`. Wenn Sie einen aufgefüllten Slot-Wert erneut ermitteln müssen, müssen Sie den Wert auf einstellen, `null` bevor Sie den Kunden erneut zur Eingabe des Slot-Werts auffordern. Wenn der zugewiesene Slot nicht gefüllt ist oder der JSON-Pfad nicht analysiert werden kann, wird das Attribut auf gesetzt. `null`

- Sitzungsattribute — umgeben Sie den Attributnamen mit eckigen Klammern („[]“). Beispiel, [sessionAttribute].

Beispiele:

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

Der Wert des Sitzungsattributs kann einer der folgenden sein:

- eine konstante Zeichenfolge
- ein JSON-Pfad, der auf den Transkriptionsblock in der Amazon Lex Lex-Antwort verweist (für en-US und en-GB)
- eine Referenz für einen Slot-Wert

 Note

Wenn der zugewiesene Slot nicht gefüllt ist oder wenn der JSON-Pfad nicht analysiert werden kann, wird das Attribut auf `null` gesetzt.

 Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Fügen Sie Bedingungen zu Konversationen in Filialen hinzu

Mithilfe von bedingter Verzweigung können Sie steuern, welchen Weg Ihr Kunde in der Konversation mit Ihrem Bot einschlägt. Sie können die Konversation auf der Grundlage von Slot-Werten, Sitzungsattributen, dem Inhalt des Eingabemodus und der Eingabe-Transkriptfelder oder einer Antwort von Amazon Kendra verzweigen.

Sie können bis zu vier Zweige definieren. Jeder Zweig hat eine Bedingung, die erfüllt sein muss, damit Amazon Lex V2 diesem Zweig folgen kann. Wenn die Bedingung für keinen der Zweige erfüllt ist, wird einer Standardverzweigung gefolgt.

Wenn Sie einen Zweig definieren, definieren Sie die Aktion, die Amazon Lex V2 ergreifen soll, wenn die Bedingungen, die diesem Zweig entsprechen, als wahr bewertet werden. Sie können jede der folgenden Aktionen definieren:

- Eine Antwort, die an den Benutzer gesendet wurde.
- Slot-Werte, die auf Slots angewendet werden sollen.
- Sitzungsattributwerte für die aktuelle Sitzung.
- Der nächste Schritt in der Konversation. Weitere Informationen finden Sie unter [Gesprächspfade erstellen](#).

Conditional branching Info

Jump to different parts of the conversation based on conditions you define. You can add up to 4 conditional branches.

Active

Under24 X

+ Add branch

if no matches

Default flow

Delete all

▼ Condition for Under24

If `{{age}} < 24`

Condition

If `{{age}} < 24`

► Response

Message: You are not eligible

► Set values

`[eligibility] = "false"`

Next step in conversation

End conversation

Jede bedingte Verzweigung hat einen booleschen Ausdruck, der erfüllt sein muss, damit Amazon Lex V2 der Verzweigung folgt. Es gibt Vergleichs- und Boolesche Operatoren, Funktionen und Quantifizierungsoperatoren, die Sie für Ihre Bedingungen verwenden können. Die folgende Bedingung gibt beispielsweise „true“ zurück, wenn der Wert für {age} kleiner als 24 ist.

```
{{age}} < 24
```

Die folgende Bedingung gibt „true“ zurück, wenn das Feld mit mehreren Werten {toppings} das Wort „pineapple“ enthält.

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

Für komplexere Bedingungen können Sie mehrere Vergleichsoperatoren mit einem booleschen Operator kombinieren. Die folgende Bedingung gibt beispielsweise „true“ zurück, wenn der Slot-Wert {make} „Honda“ und der Slot-Wert {model} „Civic“ ist. Verwenden Sie Klammern, um die Reihenfolge der Auswertung festzulegen.

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

Die folgenden Themen enthalten Einzelheiten zu den Operatoren und Funktionen für bedingte Verzweigungen.

Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Themen

- [Vergleichsoperatoren](#)
- [Boolesche Operatoren](#)
- [Quantifizierer-Operatoren](#)
- [Funktionen](#)
- [Beispiel für bedingte Ausdrücke](#)

Vergleichsoperatoren

Amazon Lex V2 unterstützt die folgenden Vergleichsoperatoren für Bedingungen:

- Entspricht (=)
- Nicht gleich (!=)

- Kleiner als (<)
- Kleiner als oder gleich (<=)
- größer als (>)
- Größer als oder gleich (>=)

Wenn ein Vergleichsoperator verwendet wird, werden die folgenden Regeln verwendet.

- Die linke Seite muss eine Referenz sein. Um beispielsweise auf einen Slot-Wert zu verweisen, verwenden `{slotName}` Sie. Um auf einen Sitzungsattributwert zu verweisen, verwenden `Sie[attribute]`. Für den Eingabemodus und das Eingabeprotokoll verwenden Sie `$.inputMode` und `$.inputTranscript`.
- Die rechte Seite muss eine Konstante sein und denselben Typ wie die linke Seite haben.
- Jeder Ausdruck, der auf ein Attribut verweist, das nicht festgelegt wurde, wird als ungültig behandelt und nicht ausgewertet.
- Wenn Sie einen Slot mit mehreren Werten vergleichen, ist der verwendete Wert eine durch Kommas getrennte Liste aller interpretierten Werte.

Vergleiche basieren auf dem Slot-Typ der Referenz. Sie werden wie folgt gelöst:

- Zeichenketten — Zeichenketten werden auf der Grundlage ihrer ASCII-Darstellung verglichen. Bei diesem Vergleich wird die Groß-/Kleinschreibung nicht beachtet.
- Zahlen — zahlenbasierte Slots werden von der Zeichenkettendarstellung in eine Zahl umgewandelt und dann verglichen.
- Datum/Uhrzeit — Zeitfenster werden anhand der Zeitreihen verglichen. Das frühere Datum oder die frühere Uhrzeit wird als kleiner angesehen. Bei Zeiträumen gelten kürzere Zeiträume als kleiner.

Boolesche Operatoren

Amazon Lex V2 unterstützt boolesche Operatoren, um Vergleichsoperatoren zu kombinieren. Mit ihnen können Sie Aussagen erstellen, die den folgenden ähneln:

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

Sie können die folgenden booleschen Operatoren verwenden:

- UND (&&)

- ODER (||)
- NICHT (!)

Quantifizierer-Operatoren

Quantifizierer-Operatoren werten die Elemente einer Sequenz aus und bestimmen, ob ein oder mehrere Elemente die Bedingung erfüllen.

- CONTAINS — bestimmt, ob der angegebene Wert in einem Slot mit mehreren Werten enthalten ist, und gibt „true“ zurück, wenn dies der Fall ist. Gibt beispielsweise „true“ {toppings} CONTAINS "pineapple" zurück, wenn der Benutzer Ananas auf seiner Pizza bestellt hat.

Funktionen

Den Funktionen muss die Zeichenfolge `fn.` vorangestellt werden. Das Argument der Funktion ist ein Verweis auf einen Slot, ein Sitzungsattribut oder ein Anforderungsattribut. Amazon Lex V2 bietet zwei Funktionen zum Abrufen von Informationen aus den Werten von slots, sessionAttribute oder requestAttribute.

- `fn.Count ()` — zählt die Anzahl der Werte in einem Slot mit mehreren Werten.

Wenn der Slot beispielsweise den Wert „Peperoni, Ananas“ {toppings} enthält:

```
fn.COUNT({toppings}) = 2
```

- `fn.is_Set ()` — Der Wert ist wahr, wenn in der aktuellen Sitzung ein Slot, ein Sitzungsattribut oder ein Anforderungsattribut gesetzt ist.

Basierend auf dem vorherigen Beispiel:

```
fn.IS_SET({toppings})
```

- `fn.Length ()` — Wert ist die Länge des Werts des Sitzungsattributs, des Slot-Werts oder des Slot-Attributs, der in der aktuellen Sitzung festgelegt wurde. Diese Funktion unterstützt keine Slots mit mehreren Werten oder zusammengesetzten Slots.

Beispiel:

Wenn der Slot den Wert „123456781234“ {credit-card-number} enthält:

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

Beispiel für bedingte Ausdrücke

Hier sind einige Beispiele für bedingte Ausdrücke. HINWEIS: \$. stellt den Einstiegspunkt zur JSON-Antwort dar. Der folgende Wert \$. wird in der Antwort analysiert, um den Wert abzurufen. Bedingte Ausdrücke, die den JSON-Pfadverweis auf den Transkriptionsblock in der Antwort verwenden, werden nur in denselben Gebietsschemas unterstützt, die ASR-Transkriptionswerte unterstützen.

Werttyp	Anwendungsfall	Bedingter Ausdruck
Benutzerdefinierter Steckplatz	<code>pizzaSize</code> Der Slot-Wert ist gleich groß	<code>{pizzaSize} = "large"</code>
Benutzerdefinierter Steckplatz	<code>pizzaSize</code> entspricht groß oder mittel	<code>{pizzaSize} = "large" ODER {pizzaSize} = "medium"</code>
Benutzerdefinierter Steckplatz	Ausdrücke mit () und AND/OR	<code>{pizzaType} = "pepperoni" ODER {pizzaSize} = "medium" ODER {pizzaSize} = "small"</code>
Benutzerdefinierter Steckplatz (Steckplatz mit mehreren Werten)	Überprüfe, ob es sich bei einem der Toppings um Zwiebel handelt	<code>{toppings} CONTAINS "Onion"</code>
Benutzerdefinierter Steckplatz (Steckplatz mit mehreren Werten)	Die Anzahl der Beläge beträgt mehr als 3	<code>fn.COUNT({topping}) > 2</code>
<code>AMAZON.AlphaNumeric</code>	<code>bookingID</code> ist ABC123	<code>{bookingID} = "ABC123"</code>
<code>AMAZON.Number</code>	Der Wert des Altersfensters ist größer als 30	<code>{age} > 30</code>
<code>AMAZON.Number</code>	Der Wert des Altersfensters ist gleich 10	<code>{age} = 10</code>

Werttyp	Anwendungsfall	Bedingter Ausdruck
AMAZON.Date	dateOfBirth Wert des Zeitplatzes vor 1990	{dateOfBirth} < "1990-10-01"
AMAZON.State	destinationState Der Slot-Wert entspricht Washington	{destinationState} = "washington"
AMAZON.Country	destinationCountry Der Slot-Wert ist nicht Vereinigte Staaten	{destinationCountry} != "united states"
AMAZON.FirstName	firstName Der Slot-Wert ist John	{firstName} = "John"
AMAZON.PhoneNumber	phoneNumber Der Slot-Wert ist 716767891932	{phoneNumber} = 716767891932
AMAZON.Percentage	Prüfen Sie, ob der prozentuale Slot-Wert größer oder gleich 78 ist	{percentage} >= 78
AMAZON.EmailAddress	emailAddress Der Slot-Wert ist userA@hmail.com	{emailAddress} = "userA@hmail.com"
AMAZON.LastName	lastNameDer Slot-Wert ist Doe	{lastName} = "Doe"
AMAZON.City	Der Wert des Stadt-Slots entspricht dem Wert von Seattle	{city} = "Seattle"
AMAZON.Time	Es ist nach 20 Uhr	{time} > "20:00"
AMAZON.StreetName	streetName Der Slot-Wert ist Boren Avenue	{streetName} = "boren avenue"

Werttyp	Anwendungsfall	Bedingter Ausdruck
AMAZON.Duration	travelDuration Der Slot-Wert beträgt weniger als 2 Stunden	{travelDuration} < P2H
Eingabemodus	Eingabemodus ist Sprache	\$.inputMode = "Speech"
Eingangstranskript	Das Eingabeprotokoll entspricht „Ich möchte eine große Pizza“	\$.inputTranscript = "I want a large pizza"
Sitzungsattribut	überprüfen Sie das Attribut customer_subscription_type	[customer_subscription_type] = "yearly"
Attribut anfordern	überprüfen Sie das Flag retry_enabled	((retry_enabled)) = "TRUE"
Antwort von Kendra	Die Antwort von Kendra enthält häufig gestellte Fragen	fn.IS_SET(((x-amz-lex:kendra-search-response-question-answer-question-1)))
Bedingter Ausdruck mit Transkriptionen	Bedingte Ausdrücke, die den JSON-Pfad für Transkriptionen verwenden	\$.transcriptions[0].transcriptionConfidence < 0.8 AND \$.transcriptions[1].transcriptionConfidence > 0.5
Sitzungsattribute festlegen	Legen Sie Sitzungsattribute mithilfe von JSON-Pfad- und Slot-Werten für Transkriptionen fest	[sessionAttribute] = "\$.transcriptions.." AND [sessionAttribute] = "{<slotName>}"

Werttyp	Anwendungsfall	Bedingter Ausdruck
Legen Sie Slot-Werte fest	Legen Sie Slot-Werte mithilfe des JSON-Pfads für Sitzungsattribute und Transkriptionen fest	<code>{slotName} = [<sessionAttribute>] AND {slotName} = "\$.transcriptions. .."</code>

Note

`slotName` bezieht sich auf den Namen eines Slots im Bot. Wenn der Slot nicht aufgelöst ist (Null) oder wenn der Slot nicht existiert, werden die Zuweisungen zur Laufzeit ignoriert. `sessionAttribute` bezieht sich auf den Namen des Sitzungsattributs, das vom Kunden bei der Erstellung festgelegt wurde.

Rufen Sie den Code-Hook des Dialogs auf

Bei jedem Schritt in der Konversation, wenn eine Nachricht an den Benutzer gesendet wird, können Sie eine Lambda-Funktion als nächsten Schritt in der Konversation verwenden. Sie können die Funktion verwenden, um Geschäftslogik auf der Grundlage des aktuellen Status der Konversation zu implementieren.

Die Lambda-Funktion ist mit dem Bot-Alias verknüpft, den Sie verwenden. Um die Lambda-Funktion für alle Dialogcode-Hooks in Ihrer Absicht aufzurufen, müssen Sie die Option Lambda-Funktion für die Initialisierung und Validierung der Absicht verwenden auswählen. Weitere Informationen zur Auswahl einer Lambda-Funktion finden Sie unter [Eine AWS Lambda Funktion für Ihren Bot erstellen](#).

Die Verwendung einer Lambda-Funktion besteht aus zwei Schritten. Zunächst müssen Sie den Dialog-Code-Hook zu einem beliebigen Zeitpunkt in der Konversation aktivieren. Zweitens müssen Sie den nächsten Schritt in der Konversation festlegen, um den Dialogcode-Hook zu verwenden.

Die folgende Abbildung zeigt den aktivierten Dialog-Code-Hook.

Dialog code hook [Info](#)

You can enable Lambda functions to manage initialize the conversation.

☒ **Invoke Lambda for user request validation**

Invocation label - *optional*

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, ~, _

Als Nächstes legen Sie den Code-Hook als nächste Aktion für den Konversationsschritt fest. Sie können dies tun, indem Sie den nächsten Schritt in der Konversation so konfigurieren, dass er den Dialog-Code-Hook aufrufen soll. Die folgende Abbildung zeigt eine bedingte Verzweigung, bei der das Aufrufen des Dialog-Code-Hooks der nächste Schritt für den Standardpfad der Konversation ist.

Conditional branching [Info](#)

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

◆

► **Response**

Message: -

▼ **Set values**

-

Slot values - optional

Add slot values as: {slot} = "value"

{slot} = "value"

Separate values with a new line.

Session attributes - optional

Add session attributes as: [session attribute] = "value"

[session attribute] = "value"

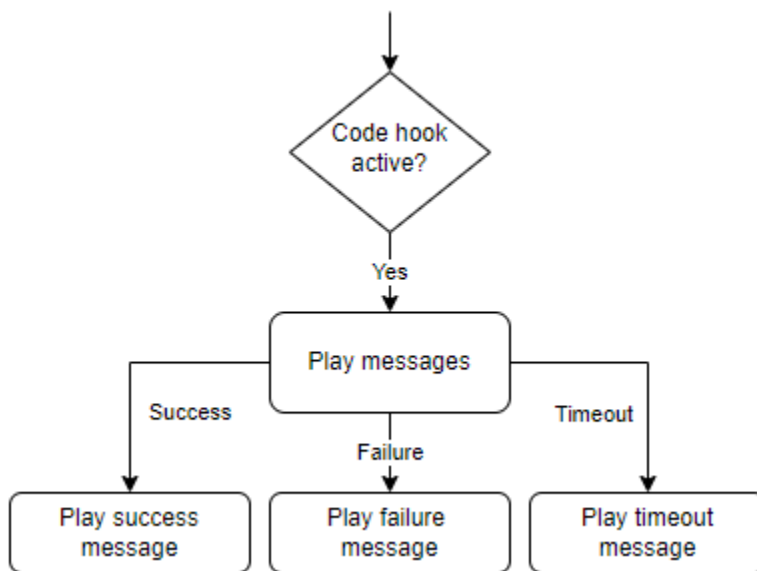
Separate values with a new line.

Next step in conversation

Invoke dialog code hook ▼

Wenn Code-Hooks aktiv sind, können Sie drei Antworten festlegen, die an den Benutzer zurückgegeben werden:

- **Erfolgreich** — Wird gesendet, wenn die Lambda-Funktion erfolgreich abgeschlossen wurde.
- **Fehler** — Wird gesendet, wenn bei der Ausführung der Lambda-Funktion ein Problem aufgetreten ist oder die Lambda-Funktion einen `intent.state` Wert von zurückgegeben hat. `Failed`
- **Timeout** — Wird gesendet, wenn die Lambda-Funktion innerhalb des konfigurierten Timeout-Zeitraums nicht abgeschlossen wurde.



Wählen Sie Lambda Dialog Code Hook und dann Erweiterte Optionen, um die drei Optionen für Antworten zu sehen, die dem Lambda-Funktionsaufruf entsprechen. Sie können Werte festlegen, die nächsten Schritte konfigurieren und Bedingungen anwenden, die jeder Antwort entsprechen, um den Konversationsablauf zu gestalten. In Ermangelung einer Bedingung oder eines ausdrücklichen nächsten Schritts entscheidet Amazon Lex V2 auf der Grundlage des aktuellen Status der Konversation über den nächsten Schritt.

Auf der Seite Erweiterte Optionen können Sie auch wählen, ob Sie Ihren Lambda-Funktionsaufruf aktivieren oder deaktivieren möchten. Wenn die Funktion aktiviert ist, wird der Dialogcode-Hook mit einem Lambda-Aufruf aufgerufen, gefolgt von der Erfolgs-, Fehler- oder Timeout-Meldung, die auf den Ergebnissen des Lambda-Aufrufs basiert. Wenn die Funktion deaktiviert ist, führt Amazon Lex V2 die Lambda-Funktion nicht aus und fährt fort, als ob der Dialogcode-Hook erfolgreich wäre.

Sie können auch ein Aufruf-Label festlegen, das an die Lambda-Funktion gesendet wird, wenn sie durch diese Nachricht aufgerufen wird. Sie können dies verwenden, um den Abschnitt Ihrer Lambda-Funktion zu identifizieren, der ausgeführt werden soll.

Note

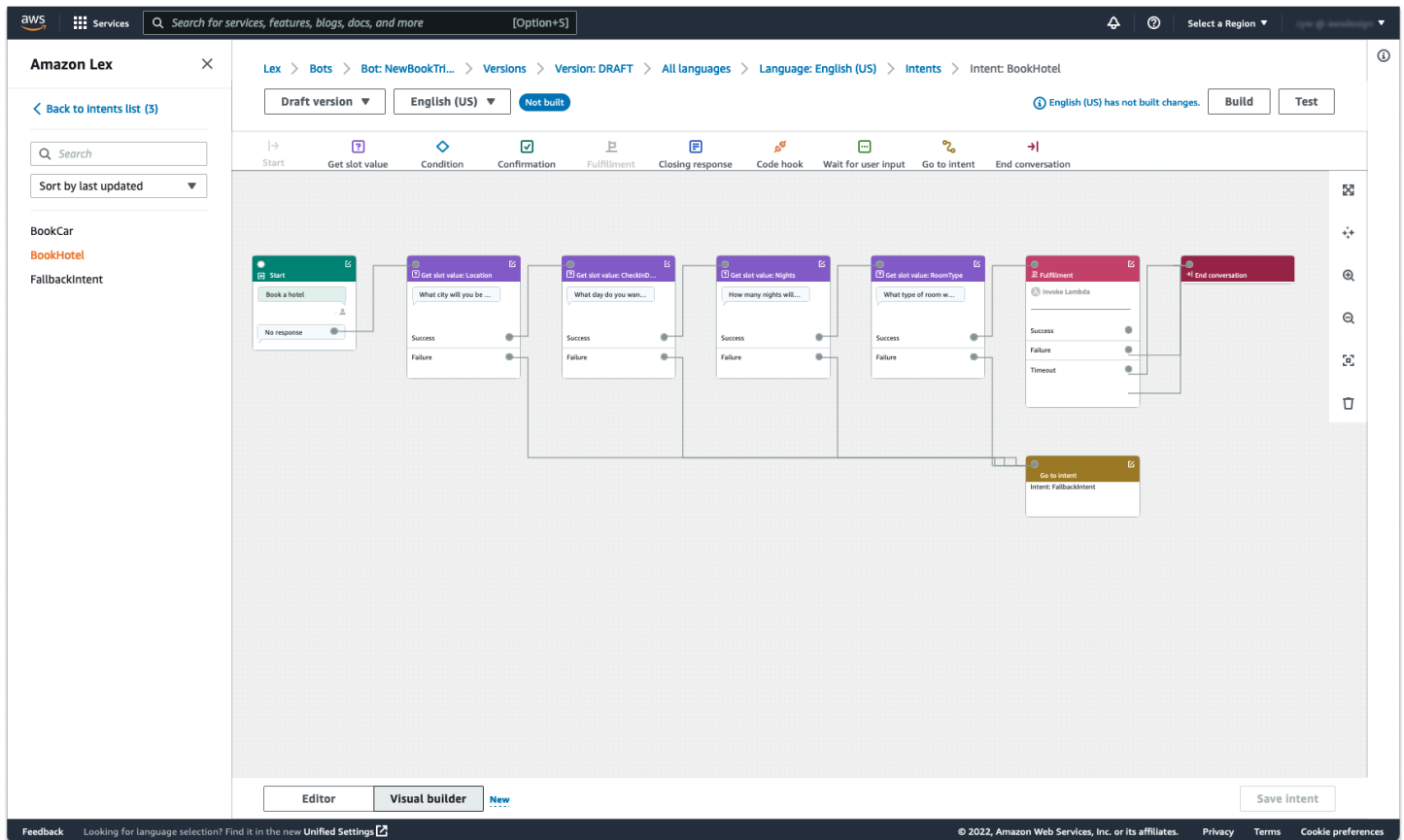
Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Verwenden von Visual Conversation Builder

Visual Conversation Builder ist ein Drag-and-Drop-Konversationsgenerator, mit dem Sie Konversationspfade einfach entwerfen und visualisieren können, indem Sie Absichten in einer reichhaltigen visuellen Umgebung verwenden.

Um auf den Visual Conversation Builder zuzugreifen

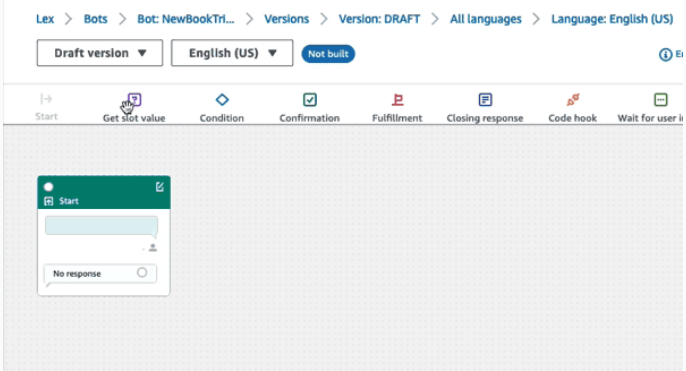
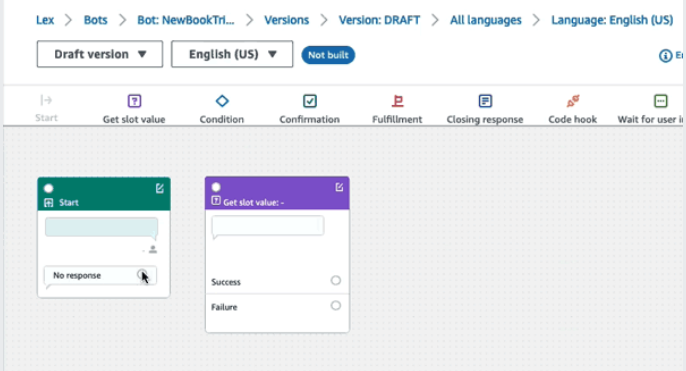
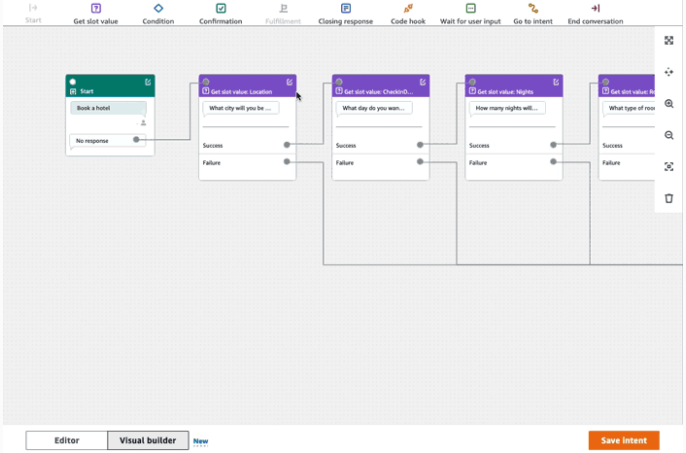
1. Wählen Sie in der Amazon Lex V2-Konsole einen Bot aus und wählen Sie im linken Navigationsbereich Intents aus.
2. Gehen Sie auf eine der folgenden Arten zum Intent-Editor:
 - Wählen Sie in der oberen rechten Ecke des Abschnitts Absichten die Option Absicht hinzufügen aus und wählen Sie dann, ob Sie entweder eine leere Absicht oder eine integrierte Absicht hinzufügen möchten.
 - Wählen Sie im Abschnitt Absichten den Namen einer Absicht aus.
3. Wählen Sie im Intent-Editor im Bereich am unteren Bildschirmrand Visual Builder aus, um auf den Visual Conversation Builder zuzugreifen.
4. Um zur Benutzeroberfläche des Menü-Intent-Editors zurückzukehren, wählen Sie Editor aus.

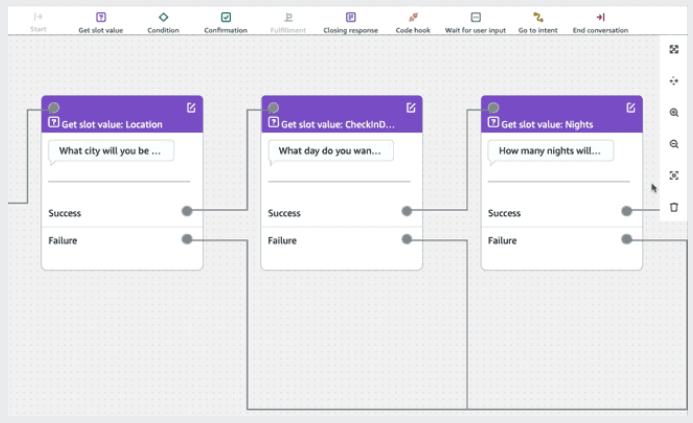
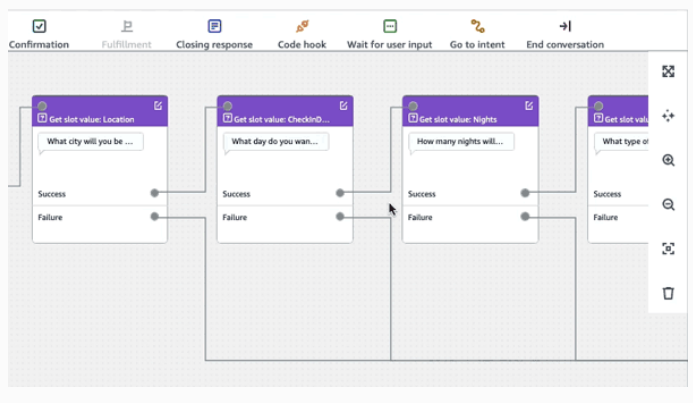
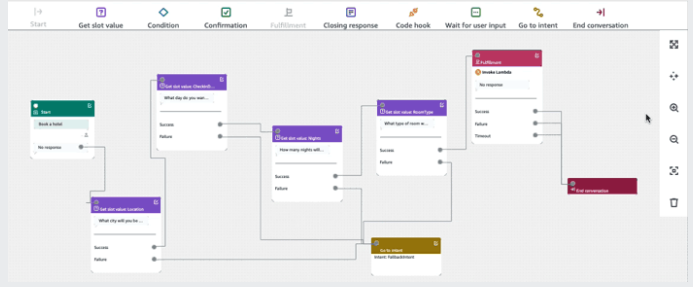


Visual Conversation Builder bietet eine intuitivere Benutzeroberfläche mit der Möglichkeit, den Konversationsfluss zu visualisieren und zu ändern. Durch Ziehen und Ablegen der Blöcke können Sie einen vorhandenen Flow erweitern oder die Konversationsschritte neu anordnen. Sie können einen Konversationsfluss mit komplexen Verzweigungen entwickeln, ohne Lambda-Code schreiben zu müssen.

Diese Änderung trägt dazu bei, das Design des Konversationsflusses von der anderen Geschäftslogik in Lambda zu entkoppeln. Visual Conversation Builder kann in Verbindung mit dem vorhandenen Intent-Editor verwendet werden und kann zum Erstellen von Konversationsabläufen verwendet werden. Es wird jedoch empfohlen, die visuelle Editor-Ansicht für komplexere Konversationsabläufe zu verwenden.

Wenn Sie eine Absicht speichern, kann Amazon Lex V2 Absichten automatisch verbinden, wenn festgestellt wird, dass Verbindungen verpasst wurden, Amazon Lex V2 eine Verbindung vorschlägt oder Sie Ihre eigene Verbindung für den Block auswählen können.

Aktion	Beispiel
Einen Block zum Workspace hinzufügen	 The screenshot shows the Amazon Lex Visual Conversation Builder interface. The top navigation bar includes 'Lex > Bots > Bot: NewBookTri... > Versions > Version: DRAFT > All languages > Language: English (US)'. Below the navigation bar, there are tabs for 'Draft version', 'English (US)', and 'Not built'. The workspace contains a single 'Start' block with a 'No response' option.
Eine Verbindung zwischen Blöcken herstellen	 The screenshot shows the Amazon Lex Visual Conversation Builder interface with two blocks: a 'Start' block and a 'Get slot value' block. A connection line is drawn from the 'No response' output of the 'Start' block to the 'Success' input of the 'Get slot value' block.
Öffnen des Konfigurationsfensters auf einem Block	 The screenshot shows the Amazon Lex Visual Conversation Builder interface with a sequence of five blocks: 'Start', 'Get slot value: Location', 'Get slot value: Checkin...', 'Get slot value: Nights', and 'Get slot value: Room'. Each block has its configuration window open, showing options for 'Success' and 'Failure' paths. The 'Start' block has a 'Book a hotel' prompt. The 'Get slot value' blocks have prompts like 'What city will you be...', 'What day do you want...', 'How many nights will...', and 'What type of room...'. The bottom of the interface shows tabs for 'Editor', 'Visual builder', and 'New', along with a 'Save Intent' button.

Aktion	Beispiel
Je nach Größe zoomen	
Löscht einen Block aus dem Konversationsfluss	
Reinigen Sie den Arbeitsbereich automatisch	

Terminologie:

Block — Die grundlegende Baueinheit eines Konversationsflusses. Jeder Block hat eine spezifische Funktionalität, um verschiedene Anwendungsfälle einer Konversation zu handhaben.

Port — Jeder Block enthält Ports, über die ein Block mit einem anderen verbunden werden kann. Blöcke können Eingangs- und Ausgangsanschlüsse enthalten. Jeder Ausgangsport steht für eine bestimmte Funktionsvariante eines Blocks (z. B. Fehler, Timeouts oder Erfolg).

Kante — Eine Kante ist eine Verbindung zwischen dem Ausgangsport eines Blocks und dem Eingangsport eines anderen Blocks. Es ist Teil einer Verzweigung in einem Konversationsfluss.

Konversationsfluss — Eine Reihe von Blöcken, die durch Kanten miteinander verbunden sind und Interaktionen mit einem Kunden auf Absichtsebene beschreiben.

Blöcke

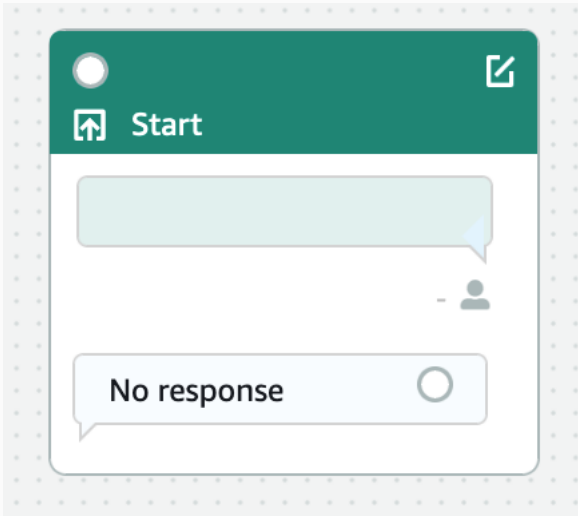
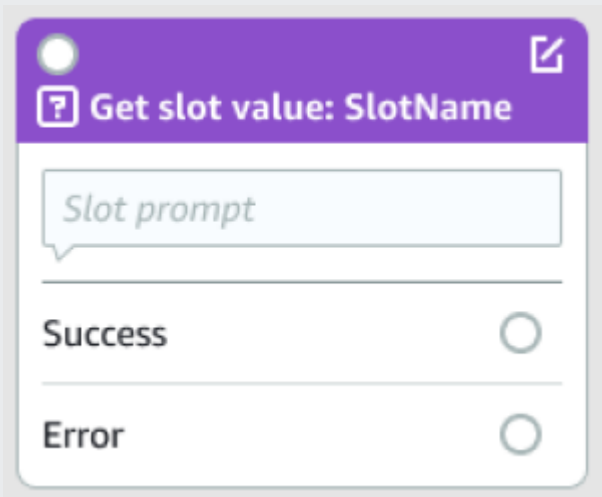
Blöcke sind die Bausteine eines Konversationsablaufdesigns. Sie stellen verschiedene Zustände innerhalb der Absicht dar, die sich vom Beginn der Absicht über Benutzereingaben bis hin zum Abschluss erstreckt.

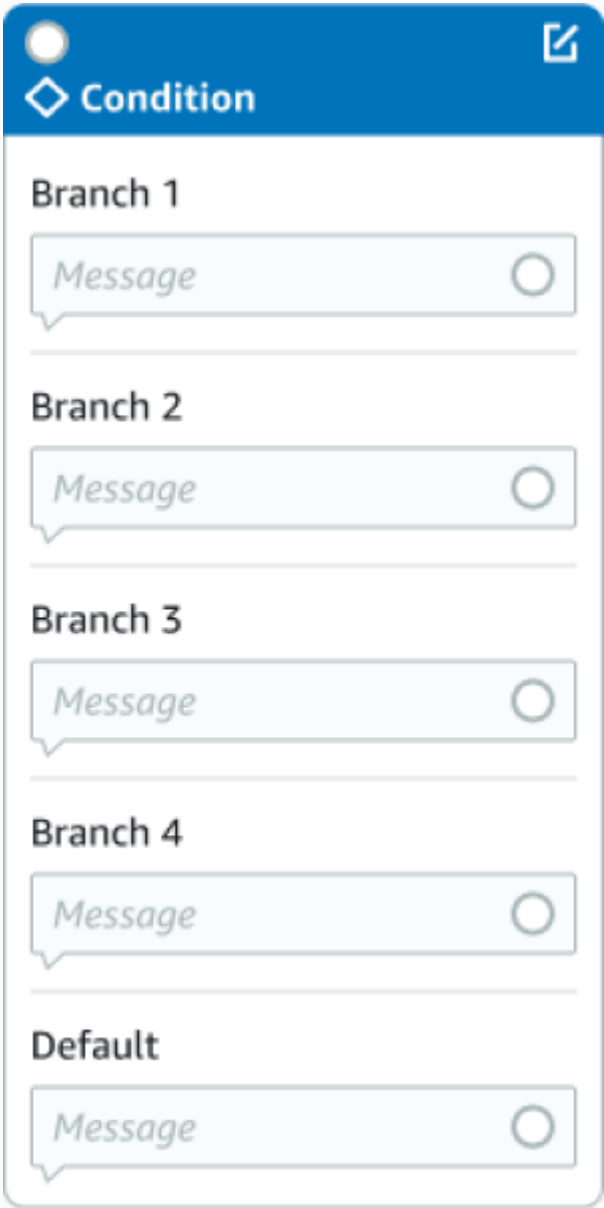
Jeder Block hat je nach Blocktyp einen Eintrittspunkt und einen oder mehrere Ausgangspunkte. Jeder Austrittspunkt kann mit einer entsprechenden Nachricht konfiguriert werden, während die Konversation über die Ausgangspunkte fortgeführt wird. Bei Blöcken mit mehreren Ausgangspunkten beziehen sich die Austrittspunkte auf den Status, der dem Knoten entspricht. Bei einem Zustandsknoten stellen die Austrittspunkte die verschiedenen Bedingungen dar.

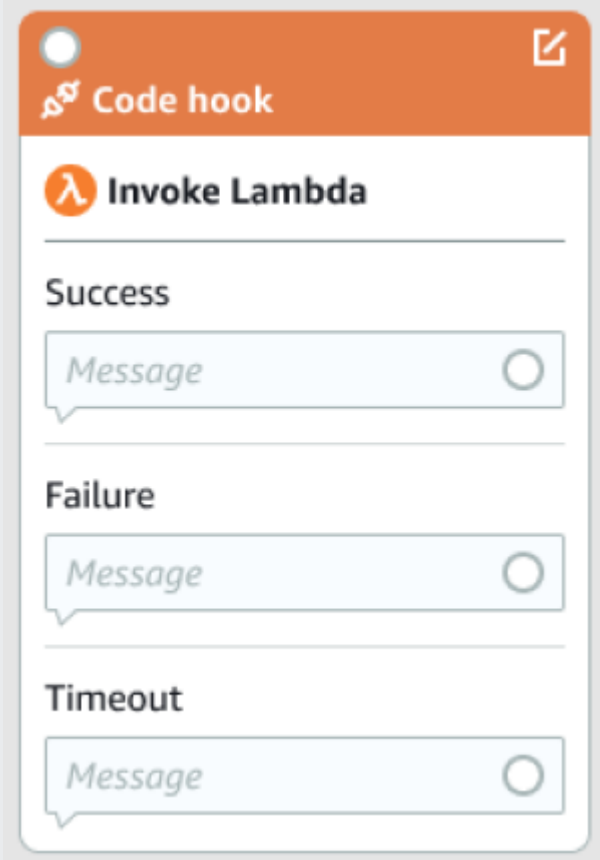
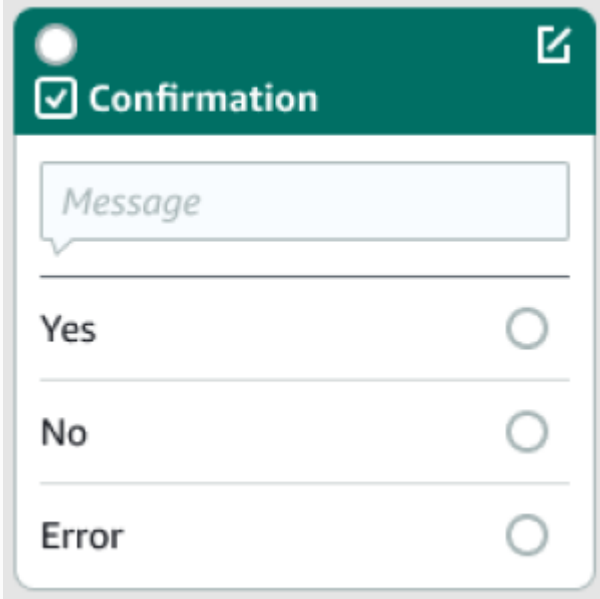
Jeder Block hat ein Konfigurationsfenster, das durch Klicken auf das Bearbeiten-Symbol in der oberen rechten Ecke des Blocks geöffnet wird. Das Konfigurationsfenster enthält detaillierte Felder, die so konfiguriert werden können, dass sie jedem Block entsprechen.

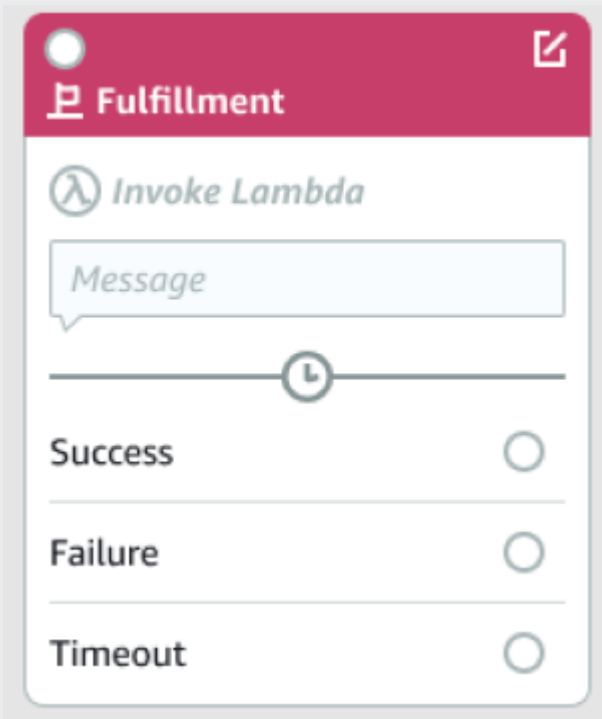
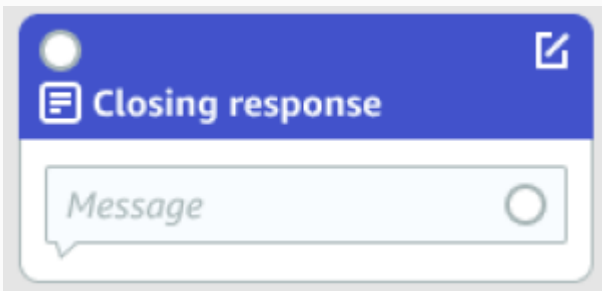
Die Bot-Eingabeaufforderungen und -nachrichten können direkt auf dem Knoten konfiguriert werden, indem ein neuer Block gezogen wird, oder sie können zusammen mit anderen Attributen des Blocks im rechten Bereich geändert werden.

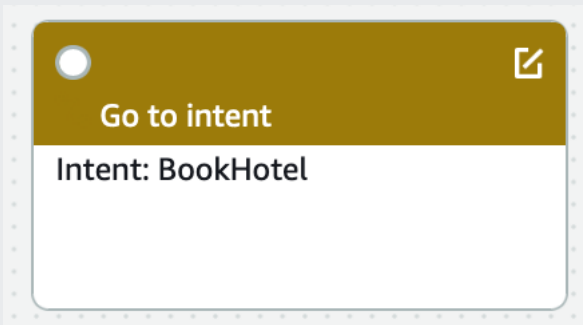
Blocktypen — Hier sind die Blocktypen, die Sie mit Visual Conversation Builder verwenden können.

Blocktyp	Blockieren
Start — Der Stamm oder der erste Block des Konversationsflusses. Dieser Block kann auch so konfiguriert werden, dass der Bot eine erste Antwort senden kann (Nachricht, dass die Absicht erkannt wurde). Weitere Informationen finden Sie unter Erste Antwort.	
Slot-Wert abrufen — Dieser Block versucht, einen Wert für einen einzelnen Slot zu ermitteln. Dieser Block ist so eingestellt, dass auf die Antwort des Kunden auf die Aufforderung zur Auswahl des Steckplatzes gewartet wird. Weitere Informationen finden Sie unter Slots.	

Blocktyp	Blockieren
Bedingung — Dieser Block enthält Bedingungen. Er enthält bis zu 4 benutzerdefinierte Zweige (mit Bedingungen) und einen Standardzweig. Weitere Informationen finden Sie unter Fügen Sie Bedingungen zu Konversationen in Filialen hinzu.	

Blocktyp	Blockieren
Dialogcode-Hook — Dieser Block verarbeitet den Aufruf der Dialog-Lambda-Funktion. Dieser Block enthält Bot-Antworten, die darauf basieren, dass die Dialog-Lambda-Funktion erfolgreich ist, fehlschlägt oder das Timeout auftritt. Weitere Informationen finden Sie unter Rufen Sie den Code-Hook des Dialogs auf.	
Bestätigung — Dieser Block fragt den Kunden ab, bevor die Absicht erfüllt wird. Er enthält Bot-Antworten, die darauf basieren, dass der Kunde die Bestätigungsaufforderung mit Ja oder Nein beantwortet. Weitere Informationen finden Sie unter Bestätigung.	

Blocktyp	Blockieren
Erfüllung — Dieser Block befasst sich mit der Erfüllung von Absichten, in der Regel nach der Eingabe von Slots. Es kann so konfiguriert werden, dass es Lambda-Funktionen aufruft und mit Nachrichten reagiert, wenn die Erfüllung erfolgreich ist oder fehlschlägt. Weitere Informationen finden Sie unter Bereitstellung.	
Antwort beenden — Dieser Block ermöglicht es dem Bot, mit einer Nachricht zu antworten, bevor er die Konversation beendet. Weitere Informationen finden Sie unter Antwort schließen.	
Konversation beenden — Dieser Block zeigt das Ende des Konversationsflusses an.	
Auf Benutzereingaben warten — Dieser Block kann verwendet werden, um Eingaben des Kunden zu erfassen und auf der Grundlage der Äußerung zu einer anderen Absicht zu wechseln.	

Blocktyp	Blockieren
Gehe zur Absicht — Dieser Block kann verwendet werden, um zu einer neuen Absicht zu wechseln oder um direkt einen bestimmten Slot dieser Absicht auszulösen.	

Port-Typen

Alle Blöcke enthalten einen Eingangsport, über den die übergeordneten Blöcke verbunden werden. Die Konversation kann nur vom Ausgangsport des übergeordneten Blocks zum Eingangsport eines bestimmten Blocks fließen. Blöcke können jedoch keinen, einen oder viele Ausgangsanschlüsse enthalten. Die Blöcke ohne Ausgangsports bedeuten das Ende des Konversationsflusses in der aktuellen Absicht (GoToIntent, EndConversation, WaitForUserInput).

Regeln für die Gestaltung von Absichten:

- Alle Flows in einer Absicht beginnen mit dem Startblock.
- Nachrichten, die den einzelnen Ausgangspunkten entsprechen, sind optional.
- Sie können die Blöcke so konfigurieren, dass sie Werte für jeden Austrittspunkt im Konfigurationsfenster festlegen.
- In einem einzigen Flow innerhalb einer Absicht können nur einzelne Start-, Bestätigungs-, Erfüllungs- und Abschlussblöcke existieren. Möglicherweise gibt es mehrere Bedingungen, einen Dialogcode-Hook, das Abrufen von Slot-Werten, das Beenden der Konversation, die Übertragung und das Warten auf Benutzereingaben.
- Ein Bedingungsblock kann keine direkte Verbindung zu einem Bedingungsblock haben. Das Gleiche gilt für den Dialogcode-Hook.
- Zirkuläre Flüsse sind in drei Blöcken zulässig, ein eingehender Anschluss an Start Intent ist jedoch nicht zulässig.
- Ein optionaler Steckplatz hat weder einen eingehenden noch einen ausgehenden Anschluss und wird hauptsächlich zur Erfassung von Daten verwendet, die während der Absichtserkennung vorhanden sind. Jeder andere Steckplatz, der Teil des Konversationspfads ist, muss ein obligatorischer Steckplatz sein.

Blöcke:

- Der Startblock muss eine Ausgangskante haben.
- Jeder Block mit Get-Slot-Werten muss eine ausgehende Flanke vom Success-Port haben, falls der Steckplatz erforderlich ist.
- Jeder Bedingungsblock muss eine ausgehende Flanke von jedem Zweig haben, wenn der Block aktiv ist.
- Ein Bedingungsblock kann nicht mehr als ein übergeordnetes Element haben.
- Ein aktiver Bedingungsblock muss eine eingehende Kante haben.
- Jeder aktive Code-Hook-Block muss über eine ausgehende Flanke von jedem Port aus verfügen: Erfolg, Fehlschlag und Timeout.
- Ein aktiver Code-Hook-Block muss eine eingehende Flanke haben.
- Ein aktiver Bestätigungsblock muss eine eingehende Kante haben.
- Ein aktiver Fulfillment-Block muss eine eingehende Flanke haben.
- Ein aktiver Abschlussblock muss eine Eingangskante haben.
- Ein Bedingungsblock muss mindestens einen Zweig haben, der nicht dem Standard entspricht.
- Für einen Block „Gehe zu Absicht“ muss eine Absicht angegeben werden.

Kanten:

- Ein Bedingungsblock kann nicht mit einem anderen Bedingungsblock verbunden werden.
- Ein Code-Hook-Block kann nicht mit einem anderen Code-Hook-Block verbunden werden.
- Ein Bedingungsblock kann nur mit Null oder einem Code-Hook-Block verbunden werden.
- Die Verbindung (Code-Hook -> Condition -> Code-Hook) ist nicht gültig.
- Ein Fulfillment-Block kann als untergeordnetes Element keinen Code-Hook-Block haben.
- Ein Bedingungsblock, der ein untergeordnetes Element des Erfüllungsblocks ist, kann keinen untergeordneten Code-Hook-Block haben.
- Ein schließender Block kann keinen Code-Hook-Block als untergeordnetes Element haben.
- Ein Bedingungsblock, der dem schließenden Block untergeordnet ist, kann keinen untergeordneten Code-Hook-Block haben.
- Ein Block mit einem Start-, Bestätigungs- oder Get-Slot-Wert kann in seiner Abhängigkeitskette nicht mehr als einen Code-Hook-Block enthalten.

 Note

Am 17. August 2022 veröffentlichte Amazon Lex V2 eine Änderung an der Art und Weise, wie Konversationen mit dem Benutzer verwaltet werden. Diese Änderung gibt Ihnen mehr Kontrolle über den Pfad, den der Benutzer durch die Konversation nimmt. Weitere Informationen finden Sie unter [Änderungen an Konversationsabläufen in Amazon Lex V2](#). Bots, die vor dem 17. August 2022 erstellt wurden, unterstützen keine Dialogcode-Hook-Nachrichten, das Setzen von Werten, das Konfigurieren der nächsten Schritte und das Hinzufügen von Bedingungen.

Integrierte Absichten

Für allgemeine Aktionen können Sie die standardmäßige integrierte Intents-Bibliothek verwenden. Um eine Absicht aus einer integrierten Absicht zu erstellen, wählen Sie eine integrierte Absicht in der Konsole und weisen ihr einen neuen Namen zu. Die neue Absicht hat die Konfiguration der Basisabsicht, wie z. B. die Beispieläußerungen.

In der aktuellen Implementierung ist Folgendes nicht möglich:

- Fügen Sie der Basisabsicht Beispieläußerungen hinzu oder entfernen Sie sie daraus
- Konfigurieren von Slots für integrierte Absichten

Um einem Bot eine integrierte Absicht hinzuzufügen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie den Bot aus, dem Sie die integrierte Absicht hinzufügen möchten.
3. Wählen Sie im linken Menü die Sprache und dann Intents aus.
4. Wählen Sie „Absicht hinzufügen“ und anschließend „Integrierte Absicht verwenden“.
5. Wählen Sie unter Integrierte Absicht die Absicht aus, die Sie verwenden möchten.
6. Geben Sie der Absicht einen Namen und wählen Sie dann Hinzufügen aus.
7. Verwenden Sie den Absichtseditor, um die Absicht so zu konfigurieren, wie es für Ihren Bot erforderlich ist.

Themen

- [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

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).
2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Aktiviert Amazon Bedrock-Agenten, die in der Absicht definiert sind, auf Kundenanfragen zu antworten und Agenten-Workflows zu aktivieren, um die definierte Aufgabe zu erfüllen. Diese Funktion ist in allen von Amazon Lex V2 unterstützten Regionen und allen Handelsregionen verfügbar, in denen sowohl Amazon Lex V2 als auch Amazon Bedrock Agents präsent sind.

Wenn diese Absicht Vorrang hat `FallbackIntent`, wird die Absicht aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet ist. Andernfalls wird sie nur aktiviert, wenn eine Äußerung dieser Absicht zugeordnet wird. Es ist wichtig zu beachten, dass diese Absicht nicht für verpasste Äußerungen aktiviert wird, wenn ein Slot-Wert ausgelöst wird.

Sobald es von Ihrem Amazon Lex V2-Bot erkannt wurde `AMAZON.BedrockAgentIntent`, aktiviert der, den definierten `BedrockAgent` oder `BedrockKnowledgeBase` um dem Kunden zu antworten. Wenn Sie Amazon Bedrock Agents verwenden, bleibt die Konversation im `BedrockAgentIntent` und die Benutzeranfragen werden an die Agenten weitergeleitet, bis der Amazon Bedrock Agent feststellt, dass die Konversation markiert ist. `FINISH` Erst danach übernimmt Amazon Lex V2 die Kontrolle über die Konversation und hält sich an die nächsten Schritte, die in der `AMAZON.BedrockAgentIntent` definiert sind.

Beantwortet Kundenfragen mithilfe von Amazon Bedrock Agents and Knowledge Bases, um Kundenfragen zu beantworten und detaillierte Antworten zu geben.

Warning

Sie können das nicht `AMAZON.BedrockAgentIntent` ohne Beispieläußerungen, `AMAZON.QnAIntent` ohne Beispieläußerungen und `AMAZON.KendraSearchIntent` im selben Bot-Gebietsschema verwenden.

Wenn Sie diese Absicht auswählen, konfigurieren Sie die folgenden Felder und wählen dann Hinzufügen aus, um die Absicht hinzuzufügen.

- Amazon Bedrock Agent Id — Die ID des Amazon Bedrock Agents. Wählen Sie den Bedrock Agent aus, den Sie verwenden möchten.
- Amazon Bedrock Agent Alias Id — Die Alias-ID des Amazon Bedrock Agents.

Important

Wenn Sie den Amazon Bedrock Agent für die Verwendung mit Amazon Lex V2 erstellen, stellen Sie sicher, dass Benutzereingaben unter Zusätzliche Einstellungen aktiviert sind `ENABLED`. Diese Einstellung ist wichtig, damit Agenten klärende oder weiterführende Fragen stellen können, und sie ermöglicht Amazon Lex V2, die Erledigung der entsprechenden Aufgabe wieder an Agenten zu delegieren.

(Optional) Sie können auch eine `BedrockAgentIntent` mit den folgenden Optionen hinzufügen:

- **Amazon Bedrock-Modell** — Wählen Sie den Anbieter und das Gründungsmodell, das Sie für diese Absicht verwenden möchten. Derzeit werden Anthropic Claude V2, Anthropic Claude 3 Haiku, Anthropic Claude 3 Haiku und Anthropic Claude Instant unterstützt.
- **Amazon Bedrock Knowledge Base** — Wenn Sie diese Option wählen, geben Sie die ID der Amazon Bedrock Knowledge Base an. Sie finden die ID auf der Detailseite der Amazon Bedrock Knowledge Base in der Konsole oder indem `GetKnowledgeBase` Sie eine Anfrage senden.

Die Antworten von `BedrockAgentIntent` werden in den Sitzungs- und Anforderungsattributen gespeichert, wie unten dargestellt:

- `x-amz-lex:bedrock-agent-search-response`— Die Antwort des Amazon Bedrock-Agenten auf die Frage oder Äußerung.
- `x-amz-lex:bedrock-knowledge-base-search-response-source`— Verweist auf das Dokument oder die Liste von Dokumenten, die zur Generierung der Antwort verwendet wurden, wenn Sie die Amazon Bedrock Knowledge Base-Konfiguration verwenden.
- `x-amz-lex:bedrock-agent-action-group-invocation-input`— Objekt, das Eingabewerte enthält, die von der Aktionsgruppe des Agenten gesammelt wurden. Weitere Informationen zu Aktionsgruppen für Agenten finden Sie unter `ActionGroupInvocationInput`.
- `x-amz-lex:bedrock-agent-knowledge-base-lookup-input`— Objekt, das die Informationen zur Suche in der Amazon Bedrock Knowledge Base enthält.
- `x-amz-lex:bedrock-agent-agent-collaborator-details`— Objekt, das Details zur Eingabe und Ausgabe der Unteragenten enthält, die im Rahmen von Aufrufen zur Zusammenarbeit mit mehreren Agenten aufgerufen wurden.

Weitere Informationen finden Sie unter [Verwenden BedrockAgentIntent eines Bedrock Agents in Amazon Lex V2](#).

AMAZON.CancelIntent

Reagiert auf Wörter und Ausdrücke, die darauf hinweisen, dass der Benutzer die aktuelle Interaktion abbrechen möchte. Ihre Anwendung kann diese Absicht nutzen, um Slot-Typ-Werte und andere Attribute zu entfernen, bevor die Interaktion mit dem Benutzer beendet wird.

Allgemeine Äußerungen:

- abbrechen
- macht nichts
- vergiss es

AMAZON.FallbackIntent

Wenn die Eingabe eines Benutzers zu einer Absicht nicht den Erwartungen eines Bot entspricht, können Sie Amazon Lex V2 so konfigurieren, dass eine Fallback-Absicht aufgerufen wird. Wenn beispielsweise die Benutzereingabe „Ich möchte Süßigkeiten bestellen“ nicht mit einer Absicht in Ihrem `OrderFlowers` Bot übereinstimmt, ruft Amazon Lex V2 die Fallback-Absicht auf, um die Antwort zu bearbeiten.

Der integrierte `AMAZON.FallbackIntent` Absichtstyp wird Ihrem Bot automatisch hinzugefügt, wenn Sie einen Bot mithilfe der Konsole erstellen oder wenn Sie einem Bot mithilfe der Operation ein Gebietsschema hinzufügen. [CreateBotLocale](#)

Das Aufrufen einer Fallback-Absicht verwendet zwei Schritte. Im ersten Schritt wird die Fallback-Absicht basierend auf der Eingabe des Benutzers abgeglichen. Wenn die Fallback-Absicht übereinstimmt, hängt das Verhalten des Bots von der Anzahl der Wiederholungen ab, die für eine Eingabeaufforderung konfiguriert wurden.

Amazon Lex V2 entspricht der Fallback-Absicht in den folgenden Situationen:

- Die Eingabe des Benutzers für eine Absicht stimmt nicht mit der Eingabe überein, die der Bot erwartet
- Audioeingabe ist Rauschen, oder Texteingaben werden nicht als Wörter erkannt.
- Die Benutzereingabe ist mehrdeutig und Amazon Lex V2 kann nicht bestimmen, welche Absicht aufgerufen werden soll.

Die Fallback-Absicht wird aufgerufen, wenn:

- Eine Absicht erkennt die Benutzereingabe nach der konfigurierten Anzahl von Versuchen nicht als Slot-Wert.
- Eine Absicht erkennt die Benutzereingabe nicht als Antwort auf eine Bestätigungsaufforderung nach der konfigurierten Anzahl von Versuchen.

Es ist nicht möglich, Folgendes zu einer Fallback-Absicht hinzuzufügen:

- Äußerungen
- Slots
- Eine Bestätigungsaufforderung

Verwenden einer Lambda-Funktion mit einer Fallback-Absicht

Wenn eine Fallback-Absicht aufgerufen wird, hängt die Antwort von der Einstellung des Parameters `fulfillmentCodeHook` für die Operation [CreateIntent](#) ab. Der Bot führt einen der folgenden Schritte aus:

- Gibt die Absicht-Informationen an die Client-Anwendung zurück.
- Ruft die Lambda-Funktion für Validierung und Erfüllung des Alias auf. Sie ruft die Funktion mit den Sitzungsvariablen auf, die für die Sitzung festgelegt sind.

Weitere Informationen zum Festlegen der Antwort, wenn eine Fallback-Absicht aufgerufen wird, finden Sie im Parameter `fulfillmentCodeHook` der [CreateIntent](#)-Operation.

Wenn Sie die Lambda-Funktion mit Ihrer Fallback-Absicht verwenden, können Sie diese Funktion verwenden, um eine andere Absicht aufzurufen oder um irgendeine Form der Kommunikation mit dem Benutzer durchzuführen, z. B. eine Rückrufnummer zu sammeln oder eine Sitzung mit einem Kundendienstmitarbeiter zu eröffnen.

Eine Fallback-Absicht kann mehrmals in derselben Sitzung aufgerufen werden. Nehmen wir beispielsweise an, dass Ihre Lambda-Funktion die `ElicitIntent` Dialogaktion verwendet, um den Benutzer zu einer anderen Absicht aufzufordern. Wenn Amazon Lex V2 nach der konfigurierten Anzahl von Versuchen nicht auf die Absicht des Benutzers schließen kann, ruft es die Fallback-Absicht erneut auf. Außerdem wird die Fallback-Absicht aufgerufen, wenn der Benutzer nach der konfigurierten Anzahl von Versuchen nicht mit einem gültigen Slot-Wert antwortet.

Sie können Ihre Lambda-Funktion so konfigurieren, dass sie mithilfe einer Sitzungsvariablen verfolgt, wie oft die Fallback-Absicht aufgerufen wird. Ihre Lambda-Funktion kann eine andere Aktion ausführen, wenn sie öfter aufgerufen wird als der Schwellenwert, den Sie in Ihrer Lambda-Funktion festgelegt haben. Weitere Informationen zu Sitzungsvariablen finden Sie unter [Sitzungsattribute für Ihren Lex V2-Bot festlegen](#).

AMAZON.HelpIntent

Reagiert auf Wörter oder Ausdrücke, die darauf hinweisen, dass der Benutzer bei der Interaktion mit Ihrem Bot Hilfe benötigt. Wenn diese Absicht aufgerufen wird, können Sie Ihre Lambda-Funktion oder -Anwendung so konfigurieren, dass sie Informationen über die Fähigkeiten Ihres Bots bereitstellt, Folgefragen zu Hilfebereichen stellt oder die Interaktion einem menschlichen Agenten übergibt.

Allgemeine Äußerungen:

- help
- hilf mir
- kannst du mir helfen

AMAZON.KendraSearchIntent

Verwenden Sie die Absicht, um nach Dokumenten zu suchen, die Sie mit Amazon Kendra indexiert haben. AMAZON.KendraSearchIntent Wenn Amazon Lex V2 die nächste Aktion in einer Konversation mit dem Benutzer nicht bestimmen kann, löst es die Suchabsicht aus.

Das AMAZON.KendraSearchIntent ist nur in den Regionen Englisch (US) (en-US) und in den Regionen USA Ost (Nord-Virginia), USA West (Oregon) und Europa (Irland) verfügbar.

Amazon Kendra ist ein machine-learning-based Suchdienst, der Dokumente in natürlicher Sprache wie PDF-Dokumente oder Microsoft Word-Dateien indexiert. Es kann indizierte Dokumente durchsuchen und die folgenden Arten von Antworten auf Fragen zurückgeben:

- Antworten
- Einträge aus häufig gestellten Fragen, die die Fragen möglicherweise beantworten
- Dokumente, die sich auf die Fragen beziehen

Ein Beispiel für die Verwendung von AMAZON.KendraSearchIntent finden Sie unter [Beispiel: Einen FAQ-Bot für einen Amazon Kendra Kendra-Index erstellen](#).

Wenn Sie eine AMAZON.KendraSearchIntent Absicht für Ihren Bot konfigurieren, ruft Amazon Lex V2 die Absicht immer dann auf, wenn die Benutzeräußerung für eine Absicht nicht bestimmt werden kann. Wenn keine Antwort von Amazon Kendra eingeht, wird die Konversation wie im Bot konfiguriert fortgesetzt.

 Note

Amazon Lex V2 unterstützt derzeit das `AMAZON.KendraSearchIntent` während der Slot-Erfassung nicht. Wenn Amazon Lex V2 die Benutzeräußerung für einen Slot nicht ermitteln kann, ruft es den `AMAZON.FallbackIntent`.

Wenn Sie `AMAZON.KendraSearchIntent` mit dem `AMAZON.FallbackIntent` im selben Bot verwenden, verwendet Amazon Lex V2 die Absichten wie folgt:

1. Amazon Lex V2 nennt das `AMAZON.KendraSearchIntent`. Die Absicht nennt Amazon Kendra Query Operation.
2. Wenn Amazon Kendra eine Antwort zurückgibt, zeigt Amazon Lex V2 dem Benutzer das Ergebnis an.
3. Wenn Amazon Kendra keine Antwort erhält, fordert Amazon Lex V2 den Benutzer erneut auf. Die nächste Aktion hängt von der Antwort des Benutzers ab.
 - Wenn die Antwort des Benutzers eine Äußerung enthält, die Amazon Lex V2 erkennt, z. B. das Ausfüllen eines Slot-Werts oder die Bestätigung einer Absicht, wird die Konversation mit dem Benutzer wie für den Bot konfiguriert fortgesetzt.
 - Wenn die Antwort des Benutzers keine Äußerung enthält, die Amazon Lex V2 erkennt, ruft Amazon Lex V2 den Query Vorgang erneut auf.
4. Wenn nach der konfigurierten Anzahl von Wiederholungen keine Antwort erfolgt, ruft Amazon Lex V2 den `AMAZON.FallbackIntent` und beendet die Konversation mit dem Benutzer.

Es gibt drei Möglichkeiten, mit dem eine Anfrage `AMAZON.KendraSearchIntent` an Amazon Kendra zu stellen:

- Lassen Sie die Suchabsicht die Anfrage für Sie stellen. Amazon Lex V2 ruft Amazon Kendra mit der Äußerung des Benutzers als Suchzeichenfolge auf. Wenn Sie die Absicht erstellen, können Sie eine Abfragefilterzeichenfolge definieren, die die Anzahl der Antworten begrenzt, die Amazon Kendra zurückgibt. Amazon Lex V2 verwendet den Filter in der Abfrageanforderung.
- Fügen Sie der Anfrage zusätzliche Abfrageparameter hinzu, um die Suchergebnisse mithilfe Ihrer Lambda-Funktion einzugrenzen. Sie fügen der `delegate` Dialogaktion ein `kendraQueryFilterString` Feld hinzu, Amazon Kendra Kendra-Abfrageparameter enthält. Wenn Sie der Anfrage mit der Lambda-Funktion Abfrageparameter hinzufügen, haben diese Vorrang vor dem Abfragefilter, den Sie bei der Erstellung der Absicht definiert haben.

- Erstellen Sie eine neue Abfrage mit der Lambda-Funktion. Sie können eine vollständige Amazon Kendra Kendra-Abfrageanforderung erstellen, die Amazon Lex V2 sendet. Sie legen die Abfrage im Feld `kendraQueryRequestPayload` in der Dialogaktion `delegate` fest. Das Feld `kendraQueryRequestPayload` hat Vorrang vor dem Feld `kendraQueryFilterString`.

Um den `queryFilterString` Parameter anzugeben, wenn Sie einen Bot erstellen, oder um das `kendraQueryFilterString` Feld anzugeben, wenn Sie die `delegate` Aktion in einer Dialog-Lambda-Funktion aufrufen, geben Sie eine Zeichenfolge an, die als Attributfilter für die Amazon Kendra Kendra-Abfrage verwendet wird. Wenn die Zeichenfolge kein gültiger Attributfilter ist, wird zur Laufzeit die Ausnahme `InvalidBotConfigException` zurückgegeben. Weitere Informationen zu Attributfiltern finden Sie unter [Verwenden von Dokumentattributen zum Filtern von Abfragen](#) im Amazon Kendra Developer Guide.

Um die Abfrage zu kontrollieren, die Amazon Lex V2 an Amazon Kendra sendet, können Sie in dem `kendraQueryRequestPayload` Feld Ihrer Lambda-Funktion eine Abfrage angeben. Wenn die Abfrage nicht gültig ist, gibt Amazon Lex V2 eine `InvalidLambdaResponseException` Ausnahme zurück. Weitere Informationen finden Sie unter [Abfragevorgang](#) im Amazon Kendra Developer Guide.

Ein Beispiel für die Verwendung von `AMAZON.KendraSearchIntent` finden Sie unter [Beispiel: Einen FAQ-Bot für einen Amazon Kendra Kendra-Index erstellen](#).

IAM-Richtlinie für Amazon Kendra Search

Um den `AMAZON.KendraSearchIntent` Intent zu verwenden, müssen Sie eine Rolle verwenden, die AWS Identity and Access Management (IAM) -Richtlinien bereitstellt, die es Amazon Lex V2 ermöglichen, eine Runtime-Rolle anzunehmen, die berechtigt ist, den Amazon Kendra Query Kendra-Intent aufzurufen. Die IAM-Einstellungen, die Sie verwenden, hängen davon ab, ob Sie die `AMAZON.KendraSearchIntent` mit der Amazon Lex V2-Konsole oder mit einem AWS-SDK oder dem AWS Command Line Interface (AWS CLI) erstellen. Wenn Sie die Konsole verwenden, können Sie wählen, ob Sie der serviceverknüpften Amazon Lex V2-Rolle die Berechtigung zum Aufrufen von Amazon Kendra hinzufügen oder eine Rolle speziell für den Aufruf des Amazon Query Kendra Kendra-Vorgangs verwenden möchten. Wenn Sie das AWS CLI oder ein SDK verwenden, um die Absicht zu erstellen, müssen Sie eine Rolle verwenden, die speziell für den Aufruf der Operation bestimmt ist. `Query`

Anfügen von Berechtigungen

Sie können die Konsole verwenden, um der standardmäßigen serviceverknüpften Amazon Lex V2-Rolle Berechtigungen für den Zugriff auf den Amazon Query Kendra-Vorgang zuzuweisen. Wenn Sie

der serviceverknüpften Rolle Berechtigungen zuordnen, müssen Sie keine spezielle Runtime-Rolle erstellen und verwalten, um eine Verbindung mit dem Amazon Kendra Kendra-Index herzustellen.

Der Benutzer, die Rolle oder die Gruppe, die Sie für den Zugriff auf die Amazon Lex V2-Konsole verwenden, muss über Berechtigungen zur Verwaltung von Rollenrichtlinien verfügen. Fügen Sie der Konsolenzugriffsrolle die folgende IAM-Richtlinie hinzu. Wenn Sie diese Berechtigungen erteilen, verfügt die Rolle über Berechtigungen zum Ändern der vorhandenen Richtlinie für die serviceverknüpfte Rolle.

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": "*"
    }
  ]
}
```

Angeben einer Rolle

Sie können die Konsole, die oder die API verwenden AWS CLI, um eine Laufzeitrolle anzugeben, die beim Aufrufen des Amazon Kendra Query Kendra-Vorgangs verwendet werden soll.

Der Benutzer, die Rolle oder die Gruppe, die Sie zur Angabe der Runtime-Rolle verwenden, muss über die `iam:PassRole` entsprechende Berechtigung verfügen. Die folgende Richtlinie definiert die Berechtigung. Sie können die Bedingungskontextschlüssel `iam:AssociatedResourceArn` und

`iam:PassedToService` verwenden, um den Umfang der Berechtigungen weiter einzuschränken. Weitere Informationen finden Sie unter [IAM und AWS STS Condition Context Keys](#) im AWS Identity and Access Management Benutzerhandbuch.

JSON

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

Die Runtime-Rolle, die Amazon Lex V2 zum Aufrufen von Amazon Kendra verwenden muss, muss über die `kendra:Query` entsprechenden Berechtigungen verfügen. Wenn Sie eine bestehende IAM-Rolle für die Erlaubnis zum Aufrufen des Amazon Kendra Query Kendra-Vorgangs verwenden, muss der Rolle die folgende Richtlinie beigefügt sein.

Sie können die IAM-Konsole, die IAM-API oder die verwenden, um eine Richtlinie AWS CLI zu erstellen und sie einer Rolle zuzuordnen. In diesen Anweisungen wird die AWS CLI zum Erstellen der Rolle und Richtlinien verwendet.

Note

Der folgende Code ist für Linux und MacOS formatiert. Ersetzen Sie unter Windows das Linux-Zeilenumbruchzeichen (`\n`) durch ein Caret-Zeichen (`^`).

So fügen Sie einer Rolle die Berechtigung für die Query-Operation hinzu

1. Erstellen Sie im aktuellen Verzeichnis ein Dokument mit dem Namen **KendraQueryPolicy.json**, fügen Sie ihm folgenden Code hinzu und speichern Sie es.

JSON

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

2. Führen Sie in der den folgenden Befehl aus AWS CLI, um die IAM-Richtlinie für die Ausführung des Amazon Kendra Query Kendra-Vorgangs zu erstellen.

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

3. Hängen Sie die Richtlinie an die IAM-Rolle an, mit der Sie den Vorgang aufrufen. Query

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

Sie können wählen, ob Sie die serviceverknüpfte Amazon Lex V2-Rolle aktualisieren oder eine Rolle verwenden möchten, die Sie bei der Erstellung der AMAZON.KendraSearchIntent für Ihren Bot erstellt haben. Das folgende Verfahren zeigt, wie Sie die zu verwendende IAM-Rolle auswählen.

Um die Runtime-Rolle für anzugeben AMAZON.KendraSearchIntent

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie den Bot, dem Sie AMAZON.KendraSearchIntent hinzufügen möchten.

3. Wählen Sie das Pluszeichen (+) neben Intents (Absichten).
4. Wählen Sie unter Add intent (Absicht hinzufügen) die Option Search existing intents (Vorhandene Absichten durchsuchen).
5. Geben Sie unter Search intents (Absichten suchen) **AMAZON.KendraSearchIntent** ein und wählen Sie dann Add (Hinzufügen).
6. Geben Sie unter Copy built-in intent (Integrierte Absicht kopieren) einen Namen für die Absicht ein, z. B. **KendraSearchIntent**, und wählen Sie dann Add (Hinzufügen).
7. Öffnen Sie den Abschnitt Amazon Kendra query (Amazon Kendra-Abfrage).
8. Wählen Sie unter IAM role (IAM-Rolle) eine der folgenden Optionen:
 - Um die serviceverknüpfte Amazon Lex V2-Rolle zu aktualisieren, sodass Ihr Bot Amazon Kendra Kendra-Indizes abfragen kann, wählen Sie Amazon Kendra Kendra-Berechtigungen hinzufügen.
 - Um eine Rolle zu verwenden, die berechtigt ist, den Amazon Kendra Query Kendra-Vorgang aufzurufen, wählen Sie Bestehende Rolle verwenden.

Verwenden von Anforderungs- und Sitzungsattributen als Filter

Um die Antwort von Amazon Kendra auf Elemente zu filtern, die sich auf die aktuelle Konversation beziehen, verwenden Sie Sitzungs- und Anforderungsattribute als Filter, indem Sie den `queryFilterString` Parameter hinzufügen, wenn Sie Ihren Bot erstellen. Sie geben einen Platzhalter für das Attribut an, wenn Sie die Absicht erstellen, und dann ersetzt Amazon Lex V2 einen Wert, bevor Amazon Kendra aufgerufen wird. Weitere Informationen zu Anforderungsattributen finden Sie unter [Einstellung von Anforderungsattributen für Ihren Lex V2-Bot](#). Weitere Informationen über Sitzungsattribute finden Sie unter [Sitzungsattribute für Ihren Lex V2-Bot festlegen](#).

Im Folgenden finden Sie ein Beispiel für einen `queryFilterString` Parameter, der eine Zeichenfolge verwendet, um die Amazon Kendra Kendra-Abfrage zu filtern.

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

Im Folgenden finden Sie ein Beispiel für einen `queryFilterString` Parameter, der ein Sitzungsattribut verwendet, das aufgerufen wird "SourceURI", um die Amazon Kendra Kendra-Abfrage zu filtern.

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

Im Folgenden finden Sie ein Beispiel für einen `queryFilterString` Parameter, der ein Anforderungsattribut verwendet, das aufgerufen wird `"DepartmentName"`, um die Amazon Kendra Kendra-Abfrage zu filtern.

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

Die `AMAZON.KendraSearchIntent` Filter verwenden dasselbe Format wie die Amazon Kendra Kendra-Suchfilter. Weitere Informationen finden Sie unter [Verwenden von Dokumentattributen zum Filtern von Suchergebnissen](#) im Amazon Kendra Developer Guide.

Die mit dem verwendete Abfragefilterzeichenfolge `AMAZON.KendraSearchIntent` muss Kleinbuchstaben für den ersten Buchstaben jedes Filters verwenden. Der folgende ist beispielsweise ein gültiger Abfragefilter für `AMAZON.KendraSearchIntent`

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

Verwenden der Suchantwort

Amazon Kendra gibt die Antwort auf eine Suche als Antwort auf die Absichtserklärung zurück. `IntentClosingSetting` Die Absicht muss eine `closingResponse` Aussage enthalten, es sei denn, eine Lambda-Funktion erzeugt eine abschließende Antwortnachricht.

Amazon Kendra bietet fünf Arten von Antworten.

- Für die folgenden beiden Antworten muss eine häufig gestellte Frage für Ihren Amazon Kendra Kendra-Index eingerichtet werden. Weitere Informationen finden Sie unter [Fragen und Antworten direkt zu einem Index hinzufügen](#).
 - `x-amz-lex:kendra-search-response-question_answer-question-<N>`— Die Frage aus einer häufig gestellten Frage, die der Suche entspricht.
 - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— Die Antwort aus einer häufig gestellten Frage, die der Suche entspricht.
- Für die folgenden drei Antworten muss eine Datenquelle für Ihren Amazon Kendra Kendra-Index eingerichtet werden. Weitere Informationen finden Sie unter [Erstellen einer Datenquelle](#).
 - `x-amz-lex:kendra-search-response-document-<N>`— Ein Auszug aus einem Dokument im Index, der sich auf den Text der Äußerung bezieht.
 - `x-amz-lex:kendra-search-response-document-link-<N>`— Die URL eines Dokuments im Index, das sich auf den Text der Äußerung bezieht.
 - `x-amz-lex:kendra-search-response-answer-<N>`— Ein Auszug aus einem Dokument im Index, das die Frage beantwortet.

Die Antworten werden in `request`-Attributen zurückgegeben. Für jedes Attribut kann es bis zu fünf Antworten geben, nummeriert von 1 bis 5. Weitere Informationen zu Antworten finden Sie unter [Antworttypen](#) im Amazon Kendra Developer Guide.

Die Anweisung `closingResponse` muss eine oder mehrere Nachrichtengruppen aufweisen. Jede Nachrichtengruppe enthält eine oder mehrere Nachrichten. Jede Nachricht kann eine oder mehrere Platzhaltervariablen enthalten, die in der Antwort von Amazon Kendra durch Anforderungsattribute ersetzt werden. In der Nachrichtengruppe muss mindestens eine Nachricht vorhanden sein, in der alle Variablen in der Nachricht durch Anforderungsattributwerte in der Laufzeitantwort ersetzt werden, oder in der Gruppe muss eine Nachricht ohne Platzhaltervariablen vorhanden sein. Die Anforderungsattribute werden durch doppelte Klammern ("`((\" \"))`") hervorgehoben. Die folgenden Nachrichtengruppennachrichten stimmen mit allen Antworten von Amazon Kendra überein:

- „Ich habe eine häufig gestellte Frage für Sie gefunden: `((x-amz-lex: kendra-search-response-question _answer-question-1))`, und die Antwort lautet `((: _answer-answer-1))`“ `x-amz-lex kendra-search-response-question`
- „Ich habe einen Auszug aus einem hilfreichen Dokument gefunden: `((: -1))`“ `x-amz-lex kendra-search-response-document`
- „Ich denke, die Antwort auf Ihre Fragen lautet `((x-amz-lex: kendra-search-response-answer -1))`“

Verwenden einer Lambda-Funktion zur Verwaltung der Anfrage und Antwort

Die `AMAZON.KendraSearchIntent` Absicht kann Ihren Dialog-Code-Hook und Ihren Fulfillment-Code-Hook verwenden, um die Anfrage an Amazon Kendra und die Antwort zu verwalten. Verwenden Sie die Dialogcode-Hook-Lambda-Funktion, wenn Sie die Anfrage ändern möchten, die Sie an Amazon Kendra senden, und die Lambda-Funktion für den Fulfillment-Code-Hook, wenn Sie die Antwort ändern möchten.

Erstellen einer Abfrage mit dem Dialogcode-Hook

Sie können den Dialog-Code-Hook verwenden, um eine Abfrage zu erstellen, die an Amazon Kendra gesendet werden soll. Die Verwendung des Dialogcode-Hooks ist optional. Wenn Sie keinen Dialog-Code-Hook angeben, erstellt Amazon Lex V2 eine Abfrage aus der Benutzeräußerung und verwendet die `queryFilterString`, die Sie bei der Konfiguration der Absicht angegeben haben, sofern Sie eine angegeben haben.

Sie können zwei Felder in der Dialog-Code-Hook-Antwort verwenden, um die Anfrage an Amazon Kendra zu ändern:

- `kendraQueryFilterString`— Verwenden Sie diese Zeichenfolge, um Attributfilter für die Amazon Kendra Kendra-Anforderung anzugeben. Sie können die Abfrage mithilfe eines beliebigen in Ihrem Index definierten Indexfelds filtern. Informationen zur Struktur der Filterzeichenfolge finden Sie unter [Verwenden von Dokumentattributen zum Filtern von Abfragen](#) im Amazon Kendra Developer Guide. Wenn die angegebene Filterzeichenfolge ungültig ist, erhalten Sie die Ausnahme `InvalidLambdaResponseException`. Die `kendraQueryFilterString`-Zeichenfolge überschreibt alle Abfragezeichenfolgen, die im für diese Absicht konfigurierten `queryFilterString` angegeben sind.
- `kendraQueryRequestPayload`— Verwenden Sie diese Zeichenfolge, um eine Amazon Kendra Kendra-Abfrage anzugeben. Ihre Anfrage kann alle Funktionen von Amazon Kendra verwenden. Wenn Sie keine gültige Abfrage angeben, erhalten Sie die Ausnahme `InvalidLambdaResponseException`. Weitere Informationen finden Sie unter [Query](#) im Amazon Kendra Developer Guide.

Nachdem Sie den Filter oder die Abfragezeichenfolge erstellt haben, senden Sie die Antwort an Amazon Lex V2, wobei das `dialogAction` Feld der Antwort auf `delegate` gesetzt ist. Amazon Lex V2 sendet die Anfrage an Amazon Kendra und sendet dann die Abfrageantwort an den Fulfillment-Code-Hook zurück.

Verwenden des Erfüllungscode-Hooks für die Antwort

Nachdem Amazon Lex V2 eine Anfrage an Amazon Kendra gesendet hat, wird die Abfrageantwort an die `AMAZON.KendraSearchIntent` Fulfillment-Lambda-Funktion zurückgegeben. Das Eingabeereignis für den Code-Hook enthält die vollständige Antwort von Amazon Kendra. Die Abfragedaten haben dieselbe Struktur wie die, die von der Amazon Kendra Query Kendra-Operation zurückgegeben wurde. Weitere Informationen finden Sie unter [Syntax der Abfrageantwort](#) im Amazon Kendra Developer Guide.

Der Erfüllungscode-Hook ist optional. Wenn keine vorhanden ist oder wenn der Code-Hook keine Nachricht in der Antwort zurückgibt, verwendet Amazon Lex V2 die `closingResponse` Anweisung für Antworten.

Beispiel: Einen FAQ-Bot für einen Amazon Kendra Kendra-Index erstellen

In diesem Beispiel wird ein Amazon Lex V2-Bot erstellt, der einen Amazon Kendra Kendra-Index verwendet, um Antworten auf Benutzerfragen zu geben. Der Bot zu häufig gestellten Fragen verwaltet den Dialog für den Benutzer. Er verwendet die Absicht `AMAZON.KendraSearchIntent`, um den Index abzufragen und die Antwort für den Benutzer bereitzustellen. Hier ist eine Zusammenfassung, wie Sie Ihren FAQ-Bot mithilfe eines Amazon Kendra Kendra-Index erstellen werden:

1. Erstellen Sie einen Bot, mit dem Ihre Kunden interagieren werden, um Antworten von diesem Bot zu erhalten.
2. Erstellen Sie eine benutzerdefinierte Absicht. Da es sich bei `AMAZON.KendraSearchIntent` und um Backup-Absichten `AMAZON.FallbackIntent` handelt, benötigt Ihr Bot mindestens eine weitere Absicht, die mindestens eine Äußerung enthalten muss. Diese Absicht ermöglicht die Entwicklung des Bots, wird anderweitig jedoch nicht verwendet. Ihr FAQ-Bot wird daher mindestens drei Absichten enthalten, wie in der Abbildung unten dargestellt:

Amazon Lex

Lex > Bots > Bot: KendraTestBot > Versions > Version: DRAFT > All languages > Language: English (US) > Intents

Draft version English (US) Successfully built English (US) has not built changes. Build Test

Intents (3) Info

An intent represents an action that the user wants to perform.

Search Intents

	Name	Description	Last edited
☐	KendraSearchIntent	Intent to ask a question. This intent searches a Kendra index for an answer to the question.	1 minute ago
☐	RequiredIntent	Intent required for bot to build	7 minutes ago
☐	FallbackIntent	Default intent when no other intent matches	1 month ago

3. Fügen Sie die AMAZON.KendraSearchIntent Absicht zu Ihrem Bot hinzu und konfigurieren Sie ihn so, dass er mit Ihrem [Amazon Kendra Kendra-Index](#) funktioniert.
4. Testen Sie den Bot, indem Sie eine Abfrage stellen und überprüfen, ob es sich bei den Ergebnissen Ihres Amazon Kendra Kendra-Index um Dokumente handelt, die die Anfrage beantworten.

Voraussetzungen

Bevor Sie dieses Beispiel verwenden können, müssen Sie einen Amazon Kendra Kendra-Index erstellen. Weitere Informationen finden Sie unter [Erste Schritte mit der Amazon Kendra-Konsole](#) im Amazon Kendra Developer Guide. Wählen Sie für dieses Beispiel den Beispieldatensatz (AWS-Beispieldokumentation) als Datenquelle aus.

Um einen FAQ-Bot zu erstellen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie im Navigationsbereich Bots.
3. Wählen Sie Create bot aus.
 - a. Wählen Sie für die Erstellungsmethode Create a blank bot aus.
 - b. Geben Sie dem Bot im Abschnitt Bot-Konfiguration einen Namen, der seinen Zweck angibt **KendraTestBot**, z. B. und eine optionale Beschreibung. Der Name muss in Ihrem Konto eindeutig sein.

- c. Wählen Sie im Abschnitt IAM-Berechtigungen die Option Rolle mit grundlegenden Amazon Lex-Berechtigungen erstellen aus. Dadurch wird eine [AWS Identity and Access Management \(IAM-\)](#) Rolle mit den Berechtigungen erstellt, die Amazon Lex V2 benötigt, um Ihren Bot auszuführen.
- d. Wählen Sie im Abschnitt Gesetz zum Schutz der Privatsphäre von Kindern im Internet (COPPA) die Option Nein aus.
- e. Behalten Sie in den Abschnitten Timeout bei inaktiver Sitzung und Erweiterte Einstellungen die Standardeinstellungen bei und wählen Sie Weiter.
- f. Jetzt befinden Sie sich im Bereich Sprache zum Bot hinzufügen. Wählen Sie im Menü unter Sprachinteraktion die Option Keine aus. Dies ist nur eine textbasierte Anwendung. Behalten Sie die Standardeinstellungen für die übrigen Felder bei.
- g. Wählen Sie Erledigt aus. Amazon Lex V2 erstellt Ihren Bot und einen Standard-Intent namens NewIntentleitet Sie zu der Seite weiter, auf der Sie diesen Intent konfigurieren können

Um erfolgreich einen Bot zu erstellen, müssen Sie mindestens eine Absicht erstellen, die von der `AMAZON.FallbackIntent` und der `getrennt istAMAZON.KendraSearchIntent`. Diese Absicht ist erforderlich, um Ihren Amazon Lex V2-Bot zu erstellen, wird jedoch nicht für die FAQ-Antwort verwendet. Diese Absicht muss mindestens eine Beispieläußerung enthalten, und die Äußerung darf sich nicht auf Fragen beziehen, die Ihr Kunde stellt.

So erstellen Sie die erforderliche Absicht:

1. Geben Sie der Absicht im Abschnitt Details zur Absicht einen Namen, z. **RequiredIntent B**.
2. Geben Sie im Abschnitt Beispieläußerungen eine Äußerung in das Feld neben Äußerung hinzufügen ein, z. B. **Required utterance** Wählen Sie dann Äußerung hinzufügen.
3. Wählen Sie Absicht speichern.

Erstellen Sie die Absicht, einen Amazon Kendra Kendra-Index zu durchsuchen, und die Antwortnachricht, die zurückgegeben werden soll.

Um ein AMAZON zu erstellen. KendraSearchIntent Absicht und Antwortnachricht:

1. Wählen Sie im Navigationsbereich Zurück zur Absichtsliste aus, um zur Absichtsseite für Ihren Bot zurückzukehren. Wählen Sie Absicht hinzufügen und wählen Sie im Drop-down-Menü die Option Integrierte Absicht verwenden aus.

2. Wählen Sie in dem sich öffnenden Feld das Menü unter Integrierte Absicht aus. Geben Sie es **AMAZON.KendraSearchIntent** in die Suchleiste ein und wählen Sie es dann aus der Liste aus.
3. Geben Sie der Absicht einen Namen, z. **KendraSearchIntent B**.
4. Wählen Sie im Dropdownmenü Amazon Kendra Index den Index aus, nach dem die Absicht suchen soll. Der Index, den Sie im Abschnitt Voraussetzungen erstellt haben, sollte verfügbar sein.
5. Wählen Sie Hinzufügen aus.
6. Scrollen Sie im Intent-Editor nach unten zum Bereich Fulfillment, klicken Sie auf den Rechtspfeil, um den Bereich zu erweitern, und fügen Sie in das Feld unter Bei erfolgreicher Erfüllung die folgende Meldung ein:

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

The screenshot displays the Amazon Lex Intent Editor interface. The top section is titled "Fulfillment" with an "Info" icon and a description: "Run a lambda function to fulfill the intent and inform users of the status when it's complete." Below this, there are two expandable sections: "On successful fulfillment" and "In case of failure", both with "Message: -" text. The bottom section is titled "Closing response" with an "Info" icon, a description: "You can define the response when closing the intent.", and an "Active" toggle switch. It contains two expandable sections: "Response sent to the user after the intent is fulfilled" with "Message: -" text, and "Set values" with "-" text. Below these is a "Next step in conversation" section with "End conversation" text. At the bottom, there is a blue plus icon and the text "Add conditional branching".

Weitere Informationen zur Amazon Kendra Search Response finden Sie unter [Verwenden der Suchantwort](#).

7. Wählen Sie **Save intent** (Absicht speichern) und anschließend **Build** (Erstellen), um den Bot zu erstellen. Wenn der Bot bereit ist, wird das Banner oben auf dem Bildschirm grün und es wird eine Erfolgsmeldung angezeigt.

Verwenden Sie schließlich das Konsolentestfenster, um Antworten von Ihrem Bot zu testen.

Um deinen FAQ-Bot zu testen:

1. Nachdem der Bot erfolgreich erstellt wurde, wählen Sie **Test**.
2. Geben Sie **What is Amazon Kendra?** in das Testfenster der Konsole ein. Stellen Sie sicher, dass der Bot mit einem Link antwortet.
3. Weitere Informationen zur Konfiguration finden `AMAZON.KendraSearchIntent` Sie unter [AMAZON.KendraSearchIntent](#) und [KendraConfiguration](#).

AMAZON.PauseIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, eine Interaktion mit einem Bot zu unterbrechen, um später darauf zurückzukommen. Ihre Lambda-Funktion oder -Anwendung muss Absichtsdaten in Sitzungsvariablen speichern, oder Sie müssen die [GetSession](#) Operation verwenden, um Absichtsdaten abzurufen, wenn Sie die aktuelle Absicht fortsetzen.

Allgemeine Äußerungen:

- Pause
- pausiere das

AMAZON.QnAIntent

Note

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock](#) — Preise.

2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Beantwortet Kundenfragen mithilfe eines Amazon Bedrock FM, um Antworten auf häufig gestellte Fragen zu suchen und zusammenzufassen. Nur in US-Englisch verfügbar. Diese Absicht wird aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet ist. Beachten Sie, dass diese Absicht nicht für verpasste Äußerungen aktiviert wird, wenn ein Slot-Wert abgerufen wird. Nach der Erkennung verwendet der das angegebene Amazon Bedrock-Modell `AMAZON.QnAIntent`, um die konfigurierte Amazon Bedrock Knowledge Base zu durchsuchen und die Kundenfrage zu beantworten.

 Warning

Sie können das `AMAZON.QnAIntent` und das nicht `AMAZON.KendraSearchIntent` im selben Bot-Gebietsschema verwenden.

Die folgenden Knowledge Store-Optionen sind verfügbar. Sie müssen den Wissensspeicher bereits erstellt und die darin enthaltenen Dokumente indiziert haben.

- OpenSearch Dienstdomäne — Enthält indizierte Dokumente. Um eine Domain zu erstellen, folgen Sie den Schritten unter [Amazon OpenSearch Service-Domains erstellen und verwalten](#).
- Amazon Kendra Kendra-Index — Enthält indizierte FAQ-Dokumente. Um einen Amazon Kendra Kendra-Index zu erstellen, folgen Sie den Schritten unter [Index erstellen](#).
- Amazon Bedrock Knowledge Base — Enthält indizierte Datenquellen. Um eine Amazon Bedrock Knowledge Base einzurichten, folgen Sie den Schritten unter [Aufbau einer Knowledge Base](#).

Wenn Sie diese Absicht auswählen, konfigurieren Sie die folgenden Felder und wählen dann Hinzufügen, um die Absicht hinzuzufügen.

- Bedrock-Modell — Wählen Sie den Anbieter und das Fundamentmodell aus, die für diese Absicht verwendet werden sollen. Derzeit werden Anthropic Claude V2, Anthropic Claude 3 Haiku, Anthropic Claude 3 Haiku und Anthropic Claude Instant unterstützt.
- Wissensspeicher — Wählen Sie die Quelle aus, aus der das Modell Informationen abrufen soll, um Kundenfragen zu beantworten. Die folgenden Quellen sind verfügbar.

- **OpenSearch**— Konfigurieren Sie die folgenden Felder.
 - **Domain-Endpunkt** — Geben Sie den Domain-Endpunkt an, den Sie für die Domain erstellt haben oder der Ihnen nach der Domain-Erstellung zur Verfügung gestellt wurde.
 - **Indexname** — Geben Sie den Index an, nach dem gesucht werden soll. Weitere Informationen finden Sie unter [Indizierung von Daten in Amazon OpenSearch Service](#).
 - Wählen Sie aus, wie Sie die Antwort an den Kunden zurücksenden möchten.
 - **Exakte Antwort** — Wenn diese Option aktiviert ist, wird der Wert im Feld Antwort wie für die Bot-Antwort verwendet. Das konfigurierte Amazon Bedrock Foundation-Modell wird verwendet, um den genauen Inhalt der Antwort unverändert auszuwählen, ohne dass eine Inhaltssynthese oder Zusammenfassung erforderlich ist. Geben Sie den Namen der Frage- und Antwortfelder an, die in der Datenbank konfiguriert wurden. OpenSearch
 - **Felder einbeziehen** — Gibt eine Antwort zurück, die vom Modell anhand der von Ihnen angegebenen Felder generiert wurde. Geben Sie den Namen von bis zu fünf Feldern an, die in der OpenSearch Datenbank konfiguriert wurden. Verwenden Sie ein Semikolon (;), um Felder zu trennen.
- **Amazon Kendra** — Konfigurieren Sie die folgenden Felder.
 - **Amazon Kendra Kendra-Index** — Wählen Sie den Amazon Kendra Kendra-Index aus, den Ihr Bot durchsuchen soll.
 - **Amazon Kendra Kendra-Filter** — Um einen Filter zu erstellen, aktivieren Sie dieses Kontrollkästchen. Weitere Informationen zum JSON-Format des Amazon Kendra Kendra-Suchfilters finden Sie unter [Verwenden von Dokumentattributen zum Filtern von Suchergebnissen](#).
 - **Exakte Antwort** — Damit Ihr Bot genau die von Amazon Kendra zurückgegebene Antwort zurückgibt, aktivieren Sie dieses Kontrollkästchen. Andernfalls generiert das von Ihnen ausgewählte Amazon Bedrock-Modell eine Antwort auf der Grundlage der Ergebnisse.

 Note

Um diese Funktion nutzen zu können, müssen Sie zunächst FAQ-Fragen zu Ihrem Index hinzufügen. Folgen Sie dazu den Schritten unter [Häufig gestellte Fragen \(FAQs\) zu einem Index hinzufügen](#).

- **Amazon Bedrock Knowledge Base** — Wenn Sie diese Option wählen, geben Sie die ID der Amazon Bedrock Knowledge Base an. Sie finden die ID auf der Detailseite der Amazon Bedrock Knowledge Base in der Konsole oder indem [GetKnowledgeBase](#) Sie eine Anfrage senden.

- **Exakte Antwort** — Wenn diese Option aktiviert ist, wird der Wert im Feld Antwort unverändert für die Bot-Antwort verwendet. Das konfigurierte Amazon Bedrock Foundation-Modell wird verwendet, um den genauen Inhalt der Antwort unverändert auszuwählen, ohne dass eine Inhaltssynthese oder Zusammenfassung erforderlich ist. Um Exact Response für die Amazon Bedrock Knowledge Base zu verwenden, müssen Sie wie folgt vorgehen:
 - Erstellen Sie individuelle JSON-Dateien, wobei jede Datei ein Antwortfeld enthält, das die genaue Antwort enthält, die an den Endbenutzer zurückgegeben werden muss.
 - Wählen Sie bei der Indexierung dieser Dokumente in der Bedrock Knowledge Base die Option Chunking Strategy als No Chunking aus.
 - Definieren Sie das Antwortfeld in Amazon Lex V2 als Antwortfeld in der Bedrock Knowledge Base.

Die Antworten von Qn AIntent werden in den Anforderungsattributen gespeichert, wie unten dargestellt:

- `x-amz-lex:qnA-search-response`— Die Antwort des Qn AIntent auf die Frage oder Äußerung.
- `x-amz-lex:qnA-search-response-source`— Verweist auf das Dokument oder die Liste der Dokumente, anhand derer die Antwort generiert wurde.

Zusätzliche Modellkonfigurationen

Wenn Amazon.qn aufgerufen AIntent wird, verwendet es eine standardmäßige Eingabeaufforderungsvorlage, die Anweisungen und Kontext mit der Benutzerabfrage kombiniert, um die Aufforderung zu erstellen, die zur Antwortgenerierung an das Modell gesendet wird. Sie können auch eine benutzerdefinierte Aufforderung angeben oder die Standardaufforderung entsprechend Ihren Anforderungen aktualisieren.

Sie können die Vorlage für die Aufforderung mit den folgenden Tools entwickeln:

Platzhalter für Eingabeaufforderungen — Vordefinierte Variablen in Amazon.qn AIntent für Amazon Bedrock, die zur Laufzeit während des Bedrock-Aufrufs dynamisch ausgefüllt werden. In der Systemaufforderung können Sie diese Platzhalter sehen, die von dem Symbol umgeben sind. \$ In der folgenden Liste werden die Platzhalter beschrieben, die Sie verwenden können:

Variable	Ersetzt durch	Modell	Erforderlich?
\$query_results\$	Die abgerufenen Ergebnisse für die Benutzerabfrage aus dem Knowledge Store	Anthropisches Claude3-Haiku, Anthropisches Claude3-Sonett	Ja
\$output_instruction\$	Grundlegende Anweisungen für die Formatierung der Antwortgenerierung und der Zitate. Unterscheidet sich je nach Modell. Wenn Sie Ihre eigenen Formatierungsanweisungen definieren, empfehlen wir Ihnen, diesen Platzhalter zu entfernen.	Anthropisches Claude-3-Haiku, Anthropisches Claude-3-Sonett	Nein

Die standardmäßig verwendete Eingabeaufforderung ist:

```
$query_results$
```

Please only follow the instructions in <instruction> tags below.

<instruction>

Given the conversation history, 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 ONLY USING the grounding content in <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-intended or off-topic;
- (4) lastly, provide the confidence level of the above prediction as LOW, MID or HIGH.


```
</instruction>

$output_instruction$
```

`$output_instruction$` wird ersetzt durch:

```
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

Wenn Sie sich entscheiden, die Standardanweisungen nicht zu verwenden, wird die vom LLM bereitgestellte Ausgabe unverändert an den Endbenutzer zurückgegeben. Die Ausgabebefehle müssen `<passageId></passageId>` Tags `<text></text>` und Anweisungen enthalten, damit das LLM die PassageIDs zurückgibt, um die Antwort und die Quellenzuweisung bereitzustellen.

Unterstützung für das Filtern von Metadaten in der Amazon Bedrock Knowledge Base anhand von Sitzungsattributen

Sie können die Metadatenfilter der Amazon Bedrock Knowledge Base als Teil des Sitzungsattributs `x-amz-lex:bkb-retrieval-filter` übergeben.

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

Note

Sie müssen die Amazon Bedrock Knowledge Base als Datenspeicher für das Qn verwenden, um diesen Filter verwenden Alntent zu können. Weitere Informationen finden Sie unter [Metadata filtering](#).

Inferenzkonfigurationen

Sie können die Inferenzkonfigurationen, die beim Aufruf von LLM verwendet werden, mithilfe des Sitzungsattributs definieren:

- Temperatur: Typ Integer
- topP
- maxTokens

Beispiel:

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

Unterstützung von Bedrock Guardrails durch Build-Time- und Session-Attribute

- Indem Sie die Konsole bei Buildtime verwenden — Stellen Sie die und die bereit. GuardrailsIdentifier GuardrailsVersion Weitere Informationen finden Sie im Abschnitt Zusätzliche Modellkonfigurationen.
- Mithilfe von Sitzungsattributen — Sie können die Guardrails-Konfiguration auch mithilfe der Sitzungsattribute definieren: x-amz-lex:bedrock-guardrails-identifier und. x-amz-lex:bedrock-guardrails-version

[Weitere Informationen zur Verwendung von Bedrock Guardrails finden Sie unter Guardrails.](#)

AMAZON.QnIntent (multiple use support)

Sie können wählen, ob mehrere Amazon.QN innerhalb eines Gebietsschemas eingerichtet werden sollen. AIntents Amazon Lex V2 unterstützt bis zu 5 Amazon.QN AIntents innerhalb eines Bot-Gebietsschemas.

Amazon.qn AIntent kann ausgelöst werden, wenn einer der folgenden Fälle zutrifft:

- Wenn ein Bot-Gebietsschema nur 1 Amazon.qn enthält AIntent und diese Absicht keine Beispieläußerungen enthält, wird sie aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet ist. Diese Absicht wird aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet ist. Beachten Sie, dass diese Absicht nicht für verpasste Äußerungen aktiviert wird, wenn ein Slot-Wert abgerufen wird.

Note

Wenn die Antwort des FM unbefriedigend ist oder der Anruf beim FM fehlschlägt, ruft Amazon Lex V2 dann den auf. `AMAZON.FallbackIntent`

- Wenn Amazon.qn Musteräußerungen AIntent enthält, wird es nur aktiviert, wenn Amazon Lex V2 erkennt, dass der Benutzer diese Absicht auf der Grundlage von Benutzereingaben initiieren möchte.

Note

Wenn die Antwort des FM unbefriedigend ist oder der Anruf beim FM fehlschlägt, ruft Amazon Lex V2 den nächsten Fehlerschritt auf, der im Fulfillment-Block definiert ist.

Note

Wenn es mehr als 1 Amazon.qn `botLocale` gibt AIntent, AIntent muss jede Amazon.QN mindestens eine Beispieläußerung haben.

AMAZON.QinConnectIntent

Note

Um generative KI-Funktionen mit Amazon Q In Connect nutzen zu können, müssen Sie die folgenden Voraussetzungen erfüllen:

1. Navigieren Sie zur Amazon Connect-Konsole und erstellen Sie Ihre Instance. Falls Sie noch keine haben, finden Sie weitere Informationen unter [Erste Schritte mit Amazon Connect](#).
2. Aktivieren Sie Amazon Q in Connect für Ihre Instance, siehe [Amazon Q in Connect für Ihre Instance aktivieren](#).

AMAZON.QinConnectIntent beantwortet Kundenfragen mithilfe der LLM-erweiterten Weiterentwicklung von Amazon Connect Wisdom, die Empfehlungen in Echtzeit liefert, um Contact-Center-Kunden und -Agenten dabei zu helfen, Kundenprobleme schnell und präzise zu lösen. Diese Absicht wird aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet wird. Beachten Sie, dass diese Absicht nicht für verpasste Äußerungen aktiviert wird, wenn ein Slot-Wert abgerufen wird. Einmal erkannt, der AMAZON.QinConnectIntent, verwendet die angegebene Q in Connect-Domain, um die konfigurierte Amazon Bedrock Knowledge Base zu durchsuchen und die Kundenfrage zu beantworten.

Note

- Sie können AMAZON nicht verwenden. QinConnectIntent zusammen mit Amazon.qnAlntent im selben Bot-Gebietsschema.
- Wenn Sie neben US-Englisch eine andere Sprache auswählen, müssen Sie die Self-Service-Eingabeaufforderungen (SELF_SERVICE_PRE_PROCESSING und SELF_SERVICE_ANSWER_GENERATION) so anpassen, dass sie in der angegebenen Sprache antworten. Weitere Informationen zum Anpassen Ihrer Aufforderung finden Sie unter [Amazon Q in Connect anpassen](#).

Wenn Sie diese Absicht auswählen, müssen Sie die folgenden Felder konfigurieren und dann Absicht speichern auswählen, um die Absicht dem Bot hinzuzufügen.

- Amazon Q In Connect-Konfiguration — Geben Sie den Amazon-Ressourcennamen (ARN) des Amazon Q in Connect-Assistenten ein. ARN-Muster des Assistenten: `^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}$>.`

Die Antworten von `QinConnectIntent` werden wie folgt in den Anforderungsattributen gespeichert:

- `x-amz-lex:q-in-connect-response`— Die Antwort von `QinConnectIntent` auf die Frage oder Äußerung.

Sitzungsattribute wurden zurückgegeben von `QinConnectIntent`

Durch die Interaktion mit der `QinConnect` Absicht werden mithilfe von Sitzungsattributen zusätzliche Daten zur Konversation bereitgestellt.

1. `x-amz-lex:q-in-connect:session-arn`— Eine eindeutige Kennung für die Sitzung, die während der Konversation mit Amazon Q In Connect erstellt wurde.
2. `x-amz-lex:q-in-connect:conversation-status`— Der aktuelle Status der Konversation mit dem `QinConnect` Assistenten oder der Domain. Für diesen Status sind drei Werte möglich:
 - `CLOSED`
 - `READY`
 - `PROCESSING`
3. `x-amz-lex:q-in-connect:conversation-status-reason`— Gibt den Grund für den aktuellen Status an, der mit dem obigen Attribut gemeldet wurde. Die möglichen Gründe sind wie folgt:
 - `SUCCESS`— Zeigt an, dass der Kunde nichts mehr zu fragen hat und die Frage erfolgreich beantwortet wurde.
 - `FAILED`— Weist auf einen Fehler bei der Beantwortung der Kundenfrage hin. Diese sind hauptsächlich darauf zurückzuführen, dass die Frage des Kunden nicht verstanden wurde.
 - `REJECTED`— Weist darauf hin, dass der Assistent sich weigert, die Kundenfrage zu beantworten, und empfiehlt, die Frage außerhalb der Bot-Interaktion zu beantworten, z. B. im Gespräch mit einer Person oder einem Agenten, um weitere Informationen zu erhalten.

 Note

Wenn bei Kundeninteraktionen, die von einer Amazon Connect Connect-Instance gesteuert werden, ein Bot mit `QinConnectIntent` aufgerufen wird, muss Ihr Sitzungs-ARN erstellt und von der Amazon Connect Connect-Instance aus übergeben werden. Um eine Sitzung zu erstellen, könnte Amazon Connect Flows im Connect-Schritt mit Amazon Q konfiguriert werden.

Einschränkungen

- Sie können `AMAZON` nicht verwenden. `QinConnectIntent` zusammen mit Absichten ohne spezifische Äußerungen wie `AlIntent Amazon.QN`, `AMAZON. BedrockAgentIntent` im gleichen Bot-Gebietsschema.
- Wenn ein Bot mit während einer Kundeninteraktion aufgerufen `QinConnectIntent` wird, die von einer Amazon Connect Connect-Instance gesteuert wird, muss Ihr Sitzungs-ARN erstellt und von der Amazon Connect Connect-Instance übergeben werden. Um eine Sitzung zu erstellen, könnte Amazon Connect Flows mit dem Schritt Amazon Q In Connect konfiguriert werden.
- Es kann nicht mehr als ein `AMAZON` geben. `QinConnectIntent` pro Bot-Gebietsschema.
- Die Amazon Q in Connect-Domain, die mit `AMAZON` verwendet wird. `QinConnectIntent` muss sich in derselben AWS-Region wie der Amazon Lex V2-Bot befinden.

Berechtigungen

Wenn der `QinConnect Intent` in einem Amazon Lex V2-Bot verwendet wird und der Bot eine Service Linked Role (SLR) verwendet, ist der Amazon Lex V2-Service berechtigt, die entsprechenden Richtlinien für die Rolle zu aktualisieren, um sie in den Q in Connect-Assistenten zu integrieren. Wenn der Bot eine benutzerdefinierte IAM-Rolle verwendet, müsste der Benutzer diese Berechtigungen manuell zu seiner IAM-Rolle hinzufügen.

Die Rolle „Service Linked“ wird mit den folgenden Berechtigungen aktualisiert, wenn die `QinConnect Absicht` hinzugefügt wird. Für den `QinConnect Zugriff` wird eine neue Richtlinie hinzugefügt:

JSON

```
{  
  "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]/*"
        ]
      }
    ]
  }
}

```

Vertrauensrichtlinie

```

{
  "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

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, die vorherige Nachricht zu wiederholen. Ihre Anwendung muss eine Lambda-Funktion verwenden, um die vorherigen Absichtsinformationen in Sitzungsvariablen zu speichern, oder Sie müssen die [GetSession](#) Operation verwenden, um die vorherigen Absichtsinformationen abzurufen.

Allgemeine Äußerungen:

- wiederholen
- sag das nochmal
- wiederhole das

AMAZON.ResumeIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, eine zuvor unterbrochene Absicht wieder aufzunehmen. Ihre Lambda-Funktion oder -Anwendung muss die Informationen verwalten, die erforderlich sind, um die vorherige Absicht wieder aufzunehmen.

Allgemeine Äußerungen:

- fortsetzen
- fortsetzen
- mach weiter

AMAZON.StartOverIntent

Reagiert auf Wörter und Ausdrücke, die es dem Benutzer ermöglichen, die Verarbeitung der aktuellen Absicht zu beenden und von vorne zu beginnen. Sie können Ihre Lambda-Funktion oder die PutSession Operation verwenden, um den ersten Slot-Wert erneut abzurufen.

Allgemeine Äußerungen:

- fange von vorne an

- Neustart
- wieder anfangen

AMAZON.StopIntent

Reagiert auf Wörter und Ausdrücke, die darauf hinweisen, dass der Benutzer die Verarbeitung der aktuellen Absicht beenden und die Interaktion mit einem Bot beenden möchte. Ihre Lambda-Funktion oder -Anwendung sollte alle vorhandenen Attribute und Slot-Typwerte löschen und dann die Interaktion beenden.

Allgemeine Äußerungen:

- stop
- aus
- Halt die Klappe

Slot-Typen hinzufügen

Slot-Typen definieren die Werte, die Benutzer für Ihre Intent-Variablen angeben können. Sie definieren Slot-Typen für jede Sprache, sodass die Werte für diese Sprache spezifisch sind. Beispielsweise könnten Sie für einen Slot-Typ, der Farbfarben auflistet, den Wert "red" in Englisch, "rouge" in Französisch und "rojo" in Spanisch angeben.

In diesem Thema wird beschrieben, wie Sie benutzerdefinierte Slot-Typen erstellen, die Werte für die Slots Ihrer Absicht bereitstellen. Sie können auch integrierte Slot-Typen für Standardwerte verwenden. Sie können beispielsweise den integrierten Slot-Typ `AMAZON.Country` für eine Liste von Ländern der Welt verwenden.

Um einen Slot-Typ zu erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Bot aus, dem Sie die Sprache hinzufügen möchten, wählen Sie dann Konversationsstruktur und dann Alle Sprachen aus.
3. Wählen Sie die Sprache aus, zu der Sie den Slot-Typ hinzufügen möchten, und wählen Sie dann Slot-Typen.

4. Wählen Sie Slot-Typ hinzufügen, geben Sie Ihrem Slot-Typ einen Namen und wählen Sie dann Hinzufügen.
5. Fügen Sie im Slot-Typ-Editor die Details Ihres Slot-Typs hinzu.
 - Auflösung von Slot-Werten — Legt fest, wie Slot-Werte aufgelöst werden. Wenn Sie Werte erweitern wählen, verwendet Amazon Lex V2 die Werte als repräsentative Werte für das Training. Wenn Sie „Auf Slot-Werte beschränken“ verwenden, sind die zulässigen Werte für den Slot auf die Werte beschränkt, die Sie angeben.
 - Slot-Typ-Werte — Die Werte für den Slot. Wenn Sie „Auf Slot-Werte beschränken“ ausgewählt haben, können Sie Synonyme für den Wert hinzufügen. Beispielsweise können Sie für den Wert „Fußball“ das Synonym „Fußball“ hinzufügen. Wenn der Nutzer in einer Konversation mit deinem Bot „Fußball“ eingibt, ist der tatsächliche Wert des Slots „Fußball“.
 - Slot-Werte als benutzerdefiniertes Vokabular verwenden — Aktivieren Sie diese Option, um die Erkennung von Slot-Werten und Synonymen in Audiokonversationen zu verbessern. Aktivieren Sie diese Option nicht, wenn es sich bei den Slot-Werten um gebräuchliche Begriffe wie „Ja“, „Nein“, „Eins“, „Zwei“, „Drei“ usw. handelt.
6. Wählen Sie Slot-Typ speichern.

Amazon Lex V2 bietet die folgenden Steckplatztypen:

Themen

- [Integrierte Steckplatztypen](#)
- [Benutzerdefinierter Slot-Typ](#)
- [Grammatik-Slot-Typ](#)
- [Steckplatztyp aus Verbundwerkstoff](#)

Integrierte Steckplatztypen

Amazon Lex unterstützt integrierte Steckplatztypen, die definieren, wie Daten im Steckplatz erkannt und verarbeitet werden. Sie können Slots dieser Art in Ihren Absichten erstellen. Es müssen dann keine Enumerationswerte für häufig verwendete Slot-Daten wie Datum, Uhrzeit und Ort mehr erstellt werden. Integrierte Slot-Typen haben keine Versionen.

Slot-Typ	Kurzbeschreibung	Unterstützte Gebietsschemata	
<u>AMAZON.AlphaNumeric</u>	Erkennt aus Buchstaben und Zahlen bestehende Wörter.		
<u>Amazon.city</u>	Erkennt Wörter, die für eine Stadt stehen.	Alle Gebietsschemas	
<u>Amazon.Bestätigung</u>	Erkennt Wörter, die „Ja“, „Nein“, „Vielleicht“ und „Weiß nicht“ bedeuten, und konvertiert sie in ein Standardformat (weiß Yes/No/Maybe/Don nicht).	Englisch (en-US, en-GB, en-AU, en-IN, en-ZA)	
<u>Amazon.country</u>	Erkennt Wörter, die für ein Land stehen.	Alle Gebietsschemas	
<u>Amazon.Currency</u>	Erkennt Wörter, die für einen Währungswert stehen, und wandelt sie in eine Standardwährungsabkürzung und einen Wert um.	Englisch (en-US, en-GB, en-AU, en-IN, en-ZA), Spanisch (es-419, es-ES, es-US)	
<u>Amazon.date</u>	Erkennt Wörter, die für ein Datum stehen, und konvertiert sie in ein Standardformat.	Alle Gebietsschemas	
<u>Amazon.Duration</u>	Erkennt Wörter, die für eine Dauer stehen,	Alle Gebietsschemas	

Slot-Typ	Kurzbeschreibung	Unterstützte Gebietsschemata	
	und konvertiert sie in ein Standardformat.		
AMAZON. EmailAddress	Erkennt Wörter, die für eine E-Mail-Adresse stehen, und wandelt sie in eine Standard-E-Mail-Adresse um.	Alle Gebietsschemas	
AMAZON. FirstName	Erkennt Wörter, die für einen Vornamen stehen.	Alle Gebietsschemas	
AMAZON. LastName	Erkennt Wörter, die für einen Nachnamen stehen.	Alle Gebietsschemas	
Amazon.Nummer	Erkennt numerische Wörter und wandelt sie in Ziffern um.	Alle Gebietsschemas	
AMAZON.Percentage	Erkennt Wörter, die eine Prozentzahl darstellen, und wandelt sie in eine Zahl und ein Prozentzeichen (%) um.	Alle Gebietsschemas	

Slot-Typ	Kurzbeschreibung	Unterstützte Gebietsschemata	
AMAZON.PhoneNumber	Erkennt Wörter, die für eine Telefonnummer stehen, und wandelt sie in eine numerische Zeichenfolge um.	Alle Gebietsschemata	
Amazon.state	Erkennt Wörter, die für einen Staat stehen.	Alle Gebietsschemata	
AMAZONAS.StreetName	Erkennt Wörter, die für einen Straßennamen stehen.	Alle Gebietsschemata	
Amazon.Zeit	Erkennt Wörter, die auf Zeiten hinweisen, und konvertiert sie in ein Zeitformat.	Alle Gebietsschemata	
AMAZON. UKPostalCode	Erkennt Wörter, die für eine britische Postleitzahl stehen, und konvertiert sie in ein Standardformular.	Nur Englisch (Britisch) (en-GB)	
AMAZON. FreeFormInput	Erkennt Zeichenketten, die aus beliebigen Wörtern oder Zeichen bestehen.	Alle Gebietsschemata	

AMAZON. AlphaNumeric

Erkennt aus Buchstaben und Zahlen bestehende Zeichenfolgen, z. B. **APQ123**.

Sie können den Slot-Typ `AMAZON.AlphaNumeric` für Zeichenfolgen verwenden, die Folgendes enthalten:

- Alphabetische Zeichen, z. B. **ABC**
- Numerische Zeichen, z. B. **123**
- Eine Kombination aus alphanumerischen Zeichen, z. B. **ABC123**

Der `AMAZON.AlphaNumeric` Slot-Typ unterstützt Eingaben mit Rechtschreibstilen. Sie können die spell-by-word Formate spell-by-letter und verwenden, um Ihren Kunden bei der Eingabe von Buchstaben zu helfen. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

Sie können dem Slot-Typ `AMAZON.AlphaNumeric` einen regulären Ausdruck hinzufügen, um die für den Slot eingegebenen Werte zu validieren. Sie können beispielsweise einen regulären Ausdruck verwenden, um Folgendes zu validieren:

- Kanadische Postleitzahlen
- Führerscheinnummern
- Fahrgestellnummern

Verwenden Sie einen regulären Standardausdruck. Amazon Lex V2 unterstützt die folgenden Zeichen im regulären Ausdruck:

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

Amazon Lex V2 unterstützt auch Unicode-Zeichen in regulären Ausdrücken. Die Form lautet `\uUnicode`. Verwenden Sie vier Ziffern, um Unicode-Zeichen darzustellen. Beispiel: `[\u0041-\u005A]` ist gleichbedeutend mit `[A-Z]`.

Die folgenden Operatoren für reguläre Ausdrücke werden nicht unterstützt:

- Endlose Wiederholer: `*`, `+`, oder `{x,}` ohne Obergrenze.
- Platzhalter (`.`)

Die maximale Länge des regulären Ausdrucks beträgt 300 Zeichen. Die maximale Länge einer Zeichenfolge, die in einem `AMAZON.AlphaNumeric Slot`-Typ gespeichert ist, der einen regulären Ausdruck verwendet, beträgt 30 Zeichen.

Im Folgenden finden Sie einige Beispiele für reguläre Ausdrücke.

- Alphanumerische Zeichenfolgen, z. B. **APQ123** oder **APQ1**: `[A-Z]{3}[0-9]{1,3}` oder eine stärker eingeschränkte `[A-DP-T]{3} [1-5]{1,3}`
- US Postal Service Priority Mail im internationalen Format, z. B. **CP123456789US**: `CP[0-9]{9}US`
- Bankleitzahlen, z. B. **123456789**: `[0-9]{9}`

Um den regulären Ausdruck für einen Slot-Typ festzulegen, verwenden Sie die Konsole oder die Operation [CreateSlotType](#). Der reguläre Ausdruck wird beim Speichern des Slot-Typs validiert. Wenn der Ausdruck nicht gültig ist, gibt Amazon Lex V2 eine Fehlermeldung zurück.

Wenn Sie einen regulären Ausdruck in einem Slot-Typ verwenden, vergleicht Amazon Lex V2 die Eingabe in Slots dieses Typs mit dem regulären Ausdruck. Wenn die Eingabe mit dem Ausdruck übereinstimmt, wird der Wert für den Slot akzeptiert. Wenn die Eingabe nicht übereinstimmt, fordert Amazon Lex V2 den Benutzer auf, die Eingabe zu wiederholen.

Amazon.city

Bietet eine Liste lokaler und weltweiter Städte. Der Slot-Typ erkennt gängige Varianten von Städtenamen. Amazon Lex V2 konvertiert nicht von einer Variante in einen offiziellen Namen.

Beispiele:

- New York
- Reykjavik
- Tokio
- Versailles

Amazon.Bestätigung

Dieser Slot-Typ erkennt eingegebene Phrasen und Wörter, die „Ja“, „Nein“, „Vielleicht“ und „Weiß nicht“ für Amazon Lex V2 entsprechen, und konvertiert ihn in einen der vier Werte. Es kann verwendet werden, um Bestätigungen oder Bestätigungen vom Benutzer zu erfassen. Auf

der Grundlage des endgültigen gelösten Werts können Sie Bedingungen erstellen, um mehrere Konversationspfade zu entwerfen.

Zum Beispiel:

wenn {confirmation} = „Ja“, erfüllen Sie die Absicht

Andernfalls rufen Sie einen anderen Slot hervor

Beispiele:

- Ja: Ja, ja, ok, klar, ich habe es, ich kann zustimmen...
- Nein: Nein, negativ, nein, vergiss es, ich werde ablehnen, auf keinen Fall...
- Vielleicht: Es ist möglich, vielleicht, manchmal, vielleicht, das könnte richtig sein...
- Weiß nicht: Keine Ahnung, Unbekannt, Keine Ahnung, bin mir nicht sicher, wer weiß...

Wenn ab dem 17. August 2023 ein benutzerdefinierter Slot-Typ mit dem Namen „Confirmation“ existiert, muss der Name geändert werden, um Konflikte mit der integrierten Slot-Bestätigung zu vermeiden. Gehen Sie in der linken Navigationsleiste der Lex-Konsole zum Slot-Typ (für einen vorhandenen benutzerdefinierten Slot-Typ namens Confirmation) und aktualisieren Sie den Slot-Typnamen. Der Name des neuen Steckplatztyps darf nicht „Confirmation“ lauten, was ein reserviertes Schlüsselwort für den integrierten Bestätigungssteckplatztyp ist.

Amazon.country

Die Namen von Ländern auf der ganzen Welt. Beispiele:

- Australien
- Deutschland
- Japan
- Vereinigte Staaten
- Uruguay

Amazon.Currency

Konvertiert Wörter, die eine Währung repräsentieren, in einen alphabetischen ISO-4217-Währungscode und eine Zahl. Erkennt Währungen, konvertiert nicht von einer Währung in eine andere.

erkennt die folgenden Währungen in diesen Sprachen Englisch und Spanisch: Englisch (en-US, en-GB, en-AU, en-IN, en-ZA), Spanisch (es-419, es-ES, es-US).

Weitere Informationen finden Sie unter [Währungscodes — ISO 4217 auf der Website der Internationalen Organisation für Normung \(ISO\)](#).

Die dargestellte Währung ist wie folgt strukturiert: {Unit} {Amount}

- {Einheit} bezieht sich auf die spezifische Währungseinheit (z. B. USD).
- {Betrag} bezeichnet den Geldwert, formatiert mit zwei Dezimalstellen (z. B. 300,00).

Beispiele (alle nachfolgenden Beispiele verwenden das Gebietsschema en-US; unterschiedliche Gebietsschemas können zu unterschiedlichen Ergebnissen führen):

- „3 USD“: 3,00 USD
- „USD300“: 300,00 USD
- „3 Cent“: 0,30 USD
- „1,56\$“: 1,56 USD
- „5 x“: 0,05 USD
- „1 Dollar“: 1,00 USD
- „fünf fünfzehn“: 515,00 USD
- „fünf Dollar fünfzehn Cent“: 5,15 USD
- „5 USD und 1/2“: 5,50 USD

Amazon.date

Konvertiert Wörter, die für Datumsangaben stehen, in ein Datumsformat.

Das Datum wird nach Ihren Wünschen im ISO-8601-Datumsformat bereitgestellt. Das Datum, an dem Ihre Absicht im Slot eingeht, kann je nach dem vom Benutzer geäußerten Satz variieren.

- Äußerungen, die einem bestimmten Datum zugeordnet sind, wie „heute“, „jetzt“ oder „fünfundzwanzigster November“, werden in ein vollständiges Datum umgewandelt: 2020-11-25
Standardmäßig werden Datumsangaben verwendet, die am oder nach dem aktuellen Datum liegen.

- Äußerungen, die einer future Woche zugeordnet sind, z. B. „nächste Woche“, werden in das Datum des letzten Tages der aktuellen Woche umgewandelt. Im ISO-8601-Format beginnt die Woche am Montag und endet am Sonntag. Wenn heute beispielsweise der 25.11.2020 ist, wird „nächste Woche“ in umgewandelt. 2020-11-29 Datumsangaben, die der aktuellen oder vorherigen Woche zugeordnet sind, werden in den ersten Tag der Woche umgewandelt. Wenn heute beispielsweise der 25.11.2020 ist, wird „letzte Woche“ in umgewandelt. 2020-11-16
- Äußerungen, die einem future Monat, aber keinem bestimmten Tag zugeordnet sind, wie z. B. „nächster Monat“, werden in den letzten Tag des Monats umgewandelt. Wenn heute beispielsweise der 25.11.2020 ist, wird „nächster Monat“ in umgewandelt. 2020-12-31 Bei Daten, die dem aktuellen oder vorherigen Monat zugeordnet sind, wird der erste Tag des Monats umgerechnet. Wenn heute beispielsweise der 25.11.2020 ist, wird „dieser Monat“ zugeordnet. 2020-11-01
- Äußerungen, die einem future Jahr, aber nicht einem bestimmten Monat oder Tag zugeordnet sind, wie z. B. „nächstes Jahr“, werden in den letzten Tag des Folgejahres umgewandelt. Wenn heute beispielsweise der 25.11.2020 ist, wird „nächstes Jahr“ in umgewandelt. 2021-12-31 Bei Datumsangaben, die dem aktuellen oder dem Vorjahr entsprechen, wird der erste Tag des Jahres umgerechnet. Wenn heute beispielsweise der 25.11.2020 ist, wird „letztes Jahr“ in umgewandelt. 2019-01-01

Amazon.Duration

Konvertiert Wörter, die eine Dauer angeben, in eine numerische Dauer.

Die Dauer wird in ein Format aufgelöst, das auf dem [ISO-8601-Dauerformat](#) basiert.

PnYnMnWnDTnHnMnS Das P gibt an, dass es sich um eine Dauer handelt, n es handelt sich um einen numerischen Wert und der darauf folgende Großbuchstabe n ist das spezifische Datums- oder Zeitelement. P3DBedeutet zum Beispiel 3 Tage. A T wird verwendet, um anzuzeigen, dass die verbleibenden Werte eher Zeitelemente als Datumselemente darstellen.

Beispiele:

- „zehn Minuten“: PT10M
- „fünf Stunden“: PT5H
- „drei Tage“: P3D
- „fünfundvierzig Sekunden“: PT45S
- „acht Wochen“: P8W
- „sieben Jahre“: P7Y

- „fünf Stunden zehn Minuten“: PT5H10M
- „zwei Jahre drei Stunden zehn Minuten“: P2YT3H10M

AMAZON. EmailAddress

Erkennt Wörter, die eine E-Mail-Adresse in der Form „Benutzername@Domäne“ darstellen. Adressen können die folgenden Sonderzeichen im Benutzernamen enthalten: Unterstrich (_), Bindestrich (-), Punkt (.) und Pluszeichen (+).

Der AMAZON.EmailAddress Slot-Typ unterstützt Eingaben mit Rechtschreibstilen. Sie können die spell-by-word Formate spell-by-letter und verwenden, um Ihren Kunden bei der Eingabe von E-Mail-Adressen zu helfen. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

AMAZON. FirstName

Häufig verwendete Vornamen. Dieser Slot-Typ erkennt formelle Namen, informelle Spitznamen und Namen, die aus mehr als einem Wort bestehen. Der Name, der zu Ihrer Absicht gesendet wurde, ist der vom Benutzer gesendete Wert. Amazon Lex V2 konvertiert nicht vom Spitznamen zum formellen Namen.

Für Vornamen, die ähnlich klingen, aber unterschiedlich geschrieben sind, sendet Amazon Lex V2 Ihrer Absicht ein einheitliches Formular.

Der AMAZON.FirstName Slot-Typ unterstützt Eingaben mit Rechtschreibstilen. Sie können die spell-by-word Formate spell-by-letter und verwenden, um Ihren Kunden bei der Eingabe von Namen zu helfen. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

Beispiele:

- Emily
- John
- Sophie
- Anil Kumar

AMAZONAS.FirstName gibt auch eine Liste eng verwandter Namen zurück, die auf dem ursprünglichen Wert basieren. Sie können die Liste der aufgelösten Werte verwenden, um Tippfehler

zu korrigieren, den Namen mit dem Benutzer zu bestätigen oder eine Datenbanksuche nach gültigen Namen in Ihrem Benutzerverzeichnis durchzuführen.

Beispielsweise kann die Eingabe „John“ dazu führen, dass zusätzliche verwandte Namen wie „John J“ und „John-Paul“ zurückgegeben werden.

Im Folgenden wird das Antwortformat für den `AMAZON.FirstName` integrierten Steckplatztyp dargestellt:

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

AMAZON.LastName

Häufig verwendete Nachnamen. Für Namen, die ähnlich klingen, aber unterschiedlich geschrieben sind, sendet Amazon Lex V2 Ihrer Absicht ein einheitliches Formular.

Der `AMAZON.LastName` Slot-Typ unterstützt Eingaben mit Rechtschreibstilen. Sie können die spell-by-word Formate spell-by-letter und verwenden, um Ihren Kunden bei der Eingabe von Namen zu helfen. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

Beispiele:

- Brosky
- Dasher
- Evers
- Parres
- Welt

`AMAZON.LastName` gibt auch eine Liste eng verwandter Namen zurück, die auf dem ursprünglichen Wert basieren. Sie können die Liste der aufgelösten Werte verwenden, um Tippfehler zu korrigieren,

den Namen mit dem Benutzer zu bestätigen oder eine Datenbanksuche nach gültigen Namen in Ihrem Benutzerverzeichnis durchzuführen.

Beispielsweise kann die Eingabe „Smith“ dazu führen, dass zusätzliche verwandte Namen wie „Smyth“ und „Smithe“ zurückgegeben werden.

Im Folgenden wird das Antwortformat für den `AMAZON.LastName` integrierten Slot-Typ dargestellt:

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

Amazon.Nummer

Konvertiert Wörter oder Zahlen, die eine Zahl ausdrücken, in Ziffern, einschließlich Dezimalzahlen. Die folgende Tabelle zeigt, wie der Slot-Typ `AMAZON.Number` numerische Wörter erfasst.

Eingabe	Antwort
one hundred twenty three point four five	123.45
one hundred twenty three dot four five	123.45
point four two	0.42
point forty two	0.42
232.998	232.998
50	50
-15	-15
minus 15	-15

Eingabe	Antwort
minus fünfzehn Punkt zwei vier fünf	-15,245

AMAZON.Percentage

Wandelt Wörter und Symbole, die einen Prozentwert repräsentieren, in einen numerischen Wert mit einem Prozentzeichen (%) um.

Wenn der Benutzer eine Zahl ohne Prozentzeichen oder das Wort "percent" eingibt, wird dem Slot-Wert diese Zahl zugewiesen. Die folgende Tabelle zeigt, wie der Slot-Typ AMAZON.Percentage Prozentwerte erfasst.

Eingabe	Antwort
50 percent	50%
0,4 Prozent	0.4%
23.5%	23.5%
fünfundzwanzig Prozent	25 %

AMAZON.PhoneNumber

Wandelt die Zahlen oder Wörter, die eine Telefonnummer repräsentieren, in ein Zeichenfolgenformat ohne Satzzeichen um, wie nachfolgend dargestellt.

Typ	Beschreibung	Eingabe	Ergebnis
Auslandsrufnummer mit vorangestelltem Pluszeichen (+)	11-stellige Rufnummer mit vorangestelltem Pluszeichen.	+61 7 4445 1061	+61744431061
		+1 (509) 555-1212	+15095551212
Auslandsrufnummer ohne vorangestelltes Pluszeichen (+)	11-stellige Rufnummer ohne	1 (509) 555-1212	15095551212
		61 7 4445 1061	61744451061

Typ	Beschreibung	Eingabe	Ergebnis
	vorangestelltes Pluszeichen		
Inlandsrufnummer	10-stellige Zahl ohne Ländervorwahl	(03) 5115 4444	0351154444
		(509) 555-1212	5095551212
Ortsrufnummer	Telefonnummer ohne internationale Vorwahl oder Ortsvorwahl	555-1212	5551212

Amazon.state

Die Namen der geografischen und politischen Regionen innerhalb der Länder.

Beispiele:

- Bayern
- Präfektur Fukushima
- Pazifischer Nordwesten
- Queensland
- Wales

AMAZONAS. StreetName

Die Straßennamen innerhalb einer typischen Straßenadresse. Dazu gehört nur der Straßename, nicht die Hausnummer.

Beispiele:

- Canberra Avenue
- Vordere Straße
- Marktstraße

Amazon.Zeit

Konvertiert Wörter, die Zeiten darstellen, in Zeitwerte. AMAZON.Time kann genaue Zeiten, mehrdeutige Werte und Zeitbereiche auflösen. Der Slot-Wert kann in die folgenden Zeitbereiche aufgelöst werden:

- AM
- PM
- MO (Morgen)
- AF (Nachmittag)
- EV (Abend)
- NI (Nacht)

Wenn ein Benutzer eine mehrdeutige Uhrzeit eingibt, verwendet Amazon Lex V2 das `slots` Attribut eines Lambda-Ereignisses, um Auflösungen für die mehrdeutigen Zeiten an Ihre Lambda-Funktion zu übergeben. Beispiel: Wenn der Bot den Benutzer zur Angabe einer Lieferzeit auffordert, kann der Benutzer mit "10 o'clock" antworten. Diese Zeitangabe ist zweideutig. Sie kann 10:00 Uhr (10:00 AM) oder 22:00 Uhr (10:00 PM) bedeuten. In diesem Fall ist der Wert im `interpretedValue` Feld `null`, und das `resolvedValues` Feld enthält die beiden möglichen Auflösungen der Zeit. Amazon Lex V2 gibt Folgendes in die Lambda-Funktion ein:

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

Wenn der Benutzer mit einer eindeutigen Uhrzeit antwortet, sendet Amazon Lex V2 die Uhrzeit an Ihre Lambda-Funktion im `interpretedValue` Feld des `slots` Attributs des Lambda-Ereignisses. Wenn Ihr Benutzer beispielsweise auf die Aufforderung zur Angabe einer Lieferzeit mit "10:00 Uhr" antwortet, gibt Amazon Lex V2 Folgendes in die Lambda-Funktion ein:

```
"slots": {
```



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

Wenn der Benutzer auf eine Aufforderung zur Angabe einer Lieferzeit mit „am Morgen“ antwortet, gibt Amazon Lex V2 Folgendes in die Lambda-Funktion ein:

```
"slots": {
  "deliveryTime": {
    "value": {
      "originalValue": "morning",
      "interpretedValue": "M0",
      "resolvedValues": [
        "M0"
      ]
    }
  }
}
```

Weitere Informationen zu den Daten, die von Amazon Lex V2 an eine Lambda-Funktion gesendet werden, finden Sie unter [AWS Lambda Eingabeereignisformat für Lex V2](#).

AMAZON.UKPostalCode

Konvertiert Wörter, die eine britische Postleitzahl darstellen, in ein Standardformat für Postleitzahlen im Vereinigtes Königreich. Der AMAZON.UKPostalCode Slot-Typ validiert die Postleitzahl und löst sie in eine Reihe von standardisierten Formaten auf, überprüft aber nicht, ob die Postleitzahl gültig ist. Ihre Bewerbung muss die Postleitzahl validieren.

Der AMAZON.UKPostalCode Steckplatztyp ist nur im Gebietsschema Englisch (UK) (en-GB) verfügbar.

Der AMAZON.UKPostalCode Slot-Typ unterstützt Eingaben mit Rechtschreibstilen. Sie können die spell-by-word Formate spell-by-letter und verwenden, um Ihren Kunden bei der Eingabe von Buchstaben zu helfen. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

Der Slot-Typ erkennt nur die unten aufgeführten gültigen Postleitzahlformate, die im Vereinigtes Königreich verwendet werden. Die gültigen Formate sind („A“ steht für einen Buchstaben und „9“ steht für eine Ziffer):

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

Für die Texteingabe kann der Benutzer eine beliebige Mischung aus Groß- und Kleinbuchstaben eingeben. Der Benutzer kann das Leerzeichen in der Postleitzahl verwenden oder weglassen. Der aufgelöste Wert enthält immer das Leerzeichen an der richtigen Stelle für die Postleitzahl.

Bei der mündlichen Eingabe kann der Benutzer die einzelnen Zeichen vorlesen oder Doppelbuchstaben-Aussprachen verwenden, z. B. „Double A“ oder „Double 9“. Sie können auch zweistellige Aussprachen verwenden, z. B. „neunundneunzig“ für „99“.

Note

Nicht alle britischen Postleitzahlen werden anerkannt. Nur die oben aufgeführten Formate werden unterstützt.

AMAZON.FreeFormInput

AMAZON.FreeFormInput kann verwendet werden, um formlose Eingaben des Endbenutzers zu erfassen. Es erkennt Zeichenketten, die aus Wörtern oder Zeichen bestehen. Der aufgelöste Wert ist die gesamte Eingabeäußerung.

Beispiel:

Bot: Bitte geben Sie Feedback zu Ihrem Anruferlebnis.

Benutzer: Ich habe Antworten auf alle meine Fragen erhalten und konnte die Transaktion abschließen.

Hinweis:

- `AMAZON.FreeFormInput` kann verwendet werden, um formlose Eingaben des Endbenutzers unverändert zu erfassen.
- `AMAZON.FreeFormInput` kann nicht in Musteräußerungen mit Absicht verwendet werden.
- `AMAZON.FreeFormInput` darf keine Stichprobenäußerungen enthalten.
- `AMAZON.FreeFormInput` wird nur erkannt, wenn es für ausgelöst wird.
- `AMAZON.FreeFormInput` unterstützt das Warten und Fortfahren nicht.
- `AMAZON.FreeFormInput` wird derzeit im Amazon Connect Chat-Kanal nicht unterstützt.
- Wenn ein `AMAZON.FreeFormInput` Slot ausgelöst wird, `FallbackIntent` wird er nicht ausgelöst.
- Wenn ein `AMAZON.FreeFormInput` Slot ausgelöst wird, erfolgt kein Intent-Switch.

Benutzerdefinierter Slot-Typ

Für jede Absicht können Sie Parameter mit den Daten angeben, die von der Absicht benötigt werden, um die Benutzeranforderung zu erfüllen. Diese Parameter (Slots) sind von einem bestimmten Typ. Ein Slot-Typ ist eine Liste von Werten, die Amazon Lex V2 verwendet, um das Machine-Learning-Modell so zu trainieren, dass es Werte für einen Slot erkennt. Sie können beispielsweise einen Slot-Typ definieren, der `Genres` mit Werten wie „Komödie“, „Abenteuer“, „Dokumentarfilm“ usw. aufgerufen wird. Sie können Synonyme für einen Slot-Typ-Wert definieren. Sie können beispielsweise die Synonyme "funny" und "humorous" für den Wert "comedy" definieren.

Slot type: Customtype [Info](#)

A slot type is a list of values used to capture values for a slot.

▼ Slot type details

Slot type name

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Description - *optional*

Helps you identify a slot type on the list

Maximum 200 characters.

Type: Custom

ID: HKGU4J6UOP

Slot value resolution

Amazon Lex resolves the slot values in an utterance to only the values you provide, or it expands the resolution to related or similar values.

☒ **Expand values (default)**
Values used as training data.

☐ **Restrict to slot values**
Use only values provided.

Slot type values

Modify the list of values used to train the machine learning model to recognize values for a slot.

No slot type values

You haven't added any slot type values yet.

Maximum 140 characters. Valid characters: A-Z, a-z, 0-9, @, #, \$

Add value

☐ Use slot values as custom vocabulary [Info](#)

Sie können den Slot-Typ konfigurieren, um die Slot-Werte zu erweitern. Slot-Werte werden als Trainingsdaten verwendet, und das Modell löst den Slot anhand des vom Benutzer angegebenen Werts auf, sofern er den Slot-Werten und Synonymen dieser Werte ähnlich ist. Dies ist das Standardverhalten. Amazon Lex V2 führt eine Liste möglicher Auflösungen für einen Steckplatz.

Jeder Eintrag in der Liste enthält einen aufgelösten Wert, den Amazon Lex V2 als zusätzliche Möglichkeiten für den Steckplatz erkannt hat. Ein aufgelöster Wert ist die beste Methode, um dem Slot-Wert zu entsprechen. Die Liste enthält bis zu fünf Werte.

Alternativ können Sie den Slot-Typ so konfigurieren, dass die Auflösung auf die Slot-Werte beschränkt wird. In diesem Fall löst das Modell einen vom Benutzer eingegebenen Slot-Wert nur dann in einen vorhandenen Slot-Wert auf, wenn er mit diesem Slot-Wert identisch ist oder es sich um ein Synonym handelt. Wenn der Benutzer beispielsweise "funny" eingibt, wird dies als Slot-Wert "comedy" aufgelöst.

Wenn der vom Benutzer eingegebene Wert ein Synonym für einen Slot-Typ-Wert ist, gibt das Modell diesen Slot-Typ-Wert als ersten Eintrag in der Liste von zurück. `resolvedValues`
Wenn der Benutzer beispielsweise „lustig“ eingibt, füllt das Modell das `originalValue` Feld mit dem Wert „lustig“ und den ersten Eintrag im Feld `ResolvedValues` mit „Comedy“. Sie können `valueSelectionStrategy` beim Erstellen oder Aktualisieren eines Slot-Typs mit der Operation [CreateSlotType](#) konfigurieren, damit der Slot-Wert mit dem ersten Wert in der Auflösungsliste gefüllt wird.

Benutzerdefinierte Slot-Typen unterstützen Eingaben mithilfe von Rechtschreibstilen. Sie können die spell-by-word Formate spell-by-letter und verwenden, um Ihren Kunden bei der Eingabe von Buchstaben zu helfen. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

Wenn Sie eine Lambda-Funktion verwenden, enthält das Eingabeereignis für die Funktion eine `resolvedValues` aufgerufene Auflösungsliste. Das folgende Beispiel zeigt den Slot-Abschnitt der Eingabe für eine Lambda-Funktion:

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

Für jeden Slot-Typ können maximal 10 000 Werte und Synonyme definiert werden. Jeder Bot kann maximal 50 000 Slot-Typenwerte und Synonyme aufweisen. Sie können beispielsweise über 5 Slot-Typen mit jeweils 5 000 Werten und 5 000 Synonymen oder über 10 Slot-Typen mit jeweils 2 500 Werten und 2 500 Synonymen verfügen.

Ein benutzerdefinierter Slot-Typ sollte nicht denselben Namen wie die integrierten Slot-Typen haben. Beispielsweise sollte ein benutzerdefinierter Slot-Typ nicht mit den reservierten Schlüsselwörtern Datum, Nummer oder Bestätigung benannt werden. Diese Schlüsselwörter sind für integrierte Slot-Typen reserviert. Eine Liste aller integrierten Steckplatztypen finden Sie unter [Integrierte Steckplatztypen](#).

Grammatik-Slot-Typ

Mit dem Grammatik-Slot-Typ können Sie Ihre eigene Grammatik im XML-Format gemäß der SRGS-Spezifikation verfassen, um Informationen in einer Konversation zu sammeln. Amazon Lex V2 erkennt Äußerungen, die den in der Grammatik festgelegten Regeln entsprechen. Sie können auch semantische Interpretationsregeln mithilfe von ECMAScript Tags in den Grammatikdateien bereitstellen. Amazon Lex gibt dann die in den Tags festgelegten Eigenschaften als aufgelöste Werte zurück, wenn eine Übereinstimmung auftritt.

Sie können Grammatik-Slottomtypen nur in den Gebietsschemas Englisch (Australien), Englisch (Großbritannien) und Englisch (USA) erstellen.

Ein Grammatik-Slot-Typ besteht aus zwei Teilen. Der erste ist die Grammatik selbst, die im SRGS-Spezifikationsformat geschrieben wurde. Die Grammatik interpretiert die Äußerung des Benutzers. Wenn die Äußerung von der Grammatik akzeptiert wird, wird sie abgeglichen, andernfalls wird sie zurückgewiesen. Wenn eine Äußerung gefunden wird, wird sie an das Skript weitergegeben, sofern es eine solche gibt.

Das zweite ist Teil eines Grammatik-Slot-Typs. Es handelt sich um ein optionales Skript ECMAScript, das die Eingabe in die vom Slot-Typ zurückgegebenen aufgelösten Werte umwandelt. Sie können beispielsweise ein Skript verwenden, um gesprochene Zahlen in Ziffern umzuwandeln. ECMAScript Anweisungen sind im <tag>Element eingeschlossen.

Das folgende Beispiel ist im XML-Format gemäß der SRGS-Spezifikation, das eine gültige Grammatik zeigt, die von Amazon Lex V2 akzeptiert wird. Es definiert einen Grammatiksteckplatztyp, der Kartennummern akzeptiert und bestimmt, ob es sich um reguläre Konten oder Premium-Konten handelt. Weitere Informationen zur akzeptablen Syntax finden Sie in den [Definition der Grammatik](#) und den [Skriptformat](#) Themen.

```

<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>

```

Die obige Grammatik akzeptiert nur zwei Arten von Kartennummern: 7241 oder 7001. Beiden kann optional das Präfix „Kartennummer“ vorangestellt werden. Es enthält auch ECMAScript Tags, die für die semantische Interpretation verwendet werden können. Bei semantischer Interpretation würde die Äußerung „Karte Nummer sieben zwei vier eins“ das folgende Objekt zurückgeben:

```

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

```

Dieses Objekt wird als JSON-seralisierte Zeichenfolge in dem `resolvedValues` Objekt zurückgegeben, das von den [RecognizeText](#) Operationen, und zurückgegeben wird.

[RecognizeUtteranceStartConversation](#)

Einen Grammatik-Slot-Typ hinzufügen

Um einen Grammatik-Slot-Typ hinzuzufügen

1. Laden Sie die XML-Definition Ihres Slot-Typs in einen S3-Bucket hoch. Notieren Sie sich den Bucket-Namen und den Pfad zur Datei.

Note

Die maximale Dateigröße beträgt 100 KB.

2. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
3. Wählen Sie im linken Menü Bots und dann den Bot aus, dem Sie den Grammatik-Slot-Typ hinzufügen möchten.
4. Wählen Sie Sprachen anzeigen und wählen Sie dann die Sprache aus, zu der der Grammatik-Slot-Typ hinzugefügt werden soll.
5. Wählen Sie Slot-Typen anzeigen.
6. Wählen Sie Slot-Typ hinzufügen und anschließend Grammatik-Slot-Typ hinzufügen.
7. Geben Sie dem Slot-Typ einen Namen und wählen Sie dann Hinzufügen.
8. Wählen Sie den S3-Bucket aus, der Ihre Definitionsdatei enthält, und geben Sie den Pfad zur Datei ein. Wählen Sie Slot-Typ speichern.

Definition der Grammatik

Dieses Thema zeigt die Teile der SRGS-Spezifikation, die Amazon Lex V2 unterstützt. Alle Regeln sind in der SRGS-Spezifikation definiert. Weitere Informationen finden Sie in der W3C-Empfehlung der [Speech Recognition Grammar Specification Version 1.0](#).

Themen

- [Header-Erklärungen](#)
- [Unterstützte XML-Elemente](#)
- [Tokens](#)
- [Referenz zur Regel](#)
- [Sequenzen und Kapselung](#)

- [Wiederholt](#)
- [Sprache](#)
- [Tags](#)
- [Gewichte](#)

[Dieses Dokument enthält Material, das aus der W3C Speech Recognition Grammar Specification Version 1.0 \(verfügbar unter <https://www.w3.org/TR/speech-grammar/>\)](#).

Die Informationen zum Zitat lauten wie folgt:

[Copyright](#) © 2004 [W3C®](#) ([MIT](#), [ERCIM](#), [Keio](#)), Alle Rechte vorbehalten. [Es gelten die W3C-Regeln zur Haftung, zum Markenrecht, zur Verwendung von Dokumenten und zur Softwarelizenzierung.](#)

Das SRGS-Spezifikationsdokument, eine [W3C-Empfehlung](#), ist beim W3C unter der folgenden Lizenz erhältlich.

Text der Lizenz

License

Durch and/or das Kopieren dieses Dokuments oder des W3C-Dokuments, von dem aus diese Erklärung verlinkt ist, erklären Sie (der Lizenznehmer), dass Sie die folgenden Nutzungsbedingungen gelesen und verstanden haben und einhalten werden:

Die Erlaubnis, den Inhalt dieses Dokuments oder des W3C-Dokuments, von dem aus diese Erklärung verlinkt ist, in jedem Medium für jeden Zweck und ohne Gebühren oder Lizenzgebühren zu kopieren und zu verteilen, vorausgesetzt, dass Sie auf ALLEN Kopien des Dokuments oder Teilen davon, die Sie verwenden, Folgendes angeben:

- Ein Link oder eine URL zum ursprünglichen W3C-Dokument.
- [Der bereits bestehende Copyright-Hinweis des ursprünglichen Autors oder, falls er nicht existiert, ein Hinweis \(Hypertext wird bevorzugt, eine Textdarstellung ist jedoch zulässig\) in der Form: „Copyright © \[\\$date-of-document\] World Wide Web Consortium, \(MIT, ERCIM, Keio, Beihang\). <http://www.w3.org/Consortium/Legal/2015/doc-Lizenz>“](#)
- Falls vorhanden, der STATUS des W3C-Dokuments.

Sofern es der Platz erlaubt, sollte der vollständige Text dieser MITTEILUNG beigelegt werden.

Wir bitten darum, dass bei Software, Dokumenten oder anderen Artikeln oder Produkten, die Sie

aufgrund der Implementierung des Inhalts dieses Dokuments oder eines Teils davon erstellen, die Urheberschaft angegeben wird.

Gemäß dieser Lizenz wird kein Recht gewährt, Änderungen oder Ableitungen von W3C-Dokumenten zu erstellen, außer in den folgenden Fällen: Um die Implementierung der in diesem Dokument dargelegten technischen Spezifikationen zu erleichtern, darf jeder abgeleitete Werke und Teile dieses Dokuments in Software, in unterstützenden Materialien zur Software und in der Dokumentation von Software erstellen und verteilen, VORAUSGESETZT, dass alle diese Werke den folgenden Hinweis enthalten. Die Veröffentlichung abgeleiteter Werke dieses Dokuments zur Verwendung als technische Spezifikation ist jedoch ausdrücklich untersagt.

Darüber hinaus sind „Codekomponenten“ — Web-IDL in Abschnitten, die deutlich als Web-IDL gekennzeichnet sind, sowie W3C-definiertes Markup (HTML, CSS usw.) und Code in Computerprogrammiersprachen, die deutlich als Codebeispiele gekennzeichnet sind — unter der W3C-Softwarelizenz lizenziert.

Der Hinweis lautet:

„Copyright © 2015 W3C® (MIT, ERCIM, Keio, Beihang). Diese Software oder dieses Dokument enthält Material, das von [Titel und URI des W3C-Dokuments] kopiert oder daraus abgeleitet wurde.“

Haftungsausschlüsse

DIESES DOKUMENT WIRD „WIE ES IST“ ZUR VERFÜGUNG GESTELLT, UND DIE URHEBERRECHTSINHABER GEBEN KEINE AUSDRÜCKLICHEN ODER STILLSCHWEIGENDEN ZUSICHERUNGEN ODER GEWÄHRLEISTUNGEN AB, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF GARANTIE DER MARKTGÄNGIGKEIT, DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DER NICHTVERLETZUNG VON RECHTEN ODER DES TITELS, DASS DER INHALT DES DOKUMENTS FÜR JEDEN ZWECK GEEIGNET IST UND DASS DIE IMPLEMENTIERUNG SOLCHER INHALTE KEINE PATENTE, URHEBERRECHTE, MARKEN ODER ANDERE RECHTE DRITTER VERLETZT.

URHEBERRECHTSINHABER HAFTEN NICHT FÜR DIREKTE, INDIREKTE, BESONDERE ODER FOLGESCHÄDEN, DIE SICH AUS DER VERWENDUNG DES DOKUMENTS ODER DER AUFFÜHRUNG ODER UMSETZUNG SEINES INHALTS ERGEBEN.

Der Name und die Marken der Urheberrechtsinhaber dürfen ohne ausdrückliche, schriftliche vorherige Genehmigung NICHT für Werbung oder Werbung im Zusammenhang mit diesem Dokument oder seinem Inhalt verwendet werden. Das Urheberrecht an diesem Dokument verbleibt zu jeder Zeit bei den Rechteinhabern.

Header-Erklärungen

Die folgende Tabelle zeigt die Header-Deklarationen, die vom Grammatik-Slot-Typ unterstützt werden. Weitere Informationen finden Sie unter [Grammatik-Header-Deklarationen](#) in der W3C-Empfehlung zur Grammatikspezifikation für Spracherkennung, Version 1.

Deklaration	Spezifikationsanforderung	XML-Formular	Amazon Lex Lex-Unterstützung	Spezifikation
Grammatik version	Erforderlich	4.3 : version Attribut auf einem grammar Element	Erforderlich	SRGS
XML-Namensraum	Erforderlich (nur XML)	4.3 : xmlns Attribut auf einem grammar Element	Erforderlich	SRGS
Dokumenttyp	Erforderlich (nur XML)	4.3 : XML-DOCTYPE	Empfohlen	SRGS
Zeichencodierung	Empfohlen	4.4 : encoding Attribut in der XML-Deklaration	Empfohlen	SRGS
Sprache	Im Sprachmodus erforderlich Wird im DTMF-Modus ignoriert	4.5 : xml:lang Attribut auf einem Element grammar	Im Sprachmodus erforderlich Wird im DTMF-Modus ignoriert	SRGS
Mode	Optional	4.6 : mode Attribut auf einem Element grammar	Optional	SRGS

Deklaration	Spezifikationsanforderung	XML-Formular	Amazon Lex Lex-Unterstützung	Spezifikation
Stammregel	Optional	4.7 : root Attribut auf einem grammar Element	Erforderlich	SRGS
Tag-Format	Optional	4.8 : tag-format Attribut auf einem grammar Element	String-Literal und ECMAScript werden unterstützt	SRGS, SISR
Basis-URI	Optional	4.9 : xml:base Attribut auf einem grammar Element	Optional	SRGS
Aussprache-Lexikon	Optional, mehrere erlaubt	4.10 : Element lexicon	Wird nicht unterstützt	SERGS, BITTE
Metadaten	Optional, mehrere erlaubt	4.11.1 : Element meta	Erforderlich	SRGS
XML-Metadaten	Optional, nur XML	4.11.2 : Element metadata	Optional	SRGS
Markierung	Optional, mehrere erlaubt	4.12 : Element tag	Globale Tags werden nicht unterstützt	SRGS

Beispiel

```
<?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">
```

Unterstützte XML-Elemente

Amazon Lex V2 unterstützt die folgenden XML-Elemente für benutzerdefinierte Grammatiken:

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

Tokens

Die folgende Tabelle zeigt die Token-Spezifikationen, die vom Grammatik-Slot-Typ unterstützt werden. Weitere Informationen finden Sie unter [Tokens](#) in der W3C-Empfehlung der Speech Recognition Grammar Specification Version 1.

Token-Typ	Beispiel	Unterstützt?
Einzelnes Token ohne Anführungszeichen	hallo	Ja
Einzelnes Token ohne Anführungszeichen: nicht alphabetisch	2	Ja
Token in einfachen Anführung szeichen, kein Leerzeichen	"hello"	Ja, lassen Sie doppelte Anführungszeichen fallen,

Token-Typ	Beispiel	Unterstützt?
		wenn es nur ein einzelnes Token enthält
Zwei durch Leerraum abgegrenzte Tokens	Gute Reise	Ja
Vier durch Leerraum abgegrenzte Spielsteine	das ist ein Test	Ja
Einfaches Zeichen in Anführungszeichen, einschließlich Leerraum	„San Francisco	Nein
Einzelnes XML-Token im <token>Tag	<token>San Francisco</token>	Nein (entspricht einem Token in einfachen Anführungszeichen mit Leerraum)

Hinweise

- Token in einfachen Anführungszeichen einschließlich Leerzeichen — Die Spezifikation verlangt, dass Wörter, die in doppelte Anführungszeichen eingeschlossen sind, als ein einzelnes Token behandelt werden. Amazon Lex V2 behandelt sie als durch Leerzeichen getrennte Token.
- Einfaches XML-Token in <token><token> — Die Spezifikation verlangt, dass Wörter, die durch getrennt sind, ein Token darstellen. Amazon Lex V2 behandelt sie als durch Leerzeichen getrennte Token.
- Amazon Lex V2 gibt einen Validierungsfehler aus, wenn eine der beiden Verwendungen in Ihrer Grammatik gefunden wird.

Beispiel

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

</rule>

Referenz zur Regel

In der folgenden Tabelle sind die verschiedenen Formen der Regelreferenz zusammengefasst, die in Grammatikdokumenten möglich sind. Weitere Informationen finden Sie unter [Regelreferenz](#) in der W3C-Empfehlung zur Grammatikspezifikation für Spracherkennung, Version 1.

Art der Referenz	XML-Formular	Unterstützt
2.2.1 Expliziter Verweis auf lokale Regeln	<code><ruleref uri="#rulename"/></code>	Ja
2.2.2 Expliziter Verweis auf eine benannte Regel einer Grammatik, die durch einen URI identifiziert wird	<code><ruleref uri="grammarURI#rulename"/></code>	Nein
2.2.2 Impliziter Verweis auf die Grundregel einer Grammatik, die durch einen URI identifiziert wird	<code><ruleref uri="grammarURI"/></code>	Nein
2.2.2 Expliziter Verweis auf eine benannte Regel einer Grammatik, die durch einen URI mit einem Medientyp identifiziert wird	<code><ruleref uri="grammarURI#rulename" type="media-type"/></code>	Nein
2.2.2 Impliziter Verweis auf die Grundregel einer Grammatik, die durch einen URI mit einem Medientyp identifiziert wird	<code><ruleref uri="grammarURI" type="media-type"/></code>	Nein
2.2.3 Spezielle Regeldefinitionen	<code><ruleref special="NULL"/></code> <code><ruleref special="VOID"/></code>	Nein

Art der Referenz	XML-Formular	Unterstützt
	<code><ruleref special="GARBAGE"/></code>	

Hinweise

1. Grammatik-URI ist eine externe URI. Beispiel, `http://grammar.example.com/world-cities.grxml`.
2. Der Medientyp kann sein:
 - `application/srgs+xml`
 - `text/plain`

Beispiel

```
<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>
```


Sequenzen und Kapselung

Das folgende Beispiel zeigt die unterstützten Sequenzen. Weitere Informationen finden Sie unter [Sequenzen und Kapselung in der W3C-Empfehlung](#) der Speech Recognition Grammar Specification Version 1.

Beispiel

```
<!-- 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>
```

Wiederholt

Die folgende Tabelle zeigt die unterstützten wiederholten Erweiterungen für Regeln. Weitere Informationen finden Sie unter [Wiederholungen](#) in der W3C-Empfehlung der Spracherkennungs-Grammatikspezifikation Version 1.

XML-Formular	Behavior	Unterstützt?
Beispiel		
wiederholen="n"	Der enthaltene Ausdruck wird exakt „n“ mal wiederholt. „n“ muss „0“ oder eine positive Ganzzahl sein.	Ja
wiederholen="6"		
repeat="m-n"	Die enthaltene Erweiterung wird zwischen „m“ und „n“ -mal wiederholt (einschließlich). „m“ und „n“ müssen beide „0“ oder eine positive Ganzzahl sein,	Ja
wiederholen="4-6"		

XML-Formular	Behavior	Unterstützt?
Beispiel		
	und „m“ muss kleiner oder gleich „n“ sein.	
repeat="m-" wiederholen="3-"	Die enthaltene Erweiterung wird „m“ -mal oder öfter wiederholt (einschließlich). „m“ muss „0“ oder eine positive Ganzzahl sein. Zum Beispiel deklariert „3-“, dass die enthaltene Erweiterung drei-, vier-, fünf- oder öfter vorkommen kann.	Ja
repeat="0-1"	Die enthaltene Erweiterung ist optional.	Ja
<item repeat="2-4" repeat-prob="0.8">		Nein

Sprache

Die folgende Diskussion bezieht sich auf Sprachkennungen, die auf Grammatiken angewendet werden. Weitere Informationen finden Sie unter [Sprache](#) in der W3C-Empfehlung der Grammatikspezifikation für Spracherkennung, Version 1.

Standardmäßig ist eine Grammatik ein einsprachiges Dokument mit einer [Sprachkennung](#), die in der Sprachdeklaration im [Grammatikkopf](#) angegeben ist. Sofern nicht anders angegeben, werden alle Tokens in dieser Grammatik entsprechend der Grammatiksprache behandelt. Sprachdeklarationen auf Grammatikebene werden nicht unterstützt.

Beachten Sie im folgenden Beispiel Folgendes:

1. Die Grammatik-Header-Deklaration für die Sprache „en-US“ wird von Amazon Lex V2 unterstützt.

2. Sprachanhänge auf Artekelebene (markiert unter **red**) werden nicht unterstützt. Amazon Lex V2 gibt einen Validierungsfehler aus, wenn sich ein Sprachanhang von der Header-Deklaration unterscheidet.

```
<?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

Die folgende Diskussion bezieht sich auf Tags, die für Grammatiken definiert sind. Weitere Informationen finden Sie unter [Tags](#) in der W3C-Empfehlung der Version 1 der Grammatikspezifikation für die Spracherkennung.

Basierend auf der SRGS-Spezifikation können Tags auf folgende Weise definiert werden:

1. Als Teil einer Header-Deklaration, wie unter beschrieben [Header-Erklärungen](#).
2. Als Teil einer <rule>Definition.

Die folgenden Tag-Formate werden unterstützt:

- semantics/1.0(SISR, ECMAScript)
- semantics/1.0-literals(SISR-Zeichenkettenlitterale)

Die folgenden Tag-Formate werden nicht unterstützt:

- swi-semantics/1.0(Proprietär von Nuance)

Beispiel

```
<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>
```

Gewichte

Sie können das Gewichtungsattribut einem Element hinzufügen. Bei der Gewichtung handelt es sich um einen positiven Fließkommawert, der den Grad angibt, in dem die Phrase im Element während der Spracherkennung verstärkt wird. Weitere Informationen finden Sie unter [Gewichte](#) in der W3C-Empfehlung der Grammatikspezifikation für Spracherkennung, Version 1.

Gewichtungen müssen größer als 0 und kleiner oder gleich 10 sein und dürfen nur eine Dezimalstelle haben. Wenn die Gewichtung größer als 0 und kleiner als 1 ist, wird die Phrase negativ verstärkt. Wenn die Gewichtung größer als 1 und kleiner oder gleich 10 ist, wird die Phrase positiv verstärkt. Eine Gewichtung von 1 bedeutet, dass der Satz überhaupt nicht gewichtet wird, und es gibt keine Verstärkung für den Ausdruck.

Es ist eine schwierige Aufgabe, Elementen eine angemessene Gewichtung zuzuweisen, um die Spracherkennungsleistung zu verbessern. Hier sind einige Tipps, die Sie beim Zuweisen von Gewichten beachten können:

- Beginnen Sie mit einer Grammatik ohne zugewiesene Gewichte für die einzelnen Elemente.
- Stellen Sie fest, welche Muster in der Sprache häufig falsch erkannt werden.
- Wenden Sie unterschiedliche Werte für Gewichtungen an, bis Sie eine Verbesserung der Spracherkennungsleistung feststellen und es keine Regressionen mehr gibt.

Beispiel 1

Wenn Sie beispielsweise eine Grammatik für Flughäfen haben und feststellen, dass New York häufig fälschlicherweise als Newark identifiziert wird, können Sie New York positiv beeinflussen, indem Sie ihm eine Gewichtung von 5 zuweisen.

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

```

    </item>
    <item>
        Newark
        <tag>out="Newark"</tag>
    </item>
</one-of>
</rule>

```

Beispiel 2

Sie haben beispielsweise eine Grammatik für den Reservierungscode der Fluggesellschaft, die mit einem englischen Alphabet beginnt, gefolgt von drei Ziffern. Der Reservierungscode beginnt höchstwahrscheinlich mit B oder D, aber Sie stellen fest, dass B häufig fälschlicherweise als P und D als T identifiziert wird. Sie können B und D positiv verstärken.

```

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

```

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

Skriptformat

Amazon Lex V2 unterstützt die folgenden ECMAScript Funktionen zum Definieren von Grammatiken.

Amazon Lex V2 unterstützt die folgenden ECMAScript Funktionen bei der Angabe von Tags in der Grammatik. tag-format muss an gesendet werden, semantics/1.0 wenn ECMAScript Tags in der Grammatik verwendet werden. Weitere Informationen finden Sie in der [Sprachspezifikation 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">
```

Themen

- [Variable Aussage](#)
- [Ausdrücke](#)
- [Wenn Aussage](#)
- [Switch-Anweisung](#)
- [Deklarationen von Funktionen](#)
- [Iterationsanweisung](#)
- [Anweisung blockieren](#)
- [Kommentare](#)
- [Nicht unterstützte Anweisungen](#)

[Dieses Dokument enthält Material aus dem ECMAScript Standard \(verfügbar unter https://www.ecma-international.org/publications-and-standards/standards/ecma-262/\)](https://www.ecma-international.org/publications-and-standards/standards/ecma-262/). Das Dokument mit den ECMAScript Sprachspezifikationen ist bei Ecma International unter der folgenden Lizenz erhältlich.

Text der Lizenz

© 2020 Ecma International

Dieses Dokument darf kopiert, veröffentlicht und an andere verteilt werden, und bestimmte davon abgeleitete Werke dürfen ganz oder teilweise erstellt, kopiert, veröffentlicht und verteilt werden, vorausgesetzt, dass der obige Copyright-Hinweis und diese Urheberrechtslizenz und dieser Haftungsausschluss auf all diesen Kopien und abgeleiteten Werken enthalten sind. Die einzigen abgeleiteten Werke, die im Rahmen dieser Urheberrechtslizenz und dieses Haftungsausschlusses zulässig sind, sind:

- (i) Werke, die dieses Dokument ganz oder teilweise zum Zweck der Kommentierung oder Erläuterung enthalten (z. B. eine kommentierte Version des Dokuments),
- (ii) Werke, die dieses Dokument ganz oder teilweise enthalten, um Barrierefreiheit zu gewährleisten,
- (iii) Übersetzungen dieses Dokuments in andere Sprachen als Englisch und in verschiedene Formate und
- (iv) verwendet diese Spezifikation in standardkonformen Produkten, indem die darin enthaltene Funktionalität implementiert wird (z. B. durch Kopieren und Einfügen ganz oder teilweise).

Der Inhalt dieses Dokuments selbst darf jedoch in keiner Weise verändert werden, auch nicht durch Entfernen des Copyright-Hinweises oder Verweise auf Ecma International, es sei denn, dies ist erforderlich, um es in eine andere Sprache als Englisch oder in ein anderes Format zu übersetzen.

Die offizielle Version eines Dokuments von Ecma International ist die englische Sprachversion auf der Website von Ecma International. Im Falle von Abweichungen zwischen einer übersetzten Version und der offiziellen Version ist die offizielle Version maßgebend.

Die oben gewährten eingeschränkten Genehmigungen sind unbefristet und können von Ecma International oder ihren Nachfolgern oder Abtretungsempfängern nicht widerrufen werden. Dieses Dokument und die darin enthaltenen Informationen werden „WIE BESEHEN“ zur Verfügung gestellt, und ECMA INTERNATIONAL LEHNT ALLE AUSDRÜCKLICHEN ODER STILLSCHWEIGENDEN GEWÄHRLEISTUNGEN AB, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF JEGLICHE GARANTIE, DASS DIE VERWENDUNG DER HIERIN ENTHALTENEN INFORMATIONEN KEINE EIGENTUMSRECHTE ODER STILLSCHWEIGENDE GEWÄHRLEISTUNGEN DER MARKTGÄNGIGKEIT ODER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK VERLETZT.“

Variable Aussage

Eine Variablenanweisung definiert eine oder mehrere Variablen.


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

Ausdrücke

Sie können Ausdruckszeichenfolgen hinzufügen, um Funktionen in Amazon Lex V2 auszuführen. Diese Tabelle zeigt die Syntax und Beispiele, die in SRGS-Ausdrücken verwendet werden können.

Typ des Ausdrucks	Syntax	Beispiel	Unterstützt?
Literaler Ausdruck für reguläre Ausdrücke	Zeichenkettenliteral, das gültige Regex-Sonderzeichen enthält	<code>"^d\.\$"</code>	Nein
Funktion	<code>function functionN ame(parameters) { functionBody }</code>	<pre>var x = function calc() { return 10; }</pre>	Nein
Löschen	<code>delete expression</code>	<code>delete obj.property;</code>	Nein
Void	<code>void expression</code>	<code>void (2 == '2');</code>	Nein
Typ von	<code>typeof expression</code>	<code>typeof 42;</code>	Nein
Index der Mitglieder	<code>expression [expressions]</code>	<pre>var fruits = ["apple"]; fruits[0];</pre>	Ja
Mitglied Punkt	<code>expression . identifier</code>	<code>out.value</code>	Ja

Typ des Ausdrucks	Syntax	Beispiel	Unterstützt?
Argumente	<code>expression (arguments)</code>	<code>new Date('1994-10-11')</code>	Ja
Erhöhung des Beitrags	<code>expression++</code>	<code>var x=10; x++;</code>	Ja
Nach dem Dekrement	<code>expression--</code>	<code>var x=10; x--;</code>	Ja
Nur inkrementiert	<code>++expression</code>	<code>var x=10; ++x;</code>	Ja
Vor dem Herunterfahren	<code>--expression</code>	<code>var x=10; --x;</code>	Ja
Unär plus//Unär minus	<code>+expression / - expression</code>	<code>+x / -x;</code>	Ja
Aber nicht	<code>~ expression</code>	<code>const a = 5; console.log(~a);</code>	Ja
Logisch nicht	<code>! expression</code>	<code>!(a > 0 b > 0)</code>	Ja
Multiplikativ	<code>expression ('*' '/' '%') expression</code>	<code>(x + y) * (a / b)</code>	Ja
Zusatzstoff	<code>expression ('+' '-') expression</code>	<code>(a + b) - (a - (a + b))</code>	Ja
Bitverschiebung	<code>expression ('<<' '>>' '>>>') expression</code>	<code>(a >> b) >>> c</code>	Ja

Typ des Ausdrucks	Syntax	Beispiel	Unterstützt?
Relativ	expression ('<' '>' '<=' '>=') expression	<pre>if (a > b) { ... }</pre>	Ja
In	expression in expression	<pre>fruits[0] in otherFruits;</pre>	Ja
Gleichheit	expression ('==' '!=' '===' '!==') expression	<pre>if (a == b) { ... }</pre>	Ja
Bit und/ xor/oder	expression ('&' '^' ' ') expression	<pre>a & b / a ^ b / a b</pre>	Ja
Logisch und/ oder	expression ('&&' ' ') expression	<pre>if (a && (b c)) { ...}</pre>	Ja
Ternär	expression ? expression : expression	<pre>a > b ? obj.prop : 0</pre>	Ja
Zuweisung	expression = expression	<pre>out.value = "string";</pre>	Ja
Zuweisungsoperator	expression ('*=' '/=' '+=' '-=' '%=') expression	<pre>a *= 10;</pre>	Ja

Typ des Ausdrucks	Syntax	Beispiel	Unterstützt?
Bitweiser Zuweisung operator	expression ('<=<' '>=>' '>>=>' '&=>' '^=>' ' =>') expression	<code>a <=< 10;</code>	Ja
Kennung	identifie rSequence wobei IdentifierSequence eine Folge gültiger Zeichen ist	<code>fruits=[10, 20, 30];</code>	Ja
Null-Literal	null	<code>x = null;</code>	Ja
Boolesches Literal	true false	<code>x = true;</code>	Ja
Zeichenketten-Literal	'string' / "string"	<code>a = 'hello', b = "world";</code>	Ja
Dezimales Literal	integer [.] digits [exponent >]	<code>111.11 e+12</code>	Ja
Hex-Literal	0 (x X)[0-9a-f A-F]	<code>0x123ABC</code>	Ja
Oktales Literalwort	0 [0-7]	<code>"051"</code>	Ja
Array-Literal	[expressio n, ...]	<code>v = [a, b, c];</code>	Ja
Objekt-Literal	{property: value, ...}	<code>out = {value: 1, flag: false};</code>	Ja

Typ des Ausdrucks	Syntax	Beispiel	Unterstützt?
In Klammern	(expressions)		Ja

Wenn Aussage

Sie können if-Anweisungen hinzufügen, um Funktionen in Amazon Lex V2 auszuführen. Dieses Beispiel zeigt die Syntax, die in SRGS-Ausdrücken verwendet werden kann.

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

Hinweis: Im vorherigen Beispiel `statements` muss es `expressions` sich um eines der unterstützten Programme in diesem Dokument handeln.

Switch-Anweisung

Sie können Switch-Anweisungen hinzufügen, um Funktionen in Amazon Lex V2 auszuführen. Dieses Beispiel zeigt die Syntax, die in SRGS-Ausdrücken verwendet werden kann.

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

Hinweis: Im vorherigen Beispiel `statements` muss es `expressions` sich um eines der unterstützten Programme in diesem Dokument handeln.

Deklarationen von Funktionen

Sie können Funktionsdeklarationen hinzufügen, um Funktionen in Amazon Lex V2 auszuführen. Dieses Beispiel zeigt die Syntax, die in SRGS-Ausdrücken verwendet werden kann.

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

Iterationsanweisung

Iterationsanweisungen können eine der folgenden sein:

```
// 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  
}
```

Anweisung blockieren

Sie können Anweisungsblöcke hinzufügen, um Funktionen in Amazon Lex V2 auszuführen. Dieses Beispiel zeigt die Syntax, die in SRGS-Ausdrücken verwendet werden kann.

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

```
}  
}
```

Hinweis: Im vorherigen Beispiel muss es sich bei den im Block statements angegebenen Angaben um eine der unterstützten Optionen aus diesem Dokument handeln.

Kommentare

Sie können Kommentare in Amazon Lex V2 hinzufügen. Dieses Beispiel zeigt die Syntax, die in SRGS-Ausdrücken verwendet werden kann.

```
// Single Line Comments  
"// <comment>"  
  
// Multiline comments  
/**  
<comment>  
**/
```

Nicht unterstützte Anweisungen

Amazon Lex V2 unterstützt die folgenden ECMAScript Funktionen nicht.

Themen

- [Leere Aussage](#)
- [Anweisung fortsetzen](#)
- [Break-Anweisung](#)
- [Erklärung zurückgeben](#)
- [Aussage werfen](#)
- [Anweisung versuchen](#)
- [Debugger-Anweisung](#)
- [Beschriftete Anweisung](#)
- [Klassendeklaration](#)

Leere Aussage

Die leere Anweisung wird verwendet, um keine Aussage abzugeben. Das Folgende ist die Syntax für eine leere Anweisung:

```
;
```

Anweisung fortsetzen

Die Continue-Anweisung ohne Label wird mit der unterstützt [Iterationsanweisung](#). Die Continue-Anweisung mit einem Label wird nicht unterstützt.

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

Break-Anweisung

Die Break-Anweisung ohne Label wird mit der unterstützt [Iterationsanweisung](#). Die Break-Anweisung mit einer Bezeichnung wird nicht unterstützt.

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

Erklärung zurückgeben

```
return expression;
```

Aussage werfen

Die Anweisung throw wird verwendet, um eine benutzerdefinierte Ausnahme auszulösen.

```
throw expression;
```

Anweisung versuchen

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



```
}  
finally {  
    statements  
}
```

Debugger-Anweisung

Die Debugger-Anweisung wird verwendet, um die von der Umgebung bereitgestellte Debugging-Funktionalität aufzurufen.

```
debugger;
```

Beschriftete Anweisung

Die beschriftete Anweisung kann mit `break` `continue` Oder-Anweisungen verwendet werden.

```
label:  
    statements  
  
// Example  
let str = '';  
  
loop1:  
for (let i = 0; i < 5; i++) {  
    if (i === 1) {  
        continue loop1;  
    }  
    str = str + i;  
}  
  
console.log(str);
```

Klassendeklaration

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

Grammatiken der Branche

Industriegrammatiken sind eine Reihe von XML-Dateien, die mit dem [Grammatik-Slot-Typ](#) verwendet werden können. Sie können diese verwenden, um schnell ein einheitliches Endbenutzererlebnis zu bieten, wenn Sie interaktive Sprachdialog-Workflows zu Amazon Lex V2 migrieren. Sie können aus einer Reihe vorgefertigter Grammatiken für drei Bereiche wählen: Finanzdienstleistungen, Versicherungen und Telekommunikation. Es gibt auch einen generischen Satz von Grammatiken, den Sie als Ausgangspunkt für Ihre eigenen Grammatiken verwenden können.

Die Grammatiken enthalten die Regeln zum Sammeln der Informationen und die [ECMAScriptTags für die semantische Interpretation](#).

[Grammatiken für Finanzdienstleistungen \(Download\)](#)

Die folgenden Grammatiken werden für Finanzdienstleistungen unterstützt: Konto- und Bankleitzahlen, Kreditkarten- und Kreditnummern, Kreditwürdigkeit, Kontoeröffnungs- und Abschlussdaten sowie Sozialversicherungsnummer.

Kontonummer

```
<?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="1-1">
    <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>

```

Bankleitzahl

```

<?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>

```

Kreditkartennummer

```

<?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>

```


Kredit-ID

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

```

```

        <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>

```

Kredit-Score

```

<?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>

```

Datum der Kontoeröffnung

```

<?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>
    </one-of>
  </rule>

```



```

        <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>

```

Automatisches Zahlungsdatum

```

<?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>
    </one-of>
</rule>

```

```

        <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>

```

Ablaufdatum der Kreditkarte

```

<?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>
  </one-of>
</rule>

```

```

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

```



```

        <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>

```

Datum der Abrechnung

```

<?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>

```

Datum der Transaktion

```

<?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>
  </one-of>
</rule>

```

```

        <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>

```

Betrag der Überweisung

```

<?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>
    </one-of>
</rule>

```

```

        <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>

```

Sozialversicherungsnummer

```

<?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>
```

Grammatiken für Versicherungen ([Download](#))

Die folgenden Grammatiken werden für den Versicherungsbereich unterstützt: Schaden- und Versicherungsnummern, Führerschein- und Nummernschilder, Ablaufdaten, Start- und Verlängerungsdaten, Schaden- und Versicherungsbeträge.

Nummer des Antrags

```
<?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>

```

Policy ID (Richtlinien-ID)

```

<?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="1-1">
        <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>

```

```

        </one-of>
      </item>
    </rule>
  </grammar>

```

Nummer des Führerscheins

```

<?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>

```

Kfz-Kennzeichen

```
<?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>
    </one-of>
  </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.firstOccurrence +=
rules.digits.numbers; out.letters += out.firstOccurrence;</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>

```

Ablaufdatum der Kreditkarte

```

<?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>

```

Ablaufdatum der Police, day/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 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>

```

Verlängerungsdatum der Police, Monat/Jahr

```

<?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>
    </one-of>
</rule>

```

```

        <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>

```

Startdatum der Police

```

<?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>
  </one-of>
</rule>

```

```

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

```

```

        <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>

```

Betrag des Antrags

```

<?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>
  </one-of>
</rule>

```

```

        <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>

```

Höhe der Prämie

```

<?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>

```

Menge der Police

```

<?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>

```

Grammatiken für Telekommunikation ([Herunterladen](#))

Die folgenden Grammatiken werden für Telekommunikation unterstützt: Telefonnummer, Seriennummer, SIM-Nummer, US-Postleitzahl, Ablaufdatum der Kreditkarte, Start-, Verlängerungs- und Ablaufdatum des Tarifs, Startdatum des Dienstes, Gerätemenge und Rechnungsbetrag.

Phone number (Telefonnummer)

```

<?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>

```

Seriennummer

```

<?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>

```

SIM-Nummer

```

<?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="1-1">
    <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>

```

US-Postleitzahl

```

<?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>0h<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>
    </one-of>
  </item>
</rule>

```

```

        <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>

```

Ablaufdatum der Kreditkarte

```

<?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>
  </one-of>
</rule>

```

```

        <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>
</rule>

```

```

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

```

Ablaufdatum des Plans, day/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 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>
    </one-of>
</rule>

```

```

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

```

```

        <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>

```

Verlängerungsdatum des Plans, Monat/Jahr

```

<?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>

```

Startdatum des Plans, Monat/Tag

```

<?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>
    </one-of>
</rule>

```

```

        <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>

```

Startdatum des Dienstes, Monat/Tag

```

<?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>
  </one-of>
</rule>

```

```

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

```



```

        <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>

```

Menge der Ausrüstung

```

<?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>

```

Rechnungsbetrag

```

<?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>
    </one-of>
    <item repeat="0-1"><ruleref uri="#currency"/></item>
</rule>

```

```

        <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>
```

Generische Grammatiken ([herunterladen](#))

Wir bieten die folgenden generischen Grammatiken an: alphanumerisch, Währung, Datum (mm/dd/yy), Zahlen, Begrüßung, Zögern und Bevollmächtigte.

Alphanumerisch

```
<?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>

```

Währung

```

<?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>

```

Datum, 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>
    </one-of>
  </rule>

```

```

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

```

```

        <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>

```


Datum, MM/JJ

```

<?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>
    </one-of>
  </rule>

```

```

        <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>

```

Datum, dd/mm/yyyy

```

<?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>
  </one-of>
</rule>

```

```
<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>

```

Zahlen, Ziffern

```

<?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>

```

Zahlen, Ordinalzahlen

```

<?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>
    </one-of>
</rule>

```



```

        <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>

```

Kundendienstmitarbeiter

```
<?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>

```

Begrüßung

```

<?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>

```

Zögern

```

<?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>

```

Steckplatztyp aus Verbundwerkstoff

Ein zusammengesetzter Steckplatz ist eine Kombination aus zwei oder mehr Steckplätzen, die mehrere Informationen in einer einzigen Benutzereingabe erfassen. Sie können den Bot beispielsweise so konfigurieren, dass er den Standort ermittelt, indem er nach „Stadt und Bundesland oder Postleitzahl“ fragt. Im Gegensatz dazu, wenn die Konversation so konfiguriert ist, dass

separate Slot-Typen verwendet werden, was zu einem starren Konversationserlebnis führt („Was ist die Stadt?“ gefolgt von „Was ist die Postleitzahl?“). Mit einem Verbundsteckplatz können Sie alle Informationen über einen einzigen Steckplatz erfassen. Ein zusammengesetzter Slot ist eine Kombination von Slots, die als Unterslots bezeichnet werden, z. B. Stadt, Bundesland und Postleitzahl.

Sie können eine Kombination aus verfügbaren Amazon Lex Lex-Slottypen (integriert) und Ihren eigenen Steckplätzen (benutzerdefinierte Steckplätze) verwenden. Sie können logische Ausdrücke entwerfen, um Informationen innerhalb der erforderlichen Teilbereiche zu erfassen. Zum Beispiel: Stadt und Bundesland oder Postleitzahl.

Der Composite-Steckplatztyp ist nur in den USA verfügbar.

Einen zusammengesetzten Steckplatztyp erstellen

Um Teilsteckplätze innerhalb eines Verbundsteckplatzes zu verwenden, müssen Sie zuerst den Verbundsteckplatztyp konfigurieren. Verwenden Sie dazu die Konsolenschritte zum Hinzufügen eines Slot-Typs oder den API-Vorgang. Nachdem Sie den Namen und eine Beschreibung für den Verbundsteckplatztyp ausgewählt haben, müssen Sie Informationen für Teilsteckplätze angeben. Weitere Informationen zum Hinzufügen eines Slot-Typs finden Sie unter [Slot-Typen hinzufügen](#)

Untersteckplätze

Ein zusammengesetzter Steckplatztyp erfordert die Konfiguration der zugrunde liegenden Steckplätze, die als Untersteckplätze bezeichnet werden. Wenn Sie in einer Anfrage mehrere Informationen von einem Kunden abrufen möchten, konfigurieren Sie eine Kombination von Teilsteckplätzen. Zum Beispiel: Stadt, Bundesland und Postleitzahl. Sie können bis zu 6 Untersteckplätze für einen zusammengesetzten Steckplatz hinzufügen.

Steckplätze einzelner Steckplatztypen können verwendet werden, um dem Verbundsteckplatztyp Untersteckplätze hinzuzufügen. Sie können jedoch einen zusammengesetzten Steckplatztyp nicht als Steckplatztyp für einen Teilsteckplatz verwenden.

Die folgenden Bilder veranschaulichen den zusammengesetzten Steckplatz „Auto“, bei dem es sich um eine Kombination von Teilsteckplätzen handelt: Farbe, Hersteller FuelType, Modell, Fahrgestellnummer und Jahr.

Slot type [Info](#)

Slot type name

Cars ▼

Create slot type

Subslots

Color, FuelType, Manufacturer, Model, VIN, Year

[View slot type details](#)

Slot expression - optional [Info](#)

Define the combination of subslots that your bot prompts for. If you don't define an expression, Amazon Lex prompts for all subslots.

(Color AND FuelType AND Manufacturer) OR (VIN AND Year)

Use , to separate different subslots; Use (), AND, OR to complete the expression.

Subslots [Info](#)

Subslot name	Subslot type	
Color	Colors	Remove
FuelType	FuelTypes	Remove
Manufacturer	Manufacturers	Remove
Model	Models	Remove
VIN	AMAZON.AlphaNumeric	Remove
Year	Years	Remove

Add new subslot

You have reached the limit of 6 subslots.

Generator für Ausdrücke

Um die Ausführung eines zusammengesetzten Slots voranzutreiben, können Sie optional den Ausdrucks-Generator verwenden. Mit dem Ausdrucks-Generator können Sie einen logischen Slot-Ausdruck entwerfen, um die erforderlichen Teilslot-Werte in der gewünschten Reihenfolge zu erfassen. Als Teil des booleschen Ausdrucks können Sie Operatoren wie AND und OR verwenden.

Basierend auf dem entworfenen Ausdruck gilt der zusammengesetzte Slot als erfüllt, wenn die erforderlichen Teilfelder erfüllt sind.

Es wird ein zusammengesetzter Steckplatztyp verwendet

In einigen Fällen möchten Sie möglicherweise verschiedene Steckplätze als Teil eines einzelnen Steckplatzes erfassen. Zum Beispiel könnte ein Bot zur Planung von Fahrzeugwartungsarbeiten eine Absicht mit der folgenden Äußerung haben:

My car is a {car}

Die Absicht geht davon aus, dass der zusammengesetzte Slot {car} eine Liste der Steckplätze enthält, die Einzelheiten des Fahrzeugs enthält. Zum Beispiel „2021 White Toyota Camry“.

Der zusammengesetzte Steckplatz unterscheidet sich von einem Steckplatz mit mehreren Werten. Der Verbundsteckplatz besteht aus mehreren Steckplätzen, von denen jeder seinen eigenen Wert hat. Bei einem Slot mit mehreren Werten handelt es sich dagegen um einen einzelnen Slot, der eine Liste von Werten enthalten kann. Weitere Informationen zu Slots mit mehreren Werten finden Sie unter [Verwendung mehrerer Werte in einem Slot](#)

Für einen zusammengesetzten Steckplatz gibt Amazon Lex in der Antwort auf die `RecognizeUtterance` Operation `RecognizeText` oder einen Wert für jeden Teilsteckplatz zurück. Die folgenden Slot-Informationen wurden für die Äußerung zurückgegeben: „Ich möchte vom Bot einen Service für meinen „weißen Toyota Camry 2021“ vereinbaren. 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"
    }
  }
},
...
}

```

Ein zusammengesetzter Slot kann in der ersten oder n-ten Runde einer Konversation ausgelöst werden. Auf der Grundlage der eingegebenen Eingabewerte kann der Verbundsteckplatz für die verbleibenden erforderlichen Teilslots eine Abfrage durchführen.

Verbundsteckplätze geben immer einen Wert für jeden Teilsteckplatz zurück. Wenn die Äußerung keinen erkennbaren Wert für einen bestimmten Teilschlitz enthält, wird für diesen bestimmten Teilschlitz keine Antwort zurückgegeben.

Composite-Slots funktionieren sowohl mit Text- als auch mit Spracheingabe.

Wenn Sie einem Intent einen Slot hinzufügen, ist ein Composite-Slot nur als benutzerdefinierter Slot-Typ verfügbar.

In Eingabeaufforderungen können Sie Verbundsteckplätze verwenden. Sie können beispielsweise die Bestätigungsaufforderung für eine Absicht festlegen.

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

Wenn Amazon Lex die Aufforderung an den Benutzer sendet, wird Folgendes gesendet: „Möchten Sie, dass ich den Service für Ihren weißen Toyota Camry 2021 plane?“

Jeder Untersteckplatz ist als Steckplatz konfiguriert. Sie können Slot-Eingabeaufforderungen hinzufügen, um die Äußerungen des Teilfensters und des Samples zu ermitteln. Sie können „Warten und Fortfahren“ für einen Teilbereich sowie Standardwerte aktivieren. Weitere Informationen finden Sie unter [Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot](#).

Cars (Composite) | Color | FuelType | Manufacturer | Model | VIN | Year

Car (Composite)

Slot prompts [Info](#)
Prompts to elicit the slot.

▶ Bot elicits information
Message: *What car do you have?*

▼ Sample utterances (0) - optional [Info](#)
Phrases that a user might use to provide the slot value. A comprehensive set of pre-defined utterances is included. You can add more if required.

No sample utterances
You haven't added any sample utterances yet.

Maximum 250 characters. Valid characters: A-Z, a-z, 0-9, @, #, \$

Sie können die Slot-Obfuscation verwenden, um den gesamten zusammengesetzten Slot in Gesprächsprotokollen zu maskieren. Bitte beachten Sie, dass die Slot-Obfuscation auf der Ebene der zusammengesetzten Slots angewendet wird. Wenn diese Option aktiviert ist, werden die Werte für Teilslots, die zu einem zusammengesetzten Slot gehören, verschleiert. Wenn Sie Slot-Werte verschleiern, wird der Wert der einzelnen Slot-Werte durch den Namen des Slots ersetzt. Weitere Informationen finden Sie unter [Verdecken von Slot-Werten in Konversationsprotokollen von Lex V2](#).

Slot info

Slot info [Info](#)

Slot name

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _.

Description - optional

Maximum 200 characters.

☒ **Required for this intent**

☐ **Enable slot obfuscation for entire slot**

☐ Color: Store as {Color}

☐ FuelType: Store as {FuelType}

☐ Manufacturer: Store as {Manufacturer}

☐ Model: Store as {Model}

☐ VIN: Store as {VIN}

☐ Year: Store as {Year}

Bearbeiten eines Steckplatztyps für Verbundwerkstoffe

Sie können einen Teilsteckplatz innerhalb der Verbundsteckplatzkonfiguration bearbeiten, um den Namen und den Steckplatztyp des Teilsteckplatzes zu ändern. Wenn jedoch ein zusammengesetzter Slot von einem Intent verwendet wird, müssen Sie die Absichten bearbeiten, bevor Sie den untergeordneten Slot ändern können.

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

Löschen eines zusammengesetzten Slot-Typs

Sie können einen Teilsteckplatz innerhalb der Verbundsteckplatzkonfiguration löschen. Bitte beachten Sie, dass, wenn ein Teilslot innerhalb eines Intents verwendet wird, die Unterslots trotzdem aus diesem Intent entfernt werden.

Delete slot type Address? ×

 This slot type is used by slots in existing intents. To build the language successfully, you may need to configure intents after deleting it.

This slot type **Address** will be deleted and cannot be recovered later.

Cancel Delete

Der Slot-Ausdruck im Expression Builder informiert mit einer Warnung über die gelöschten Teilfelder.

Slot type [Info](#)

Slot type name

Cars ▼



Create slot type

Subslots
Color, FuelType, Manufacturer, Model, VIN, Year
[View slot type details](#)

Slot expression - *optional* [Info](#)
Define the combination of subslots that your bot prompts for. If you don't define an expression, Amazon Lex prompts for all subslots.

(Color AND FuelType AND Manufacturer) OR (VIN AND Year)

Use , to separate different subslots; Use (), AND, OR to complete the expression.

Einen Bot mit der Konsole testen

Die Amazon Lex V2-Konsole enthält ein Testfenster, in dem Sie die Interaktion mit Ihrem Bot testen können. Sie verwenden das Testfenster, um eine Testkonversation mit Ihrem Bot zu führen und die Antworten zu sehen, die Ihre Anwendung vom Bot erhält.

Es gibt zwei Arten von Tests, die Sie mit Ihrem Bot durchführen können. Der erste, Express-Test, ermöglicht es Ihnen, Ihren Bot mit genau den Phrasen zu testen, die Sie bei der Erstellung des Bots verwendet haben. Wenn Sie Ihrer Absicht beispielsweise die Äußerung „Ich möchte Blumen pflücken“ hinzugefügt haben, können Sie den Bot mit genau dieser Phrase testen.

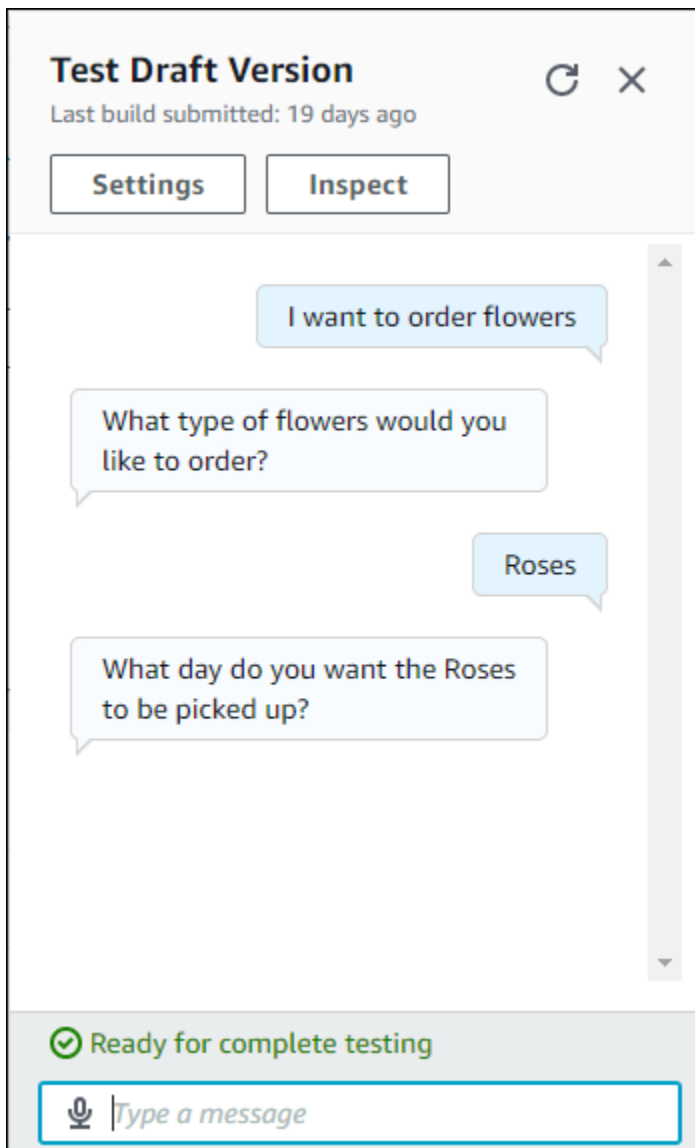
Beim zweiten Typ, dem vollständigen Testen, können Sie Ihren Bot anhand von Phrasen testen, die sich auf die von Ihnen konfigurierten Äußerungen beziehen. Sie können beispielsweise den Satz „Kann ich Blumen bestellen“ verwenden, um eine Konversation mit Ihrem Bot zu beginnen.

Sie testen einen Bot mit einem bestimmten Alias und einer bestimmten Sprache. Wenn Sie die Entwicklungsversion des Bots testen, verwenden Sie den `TestBotAlias` Alias zum Testen.

Um das Testfenster zu öffnen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie den zu testenden Bot aus der Liste der Bots aus.
3. Wählen Sie im linken Menü Aliase aus.
4. Wählen Sie aus der Liste der Aliase den Alias aus, den Sie testen möchten.
5. Wählen Sie unter Sprachen das Optionsfeld der zu testenden Sprache aus, und wählen Sie dann Test aus.

Nachdem Sie Test ausgewählt haben, wird das Testfenster in der Konsole geöffnet. Sie können das Testfenster verwenden, um mit Ihrem Bot zu interagieren, wie in der folgenden Grafik dargestellt.



Zusätzlich zur Konversation können Sie im Testfenster auch Inspect auswählen, um die vom Bot zurückgegebenen Antworten zu sehen. Die erste Ansicht zeigt Ihnen eine Zusammenfassung der Informationen, die Ihr Bot an das Testfenster zurückgegeben hat.

Inspect

Summary

JSON input and output

Intent

OrderFlowers

Slots	Elicitation
FlowerType	Roses
PickupDate	-
PickupTime	-

Active contexts	Number of turns or seconds
Weather	5 turns or 90s

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

Sie können das Testfenster auch verwenden, um die JSON-Strukturen zu sehen, die zwischen dem Bot und dem Testfenster gesendet werden. Sie können sowohl die Anfrage im Testfenster als auch die Antwort von Amazon Lex V2 sehen.

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*

Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI

Nutzen Sie die generativen KI-Funktionen von Amazon Bedrock, um Ihren Amazon Lex V2-Bot-Erstellungsprozess zu automatisieren und zu beschleunigen. Sie können die folgenden Prozesse mit Hilfe von Amazon Bedrock durchführen.

Note

Diese Funktionen verwenden generative KI. Denken Sie bei der Nutzung des Dienstes daran, dass er zu ungenauen oder unangemessenen Antworten führen kann. Weitere Informationen finden Sie in der [AWS-Richtlinie für verantwortungsvolle KI](#).

Bereitgestellt von Amazon Bedrock: AWS implementiert automatisierte Missbrauchserkennung. Da die generativen KI-Funktionen von Amazon Lex V2 auf Amazon Bedrock basieren, übernehmen Benutzer die in Amazon Bedrock implementierten Kontrollen, um Sicherheit und den verantwortungsvollen Umgang mit KI durchzusetzen.

- Erstellen Sie neue Bots und füllen Sie sie mithilfe von Beschreibungen in natürlicher Sprache effizient mit relevanten Absichten und Slot-Typen aus.
- Generieren Sie automatisch Beispieläußerungen für die Absichten Ihres Bots.
- Verbessern Sie die Leistung Ihrer Bots bei der Slot-Auflösung.
- Erstellen Sie eine Absicht, um die Fragen Ihrer Kunden zu beantworten.
- Verwenden Sie Amazon Bedrock Agents und Amazon Bedrock Knowledge Bases, um die Fragen Ihrer Kunden zu beantworten.

Sie können generative KI-Funktionen für Amazon Lex V2 entweder über die Konsole oder die API aktivieren.

Note

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).
2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Using the console

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie den Bot und das Gebietsschema im Bot aus, für das Sie generative KI-Funktionen aktivieren möchten.
3. Wählen Sie im Abschnitt Generative KI-Konfigurationen die Option Konfigurieren aus.
4. Klicken Sie für jede Funktion, die Sie aktivieren möchten, auf die Schaltfläche Aktiviert. Wählen Sie das Modell und die Version aus, die Sie für diese Funktion verwenden möchten. Durch die Aktivierung einer Funktion können zusätzliche Kosten anfallen. Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#). Um mehr über eine Funktion zu erfahren, wählen Sie das entsprechende Thema aus der folgenden Liste aus. Wählen Sie Speichern aus, nachdem Sie die Funktionen aktiviert haben, die Sie aktivieren möchten. Es erscheint ein grünes Erfolgsbanner, das bestätigt, dass die Funktionen aktiviert sind.

Using the API

1. Um generative KI-Funktionen für einen neuen Bot zu aktivieren, verwenden Sie den [CreateBot](#)Vorgang, um einen neuen Bot zu erstellen.
2. Senden Sie eine [CreateBotLocale](#)Anfrage und ändern Sie das `generativeAISettings` Objekt nach Bedarf. Wenn Sie die Funktionen für einen vorhandenen Bot aktivieren, senden Sie stattdessen eine [UpdateBotLocale](#)Anfrage.
 - a. Um die Verwendung des deskriptiven Bot-Builders zu aktivieren, ändern Sie das `descriptiveBotBuilder` Objekt. Geben Sie das Fundamentmodell an, das in dem `modelArn` Feld verwendet werden soll, und legen Sie den `enabled` Wert auf `True` fest.

- b. Um die Verbesserung der Steckplatzauflösung zu ermöglichen, ändern Sie das `slotResolutionImprovement` Objekt. Geben Sie das Fundamentmodell an, das in dem `modelArn` Feld verwendet werden soll, und legen Sie den `enabled` Wert auf `festTrue`.
- c. Um die Generierung von Beispieläußerungen zu aktivieren, ändern Sie das `sampleUtteranceGeneration` Objekt. Geben Sie das Fundamentmodell an, das in dem `modelArn` Feld verwendet werden soll, und legen Sie den `enabled` Wert auf `True` fest.

Themen

- [Verwenden Sie eine Beschreibung, um einen Bot in Lex V2 mit dem beschreibenden Bot-Builder zu erstellen](#)
- [Verwenden Sie die Generierung von Äußerungen, um Musteräußerungen zur Absichtserkennung zu generieren](#)
- [Verwendung der unterstützten Slot-Auflösung zur Verdeutlichung von Slot-Werten in Amazon Lex V2](#)
- [Verwendung `BedrockAgentIntent` eines Amazon Bedrock Agents in Amazon Lex V2](#)
- [Verbessern Sie die Absichtsklassifizierung und die Slot-Auflösung in Lex V2 mit unterstützter NLU](#)
- [Amazon.qn `AIIntent`](#)

Verwenden Sie eine Beschreibung, um einen Bot in Lex V2 mit dem beschreibenden Bot-Builder zu erstellen

Note

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).

2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Mit dem Descriptive Bot Builder können Sie den Zugriff von Amazon Bedrock auf umfangreiche Sprachmodelle nutzen, um die Effizienz des Bot-Erstellungsprozesses zu verbessern. Sie geben eine Aufforderung in natürlicher Sprache, die den Zweck des Bots und die Aktionen, die er ausführen soll, enthält. Amazon Lex V2 nutzt die Funktionen von Amazon Bedrock, um auf der Grundlage Ihrer Beschreibung relevante Absichten und Slot-Typen für Ihren Bot zu generieren. Sobald Sie die Intents und Slot-Typen ausgewählt haben, die Sie behalten möchten, können Sie den Bot testen, um ihn an Ihren spezifischen Anwendungsfall anzupassen. Der beschreibende Bot-BUILDER spart Ihnen Zeit, da Sie vermeiden müssen, Intents und Slot-Typen für den Bot manuell erstellen zu müssen.

Der Descriptive Bot Builder ist in den englischen Gebietsschemas verfügbar (die Gebietsschemas, die mit beginnen, finden Sie en_ in der Tabelle unter). [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#)

Bevor Sie Ihren Bot erstellen, gehen Sie wie folgt vor.

1. Überprüfen Sie anhand der Schritte unter, ob Ihre Rolle über die richtigen Berechtigungen verfügt [Erforderliche Berechtigungen, um einen Bot mit Beschreibung in natürlicher Sprache in Lex V2 zu erstellen](#).
2. Entscheiden Sie sich für die Beschreibung, die Sie verwenden möchten. Beispielbeschreibungen [Beispiele für Bot-Beschreibungen für einen beschreibenden Bot-BUILDER](#) für Bots finden Sie unter.

Erstellen Sie einen Bot, indem Sie in natürlicher Sprache beschreiben, wozu der Bot in der Lage sein sollte. Amazon Lex V2 ruft Amazon Bedrock-Modelle auf, um Intents und Slot-Typen zu generieren, die dem Anwendungsfall Ihres Bots entsprechen. Sie können den Bot entweder mit der Konsole oder der API erstellen.

Console

Erstellen Sie einen Bot mithilfe des beschreibenden Bot-Builders

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie auf der Seite Bots die Option Bot erstellen aus.

3. Wählen Sie für die Erstellungsmethode Descriptive Bot Builder - GenAI aus.
4. Geben Sie Ihrem Bot einen Namen und eine optionale Beschreibung, konfigurieren Sie die IAM-Berechtigungen und wählen Sie aus, ob Ihr Bot dem COPPA unterliegt oder nicht. Wählen Sie dann Weiter aus.
5. Wählen Sie eine Sprache, in der der Bot erstellt werden soll, eine Stimme für den Bot und einen Vertrauensschwellenwert für die Klassifizierung von Absichten aus (weitere Informationen finden Sie unter [Verwendung von Intent-Konfidenzwerten zur Verbesserung der Absichtsauswahl mit Lex V2](#)).
6. Geben Sie unter Descriptive Bot Builder — GenAI eine Beschreibung für den Bot ein, den Sie erstellen möchten. Ihre Beschreibung sollte sowohl detailliert als auch präzise sein, um angemessene und ausreichende Absichten für Ihren Bot zu generieren. Füge eine Liste von Maßnahmen hinzu, um den Prozess der Absichtserstellung zu verbessern.
7. Wählen Sie unter Modell auswählen einen Modellanbieter und ein Modell aus.
8. Um den Bot in einem anderen Gebietsschema zu erstellen, wählen Sie Weitere Sprache hinzufügen aus. Wenn Sie mit dem Hinzufügen von Sprachen fertig sind, wählen Sie Fertig aus. Amazon Lex V2 erstellt Ihren Bot und der Descriptive Bot Builder generiert Intents und Slots dafür. Wenn das Gebietsschema generiert wurde, wechselt das Banner von blau zu grün. Wähle „Überprüfen“ aus, um die generierten Absichten und Slot-Typen zu sehen.

 Note

Der beschreibende Bot-BUILDER ist derzeit nur in englischer Sprache verfügbar. Sie können einen Bot jedoch nach der Erstellung in ein anderes Gebietsschema als Englisch kopieren.

Überprüfe die generierten Intents und Slot-Typen und füge sie deinem Bot hinzu

1. Wenn es genügend Intents und Slot-Typen gibt, die für den Anwendungsfall Ihres Bots geeignet sind, können Sie die generierten Intents überprüfen.
 - a. Überprüfe die generierten Absichten.
 - i. Wählen Sie ein Kontrollkästchen neben einer Absicht aus, um sie aus der Liste der Absichten zu entfernen, die Sie dem Bot hinzufügen möchten.
 - ii. Wählen Sie einen Namen für die Absicht, um die für die Absicht generierten Beispieläußerungen und Slots anzuzeigen.

- iii. Standardmäßig sind alle Äußerungen und Slots ausgewählt. Wählen Sie ein Kontrollkästchen, um dieses Element aus der Absicht zu entfernen. Wählen Sie Zur Auswahl hinzufügen aus, um die markierten Elemente in der Absicht beizubehalten.
 - b. Überprüfen Sie die generierten Slot-Typen.
 - i. Wählen Sie ein Kontrollkästchen neben einem Slot-Typ, um ihn aus der Liste der Absichten zu entfernen, die dem Bot hinzugefügt werden sollen.
 - ii. Sie können einem Slot-Typ Werte hinzufügen, nachdem Sie ihn dem Bot hinzugefügt haben
2. Wenn Sie mit Ihren Absichten und Slot-Typen zufrieden sind, wählen Sie oben auf der Seite Intents und Slot-Typen hinzufügen aus, um die Absichten und Slot-Typen zu Ihrem Bot hinzuzufügen.
3. Wenn das Hinzufügen der Ressourcen abgeschlossen ist, erscheint ein grünes Erfolgsbanner. Gehen Sie zu Intents und Slot-Typen, um die generierten Intents zu bearbeiten und weitere Werte hinzuzufügen.
4. Wenn die Slot-Typen Generierte Absichten und Generierte Slot-Typen für den Bot, den Sie erstellen möchten, größtenteils nicht zutreffen, führen Sie die folgenden Schritte aus.
 - a. Wählen Sie im Abschnitt Descriptive Bot Builder Details die Option Neue Generation aus.
 - b. Schreiben Sie die Aufforderung neu und wählen Sie Erneut generieren, um neue Absichten und Slot-Typen zu generieren. Die Ergebnisse unterscheiden sich, wenn Sie ein anderes Modell verwenden.

**Important**

Es gibt keine Garantie dafür, dass dieselben Absichten und Slots generiert werden. Ihnen wird jedes Mal eine Gebühr berechnet, wenn Sie die Intents und Slot-Typen neu generieren.

API

Erstellen Sie den Bot mit einer Beschreibung in natürlicher Sprache

Wenn Sie den Descriptive Bot Builder über die API verwenden, erstellt er eine Bot-Definition in einer ZIP-Datei in einem Amazon S3 S3-Bucket. Sie laden diese Datei herunter und importieren die Bot-Definition in Amazon Lex V2, um Ihren Bot zu erstellen.

1. Senden Sie eine [CreateBot](#)Anfrage, um einen neuen Bot zu erstellen. Senden Sie dann eine [CreateBotLocale](#)Anfrage, um ein Gebietsschema für den Bot zu erstellen.
2. Senden Sie eine [StartBotResourceGeneration](#)Anfrage mit Angabe der ID, Version und des Gebietsschemas des Bots. Sie können DRAFT für die Bot-Version verwenden. Geben Sie Ihre Eingabeaufforderung in das `generationInputPrompt` Feld ein. Ihre Beschreibung sollte sowohl detailliert als auch präzise sein, um angemessene und ausreichende Absichten für Ihren Bot zu generieren. Füge eine Liste von Maßnahmen hinzu, um den Prozess der Absichtserstellung zu verbessern.
3. Notieren Sie sich das `generationId` in der Antwort.
4. Senden Sie eine [DescribeBotResourceGeneration](#)Anfrage mit der, die `generationId` Sie in der `StartBotResourceGeneration` Antwort erhalten haben. Geben Sie die Bot-ID, die Version und das Gebietsschema an.
5. Wenn das `generationStatus` in der `DescribeBotResourceGeneration` Antwort `Complete` steht, wird das `generatedBotLocaleUrl` Feld ebenfalls ausgefüllt. Verwenden Sie diesen Amazon S3 S3-URI, um die Bot-Definition [herunterzuladen, indem Sie den Schritten unter Objekt herunterladen](#) folgen.

Überprüfen Sie die generierte Bot-Definition und importieren Sie sie

1. Verwenden Sie den Amazon S3 S3-URI aus der `generationStatus` `DescribeBotResourceGeneration` Antwort, um die Bot-Definition [herunterzuladen, indem Sie den Schritten unter Objekt herunterladen](#) folgen.
2. Sie können den generierten Inhalt direkt für den spezifischen Anwendungsfall Ihres Bots ändern, indem Sie die Datei bearbeiten. Du kannst auch eine weitere `StartBotResourceGeneration` Anfrage senden, um Intents und Slots neu zu generieren.



Important

Es gibt keine Garantie dafür, dass dieselben Intents und Slots generiert werden. Ihnen wird jedes Mal eine Gebühr berechnet, wenn Sie die Intents und Slot-Typen neu generieren.

3. Um die Bot-Definition zu importieren, folgen Sie den Schritten unter. [Bots in Lex V2 importieren](#)
4. Nach dem Import können Sie die generierten Absichten und Slots mithilfe der [UpdateSlotType](#) Operationen [UpdateIntent](#) [UpdateSlot](#), und ändern.

Verwenden Sie den Vorgang, um Metadaten zu allen generierten Elementen für ein Bot-Gebietsschema aufzulisten. [ListBotResourceGenerations](#) Verwenden Sie einen der zurückgegebenen `generationId` Werte in einer `DescribeBotResourceGeneration` Anfrage, um den Amazon S3 S3-URI für eine generierte Bot-Definition abzurufen.

Themen

- [Beispiele für Bot-Beschreibungen für einen beschreibenden Bot-BUILDER](#)
- [Erforderliche Berechtigungen, um einen Bot mit Beschreibung in natürlicher Sprache in Lex V2 zu erstellen](#)

Beispiele für Bot-Beschreibungen für einen beschreibenden Bot-BUILDER

Hier sind einige hilfreiche Beispiel-Bot-Beschreibungen, die Sie mit dem Descriptive Bot Builder in Amazon Lex V2 verwenden können.

Industry	Beispiel für eine Aufforderung
Finanzdienstleistungen	Wir sind ein Service für Finanzkarten, der Benutzern hilft, Aufgaben zu erledigen, wenn sie eine neue Karte erhalten, wie z. B. die Aktivierung der Karte, das Senden einer PIN per E-Mail oder Post oder die Verifizierung einer neuen Karte (anhand einer Postleitzahl). Wir helfen ihnen auch bei Aufgaben im Zusammenhang mit ihrer bestehenden Karte, z. B. bei der Anfrage nach Kreditkartenvorteilen, beim Melden eines Kartenverlusts, beim Beantragen einer neuen Karte, beim Zurücksetzen einer Karten-PIN oder beim Bezahlen einer Kartenrechnung.
Verpflegungsdienstleistungen	Ich möchte, dass ein Bot Kunden hilft, Lebensmittel zu bestellen (anhand von Artikelnummer, Menge, Größe), den Bestellstatus zu überprüfen und eine Bestellung zu

Industry	Beispiel für eine Aufforderung
	stornieren. Verwende die Bestell-ID für die Indexierung von Bestellungen.
Fluggesellschaft	Wir sind eine Airline-Domain, die Benutzern hilft, Flugtickets zu buchen, Details einer Reservierung zu überprüfen, eine Quittung für einen gebuchten Flug zu erhalten, den Flugstatus abzufragen, gebuchte Flüge zu verschieben, Flugdetails zu ermitteln und gebuchte Flüge zu stornieren. Sie können auch zusätzliche Intents generieren, wenn diese Funktionen in der Domainbeschreibung unterstützen.
Versicherung	Ziel: Wir sind eine Versicherungsgesellschaft, die Auto-, Haus- und Rentenversicherungen verkauft. Ich möchte einen Bot, der den Status eines Antrags überprüfen, einen Anspruch geltend machen, Versicherungszahlungen leisten und eine Police kündigen kann. Wir verwenden policy_id und die letzten 4 von SSN für die Kontoidentifikation und -validierung. Ich erwarte, dass der Bot mindestens die folgenden Absichten und Slots hat: authentication - policy_id, SSNpolicy last4-Typ: Auto, Haus, Rentenversicherung Status: Saldo, Scheckfälligkeitsdatum, ScheckdeckungZahlung tätigen: einmalige Zahlung, Raten, Betrag

Industry	Beispiel für eine Aufforderung
Fahrzeugmanagement	Wir entwickeln einen Bot zur Suche nach abgeschleppten Autos, der Fahrern in einer Stadt, mit deren Auto abgeschleppt wurde, hilft, herauszufinden, wo sich das Auto befindet. Dieser Bot sollte nach der Adresse oder dem Ort fragen, von dem das Auto abgeschleppt wurde, sowie nach Einzelheiten zum Fahrzeug wie Nummernschild und Marke, Modell und Baujahr des Fahrzeugs. Der Bot sollte mit dem Standort des abgeschleppten Parkplatzes und den Öffnungszeiten antworten.
Reisen	Ich bin ein Reisebüro und möchte, dass ein Bot meinen Kunden hilft, eine Reise nach Disney zu buchen. Disney hat mehrere Parks auf der ganzen Welt zur Auswahl und bietet auch Hotels, Restaurants und besondere Unterhaltungsmöglichkeiten, die reserviert werden können. Benutzer des Bots sollten in der Lage sein, ihre Buchung zu ändern oder zu stornieren. Buchungen müssen mindestens den Park, die Daten und das Hotel beinhalten. Das Einbeziehen von Restaurants oder Unterhaltungsangeboten ist optional und kann später hinzugefügt oder geändert werden.

Erforderliche Berechtigungen, um einen Bot mit Beschreibung in natürlicher Sprache in Lex V2 zu erstellen

- Um auf diese Funktion auf der Amazon Lex V2-Konsole zugreifen zu können, stellen Sie sicher, dass Ihre Konsolenrolle über die entsprechenden `bedrock:ListFoundationModels` Berechtigungen verfügt.
- Die dem Bot zugeordnete IAM-Rolle sollte über eine `bedrock:InvokeModel` entsprechende Berechtigung verfügen. Wenn Sie die Funktion mit der Amazon Lex-Konsole aktivieren, wird die

Richtlinie automatisch zur Bot-Rolle hinzugefügt, sofern Ihr Bot eine von Amazon Lex generierte serviceverknüpfte Rolle verwendet.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:region::foundation-model/model-id"
      ]
    }
  ]
}
```

Verwenden Sie die Generierung von Äußerungen, um Musteräußerungen zur Absichtserkennung zu generieren

Note

Bevor Sie die Vorteile der generativen KI nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).
2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Verwenden Sie die Generierung von Äußerungen, um die Erstellung von Beispieläußerungen für Ihre Zwecke zu automatisieren. Anstatt Musteräußerungen manuell einzugeben, generiert Amazon Lex V2 Beispieläußerungen für Sie auf der Grundlage des Namens, der Beschreibung und vorhandener Beispieläußerungen, sodass Sie den Zeit- und Arbeitsaufwand für die Suche und das Schreiben Ihrer eigenen Beispieläußerungen reduzieren können. Nachdem Amazon Lex V2 Äußerungen generiert hat, können Sie die Äußerungen bearbeiten und löschen. Verwenden Sie dieses Tool, um die Erstellung von Musteräußerungen für den Prozess der Absichtserkennung zu beschleunigen.

Um die Generierung von Äußerungen zu ermöglichen, folgen Sie den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#). So aktivieren Sie die Funktionen der generativen KI.

Um auf diese Funktion auf der Amazon Lex V2-Konsole zugreifen zu können, stellen Sie sicher, dass Ihre Konsolenrolle über die `bedrock:InvokeModel` erforderlichen Berechtigungen verfügt `bedrock:ListFoundationModels`.

Sie können Äußerungen entweder mit der Konsole oder der API generieren.

Console

1. Navigiere zum Abschnitt Beispieläußerungen für jede Absicht in deinem Bot (im Visual Conversation Builder befindet sie sich im Startblock).
2. Wählen Sie die Schaltfläche „Äußerungen generieren“, um 5 Beispieläußerungen zu generieren. Wenn Ihre Absicht mehr als 25 Beispieläußerungen umfasst, wird die Schaltfläche „Äußerungen generieren“ deaktiviert.
3. Generierte Äußerungen werden mit einem grünen Banner angezeigt, das die generierten Äußerungen von den vorhandenen Äußerungen unterscheidet.
4. Bewegen Sie den Mauszeiger über eine Äußerung, um die Optionen zum Bearbeiten, Löschen und Sortieren der generierten Äußerungen anzuzeigen.

API

1. Senden Sie eine [GenerateBotElement](#)Anfrage und geben Sie die Absicht und die Bot-ID, die Version und das Gebietsschema ein, für das Sie Beispieläußerungen generieren möchten.
2. Die Antwort gibt eine Liste von [SampleUtterance](#)Objekten zurück, von denen jedes eine generierte Äußerung enthält.
3. Um die Äußerungen zur Absicht hinzuzufügen, senden Sie eine [UpdateIntent](#)Anfrage und fügen Sie die Äußerungen dem Feld hinzu. `sampleUtterances`

Verwendung der unterstützten Slot-Auflösung zur Verdeutlichung von Slot-Werten in Amazon Lex V2

Note

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).
2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Sie können die Genauigkeit einiger integrierter Slots im Konversationsablauf Ihres Bots verbessern, indem Sie die unterstützte Slot-Auflösung verwenden. Die unterstützte Slot-Auflösung verwendet große Sprachmodelle von Amazon Bedrock (LLMs), um die Erkennung einiger integrierter Slots zu verbessern, was zu einer besseren Interpretation der Kundenantworten bei der Slot-Abfrage führt. Bei Äußerungen, die nicht normal gelöst werden konnten, wird versucht, sie ein zweites Mal mit Amazon Bedrock zu lösen.

Mit der unterstützten Steckplatzauflösung können Sie die Leistung der Amazon Bedrock Foundation-Modelle nutzen, um die Genauigkeit der folgenden integrierten Steckplätze zu verbessern:

- `AMAZON.Alphanumericohne Regex`-Unterstützung
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

Sie können die unterstützte Steckplatzauflösung für jede Absicht aktivieren, die die oben aufgeführten integrierten Steckplätze verwendet. Die unterstützte Steckplatzauflösung gilt nicht für benutzerdefinierte Steckplätze oder integrierte Amazon-Steckplätze, die oben nicht aufgeführt sind.

Nachdem Sie die unterstützte Slot-Auflösung in Ihrem Bot aktiviert haben, können Sie mithilfe von Konversationsprotokollen und Messwerten Daten zu den Genauigkeitsverbesserungen sammeln.

- Gesprächsprotokolle — Interpretationen werden `interpretationSource` so aussehen, als ob Amazon Bedrock verwendet wurde, um den Slot zu lösen.
- CloudWatch Metriken — Metriken werden unter den unter Metrik aufgeführten Dimensionen veröffentlicht. CloudWatch Weitere Informationen finden Sie unter [Amazon Lex mit Amazon überwachen CloudWatch](#).

Um den beschreibenden Bot-Builder zu verwenden, stellen Sie sicher, dass Ihre IAM-Rolle über die richtigen Berechtigungen verfügt, indem Sie die Schritte unter ausführen. [In Lex V2 sind für die unterstützte Slot-Auflösung Berechtigungen erforderlich](#)

Themen

- [Beispiele für unterstützte Schlitzauflösung, die in Lex V2 verwendet werden](#)
- [Aktivieren Sie die unterstützte Steckplatzauflösung im Generative AI-Konfigurationsbildschirm](#)
- [Aktivieren Sie die unterstützte Steckplatzauflösung in den Steckplatzeinstellungen in Lex V2](#)
- [In Lex V2 sind für die unterstützte Slot-Auflösung Berechtigungen erforderlich](#)

Beispiele für unterstützte Schlitzauflösung, die in Lex V2 verwendet werden

Im Folgenden finden Sie einige Beispiele, bei denen die unterstützte Slot-Auflösung Benutzeräußerungen intelligent in einen Wert umwandeln kann.

Amazon.Nummer

Vertical	Steckplatztyp	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
Reisen	Amazon-Nummer	numberOfNightsBlieb	Wie viele Nächte warst du für	Eine ganze Woche, 7 Nächte.	7

Vertical	Steckplatztyp	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
			die Reise geblieben?		
Bankwesen	Amazon.Nummer	numberOfPeopleOnTheAccount	Wie viele Personen sind auf dem Konto?	Ich und meine Frau.	2
Reisen	Amazon-Nummer	numberOfStops	Wie viele Haltestellen?	Einmal in Japan. Einmal in LA.	2

AMAZONAS. AlphaNumeric

Vertical	Slot-Typ	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
Autoverleih	Amazon.Alphanumerisch	Transaktions-ID	Was ist Ihre Transaktions-ID?	Ich glaube, es war Alpha Whiskey Echo Acht Drei Vier Neun Romeo Julia.	AWE8349R. J.
Reisen	Amazon.Alphanumerisch	Bestätigungscode	Was ist die Bestätigungsnummer für Ihre Reservierung?	Die Bestätigungsnummer lautet BLT2 UE.	BLT2UE

Amazon.Datum

Vertical	Slot-Typ	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert	Aktuelles Datum
Autoverleih	Amazon.datum	Fälligkeitsdatum	Wann läuft der Mietvertrag aus?	Der Mietvertrag läuft am 1. des nächsten Monats aus.	2023-12-01	2023-11-09
Reisen	Amazon.datum	Datum der Rückgabe	Wann kehrst du zurück?	Später heute gegen 7 Uhr.	2023-11-09	2023-11-09

AMAZONAS. PhoneNumber

Vertical	Slot-Typ	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
Versicherung	AMAZON.PhoneNumber	Versicherungsnehmer	Wie lautet die Telefonnummer des Versicherungsnehmers?	Die Telefonnummer des Versicherungsnehmers lautet 123-456-7890.	1234567890
Einzelhandel	AMAZON.PhoneNumber	Telefon-Suche	Wie lautet Ihre Telefonnummer	Ich glaube, es ist unter 413-570-9	4135709617

Vertical	Slot-Typ	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
			mmer, damit ich Ihr Konto finden kann?	617, lassen Sie mich das noch einmal überprüfen.	

Amazon.Land

Vertical	Typ des Steckplatzes	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
Reisen	Amazon.country	Heimatland	Was ist dein Herkunftsland?	Ich bin Inder.	Indien
Bankwesen	Amazon.country	Reiseroute des Landes	In welche Länder werden Sie mit Ihrer Debitkarte reisen?	Ich werde nach Neu-Delhi reisen.	Indien

Amazon.city

Vertical	Slot-Typ	Absicht	Frage	Antwort	Gelöster Wert
Versicherung	Amazon.city	policyHolderCity	In welcher Stadt wohnt der Versicherungsnehmer?	Ich lebe in Springfield.	Springfield

Vertical	Slot-Typ	Absicht	Frage	Antwort	Gelöster Wert
Reisen	Amazon.city	Zielstadt	In welche Stadt reist du?	Ich reise nach Tokio.	Tokio

Amazon. Bestätigung

Vertical	Slot-Typ	slotName	Slot-Eingabeaufforderung	Äußerung	Gelöster Wert
Versicherung	Amazon.Bestätigung	Die Richtlinie ist abgelaufen	Ist die Versicherungspolice abgelaufen?	Ja, leider ist sie abgelaufen.	Ja
Bankwesen	Amazon.Bestätigung	Hat Investitionen	Haben Sie irgendwelche Investitionen?	Ich habe noch in nichts investiert.	Nein

Aktivieren Sie die unterstützte Steckplatzauflösung im Generative AI-Konfigurationsbildschirm

Sie können die unterstützte Steckplatzauflösung für unterstützte integrierte Steckplätze aktivieren, indem Sie zum Generativen AI-Bildschirm navigieren.

Wenn es sich bei dem Steckplatz um einen unterstützten integrierten Steckplatz handelt, haben Sie die Möglichkeit, die unterstützte Steckplatzauflösung auf Steckplatzebene zu aktivieren.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie im Navigationsbereich unter Bots den Bot aus, den Sie für die unterstützte Slot-Auflösung verwenden möchten.
3. Wählen Sie die Sprache Englisch (USA) für den Bot aus, den Sie aktivieren möchten.

4. Gehen Sie auf dem Bildschirm zum Abschnitt Generative AI-Konfiguration.
5. Wählen Sie Gehe zu Amazon Bedrock, um sich zu registrieren und die Funktion zu aktivieren, falls die Funktion nicht aktiviert wurde.

 Note

Wenn Sie keinen Zugriff auf Amazon Bedrock Foundation-Modelle haben, sollten Sie den Artikel Gehe zu Amazon Bedrock lesen. Klicken Sie auf Go to Amazon Bedrock, um zur Amazon Bedrock-Seite zu gelangen, auf der Sie sich für den Zugriff auf Foundation-Modelle registrieren können. Die unterstützte Slot-Auflösung unterstützt derzeit Claude V2 und Claude Instant V1. Für beste Ergebnisse empfehlen wir die Verwendung von Claude V2.

6. Wenn Sie bereits Zugriff auf Amazon Bedrock Foundation-Modelle haben, sollte die Schaltfläche Konfigurieren angezeigt werden. Klicken Sie auf diese Schaltfläche, um zur generativen KI-Konfigurationsseite zu gelangen und die generativen KI-Funktionen in Lex zu aktivieren.

Generative AI configurations [Info](#)

Improve Lex bot performance in this language with generative AI features powered by Amazon Bedrock.

Configure

Generative AI features have not been configured

Configure

7. Bewegen Sie den Schieberegler in der oberen rechten Ecke des Felds nach rechts, um die Einstellung Aktiviert auszuwählen.
8. Wählen Sie die Schaltfläche „Aktivieren“, um die unterstützte Slot-Auflösung für die ausgewählten Slots zu aktivieren.
9. Sie können die unterstützte Steckplatzauflösung deaktivieren, indem Sie die Steckplätze aus der Liste auswählen und auf Deaktivieren klicken.

Aktivieren Sie die unterstützte Steckplatzauflösung in den Steckplatzeinstellungen in Lex V2

Sie können die unterstützte Steckplatzauflösung für unterstützte integrierte Steckplätze aktivieren, indem Sie für jeden Intent, der Steckplätze hat, zur Slot-Ebene navigieren. Damit Sie die unterstützte Steckplatzauflösung aktivieren können, muss es sich bei den Steckplätzen um einen der oben

aufgeführten unterstützten integrierten Steckplätze handeln. Wenn der Steckplatz nicht über die Option zur Aktivierung der unterstützten Steckplatzauflösung verfügt, ist die Option ausgegraut.

 Note

Sie müssen zuerst die Funktion zur unterstützten Slot-Auflösung im Generative AI-Bedienfeld aktivieren, um die Funktion für einzelne Slots zu aktivieren.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/> Hause.
2. Wählen Sie im Navigationsbereich unter Bots den Bot aus, den Sie für die unterstützte Slot-Auflösung verwenden möchten.
3. Wählen Sie unter Alle Sprachen die Option Englisch (USA) aus, um die Liste zu erweitern.
4. Wählen Sie in der linken Seitenleiste Intents aus, um eine Liste der Absichten in dem von Ihnen ausgewählten Bot anzuzeigen.
5. Wählen Sie auf dem Bildschirm Intents den Intent aus, der die Slots enthält, die Sie ändern möchten.
6. Wählen Sie den Namen der Absicht aus, um die Slots für diese Absicht anzuzeigen.
7. Wählen Sie im Bereich Slots die Schaltfläche Erweiterte Optionen aus.
8. Aktivieren Sie das Kontrollkästchen Unterstützte Steckplatzauflösung aktivieren, um die Funktion zu aktivieren.

Slot: NumberOfPeople Info

Slot info Info

Slot name

NumberOfPeople

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Description - *optional*

Maximum 200 characters.

☒ Required for this intent

☐ Enable slot obfuscation: Store as {NumberOfPeople}

☐ ⚡ Enable assisted slot resolution - GenAI
The bot will use generative AI to further assist slot resolution. [Learn more](#)

Additional charges may be incurred based on the usage of generative AI features [Learn more](#)

- Wählen Sie in der unteren rechten Ecke des Bildschirms die Schaltfläche „Steckplatz aktualisieren“. Dadurch wird die unterstützte Slot-Auflösung für die von Ihnen ausgewählten Slots aktiviert.

Sie können die unterstützte Steckplatzauflösung für unterstützte integrierte Steckplätze aktivieren, indem Sie API-Aufrufe tätigen.

- Folgen Sie den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#), um die unterstützte Slot-Auflösung für Ihr Bot-Gebietsschema zu aktivieren.
- Senden Sie eine [UpdateSlot](#)Anfrage und geben Sie den Slot an, für den Sie die unterstützte Slot-Auflösung aktivieren möchten. Stellen Sie in dem `slotResolutionSetting` Feld den `slotResolutionStrategy` Wert als `enhancedFallback`. Um einen neuen Slot mit aktivierter unterstützter Slot-Auflösung zu erstellen, senden Sie stattdessen eine [CreateSlot](#)Anfrage.

In Lex V2 sind für die unterstützte Slot-Auflösung Berechtigungen erforderlich

- Um auf diese Funktion auf der Amazon Lex V2-Konsole zugreifen zu können, stellen Sie sicher, dass Ihre Konsolenrolle über die entsprechenden `bedrock:ListFoundationModels` Berechtigungen verfügt.
- Die dem Bot zugeordnete IAM-Rolle sollte über eine `bedrock:InvokeModel` entsprechende Berechtigung verfügen. Wenn Sie die Funktion mit der Konsole aktivieren, wird die Richtlinie automatisch zur Bot-Rolle hinzugefügt, sofern Ihr Bot eine dienstbezogene Rolle verwendet, die von generiert wurde.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:Region::foundation-model/modelId"
      ]
    }
  ]
}
```

Verwendung BedrockAgentIntent eines Amazon Bedrock Agents in Amazon Lex V2

Note

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).
2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Sie können Amazon Bedrock Agents nutzen, um komplexe Workloads zu bewältigen, die von Kunden angefordert werden, ohne einen umfassenden Aufgabendefinitionsprozess durchlaufen zu müssen. Amazon Lex V2 bietet eine integrierte `AMAZON.BedrockAgentIntent` Funktion, die Sie Ihrem Bot hinzufügen können. Diese Absicht nutzt die generativen KI-Funktionen von Amazon Bedrock, indem Kundenanfragen erkannt, analysiert, begründet und schließlich beantwortet werden. Es bietet auch die Möglichkeit, weitere Fragen zu stellen, um die erforderliche Aufgabe zu erledigen (stellen Sie sich beispielsweise vor, Sie haben einen Einzelhandelsvertreter definiert, der den Bestellstatus des Kunden überprüfen kann). Wenn der Kunde nach dem Status der Bestellung fragt, bittet `customerId` der Mitarbeiter zunächst `darumemailId`, die Details abzurufen, und antwortet schließlich mit dem korrekten Bestellstatus). Sie können sich auch dafür entscheiden, Ihr AMAZON-Konto zu integrieren. `BedrockAgentIntent` mit einer Bedrock Knowledge Base, um Kundenanfragen direkt zu beantworten.

Stellen Sie sicher, dass Ihre IAM-Rolle über die richtigen Berechtigungen für den Zugriff auf `AMAZON.BedrockAgentIntent` verfügt. BedrockAgentIntent indem Sie die Schritte unter Berechtigungen für den AMAZON befolgen.

Um den `AMAZONAS` zu nutzen. `BedrockAgentIntent` Sie müssen einen der folgenden Wissensspeicher eingerichtet haben.

- Amazon Bedrock Agents — Weitere Informationen finden Sie unter [Bedrock Agents erstellen](#).
- Amazon Bedrock Knowledge Base — Weitere Informationen finden Sie unter [Aufbau einer Wissensdatenbank](#).

Um den `AMAZON` zu verwenden. `BedrockAgentIntent`, stellen Sie sicher, dass Ihre IAM-Rolle über die richtigen Berechtigungen verfügt, indem Sie die Schritte unter Erforderliche [Berechtigungen in Lex V2 for Bedrock Agent](#) Intent befolgen.

Themen

- [Aktivieren Sie Bedrock Agent Intent im Generative AI-Konfigurationsbildschirm](#)
- [Aktivieren Sie Bedrock Agent Intent, indem Sie Ihrem Bot eine integrierte Absicht hinzufügen](#)
- [In Lex V2 für Bedrock Agent Intent benötigte Berechtigungen](#)
- [Beispielanfrage mit Sitzungsattributen](#)

Aktivieren Sie Bedrock Agent Intent im Generative AI-Konfigurationsbildschirm

Sie können Bedrock Agent Intent aktivieren, indem Sie zum Generative AI-Bildschirm navigieren.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie im Navigationsbereich unter Bots den Bot aus, den Sie für Bedrock Agent Intent verwenden möchten.
3. Wählen Sie die Sprache für den Bot aus, den Sie aktivieren möchten.
4. Gehen Sie auf dem Bildschirm zum Abschnitt Generative AI-Konfiguration und wählen Sie Konfigurieren aus.
5. Wählen Sie im Abschnitt BedrockAgentIntent Konfiguration die Option Create BedrockAgent Intent aus.

Aktivieren Sie Bedrock Agent Intent, indem Sie Ihrem Bot eine integrierte Absicht hinzufügen

Sie können Bedrock Agent Intent aktivieren, indem Sie Ihrem Amazon Lex V2-Bot eine integrierte Absicht hinzufügen.

Note

Sie müssen zuerst die Bedrock Agent Intent-Funktion im Generative AI-Panel aktivieren, um die Funktion für einzelne Bots zu aktivieren.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.

2. Wählen Sie im Navigationsbereich unter Bots den Bot aus, den Sie für Bedrock Agent Intent verwenden möchten.
3. Wählen Sie unter Alle Sprachen die Option Englisch (USA) aus, um die Liste zu erweitern.
4. Wählen Sie Absicht hinzufügen und wählen Sie im Dropdownmenü die Option Integrierte Absicht verwenden aus.
5. Weitere Informationen zu den Konfigurationen für den AMAZON.BedrockAgentIntent, siehe [AMAZON.BedrockAgentIntent](#).

In Lex V2 für Bedrock Agent Intent benötigte Berechtigungen

- Um auf diese Funktion auf der Amazon Lex V2-Konsole zugreifen zu können, stellen Sie sicher, dass Ihre Konsolenrolle über die entsprechenden `bedrock:ListFoundationModels` Berechtigungen verfügt.
- Die dem Bot zugeordnete IAM-Rolle sollte über die erforderlichen Berechtigungen verfügen. `AMAZON.BedrockAgentIntent`

Die Bot-Rolle sollte über Anrufberechtigungen verfügen, `bedrock:InvokeAgent` wenn sie Amazon Bedrock Agents verwendet `bedrock:InvokeModel` und `bedrock:RetrieveAndGenerate` wenn Amazon Bedrock Knowledge Bases in der Absicht verwendet wird. Sie sollten auch eine Erklärung für jeden der Agenten beifügen, die Sie in Ihren Bots angeben `AMAZON.BedrockAgentIntent` (siehe die Erklärung Berechtigungen für den Zugriff auf Amazon Bedrock Agent oder Berechtigungen für den Zugriff auf Amazon Bedrock Knowledge Bases in der Richtlinie unten).

Wenn Sie die Funktion mit der Amazon Lex-Konsole aktivieren, wird die Richtlinie automatisch zur Bot-Rolle hinzugefügt, sofern Ihr Bot eine von Amazon Lex generierte serviceverknüpfte Rolle verwendet.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Permissions to access Amazon Bedrock Agent",
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeAgent"
      ],
      "Resource": [
```



```

        "arn:aws:bedrock:region:accountId:agent-alias/agentId/aliasId"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "accountId"
        }
    }
},
{
    "Sid": "Permissions to access Knowledge Base in Amazon Bedrock",
    "Effect": "Allow",
    "Action": [
        "bedrock:RetrieveAndGenerate"
    ],
    "Resource": [
        "arn:aws:bedrock:region:accountId:knowledge-base/knowledgeBaseId"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "accountId"
        }
    }
},
{
    "Sid": "Permissions to invoke Amazon Bedrock foundation models",
    "Effect": "Allow",
    "Action": [
        "bedrock:InvokeModel"
    ],
    "Resource": [
        "arn:aws:bedrock:region::foundation-model/modelId"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "accountId"
        }
    }
}
]
}

```

Beispielanfrage mit Sitzungsattributen

Das folgende Beispiel zeigt, wie AMAZON aufgerufen wird. BedrockAgentIntent und demonstriert die Sitzungs- und Anforderungsattribute, die in der Antwort angegeben sind. Diese Attribute enthalten die Antwortdaten des Bedrock Agents und können verwendet werden, um auf die Ausgabe des Agenten, auf die Amazon Bedrock Knowledge Base-Quellen und auf Details zum Aufruf von Aktionsgruppen zuzugreifen.

```
{
  "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}"
  }
}

```

In diesem Beispiel zeigen die Sitzungsattribute, wie die Antwortdaten `BedrockAgentIntent` aufgefüllt werden, einschließlich der Antwort des Agenten, der verwendeten Amazon Bedrock Knowledge Base-Quellen, Aktionsgruppeneufrufen und der Amazon Bedrock Knowledge Base-Suchdetails, die zur Generierung der Antwort verwendet wurden.

[Weitere Informationen finden Sie unter AMAZON. BedrockAgentIntent.](#)

Verbessern Sie die Absichtsklassifizierung und die Slot-Auflösung in Lex V2 mit unterstützter NLU

Assisted NLU ist eine Funktion, die Large Language Models (LLMs) verwendet, um die Funktionen zur Absichtsklassifizierung und Slot-Auflösung von Amazon Lex V2 zu verbessern. Es verbessert die Genauigkeit und hält gleichzeitig die von Ihrem Bot konfigurierten Absichten und Slots ein. Die Funktion generiert oder modifiziert keine Bot-Inhalte. Diese Funktion trägt dazu bei, die Gesamtgenauigkeit des NLU-Systems zu verbessern, was zu einem reibungsloseren und effektiveren Konversationserlebnis für Benutzer führt.

Die unterstützte NLU-Funktion ist in englischer und spanischer Sprache verfügbar (die Sprachumgebungen, die mit `en_` und `beginnen`, finden Sie `es_` in der Tabelle unter). [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#)

Verwenden Sie unterstütztes NLU, um die Klassifizierung von Absichten und die Auflösung von Zeitfenstern zu verbessern. Amazon Lex V2 verwendet Amazon Bedrock-Modelle, um Absichten zu klassifizieren und Slot-Typen zu lösen, die für den Anwendungsfall Ihres Bots geeignet sind. Sie können Assisted NLU für Ihren Bot mit der Konsole aktivieren.

Console

Verwenden von Assisted NLU mit Ihrem Amazon Lex V2-Bot

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie auf der Seite Bots den Bot aus, den Sie mit unterstützter NLU verwenden möchten.
3. Klicken Sie auf der Seite Bot Locale im Abschnitt Assisted NLU auf Configure.
4. Im Abschnitt Generative AI-Funktionen zur Runtime findest du die Funktion Assisted NLU. Verwenden Sie die Umschaltfläche daneben, um die LLM Assisted NLU-Funktion zu aktivieren, und klicken Sie auf Speichern.
5. Vergewissern Sie sich, dass die Funktion LLM Assisted NLU im Abschnitt Assisted NLU auf der Seite Bot Locale aktiviert ist.
6. Erstellen Sie den Bot, um zu sehen, ob sich die Änderungen in Runtime in Ihrem Bot widerspiegeln.
7. Sobald der Bot-Build abgeschlossen ist, können Sie das Testfenster in der Konsole verwenden oder ein Testset ausführen, um sich die Verbesserungen anzusehen, nachdem Sie die LLM Assisted NLU-Funktion aktiviert haben.

Anleitung zur Verbesserung der Genauigkeit Ihres Bots bei der Verwendung der LLM Assisted NLU-Funktion

Die folgenden bewährten Methoden können Ihnen dabei helfen, die Effektivität der Assisted NLU-Funktion zu maximieren:

1. Machen Sie die Namen der Absicht selbsterklärend — Verwenden Sie Namen, die unmittelbar die Handlung oder den Zweck der Absicht vermitteln. Wenn Sie beispielsweise eine Absicht für die Buchung von Flügen erstellen, nennen Sie sie einfach "BookFlight".
2. Halte deine Namen klar und einfach — Vermeide es, Präfixe, Suffixe oder unnötige Wörter zu deiner Absicht und deinen Slot-Namen hinzuzufügen. Zusätzliche Elemente wie „Dev“ oder „Test“ können das LLM verwirren und den Zweck weniger klar machen.

3. Geben Sie detaillierte Beschreibungen an — Fügen Sie für jede benutzerdefinierte Absicht und jeden Slot eine kurze, aber informative Beschreibung bei. Dies hilft, die spezifische Verwendung und den Kontext zu erklären, sodass es sowohl für Menschen als auch für LLM einfacher ist, ihren Zweck zu verstehen.

Note

Wenn Sie diese Funktion aktivieren, werden Ihre Daten möglicherweise regionsübergreifend AWS verarbeitet. Weitere Informationen zur regionsübergreifenden Inferenz finden Sie unter <https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html>

Important

Aktivieren Sie diese Funktion in einer Entwurfsversion des Bots. Testen Sie es, bevor Sie es in einem Produktionsalias verwenden.

Deaktivieren von Assisted NLU

Gehen Sie wie folgt vor, um die Assisted NLU-Funktion zu deaktivieren:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie auf der Seite Bots Ihren Bot aus.
3. Klicken Sie auf der Seite Bot Locale im Abschnitt Assisted NLU auf Configure.
4. Schalten Sie im Bereich Generative AI-Funktionen zur Laufzeit die Funktion Assisted NLU aus und klicken Sie auf Speichern.
5. Erstellen Sie den Bot, um die Änderungen zu übernehmen.

Amazon.qn Alintent

Note

Bevor Sie die generativen KI-Funktionen nutzen können, müssen Sie die folgenden Voraussetzungen erfüllen

1. Navigieren Sie zur [Amazon Bedrock-Konsole](#) und registrieren Sie sich für den Zugriff auf das Anthropic Claude-Modell, das Sie verwenden möchten (weitere Informationen finden Sie unter [Modellzugriff](#)). Informationen zu den Preisen für die Nutzung von Amazon Bedrock finden Sie unter [Amazon Bedrock — Preise](#).
2. Schalten Sie die generativen KI-Funktionen für Ihr Bot-Gebietsschema ein. Folgen Sie dazu den Schritten unter [Optimieren Sie die Erstellung und Leistung von Lex V2-Bots mithilfe generativer KI](#).

Sie können Amazon Bedrock nutzen, FMs um Kundenfragen in einer Bot-Konversation zu beantworten. Amazon Lex V2 bietet eine integrierte AMAZON.QnAIntent Funktion, die Sie Ihrem Bot hinzufügen können. Diese Absicht nutzt die generativen KI-Funktionen von Amazon Bedrock, indem Kundenfragen erkannt und in den folgenden Wissensspeichern nach einer Antwort gesucht wird (z. B. **Can you provide me details on the baggage limits for my international flight?**). Diese Funktion reduziert die Notwendigkeit, Fragen und Antworten mithilfe eines aufgabenorientierten Dialogs innerhalb von Amazon Lex V2 Intents zu konfigurieren. Dieser Intent erkennt auch Folgefragen (z. B. **What about domestic flight?**) auf der Grundlage des Konversationsverlaufs und gibt die Antwort entsprechend.

Stellen Sie sicher, dass Ihre IAM-Rolle über die entsprechenden Zugriffsberechtigungen verfügt, AMAZON.QnAIntent indem Sie die Schritte unter [Berechtigungen für AMAZON.QnAIntent](#) befolgen.

Um die Vorteile nutzen zu können, müssen AMAZON.QnAIntent Sie einen der folgenden Wissenspeicher eingerichtet haben.

- Amazon OpenSearch Service-Datenbank — Weitere Informationen finden Sie unter [Amazon OpenSearch Service-Domains erstellen und verwalten](#).
- Amazon Kendra Kendra-Index — Weitere Informationen finden Sie unter [Index erstellen](#).
- Amazon Bedrock-Wissensdatenbank — Weitere Informationen finden Sie unter [Aufbau einer Wissensdatenbank](#).

Sie können die AMAZON.QnAIntent auf zwei Arten einrichten:

Zur Einrichtung mithilfe generativer KI-Konfigurationen

1. Wählen Sie in der Amazon Lex V2-Konsole im linken Navigationsbereich Bots aus und wählen Sie im Abschnitt Bots den Bot aus, für den Sie die Absicht hinzufügen möchten.

2. Wählen Sie im linken Navigationsbereich die Sprache aus, für die Sie die Absicht hinzufügen möchten.
3. Wählen Sie im Abschnitt Generative KI-Konfigurationen die Option Konfigurieren aus.
4. Wählen Sie im Abschnitt QnA-Konfigurationen die Option QnA-Absicht erstellen aus.

Zur Einrichtung fügen Sie Ihrem Bot eine integrierte Absicht hinzu

1. Wählen Sie in der Amazon Lex V2-Konsole im linken Navigationsbereich Bots aus und wählen Sie im Abschnitt Bots den Bot aus, für den Sie die Absicht hinzufügen möchten.
2. Wählen Sie im linken Navigationsbereich unter der Sprache, für die Sie die Absicht hinzufügen möchten, die Option Intents aus.
3. Wählen Sie Absicht hinzufügen aus und wählen Sie im Dropdownmenü die Option Integrierte Absicht verwenden aus.
4. Weitere Informationen zu den Konfigurationen für finden Sie `AMAZON.QnAIntent` unter [AMAZON.QnAIntent](#).

Note

Der `AMAZON.QnAIntent` wird aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet ist. Diese Absicht wird aktiviert, wenn eine Äußerung keiner der anderen im Bot vorhandenen Absichten zugeordnet ist. Beachten Sie, dass diese Absicht nicht für verpasste Äußerungen aktiviert wird, wenn ein Slot-Wert abgerufen wird. Nach der Erkennung `AMAZON.QnAIntent` verwendet der das angegebene Amazon Bedrock-Modell, um die konfigurierte Wissensdatenbank zu durchsuchen und die Kundenfrage zu beantworten.

Themen

- [Berechtigungen für AMAZON.QnAIntent](#)

Berechtigungen für AMAZON.QnAIntent

Um auf diese Funktion auf der Amazon Lex V2-Konsole zugreifen zu können, stellen Sie sicher, dass Ihre Konsolenrolle über `bedrock:ListFoundationModels` Berechtigungen verfügt.

Die dem Bot zugeordnete IAM-Rolle sollte über die folgenden erforderlichen Berechtigungen verfügen. AMAZON.QnAIntent Die Bot-Rolle sollte über `bedrock:InvokeModel` Anrufberechtigungen verfügen. Sie sollten auch für jeden Datenspeicher, den Sie in Ihren Bots angeben, eine Erklärung beifügen AMAZON.QnAIntent (siehe die `Permissions to access knowledge base in Amazon Bedrock` Anweisungen `Permissions to access Amazon Kendra index` `Permissions to access OpenSearch Service index`, und in der Richtlinie unten). Wenn Sie die Funktion mit der Amazon Lex-Konsole aktivieren, werden die Richtlinien automatisch zur Bot-Rolle hinzugefügt, sofern Ihr Bot eine von Amazon Lex generierte serviceverknüpfte Rolle verwendet.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Permissions to invoke Amazon Bedrock foundation models",
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:region::foundation-model/model-id"
      ]
    },
    {
      "Sid": "Permissions to access Amazon Kendra index",
      "Effect": "Allow",
      "Action": [
        "kendra:Query",
        "kendra:Retrieve"
      ],
      "Resource": [
        "arn:aws:kendra:region:account-id:index/kendra-index"
      ]
    },
    {
      "Sid": "Permissions to access OpenSearch Service index",
      "Effect": "Allow",
      "Action": [
        "es:ESHttpGet",
```



```
        "es:ESHttpPost"
      ],
      "Resource": [
        "arn:aws:es:region:account-id:domain/domain-name/index-name/_search"
      ]
    },
    {
      "Sid": "Permissions to access knowledge base in Amazon Bedrock",
      "Effect": "Allow",
      "Action": [
        "bedrock:Retrieve"
      ],
      "Resource": [
        "arn:aws:bedrock:region:account-id:knowledge-base/knowledge-base"
      ]
    }
  ]
}
```

Erstellen Sie ein Netzwerk von Bots für Ihre Lex V2-Bots

Network of Bots ermöglicht es Unternehmen, ein einheitliches Benutzererlebnis für mehrere Bots bereitzustellen. Mit Network of Bots können Unternehmen mehrere Bots zu einem einzigen Netzwerk hinzufügen, um ein flexibles und unabhängiges Bot-Lebenszyklusmanagement zu ermöglichen. Das Netzwerk stellt dem Endbenutzer eine einzige einheitliche Oberfläche zur Verfügung und leitet die Anfrage auf der Grundlage von Benutzereingaben an den entsprechenden Bot weiter.

Teams können zusammenarbeiten, um ein Netzwerk von Bots aufzubauen, das unterschiedlichen Geschäftsanforderungen gerecht wird, indem sie Bots verwalten und dem Netzwerk hinzufügen, sobald verbesserte Bots in der Produktion eingesetzt werden. Entwickler können die Bereitstellung und Verbesserung vereinfachen und beschleunigen, indem sie mehrere Bots in ein einziges Netzwerk integrieren.

Network of Bots ist derzeit nur in der Sprache en-US verfügbar.

Note

Derzeit ist ein Netzwerk von Bots auf ein Konto beschränkt. Sie können keine Bots von anderen Konten hinzufügen.

The screenshot displays the Amazon Lex V2 console interface for managing a 'Network of bots'. The main heading is 'BankingBots'. Below it, there are tabs for 'Details' and 'Info'. The 'Details' tab is active, showing a table with the following information:

Name	Language	Description	Last edited
BankingBots	English (US)	Newly created network of bots.	1 minute ago

Below the details section, there is a 'Bots (4)' section with a search bar and a table listing the bots in the network:

Name	Status	Alias	Associated version	Description
CreditCardBot	Available	Prod	Version 2	-
ServiceBot	Available	Prod	Version 3	-
DebitCardBot	Available	Beta	Version 3	-
LoanBot	Available	Prod	Version 1	Description text

On the left side of the console, there is a sidebar with navigation links: 'Bots', 'Network of bots', 'BankingBots', 'Versions', 'Deployment', 'Aliases', 'Channel integrations', 'Related resources', and 'Return to the V1 console'. At the top right of the main content area, there are buttons for 'Draft version', 'Ready', 'Build', and 'Test'. Below the 'Details' tab, there are buttons for 'Delete' and 'Edit'.

Erstellen Sie ein Netzwerk von Bots für Ihre Lex V2-Bots

Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>. Wählen Sie im Seitenmenü Network of Bots aus. Sie müssen mindestens einen Bot erstellt haben, um ein Netzwerk von Bots zu erstellen.

Schritt 1: Konfigurieren Sie die Einstellungen für das Bot-Netzwerk

1. Geben Sie im Abschnitt Details den Namen Ihres Netzwerks ein und geben Sie ihm optional eine Beschreibung.
2. Wählen Sie im Abschnitt IAM-Berechtigungen eine AWS Identity and Access Management (IAM-) Rolle aus, die Amazon Lex V2-Berechtigungen für den Zugriff auf andere AWS Dienste wie Amazon gewährt. CloudWatch Sie können Amazon Lex V2 die Rolle erstellen lassen, oder Sie können eine vorhandene Rolle mit CloudWatch Berechtigungen auswählen. Weitere Informationen finden Sie unter [Identitäts- und Zugriffsmanagement für Amazon Lex V2](#).
3. Wählen Sie im Abschnitt Gesetz zum Schutz der Privatsphäre von Kindern im Internet (COPPA) die entsprechende Antwort aus. Weitere Informationen finden Sie unter [DataPrivacy](#).
4. Wählen Sie im Abschnitt Timeout für Leerlaufsitzen die Dauer aus, für die Amazon Lex V2 eine Sitzung mit einem Benutzer geöffnet hält. Amazon Lex V2 verwaltet Sitzungsvariablen für die Dauer der Sitzung, sodass Ihr Bot eine Konversation mit denselben Variablen wieder aufnehmen kann. Weitere Informationen finden Sie unter [Einstellung des Sitzungs-Timeouts](#).
5. Wählen Sie im Abschnitt Spracheinstellungen hinzufügen eine Stimme aus, mit der Ihr Bot mit Benutzern interagieren soll. Sie können eine Phrase in das Sprachbeispiel eingeben und Abspielen auswählen, um sich die Stimme anzuhören.
6. Fügen Sie im Bereich Erweiterte Einstellungen optional Tags hinzu, mit deren Hilfe der Bot identifiziert werden kann. Tags können verwendet werden, um den Zugriff zu kontrollieren und Ressourcen zu überwachen. Weitere Informationen [finden Sie unter Ressourcen](#) kennzeichnen.
7. Wählen Sie Weiter, um das Bot-Netzwerk zu erstellen, und fahren Sie mit dem Hinzufügen von Bots fort.

Schritt 2: Bots hinzufügen

1. Wählen Sie im Bereich Bots die Option + Bots hinzufügen aus.
2. Ein Modal zum Hinzufügen von Bots wird angezeigt. Wählen Sie im Dropdownmenü Bot einen Bot aus, den Sie hinzufügen möchten, und wählen Sie im Dropdownmenü Alias den Alias des Bots aus, den Sie verwenden möchten.

Der Alias muss auf eine nummerierte Version des Bots verweisen und nicht auf die Entwurfsversion. Sie können bis zu 5 Bots hinzufügen. Ein Bot kann zu bis zu 25 verschiedenen Netzwerken hinzugefügt werden.

3. Wählen Sie + Bot hinzufügen, um Ihrem Netzwerk weitere Bots hinzuzufügen. Um einen Bot zu entfernen, wählen Sie neben dem Bot, den Sie entfernen möchten, die Option Entfernen aus. Wenn Sie mit dem Hinzufügen von Bots fertig sind, wählen Sie Speichern, um das Modal zu schließen.
4. Wählen Sie Speichern, um die Erstellung Ihres Netzwerks abzuschließen.

Verwalte dein Netzwerk von Bots für deine Lex V2 Bots

Nachdem Sie Ihr Netzwerk von Bots erstellt haben, werden Sie zu einer Seite weitergeleitet, auf der Sie Ihr Netzwerk verwalten und aufbauen können. Sie können diese Seite auch erreichen, indem Sie im Seitenmenü Netzwerk von Bots auswählen und den Namen des zu verwaltenden Netzwerks auswählen.

1. Um Informationen für Ihr Netzwerk zu bearbeiten, wählen Sie über dem Abschnitt Details die Option Bearbeiten aus. Um das Netzwerk zu löschen, wählen Sie über dem Abschnitt Details die Option Löschen aus.
2. Im Bereich Bots können Sie weitere Bots hinzufügen, indem Sie + Bots hinzufügen auswählen. Sie können Bots auch hinzufügen, wenn Sie im Seitenmenü der Amazon Lex V2-Konsole zur Seite Bots navigieren. Aktivieren Sie das Optionsfeld neben dem Bot, den Sie hinzufügen möchten, und wählen Sie im Dropdownmenü Aktionen die Option Zu einem Netzwerk von Bots hinzufügen aus.

Wählen Sie im Dropdownmenü Netzwerk der Bots in dem sich öffnenden Modal das Netzwerk aus, zu dem Sie den Bot hinzufügen möchten. Wählen Sie dann im Drop-down-Menü Bot-Alias den Alias des Bots aus, den Sie verwenden möchten. Wählen Sie Hinzufügen aus, um den Bot dem ausgewählten Netzwerk hinzuzufügen.

3. Sie können Bots aus Ihrem Netzwerk entfernen, indem Sie das Optionsfeld neben einem Bot aktivieren und Entfernen auswählen.
4. Wenn Sie mit der Konfiguration Ihres Netzwerks fertig sind, wählen Sie oben rechts Build aus, um Ihr Netzwerk aufzubauen. Der Aufbau kann einige Minuten dauern. Wenn der Build erfolgreich ist, wird oben auf der Seite ein grünes Erfolgsbanner angezeigt.

5. Sobald das Netzwerk aufgebaut ist, können Sie oben rechts „Testen“ auswählen, sodass in der unteren rechten Ecke ein Chat-Fenster angezeigt wird. Sie können dieses Chat-Fenster verwenden, um mit den Bots Ihres Netzwerks zu kommunizieren und sicherzustellen, dass die Konversationsabläufe und -übergänge korrekt konfiguriert sind.

Note

Wenn Sie Bots in Ihrem Netzwerk hinzufügen, entfernen oder aktualisieren, müssen Sie das Netzwerk neu aufbauen.

Versionen Ihres Bot-Netzwerks für Lex V2

Sie können verschiedene Versionen Ihres Bot-Netzwerks erstellen. Um Versionen zu verwalten, wählen Sie Ihr Netzwerk im Seitenmenü der Amazon Lex V2-Konsole aus und wählen Sie Versionen aus.

1. Wählen Sie Version erstellen, um eine neue Version Ihres Bot-Netzwerks zu erstellen. Sie können eine optionale Beschreibung hinzufügen. Wählen Sie Erstellen, um die Version zu erstellen.
2. Wenn Sie das Optionsfeld neben einer Version Ihres Bot-Netzwerks aktivieren, können Sie Alias mit Version verknüpfen auswählen, um einen Alias auf diese Version zu verweisen.
3. Um eine Version Ihres Netzwerks zu verwalten, wählen Sie den Namen der Version im Abschnitt Versionen aus. Auf der folgenden Seite können Sie die Details der Version bearbeiten und die Bots innerhalb der Version und den zugehörigen Alias verwalten.

Aliase für Ihr Netzwerk von Bots für Lex V2

Sie können Aliase verwenden, um Ihre Netzwerke bereitzustellen. Um Aliase zu verwalten, wählen Sie Ihr Netzwerk im Seitenmenü der Amazon Lex V2-Konsole aus und wählen Sie Aliase aus.

1. Wählen Sie Alias erstellen aus, um einen neuen Alias zu erstellen.
2. Geben Sie dem Alias im Abschnitt Aliasdetails einen Namen und optional eine Beschreibung. Sie können im Abschnitt Mit einer Version verknüpfen eine Version auswählen, um den Alias zu verknüpfen, und im Abschnitt Tags Tags hinzufügen. Wählen Sie Erstellen, um den Alias zu erstellen.

3. Um einen Alias für Ihr Netzwerk zu verwalten, wählen Sie den Namen des Alias im Abschnitt Aliase aus. Auf der folgenden Seite können Sie die Details des Alias bearbeiten und seine Tags, Kanalintegrationen und ressourcenbasierte Richtlinien verwalten. Sie können sich auch den Verlauf der Verknüpfung mit Versionen des Netzwerks ansehen.

Kanalintegrationen für Ihr Lex V2-Bot-Netzwerk

Um Ihr Bot-Netzwerk in eine Messaging-Plattform zu integrieren, wählen Sie Ihr Bot-Netzwerk aus dem Seitenmenü der Amazon Lex V2-Konsole aus. Wählen Sie dann Kanalintegrationen aus.

1. Wählen Sie Kanal hinzufügen aus, um Ihr Netzwerk in einen neuen Kanal zu integrieren.
2. Wählen Sie im Bereich Plattform auswählen die Plattform aus, auf der Sie Ihren Bot bereitstellen möchten. Eine IAM-Rolle wird erstellt. Wählen Sie im Drop-down-Menü unter KMS-Schlüssel einen Schlüssel aus, um Ihre Daten zu schützen.
3. Geben Sie im Integrationskonfigurationskanal den Namen und optional eine Beschreibung ein. Wählen Sie einen Alias aus dem Drop-down-Menü aus.
4. Holen Sie sich Ihre Konto-SID und Ihr Authentifizierungstoken von der Plattform und füllen Sie die Felder Konto-SID und Authentifizierungstoken aus. Weitere Informationen finden Sie unter [Integrieren Ihrer Bots](#).
5. Wählen Sie Erstellen aus, um die Kanalintegration abzuschließen.

Note

Network of Bots ist derzeit nicht in Amazon Connect Voice oder Chat verfügbar.

Bereitstellung von Bots von Lex V2 für Ihre Produktionsumgebung

Nachdem Sie Ihren Bot erstellt und getestet haben, kann er für die Interaktion mit Ihren Kunden eingesetzt werden. In diesem Abschnitt erfahren Sie, wie Sie Versionen Ihres Bots erstellen, wenn Sie ein Update vorgenommen haben. Verwenden Sie Aliase, um auf verschiedene Versionen Ihres Bots zu verweisen, wenn diese für die Bereitstellung bereit sind. Erfahren Sie, wie Sie Ihre Bots in Messaging-Plattformen, mobile Anwendungen und Websites integrieren können.

Themen

- [Versionierung und Aliase mit Ihrem Lex V2-Bot](#)
- [Verwenden einer Java-Anwendung zur Interaktion mit einem Amazon Lex V2-Bot](#)
- [Verwenden Sie Global Resiliency, um Bots in anderen Regionen einzusetzen](#)
- [Integration eines Amazon Lex V2-Bots in eine Messaging-Plattform](#)
- [Integration eines Amazon Lex V2-Bots in ein Kontaktzentrum](#)

Versionierung und Aliase mit Ihrem Lex V2-Bot

Amazon Lex V2 unterstützt die Erstellung von Versionen und Aliasnamen von Bots und Bot-Netzwerken, sodass Sie die Implementierung kontrollieren können, die Ihre Client-Anwendungen verwenden. Eine Version dient als nummerierte Momentaufnahme Ihrer Arbeit. Sie können einen Alias auf die Version Ihres Bots verweisen, die Ihren Kunden zur Verfügung stehen soll. Zwischen der Erstellung von Versionen können Sie die Draft Version Ihres Bots weiter aktualisieren, ohne die Benutzererfahrung zu beeinträchtigen.

Versionen

Amazon Lex V2 unterstützt die Erstellung von Versionen von Bots, sodass Sie die Implementierung kontrollieren können, die Ihre Client-Anwendungen verwenden. Eine Version ist eine nummerierte Momentaufnahme Ihrer Arbeit, die Sie für die Verwendung in verschiedenen Teilen Ihres Workflows erstellen können, z. B. in der Entwicklung, Betabereitstellung und Produktion.

Die Entwurfsversion Ihres Lex V2-Bots

Wenn Sie einen Amazon Lex V2-Bot erstellen, gibt es nur eine Version, die Draft Version.

Draft ist die Arbeitskopie Ihres Bots. Sie können nur die **Draft** Version aktualisieren und bis Sie Ihre erste Version erstellt haben, **Draft** ist die einzige Version des Bots, die Sie haben.

Die **Draft** Version Ihres Bots ist mit dem verknüpft **TestBotAlias**. Das **TestBotAlias** sollte nur für manuelle Tests verwendet werden. Amazon Lex V2 begrenzt die Anzahl der Runtime-Anfragen, die Sie an den **TestBotAlias** Alias des Bots stellen können.

Eine Version für Ihren Lex V2-Bot erstellen

Wenn Sie einen Amazon Lex V2-Bot versionieren, erstellen Sie einen nummerierten Snapshot des Bots, sodass Sie den Bot so verwenden können, wie er zum Zeitpunkt der Erstellung der Version vorhanden war. Sobald Sie eine numerische Version erstellt haben, bleibt diese unverändert, während Sie weiter an der Entwurfsversion Ihrer Anwendung arbeiten.

Wenn Sie eine Version erstellen, können Sie die Gebietsschemas auswählen, die in die Version aufgenommen werden sollen. Sie müssen nicht alle Gebietsschemas in einem Bot auswählen. Wenn Sie eine Version erstellen, können Sie auch ein Gebietsschema aus einer früheren Version auswählen. Wenn Sie beispielsweise drei Versionen eines Bots haben, können Sie bei der Erstellung von Version vier ein Gebietsschema aus der **Draft** Version und eines aus Version zwei auswählen.

Wenn Sie ein Gebietsschema aus der **Draft** Version löschen, wird es nicht aus einer nummerierten Version gelöscht.

Wenn eine Bot-Version sechs Monate lang nicht verwendet wird, markiert Amazon Lex V2 die Version als inaktiv. Wenn eine Version inaktiv ist, können Sie keine Runtime-Operationen mit dem Bot verwenden. Um den Bot aktiv zu machen, müssen Sie alle mit der Version verknüpften Sprachen neu erstellen.

Aktualisierung eines Amazon Lex V2-Bots

Sie können nur die **Draft** Version eines Amazon Lex V2-Bots aktualisieren. Versionen können nicht geändert werden. Sie können jederzeit eine neue Version erstellen, nachdem Sie eine Ressource in der Konsole oder während des [CreateBotVersion](#) Vorgangs aktualisiert haben.

Löschen eines Amazon Lex V2-Bots oder einer Amazon Lex V2-Version

Amazon Lex V2 unterstützt das Löschen eines Bots oder einer Version mithilfe der Konsole oder einer der API-Operationen:

- [DeleteBot](#)

- [DeleteBotVersion](#)

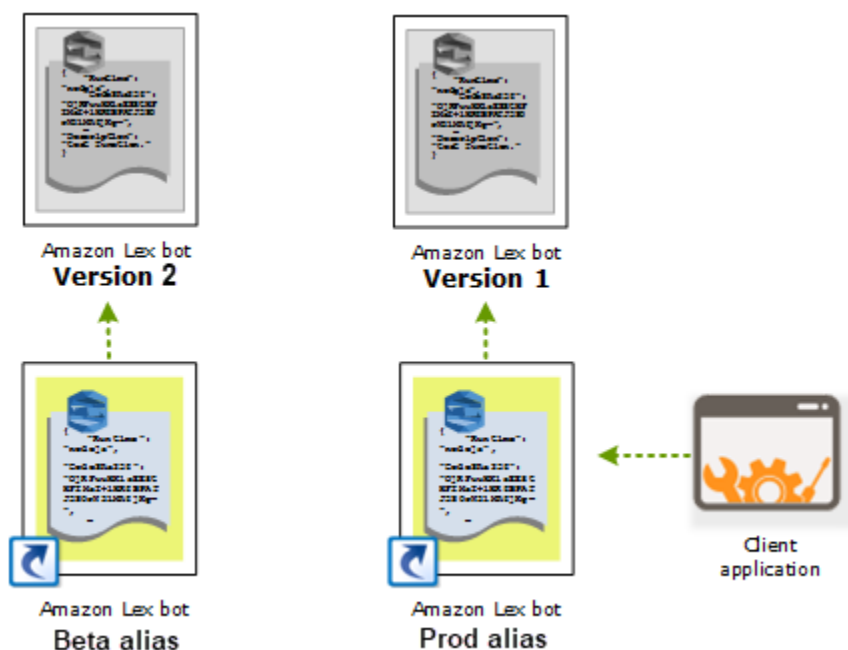
Aliase für deinen Lex V2-Bot

Amazon Lex V2-Bots unterstützen Aliase. Ein Alias ist ein Zeiger auf eine bestimmte Bot-Version. Mit einem Alias können Sie einfach die Version aktualisieren, die Ihre Clientanwendungen verwenden. Beispielsweise können Sie einen Alias auf Version 1 Ihres Bot zeigen lassen. Wenn Sie bereit sind, den Bot zu aktualisieren, erstellen Sie Version 2 und ändern den Alias so, dass er auf die neue Version verweist. Da Ihre Anwendungen den Alias anstelle einer bestimmten Version verwenden, erhalten alle Ihre Clients die neuen Funktionen, ohne dafür aktualisiert werden zu müssen.

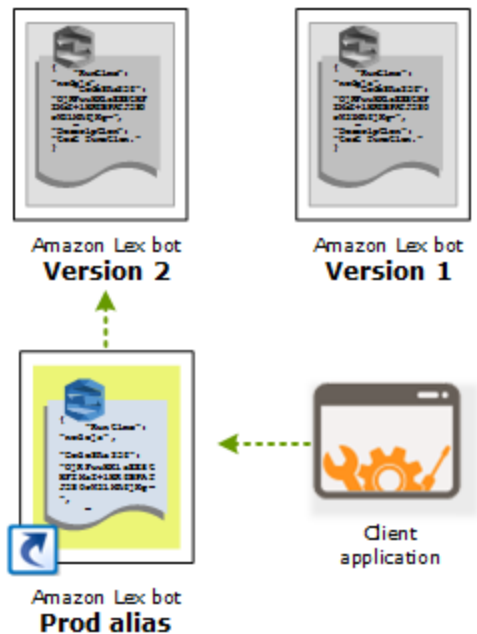
Ein Alias ist ein Verweis auf eine bestimmte Version eines Amazon Lex V2-Bots. Verwenden Sie einen Alias, um Clientanwendungen zu erlauben, eine bestimmte Version des Bots zu verwenden, ohne dass die Anwendung nachverfolgen muss, um welche Version es sich handelt.

Wenn Sie einen Bot erstellen, erstellt Amazon Lex V2 einen Alias namens `TestBotAlias`, den Sie zum Testen Ihres Bots verwenden können. Der `TestBotAlias` Alias ist immer mit der Draft Version Ihres Bots verknüpft. Sie sollten den `TestBotAlias` Alias nur zu Testzwecken verwenden. Amazon Lex V2 begrenzt die Anzahl der Laufzeitanforderungen, die Sie an den Alias stellen können.

Das folgende Beispiel zeigt zwei Versionen eines Amazon Lex V2-Bots, Version 1 und Version 2. Jeder dieser Bot-Versionen ist ein Alias zugeordnet, BETA bzw. PROD. Clientanwendungen greifen auf den Bot mithilfe des PROD-Alias zu.



Wenn Sie eine zweite Version des Bots erstellen, können Sie den Alias mit der Konsole oder der [UpdateBotAlias](#)-Operation so aktualisieren, dass er auf die neue Version des Bots zeigt. Wenn Sie den Alias ändern, verwenden alle Ihre Clientanwendungen die neue Version. Wenn es mit der neuen Version ein Problem gibt, können Sie einfach zu der vorhergehenden Version zurückkehren, indem Sie den Alias so ändern, dass er auf diese Version zeigt.



Wenn Sie Ihre Client-Anwendungen so einrichten, dass sie [Amazon Lex Runtime V2](#) aufrufen, APIs damit Kunden mit Ihrem Bot interagieren können, verwenden Sie den Alias, der auf die Version verweist, die Ihre Kunden verwenden sollen.

Note

Sie können zwar die Draft Version eines Bots in der Konsole testen, wir empfehlen jedoch, dass Sie, wenn Sie einen Bot in Ihre Client-Anwendung integrieren, zunächst eine Version und einen Alias erstellen, der auf diese Version verweist. Verwenden Sie den Alias in Ihrer Clientanwendung aus den Gründen, die in diesem Abschnitt erklärt werden. Wenn Sie einen Alias aktualisieren, verwendet Amazon Lex V2 die aktuelle Version für alle laufenden Sitzungen. Neue Sitzungen verwenden die neue Version.

Verwenden einer Java-Anwendung zur Interaktion mit einem Amazon Lex V2-Bot

[AWS SDK für Java 2.0](#) bietet eine Schnittstelle, die Sie von Ihren Java-Anwendungen aus verwenden können, um mit Ihren Bots zu interagieren. Verwenden Sie das SDK for Java, um Client-Anwendungen für Benutzer zu erstellen.

Die folgende Anwendung interagiert mit dem OrderFlowers Bot, in [Übung 1: Einen Bot anhand eines Beispiels erstellen](#) dem Sie erstellt haben. Es verwendet das SDK `LexRuntimeV2Client` aus dem SDK for Java, um die `RecognizeText` Operation aufzurufen, um eine Konversation mit dem Bot zu führen.

Die Ausgabe der Konversation sieht wie folgt aus:

```
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.
```

Informationen zu den JSON-Strukturen, die zwischen der Client-Anwendung und dem Amazon Lex V2-Bot gesendet werden, finden Sie unter [Übung 2: Überprüfen Sie den Gesprächsablauf](#).

Um die Anwendung auszuführen, müssen Sie die folgenden Informationen angeben:

- `BotID` — Die Kennung, die dem Bot zugewiesen wurde, als Sie ihn erstellt haben. Sie können die Bot-ID in der Amazon Lex V2-Konsole auf der Seite mit den Bot-Einstellungen sehen.
- `botAliasId` — Die Kennung, die dem Bot-Alias zugewiesen wurde, als Sie ihn erstellt haben. Sie können die Bot-Alias-ID in der Amazon Lex V2-Konsole auf der Seite Aliase sehen. Wenn Sie die Alias-ID nicht in der Liste sehen können, wählen Sie das Zahnradsymbol oben rechts und aktivieren Sie die Alias-ID.

- **LocaleID** — Die Kennung des Gebietsschemas, das Sie für Ihren Bot verwendet haben. Eine Liste der Gebietsschemas finden Sie unter. [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#)
- **AccessKey** und **secretKey** — Die Authentifizierungsschlüssel für Ihr Konto. Wenn Sie nicht über einen Schlüsselsatz verfügen, erstellen Sie ihn mit der Konsole. AWS Identity and Access Management
- **sessionId** — Eine Kennung für die Sitzung mit dem Amazon Lex V2-Bot. In diesem Fall verwendet der Code eine zufällige UUID.
- **Region** — Wenn sich Ihr Bot nicht in der Region USA Ost (Nord-Virginia) befindet, stellen Sie sicher, dass Sie die Region ändern.

Die Anwendung verwendet eine aufgerufene Funktion `getRecognizeTextRequest`, um individuelle Anfragen an den Bot zu erstellen. Die Funktion erstellt eine Anfrage mit den erforderlichen Parametern, die an Amazon Lex V2 gesendet werden soll.

```
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;
}
}
```

Verwenden Sie Global Resiliency, um Bots in anderen Regionen einzusetzen

Mit Global Resiliency können Sie einen Bot in einer sekundären Region replizieren. Die sekundäre Region kann durch die automatische Replikation des Bots des Benutzers in beiden Regionen aktiviert werden. Für den Fall eines regionalen Ausfalls steht Ihnen eine Backup-Region zur Verfügung. Sobald Global Resiliency aktiv ist, werden neu erstellte Bots in einer zweiten Region repliziert. AWS

Note

Diese Funktion ist nur für Amazon Connect- und Amazon Lex V2-Instances verfügbar, die in den Regionen USA Ost (Nord-Virginia) und USA West (Oregon) sowie in den Regionen Europa West (London) und Europa Zentral (Frankfurt) erstellt wurden.

Um Zugriff auf diese Funktion zu erhalten, wenden Sie sich an Ihren Amazon Connect Solutions Architect oder Technical Account Manager.

Nachdem Sie diese Funktion aktiviert haben, können Sie die Replikation von Amazon Lex V2-Bots und ihren Ressourcen, Versionen und Aliasen in einer gekoppelten AWS Region nahezu in Echtzeit automatisieren. Mit dieser Funktion können Sie die Versionsnummer des Original- und des Replikat-Bots überwachen, um sicherzustellen, dass das Bot-Replikat mit dem ursprünglichen Bot synchron bleibt. Wenn Sie die Replikation aktivieren, können Sie die vordefinierte AWS Region aktivieren, in der der Bot repliziert werden soll (Regionen basieren auf vordefinierten Paaren). Alle Aktualisierungen des Quell-Bot in der Quellregion werden automatisch auf den replizierten Bot in der zweiten Region aktualisiert.

Note

Wenn Global Resiliency für einen Bot aktiviert ist, werden alle vorhandenen Aliase und die zugehörigen Versionen in der Replikatregion repliziert. Versionen, die vor der Aktivierung der Replikation keinem Alias zugeordnet wurden, werden repliziert, wenn sie einem Alias zugeordnet werden. Jede Version und jeder Alias, die nach der Aktivierung der Replikation erstellt wurden, werden automatisch repliziert. Benutzer können den Replikationsstatus jeder einzelnen Version `ListBotVersionReplicas` und `ListBotAliasReplicas` jedes Alias verwenden und überprüfen. Bot-Mutationen verlaufen unidirektional vom Bot zu

seinem Replikat. Benutzer können den Replik-Bot nicht ändern, da er immer mit dem Bot synchronisiert wird.

Zusätzliche Informationen zur Verwendung von Global Resiliency:

- Global Resiliency funktioniert derzeit nur mit vorab festgelegten Regionspaaren.

us-east-1	us-west-2
eu-west-2	eu-central-1

- Wenn Global Resiliency für einen Bot aktiviert ist, werden alle vorhandenen Aliase und die zugehörigen Versionen in der Replikatregion repliziert. Versionen, die vor der Aktivierung der Replikation keinem Alias zugeordnet wurden, werden repliziert, wenn sie einem Alias zugeordnet sind. Jede Version und jeder Alias, die nach der Aktivierung der Replikation erstellt wurden, werden automatisch repliziert. Benutzer können den Replikationsstatus für jede einzelne Version `ListBotVersionReplicas` und `ListBotAliasReplicas` jeden Alias verwenden und sich über den Status der Replikation informieren. Bot-Mutationen verlaufen unidirektional vom Bot zu seinem Replikat. Benutzer können den Replik-Bot nicht ändern, da er immer mit dem Bot synchron gehalten wird.
- Jeder Alias kann mit jeder Version verknüpft werden. Wenn die Version noch nicht repliziert wurde, wird sie während der Verknüpfung mit dem Alias repliziert.

Einschränkungen:

- Global Resiliency repliziert keine Bots, die mit Slots erstellt wurden, die LLM wie CFAQ und Utterance Generation verwenden.
- Global Resiliency repliziert kein Netzwerk von Bots, aber jeder Bot, der Teil des Netzwerks der Bots ist, kann trotzdem einzeln repliziert werden.

Themen

- [Berechtigungen zur Replikation von Bots und zur Verwaltung von Bot-Repliken in Lex V2](#)
- [Bereitstellung globaler Resilienz mit Ihrem Lex V2-Bot](#)

Berechtigungen zur Replikation von Bots und zur Verwaltung von Bot-Repliken in Lex V2

Wenn eine IAM-Rolle mit der [AmazonLexFullAccess](#) Richtlinie verknüpft ist, kann sie Bot-Repliken erstellen und verwalten.

Wenn Sie es vorziehen, eine Rolle mit minimalen Berechtigungen für Global Resiliency zu erstellen, verwenden Sie die folgende Richtlinie, die die folgenden Aussagen enthält.

- Berechtigungen für den Zugriff auf die [serviceverknüpfte Amazon Lex V2-Rolle für die Bot-Replikation](#).
- Berechtigungen, die es Amazon Lex V2 ermöglichen, in [Ihrem Namen eine serviceverknüpfte Rolle für die Bot-Replikation zu erstellen](#).
- Berechtigungen zum Aufrufen der Bot-Replikation APIs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetReplicationSLR",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
      ]
    },
    {
      "Sid": "CreateReplicationSLR",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
      ],
      "Condition": {
```

```

        "StringEquals": {
            "iam:AWSServiceName": "lexv2.amazonaws.com"
        }
    },
    {
        "Sid": "AllowBotReplicaActions",
        "Effect": "Allow",
        "Action": [
            "lex:CreateBotReplica",
            "lex:DescribeBotReplica",
            "lex:ListBotReplica",
            "lex:ListBotVersionReplicas",
            "lex:ListBotAliasReplicas",
            "lex>DeleteBotReplica"
        ],
        "Resource": [
            "arn:aws:lex:*:*:bot/*",
            "arn:aws:lex:*:*:bot-alias/*"
        ]
    }
]
}

```

Sie können die Berechtigungen weiter einschränken, indem Sie sie wie folgt ändern.

- Ersetzen Sie `*` durch einen bestimmten Bot oder Bot-Alias IDs, um die Berechtigungen auf bestimmte Bots oder Bot-Aliase zu beschränken.
- Verwenden Sie eine Teilmenge der `lex BotReplica` Aktionen, um die Rolle auf bestimmte Aktionen zu beschränken.

Ein Beispiel finden Sie unter [Erlaubt Benutzern, Bot-Replikate zu erstellen und anzusehen, sie jedoch nicht zu löschen](#).

Bereitstellung globaler Resilienz mit Ihrem Lex V2-Bot

Mit Global Resiliency können Sie einen Bot in einer sekundären Region replizieren. Die sekundäre Region kann durch die automatische Replikation des Bots des Benutzers in beiden Regionen aktiviert werden. Für den Fall eines regionalen Ausfalls steht Ihnen eine Backup-Region zur Verfügung. Sobald Global Resiliency aktiv ist, werden neu erstellte Bots in einer zweiten AWS-Region repliziert.

Das Informationsfenster „Global Resiliency“ zeigt Details zu Ihren Bereitstellungen

Im Bereich Global Resiliency können Sie auf die folgenden Informationen zugreifen:

- **Quelldetails** — Informationen zur Quellregion Ihres Bots, zum Replikattyp, zum Aktivierungsdatum der Replikation und zur zuletzt erstellten Version. Verwenden Sie diese Informationen, um die Iterationen Ihres Bots zu verfolgen.
- **Replikationsdetails** — Nachdem Sie Ihr Bot-Replikat erstellt haben, können Sie die replizierte Region, den Replikattyp, das Synchronisierungsdatum der letzten Version und die letzte replizierte Version verfolgen. Verwenden Sie diese Informationen, um die Synchronisierung Ihres Bot-Replikats zu verfolgen.
- **Quellregion** — Die Region, in der Global Resiliency aktiviert ist. Sie können Änderungen in der Quellregion vornehmen, um den Bot in beiden Regionen zu replizieren.
- **Replikattyp** — Gibt an, ob der Bot nur lesen kann oder je nach Region lesen und schreiben kann.
- **Replikatregion** — Die sekundäre Region, die verwendet wird, um Ihren Quell-Bot aus Gründen der globalen Resilienz zu replizieren. Global Resiliency funktioniert derzeit nur mit IAD/PDX und regionalen Paaren. LDN/FRA
- **Datum der Aktivierung der Replikation** — Datum und Uhrzeit der Aktivierung des Bot-Replikats.
- **Zuletzt erstellte Version** — Die letzte Bot-Version, die dem Replikat in der Quellregion zugeordnet ist.

Aktivierung der globalen Resilienz für Ihre Lex V2-Bots

Bevor Sie Global Resiliency in der Amazon Lex V2-Konsole aktivieren, müssen Sie sicherstellen, dass der Benutzer, der die Bot-Replikation aktiviert, berechtigt ist, Service Linked Roles (SLR) zu erstellen. Global Resiliency verwendet diese FAS-Anmeldeinformationen, um eine SLR im aktivierten Konto zu erstellen, wenn sie aufgerufen wird. CreateReplica Weitere Informationen zur Einrichtung des SLR für Global Resiliency in Amazon Lex V2 finden Sie unter [AWS-verwaltete Richtlinie](#):
AmazonLexFullAccess

Note

Diese Funktion ist nur für Amazon Connect- und Amazon Lex V2-Instances verfügbar, die in den Regionen USA Ost (Nord-Virginia) und USA West (Oregon) sowie in den Regionen Europa West (London) und Europa Zentral (Frankfurt) erstellt wurden.

Um Zugriff auf diese Funktion zu erhalten, wenden Sie sich an Ihren Amazon Connect Solutions Architect oder Technical Account Manager.

Aktivieren Sie Global Resiliency und richten Sie die Bot-Replikation für eine zweite Region ein:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie in der Bots-Navigation im linken Navigationsbereich den Bot aus, den Sie replizieren möchten.
3. Wählen Sie Deployment > Global Resiliency.
4. Klicken Sie in der oberen rechten Ecke des Fensters auf die Schaltfläche Replikat erstellen, um eine Entwurfsversion Ihres Bots zu erstellen.

 Note

Stellen Sie sicher, dass Sie in der sekundären Region keine Bots mit demselben Namen wie der Bot haben, den Sie replizieren möchten. (Ihr Bot muss einen eindeutigen Namen haben).

5. Gehen Sie zu Global Resiliency und klicken Sie auf Create Replica. Diese Aktion erstellt eine Entwurfsversion Ihres Bots. (Sie müssen nicht zur Registerkarte Global Resiliency zurückkehren, außer um den Status zu überprüfen oder Details zukünftiger Builds zu sehen).

 Note

Sie können auch einen Alias-Bot für die Replikation in Global Resiliency erstellen, indem Sie zu Alias gehen und Neuen Alias für Bot mit aktiviertem Global Resiliency erstellen auswählen. Alle Aliase werden repliziert, auch wenn sie vor der Aktivierung der Replikation erstellt wurden.

6. Gehen Sie zu Alias — Neuen Alias für den Bot mit aktivierter Global Resiliency-Funktion erstellen. Alle Aliase werden repliziert, auch wenn sie vor der Aktivierung der Replikation erstellt wurden.
7. Gehe zu Version — Neue Version für den Bot mit aktiviertem Global Resiliency erstellen. Jede Version, die einem Alias zugeordnet ist, wird repliziert, auch wenn sie vor der Aktivierung der Replikation erstellt wurde.

 Note

Kunden haben weiterhin die volle Kontrolle über die Verwaltung ihrer ressourcenbasierten Richtlinien und Tags für replizierte Bots. Lambda-Funktionen und CloudWatch Logs Groups müssen in beiden Regionen mit denselben Identifikatoren bereitgestellt werden. Benutzer müssen die Lambda-Funktion in der Replikatregion nicht erneut zuordnen.

Deaktivierung der globalen Resilienz

Sie können Global Resiliency jederzeit deaktivieren, indem Sie auf die Schaltfläche Disable Global Resiliency klicken. Diese Aktion verhindert, dass Ihr Quell-Bot und alle mit ihm verknüpften Aliase und Versionen in anderen Regionen repliziert werden.

Verwendung APIs mit Global Resiliency für Ihre Lex V2-Bots

Sie können API-Aufrufe in Global Resiliency wie folgt tätigen. APIs Weitere Informationen zu Global Resiliency APIs und Amazon Lex V2 finden Sie im [Amazon Lex V2 API-Leitfaden](#).

- **CreateBotReplica**

Aktivieren Sie Global Resiliency und erstellen Sie einen replizierten Bot. Erfordert `replicaRegion`.

Weitere Informationen finden Sie [CreateBotReplica](#) im API-Leitfaden.

- **DeleteBotReplica**

Deaktivieren Sie Global Resiliency und löschen Sie den replizierten Bot. Erfordert `replicaRegion` und `botId`.

Weitere Informationen finden Sie [DeleteBotReplica](#) im API-Leitfaden.

- **ListBotReplicas**

Listet die replizierten Bots in der sekundären Zone auf. Erfordert `botId`.

Weitere Informationen finden Sie [ListBotReplicas](#) im API-Leitfaden.

- **DescribeBotReplica**

Zusammenfassung der Informationen für den replizierten Bot. Erfordert `replicaRegion` und `botId`.

Weitere Informationen finden Sie [DescribeBotReplica](#) im API-Leitfaden.

Integration eines Amazon Lex V2-Bots in eine Messaging-Plattform

In diesem Abschnitt wird erklärt, wie Amazon Lex V2-Bots in die Messaging-Plattformen Facebook, Slack und Twilio integriert werden. Wenn Sie noch keinen Amazon Lex V2-Bot haben, erstellen Sie einen. In diesem Thema gehen wir davon aus, dass Sie den Bot verwenden, in dem Sie ihn erstellt haben [Übung 1: Einen Bot anhand eines Beispiels erstellen](#). Sie können jedoch jeden beliebigen Bot verwenden.

Note

Beim Speichern Ihrer Facebook-, Slack- oder Twilio-Konfigurationen verwendet Amazon Lex V2 eine, um Informationen AWS KMS key zu verschlüsseln. Wenn Sie zum ersten Mal einen Kanal zu einer dieser Messaging-Plattformen einrichten, erstellt Amazon Lex V2 einen vom Kunden verwalteten Standardschlüssel (aws/lex) in Ihrem AWS Konto, oder Sie können Ihren eigenen vom Kunden verwalteten Schlüssel auswählen. Amazon Lex V2 unterstützt nur symmetrische Schlüssel. Weitere Informationen finden Sie im [AWS Key Management Service -Entwicklerhandbuch](#).

Wenn eine Messaging-Plattform eine Anfrage an Amazon Lex V2 sendet, enthält sie plattformspezifische Informationen als Anforderungsattribut für Ihre Lambda-Funktion. Verwenden Sie dieses Attribut, um das Verhalten Ihres Bots anzupassen. Weitere Informationen finden Sie unter [Einstellung von Anforderungsattributen für Ihren Lex V2-Bot](#).

Allgemeine Anforderungsattribute für Messaging-Plattformen

Attribut	Beschreibung
x-amz-lexAllgemeine Anforderungsattribute für Messaging-Plattformen ----sep----:channel:platform	Einer der folgenden Werte: • Facebook • Slack • Twilio

Integrieren eines Amazon Lex V2-Bots mit Facebook Messenger

Sie können Ihren Amazon Lex V2-Bot im Facebook Messenger hosten. Wenn Sie dies tun, können Facebook-Nutzer mit Ihrem Bot interagieren, um Absichten zu erfüllen.

Bevor Sie beginnen, müssen Sie sich unter <https://developers.facebook.com> für ein Facebook-Entwicklerkonto anmelden.

Sie müssen die folgenden Schritte ausführen:

Themen

- [Schritt 1: Erstellen Sie eine Facebook-Anwendung](#)
- [Schritt 2: Integrieren Sie Facebook Messenger in den Amazon Lex V2-Bot](#)
- [Schritt 3: Schließen Sie die Facebook-Integration mit Ihrem Lex V2-Bot ab](#)
- [Schritt 4: Testen Sie die Integration mit Facebook Messenger](#)

Schritt 1: Erstellen Sie eine Facebook-Anwendung

Klicken Sie auf dem Facebook-Developer-Portal, erstellen Sie eine Facebook-Anwendung und eine Facebook-Seite.

Um eine Facebook-Anwendung zu erstellen

1. Öffnen Sie <https://developers.facebook.com/apps>
2. Wählen Sie Create app (App erstellen).
3. Wählen Sie auf der Seite „App erstellen“ die Option Business und dann Weiter aus.
4. Treffen Sie in den Feldern App-Name hinzufügen, App-Kontakt-E-Mail und Unternehmenskonto die entsprechenden Optionen für Ihre App. Wählen Sie App erstellen, um fortzufahren.
5. Wählen Sie unter Produkte zu Ihrer App hinzufügen auf der Messenger-Kachel die Option Einrichten aus.
6. Wählen Sie im Abschnitt Zugriffstoken die Option Seiten hinzufügen oder entfernen aus.
7. Wählen Sie eine Seite aus, die Sie mit Ihrer App verwenden möchten, und wählen Sie dann Weiter.
8. Behalten Sie unter Was darf die App tun? die Standardeinstellungen bei und wählen Sie dann „Fertig“.
9. Klicken Sie auf der Bestätigungsseite auf OK.

10. Wählen Sie im Abschnitt Zugriffstoken die Option Generate Token aus und kopieren Sie dann das Token. Sie geben dieses Token in der Amazon Lex V2-Konsole ein.
11. Wählen Sie im linken Menü Einstellungen und dann Basic.
12. Wählen Sie für App Secret die Option Show aus und kopieren Sie dann das Secret. Sie geben dieses Token in der Amazon Lex V2-Konsole ein.

Nächster Schritt

Schritt 2: Integrieren Sie Facebook Messenger in den Amazon Lex V2-Bot

Schritt 2: Integrieren Sie Facebook Messenger in den Amazon Lex V2-Bot

In diesem Schritt verknüpfen Sie Ihren Amazon Lex V2-Bot mit Facebook.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Amazon Lex V2-Bot aus, den Sie erstellt haben.
3. Wählen Sie im linken Menü Kanalintegrationen und dann Kanal hinzufügen aus.
4. Gehen Sie unter Kanal erstellen wie folgt vor:
 - a. Wählen Sie als Plattform Facebook aus.
 - b. Wählen Sie für Identitätsrichtlinien den AWS KMS Schlüssel zum Schutz der Kanalinformationen aus. Der Standardschlüssel wird von Amazon Lex V2 bereitgestellt.
 - c. Geben Sie dem Kanal für die Integrationskonfiguration einen Namen und eine optionale Beschreibung. Wählen Sie den Alias, der auf die Version des Bots verweist, die verwendet werden soll, und wählen Sie die Sprache aus, die der Kanal unterstützt.
 - d. Geben Sie für die zusätzliche Konfiguration Folgendes ein:
 - Alias — Eine Zeichenfolge, die die App identifiziert, die Amazon Lex V2 aufruft. Sie können eine beliebige Zeichenfolge verwenden. Nehmen Sie diese Zeichenfolge auf und geben Sie sie in der Facebook-Entwicklerkonsole ein.
 - Seitenzugriffstoken — Das Seitenzugriffstoken, das Sie von der Facebook-Entwicklerkonsole kopiert haben.
 - Geheimer App-Schlüssel — Der geheime Schlüssel, den Sie von der Facebook-Entwicklerkonsole kopiert haben.
 - e. Wählen Sie Create (Erstellen) aus.

- f. Amazon Lex V2 zeigt die Liste der Kanäle für Ihren Bot. Wählen Sie aus der Liste den Kanal aus, den Sie gerade erstellt haben.
- g. Notieren Sie sich unter Rückruf-URL die Rückruf-URL. Sie geben diese URL in der Facebook-Entwicklerkonsole ein.

Nächster Schritt

Schritt 3: Schließe die Facebook-Integration mit deinem Lex V2-Bot ab

Schritt 3: Schließe die Facebook-Integration mit deinem Lex V2-Bot ab

Verwenden Sie in diesem Schritt die Facebook-Entwicklerkonsole, um die Integration mit Amazon Lex V2 abzuschließen.

Um die Facebook Messenger-Integration abzuschließen

1. Öffnen Sie <https://developers.facebook.com/apps>
2. Wählen Sie aus der Liste der Apps die App aus, die Sie in Facebook Messenger integrieren möchten.
3. Wähle im linken Menü Messenger und dann Einstellungen.
4. Im Bereich Webhooks:
 - a. Wählen Sie Rückruf-URL hinzufügen.
 - b. Geben Sie unter Rückruf-URL bearbeiten Folgendes ein:
 - Callback-URL — Geben Sie die Callback-URL ein, die Sie über die Amazon Lex V2-Konsole aufgezeichnet haben.
 - Token verifizieren — Geben Sie den Alias ein, den Sie in der Amazon Lex V2-Konsole eingegeben haben.
 - c. Wählen Sie Verify and Save aus.
 - d. Wählen Sie unter Webhooks neben Ihrer Seite die Option Abonnements hinzufügen aus.
 - e. Wählen Sie in dem sich öffnenden Fenster die Option **messages** und klicken Sie dann auf Speichern.

Nächster Schritt

Schritt 4: Testen Sie die Integration mit Facebook Messenger

Schritt 4: Testen Sie die Integration mit Facebook Messenger

Sie können jetzt mit Ihrem Amazon Lex V2-Bot eine Konversation über Facebook Messenger starten.

Um die Integration zwischen Facebook Messenger und einem Amazon Lex V2-Bot zu testen

1. Öffnen Sie die Facebook-Seite, die Sie in Schritt 1 mit Ihrem Bot verknüpft haben.
2. Verwenden Sie im Messenger-Fenster die unter angegebenen Testäußerungen. [Übung 1: Einen Bot anhand eines Beispiels erstellen](#)

Einen Amazon Lex V2-Bot mit Slack integrieren

Dieses Thema enthält Anweisungen zur Integration eines Amazon Lex V2-Bots in die Slack-Messaging-Anwendung. Führen Sie die folgenden Schritte aus:

Themen

- [Schritt 1: Melde dich bei Slack an und erstelle ein Slack-Team](#)
- [Schritt 2: Erstelle eine Slack-Anwendung](#)
- [Schritt 3: Integrieren Sie die Slack-Anwendung in den Amazon Lex V2-Bot](#)
- [Schritt 4: Schließe die Slack-Integration mit deinem Lex V2-Bot ab](#)
- [Schritt 5: Teste die Integration zwischen deinem Lex V2-Bot und Slack](#)

Schritt 1: Melde dich bei Slack an und erstelle ein Slack-Team

Melden Sie sich für ein Slack-Konto an und erstellen Sie ein Slack-Team. Anweisungen hierzu finden Sie unter [Verwenden von Slack](#). Im nächsten Abschnitt erstellst du eine Slack-Anwendung, die jedes Slack-Team installieren kann.

Nächster Schritt

[Schritt 2: Erstelle eine Slack-Anwendung](#)

Schritt 2: Erstelle eine Slack-Anwendung

In diesem Abschnitt führen Sie folgenden Aufgaben aus:

1. Erstelle eine Slack-Anwendung in der Slack-API-Konsole.
2. Konfiguriere die Anwendung so, dass sie deinem Bot interaktive Nachrichten hinzufügt.

Am Ende dieses Abschnitts erhalten Sie Anmeldeinformationen für die Anwendung (Client-ID, Client Secret und Verification Token). Im nächsten Schritt verwenden Sie diese Informationen, um den Bot in die Amazon Lex V2-Konsole zu integrieren.

Um eine Slack-Anwendung zu erstellen

1. Melde dich bei der Slack API-Konsole an unter. <https://api.slack.com>
2. Erstellen Sie eine -Anwendung.

Nachdem Sie die Anwendung erfolgreich erstellt haben, zeigt Slack die Seite Basic Information für die Anwendung an.

3. Konfigurieren Sie die Anwendungsfunktionen wie folgt:
 - Wähle im linken Menü Interaktivität und Kurzbefehle.
 - Aktivieren Sie die Option, um interaktive Komponenten zu aktivieren.
 - Geben Sie im Feld Request URL eine gültige URL an. Sie können beispielsweise die Datei **`https://slack.com`** verwenden.

 Note

Geben Sie zunächst eine gültige URL ein, sodass Sie das Verifizierungstoken erhalten, das Sie im nächsten Schritt benötigen. Sie werden diese URL aktualisieren, nachdem Sie die Bot-Channel-Verknüpfung in der Amazon Lex-Konsole hinzugefügt haben.

- Wählen Sie Save Changes.
4. Klicken Sie im linken Menü unter Settings auf Basic Information. Notieren Sie die folgenden Anwendungs-Anmeldeinformationen:
 - Client-ID
 - Clientschlüssel
 - Verifizierungstoken

Nächster Schritt

[Schritt 3: Integrieren Sie die Slack-Anwendung in den Amazon Lex V2-Bot](#)

Schritt 3: Integrieren Sie die Slack-Anwendung in den Amazon Lex V2-Bot

Integrieren Sie in diesem Abschnitt die von Ihnen erstellte Slack-Anwendung mit dem Amazon Lex V2-Bot, den Sie mithilfe von Channel-Integrationen erstellt haben.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
 2. Wählen Sie aus der Liste der Bots den Amazon Lex V2-Bot aus, den Sie erstellt haben.
 3. Wählen Sie im linken Menü Kanalintegrationen und dann Kanal hinzufügen aus.
 4. Gehen Sie unter Kanal erstellen wie folgt vor:
 - a. Wähle Slack als Plattform aus.
 - b. Wähle für Identitätsrichtlinien den AWS KMS Schlüssel zum Schutz der Kanalinformationen aus. Der Standardschlüssel wird von Amazon Lex V2 bereitgestellt.
 - c. Geben Sie dem Kanal für die Integrationskonfiguration einen Namen und eine optionale Beschreibung. Wählen Sie den Alias, der auf die Version des Bots verweist, die verwendet werden soll, und wählen Sie die Sprache aus, die der Kanal unterstützt.
-  **Note**

Wenn Ihr Bot in mehreren Sprachen verfügbar ist, müssen Sie für jede Sprache einen anderen Kanal und eine andere Anwendung erstellen.
- d. Geben Sie für die zusätzliche Konfiguration Folgendes ein:
 - Client-ID — gib die Client-ID von Slack ein.
 - Kundengeheimnis — gib das Client-Geheimnis von Slack ein.
 - Verifizierungstoken — gib das Verifizierungstoken von Slack ein.
 - URL der Erfolgsseite — Die URL der Seite, die Slack öffnen soll, wenn der Benutzer authentifiziert ist. Normalerweise lässt du dieses Feld leer.
 5. Wählen Sie Erstellen, um den Kanal zu erstellen.
 6. Amazon Lex V2 zeigt die Liste der Kanäle für Ihren Bot. Wählen Sie aus der Liste den Kanal aus, den Sie gerade erstellt haben.
 7. Notieren Sie sich unter Callback-URL den Endpunkt und den OAuth Endpunkt.

Nächster Schritt

Schritt 4: Schließe die Slack-Integration mit deinem Lex V2-Bot ab

Schritt 4: Schließe die Slack-Integration mit deinem Lex V2-Bot ab

Verwenden Sie in diesem Abschnitt die Slack-API-Konsole, um die Integration mit der Slack-Anwendung abzuschließen.

1. Melde dich bei der Slack-API-Konsole an unter <https://api.slack.com>. Wählen Sie die App aus, die Sie in [Schritt 2: Erstelle eine Slack-Anwendung](#) erstellt haben.
2. Aktualisiere die Funktion OAuth & Berechtigungen wie folgt:
 - a. Wählen Sie im linken Menü die Option OAuth & Berechtigungen aus.
 - b. Fügen Sie im URLs Abschnitt Umleitung den OAuth Endpunkt hinzu, den Amazon Lex im vorherigen Schritt bereitgestellt hat. Wählen Sie Add und dann Save URLs aus.
 - c. Fügen Sie im Abschnitt Bot-Token-Bereiche zwei Berechtigungen mit der Schaltfläche OAuth Bereich hinzufügen hinzu. Filtern Sie die Liste nach dem folgenden Text:
 - **chat:write**
 - **team:read**
3. Aktualisieren Sie die Funktion Interaktivität und Kurzbefehle, indem Sie den Wert der Anforderungs-URL auf den Endpunkt aktualisieren, den Amazon Lex im vorherigen Schritt bereitgestellt hat. Geben Sie den Endpunkt ein, den Sie in Schritt 3 gespeichert haben, und wählen Sie dann Änderungen speichern.
4. Abonnieren Sie die Funktion Event Subscriptions- wie folgt:
 - Aktivieren Sie Ereignisse, indem Sie die Option On auswählen.
 - Setzen Sie den Wert für die Anforderungs-URL auf den Endpunkt, den Amazon Lex im vorherigen Schritt bereitgestellt hat.
 - Wählen Sie im Abschnitt Bot-Ereignisse abonnieren die Option Bot-Benutzerereignis hinzufügen aus und fügen Sie das **message.im** Bot-Ereignis hinzu, um Direktnachrichten zwischen dem Endbenutzer und dem Slack-Bot zu aktivieren.
 - Speichern Sie die Änderungen.
5. Aktiviere das Senden von Nachrichten über den Nachrichten-Tab wie folgt:
 - Wählen Sie im linken Menü App Home.

- Wählen Sie im Bereich „Tabs anzeigen“ auf dem Nachrichten-Tab die Option Benutzern das Senden von Slash-Befehlen und Nachrichten erlauben aus.
6. Wählen Sie Manage Distribution unter Settings aus. Wählen Sie Add to Slack aus, um die Anwendung zu installieren. Wenn du bei mehreren Workspaces authentifiziert bist, wähle zuerst den richtigen Workspace in der oberen rechten Ecke aus der Drop-down-Liste aus. Wählen Sie als Nächstes Zulassen aus, um den Bot zu autorisieren, auf Nachrichten zu antworten.

 Note

Wenn du später Änderungen an den Einstellungen deiner Slack-Anwendung vornimmst, musst du diesen Teilschritt wiederholen.

Nächster Schritt

[Schritt 5: Teste die Integration zwischen deinem Lex V2-Bot und Slack](#)

Schritt 5: Teste die Integration zwischen deinem Lex V2-Bot und Slack

Verwenden Sie jetzt ein Browserfenster, um die Integration von Slack mit Ihrem Amazon Lex V2-Bot zu testen.

Um deine Slack-Anwendung zu testen

1. Starte Slack. Wähle im linken Menü im Bereich Direktnachrichten deinen Bot aus. Wenn Sie Ihren Bot nicht sehen, klicken Sie auf das Plus-Symbol (+) neben Direct Messages, um danach zu suchen.
2. Nehmen Sie mit Ihrer Slack-Anwendung an einem Chat teil. Dein Bot reagiert auf Nachrichten.

Wenn Sie den Bot mithilfe von erstellt haben [Übung 1: Einen Bot anhand eines Beispiels erstellen](#), können Sie die Beispielkonversationen aus dieser Übung verwenden.

Integration eines Amazon Lex V2-Bots mit Twilio SMS

Dieses Thema enthält Anweisungen zur Integration eines Amazon Lex V2-Bots in den Twilio Simple Message Service (SMS). Führen Sie die folgenden Schritte aus:

Themen

- [Schritt 1: Erstellen Sie ein Twilio SMS-Konto](#)
- [Schritt 2: Integrieren Sie den Twilio Message Service-Endpunkt in den Amazon Lex V2-Bot](#)
- [Schritt 3: Vollständige Twilio-Integration zwischen Ihrem Lex V2-Bot und Twilio](#)
- [Schritt 4: Testen Sie die Integration zwischen Ihrem Lex V2-Bot und Twilio](#)

Schritt 1: Erstellen Sie ein Twilio SMS-Konto

Melden Sie sich für ein Twilio-Konto an und erfassen Sie die folgenden Kontoinformationen:

- ACCOUNT SID
- AUTH TOKEN

[Anweisungen zur Registrierung finden Sie unter https://www.twilio.com /console.](https://www.twilio.com/console)

Nächster Schritt

[Schritt 2: Integrieren Sie den Twilio Message Service-Endpunkt in den Amazon Lex V2-Bot](#)

Schritt 2: Integrieren Sie den Twilio Message Service-Endpunkt in den Amazon Lex V2-Bot

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Amazon Lex V2-Bot aus, den Sie erstellt haben.
3. Wählen Sie im linken Menü Kanalintegrationen und dann Kanal hinzufügen aus.
4. Gehen Sie unter Kanal erstellen wie folgt vor:
 - a. Wählen Sie als Plattform Twilio aus.
 - b. Wählen Sie für Identitätsrichtlinien den AWS KMS Schlüssel zum Schutz der Kanalinformationen aus. Der Standardschlüssel wird von Amazon Lex V2 bereitgestellt.
 - c. Geben Sie dem Kanal für die Integrationskonfiguration einen Namen und eine optionale Beschreibung. Wählen Sie den Alias, der auf die zu verwendende Version des Bots verweist, und wählen Sie die Sprache aus, die der Kanal unterstützt.
 - d. Geben Sie für die zusätzliche Konfiguration die Konto-SID und das Authentifizierungstoken aus dem Twilio-Dashboard ein.
5. Wählen Sie Erstellen aus.

6. Wählen Sie aus der Liste der Kanäle den Kanal aus, den Sie gerade erstellt haben.
7. Kopieren Sie die Callback-URL.

Nächster Schritt

Schritt 3: Vollständige Twilio-Integration zwischen Ihrem Lex V2-Bot und Twilio

Schritt 3: Vollständige Twilio-Integration zwischen Ihrem Lex V2-Bot und Twilio

Verwenden Sie die Twilio-Konsole, um die Integration Ihres Amazon Lex V2-Bots mit Twilio SMS abzuschließen.

1. Öffnen Sie die Twilio-Konsole unter /console. <https://www.twilio.com>
2. Wählen Sie im linken Menü „Alle Produkte und Dienste“ und anschließend „Telefonnummer“.
3. Wenn du eine Telefonnummer hast, wähle sie aus. Wenn Sie keine Telefonnummer haben, wählen Sie Nummer kaufen, um eine zu erhalten.
4. Geben Sie im Bereich Messaging unter A MESSAGE COMES IN die Callback-URL von der Amazon Lex V2-Konsole ein.
5. Wählen Sie Speichern.

Nächster Schritt

Schritt 4: Testen Sie die Integration zwischen Ihrem Lex V2-Bot und Twilio

Schritt 4: Testen Sie die Integration zwischen Ihrem Lex V2-Bot und Twilio

Verwenden Sie Ihr Mobiltelefon, um die Integration zwischen Twilio SMS und Ihrem Bot zu testen. Senden Sie von Ihrem Mobiltelefon aus Nachrichten an die Twilio-Nummer.

Wenn Sie den Bot mit erstellt haben [Übung 1: Einen Bot anhand eines Beispiels erstellen](#), können Sie die Beispielkonversationen aus dieser Übung verwenden.

Integration eines Amazon Lex V2-Bots in ein Kontaktzentrum

Sie können Amazon Lex V2-Bots in Ihre Kontaktzentren integrieren, um Self-Service-Anwendungsfälle mithilfe der Amazon Lex V2-Streaming-API zu ermöglichen. Verwenden Sie diese

Bots als Agenten für interaktive Sprachantwort (IVR) in der Telefonie oder als textbasierten Chatbot, der in Ihr Kontaktzentrum integriert ist. Weitere Informationen zum Streaming APIs finden Sie unter.

[Konversationen an einen Amazon Lex V2-Bot streamen](#)

Beim Streaming APIs können Sie die folgenden Funktionen aktivieren:

- Unterbrechungen („Barge-In“) — Anrufer können den Bot unterbrechen und eine Frage beantworten, bevor die Aufforderung abgeschlossen ist. Weitere Informationen finden Sie unter [So aktivieren, dass Ihr Amazon Lex V2-Bot vom Benutzer unterbrochen wird](#).
- Warten und weitermachen — Anrufer können den Bot anweisen, zu warten, wenn sie während eines Anrufs Zeit benötigen, um zusätzliche Informationen abzurufen, z. B. eine Kreditkartennummer oder eine Buchungsnummer. Weitere Informationen finden Sie unter [Den Amazon Lex V2-Bot so aktivieren, dass er während einer Pause darauf wartet, dass der Benutzer weitere Informationen bereitstellt](#).
- DTMF-Unterstützung — Anrufer können Informationen synonym per Sprache oder DTMF bereitstellen.
- SSML-Unterstützung — Sie können Amazon Lex V2-Bot-Eingabeaufforderungen mithilfe von SSML-Tags konfigurieren, um eine bessere Kontrolle über die Sprachgenerierung aus Text zu erhalten. Weitere Informationen finden Sie unter [Generieren von Sprache aus SSML-Dokumenten](#) im Amazon Polly Developer Guide.
- Konfigurierbare Timeouts — Sie können konfigurieren, wie lange es dauern soll, bis Kunden ihre Rede beendet haben, bevor Amazon Lex V2 ihre Spracheingaben erfasst, z. B. die Beantwortung einer Ja- oder Nein-Frage oder die Angabe eines Datums oder einer Kreditkartennummer. Weitere Informationen finden Sie unter [Konfiguration von Timeouts für die Erfassung von Benutzereingaben mit einem Lex V2-Bot](#).
- Aktualisierungen des Fulfillment-Fortschritts — Sie können den Bot so konfigurieren, dass er während der Ausführung der Geschäftslogik für die Erfüllung von Absichtserfüllungen je nach Erfüllungsstatus mehrere Nachrichten sendet. Sie können den Bot so einrichten, dass er mit Nachrichten reagiert, wenn die Erfüllung beginnt und abgeschlossen ist, und dass er regelmäßige Updates für Lambda-Funktionen mit langer Laufzeit bereitstellt. Weitere Informationen finden Sie unter [Updates zum Fulfillment-Fortschritt für Ihren Lex V2-Bot konfigurieren](#).

Amazon Chime SDK

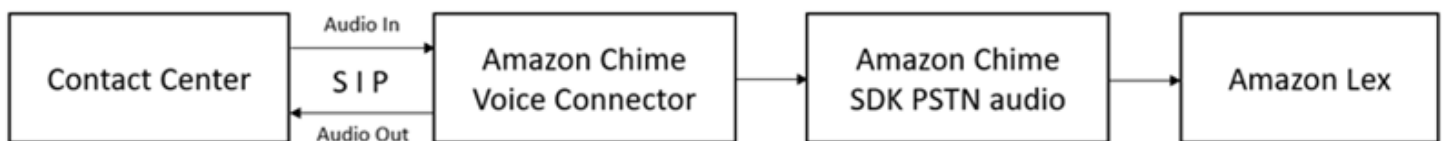
Verwenden Sie das Amazon Chime SDK, um Ihren Web- oder Mobilanwendungen Audio-, Video-, Screensharing- und Messaging-Funktionen in Echtzeit hinzuzufügen. Das Amazon Chime SDK

bietet einen PSTN-Audiodienst (Public Switched Telephone Network), mit dem Sie benutzerdefinierte Telefonieanwendungen mit einer Funktion erstellen können. AWS Lambda

Amazon Chime PSTN-Audio ist in Amazon Lex V2 integriert. Sie können diese Integration verwenden, um auf Amazon Lex V2-Bots als interaktive Sprachantwortsysteme (IVR) in Kontaktzentren für Audiointeraktionen zuzugreifen. Verwenden Sie dies, um Amazon Lex V2 mithilfe von PSTN-Audiodiensten in den folgenden Szenarien zu integrieren.

Contact-Center-Integrationen — Sie können den Amazon Chime Voice Connector und den Amazon Chime SDK PSTN-Audioservice verwenden, um auf Amazon Lex V2-Bots zuzugreifen. Verwenden Sie sie in jeder Contact-Center-Anwendung, die das Session Initiation Protocol (SIP) für die Sprachkommunikation verwendet. Diese Integration erweitert Ihr vorhandenes lokales oder cloudbasiertes Contact Center mit SIP-Unterstützung um Sprachgespräche in natürlicher Sprache. Eine Liste der unterstützten Contact-Center-Plattformen finden Sie unter [Amazon Chime Voice Connector-Ressourcen](#).

Das folgende Diagramm zeigt die Integration zwischen einem Contact Center, das SIP verwendet, und Amazon Lex V2.



Direkter Telefonie-Support — Sie können maßgeschneiderte IVR-Lösungen erstellen, um über eine im Amazon Chime SDK bereitgestellte Telefonnummer direkt auf Amazon Lex V2-Bots zuzugreifen.

Weitere Informationen finden Sie in den folgenden Themen im Amazon Chime SDK-Handbuch.

- [SIP-Integration mithilfe eines Amazon Chime Chime-Sprachanschlusses](#)
- [Verwenden des Amazon Chime SDK PSTN-Audiodienstes](#)
- [Integrieren von Amazon Chime PSTN-Audio mit Amazon Lex V2](#)

Wenn das Amazon Chime SDK eine Anfrage an Amazon Lex V2 sendet, enthält es plattformspezifische Informationen für Ihre Lambda-Funktion und Konversationsprotokolle. Verwenden Sie diese Informationen, um die Contact-Center-Anwendung zu ermitteln, die Traffic an Ihren Bot sendet.

Gemeinsames Anforderungsattribut	Wert
x-amz-lexAllgemeines Anforderungsattribut ---- sep----:channels:platform	Amazon Chime SDK PSTN Audio

Amazon Connect

Amazon Connect ist ein Omnichannel-Cloud-Contact Center. Sie können in wenigen Schritten ein Kontaktzentrum einrichten, Agenten hinzufügen, die sich überall befinden, und mit Ihren Kunden in Kontakt treten. Weitere Informationen finden [Sie unter Erste Schritte mit Amazon Connect](#) im Amazon Connect Connect-Administratorhandbuch.

Durch die Kommunikation über unterschiedliche Kanäle profitieren Ihre Kunden von einer personalisierten Erfahrung. Sie können beispielsweise Chat- und Sprachkontakt anbieten, die auf Kundenpräferenzen und geschätzten Wartezeiten basieren. In der Zwischenzeit können Agenten alle Kunden von nur einer Oberfläche aus betreuen. Sie können beispielsweise mit Kunden chatten und Aufgaben erstellen oder beantworten, während diese an sie weitergeleitet werden.

Sie können Amazon Connect für Audiointeraktionen mit Ihren Kunden oder Amazon Connect Chat für reine Textinteraktionen verwenden.

Weitere Informationen finden Sie in den folgenden Themen im Amazon Connect Connect-Administratorhandbuch.

- [Was ist Amazon Connect](#)
- [Einen Amazon Lex V2-Bot hinzufügen](#)
- [Amazon Connect Kontaktblock für Kundeneingaben abrufen](#)

Wenn ein Contact Center eine Anfrage an Amazon Lex V2 sendet, enthält diese plattformspezifische Informationen als Anforderungsattribut für Ihre Lambda-Funktion und Konversationsprotokolle. Verwenden Sie diese Informationen, um festzustellen, welche Contact-Center-Anwendung Traffic an Ihren Bot sendet.

Allgemeine Anforderungsattribute für Amazon Connect

Attribut	Wert
x-amz-lexAllgemeine Anforderungsattribute für Amazon Connect ----sep----:channels:platform	Einer der folgenden Werte: • Connect • Connect Chat

Genesys Cloud

Genesys Cloud ist eine Suite von Cloud-Diensten für Unternehmenskommunikation, Zusammenarbeit und Contact-Center-Management. Genesys Cloud basiert auf einer verteilten Cloud-Umgebung AWS und nutzt diese, die Organisationen rund um den Arbeitsplatz sicheren Zugriff bietet.

Weitere Informationen finden Sie auf den folgenden Seiten der Genesys Cloud-Website.

- [Über das Genesys Cloud-Kontaktzentrum](#)
- [Über die Amazon Lex V2-Integration](#)

Wenn ein Contact Center eine Anfrage an Amazon Lex V2 sendet, enthält es plattformspezifische Informationen als Anforderungsattribut für Ihre Lambda-Funktion und Konversationsprotokolle. Verwenden Sie diese Informationen, um festzustellen, welche Contact-Center-Anwendung Traffic an Ihren Bot sendet.

Allgemeine Anforderungsattribute für Genesys Cloud

Attribut	Wert
x-amz-lexAllgemeine Anforderungsattribute für Genesys Cloud ----sep----:channels:platform	• Genesys Cloud

Weitere Informationen

- [Stärken Sie Ihr Kontaktzentrum mit Amazon Lex und Genesys Cloud](#)

Bot-Konversationen verstehen

Nachdem Sie einen Bot erstellt haben, integrieren Sie Ihre Client-Anwendung in die Amazon Lex V2-Laufzeitoperationen, um Konversationen mit Ihrem Bot zu führen. Wenn ein Benutzer eine Konversation mit Ihrem Bot beginnt, erstellt Amazon Lex V2 eine Sitzung. Eine Sitzung kapselt die Informationen, die zwischen Ihrer Anwendung und dem Bot ausgetauscht werden. Weitere Informationen finden Sie unter [Grundlegendes zu Amazon Lex V2-Bot-Sitzungen](#).

Eine typische Konversation beinhaltet einen Hin- und Herfluss zwischen dem Benutzer und einem Bot. Zum Beispiel:

```
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.
```

Verwenden Sie die [RecognizeText](#) oder [RecognizeUtterance](#) API-Operationen, um die Konversationen selbst zu verwalten. Verwenden Sie den [StartConversation](#) API-Vorgang, damit Amazon Lex V2 die Konversation für Sie verwalten kann.

Um die Konversation zu verwalten, müssen Sie Benutzeräußerungen an den Bot senden, bis die Konversation ein logisches Ende erreicht hat. Die aktuelle Konversation wird im Sitzungsstatus aufgezeichnet. Der Sitzungsstatus wird nach jeder Benutzeräußerung aktualisiert. Der Sitzungsstatus enthält den aktuellen Status der Konversation und wird vom Bot als Antwort auf jede Benutzeräußerung zurückgegeben.

Eine Konversation kann sich in einem der folgenden Zustände befinden:

- **ElicitIntent**— Zeigt an, dass der Bot die Absicht des Benutzers noch nicht bestimmt hat.
- **ElicitSlot**— Zeigt an, dass der Bot die Absicht des Benutzers erkannt hat und die erforderlichen Informationen sammelt, um die Absicht zu erfüllen.
- **ConfirmIntent**— Zeigt an, dass der Bot darauf wartet, dass der Benutzer bestätigt, dass die gesammelten Informationen korrekt sind.

- **Geschlossen** — Zeigt an, dass die Absicht des Benutzers abgeschlossen ist und dass die Konversation mit dem Bot ein logisches Ende erreicht hat.

Ein Benutzer kann eine neue Absicht angeben, nachdem die erste Absicht abgeschlossen ist. Weitere Informationen finden Sie unter [Konversationskontext mit Ihren Lex V2-Bots](#).

Eine Absicht kann einen der folgenden Zustände haben:

- **InProgress**— Zeigt an, dass der Bot Informationen sammelt, die zur Ausführung der Absicht erforderlich sind. Dies steht in Verbindung mit dem `ElicitSlot` Konversationsstatus.
- **Wartend** — Zeigt an, dass der Benutzer den Bot gebeten hat, zu warten, als der Bot nach Informationen für einen bestimmten Slot gefragt hat.
- **Erfüllt** — Zeigt an, dass die Geschäftslogik in einer Lambda-Funktion, die der Absicht zugeordnet ist, erfolgreich ausgeführt wurde.
- **ReadyForFulfillment**— Zeigt an, dass der Bot alle Informationen gesammelt hat, die zur Erfüllung der Absicht erforderlich sind, und dass die Client-Anwendung die Fulfillment-Geschäftslogik ausführen kann.
- **Fehlgeschlagen** — Zeigt an, dass eine Absicht fehlgeschlagen ist.

In den folgenden Themen erfahren Sie, wie Sie Amazon Lex V2 verwenden, APIs um den Konversationskontext und die Sitzungen zwischen Ihrem Bot und Benutzern zu verwalten.

Themen

- [Konversationskontext mit Ihren Lex V2-Bots](#)
- [Grundlegendes zu Amazon Lex V2-Bot-Sitzungen](#)

Konversationskontext mit Ihren Lex V2-Bots

Konversationskontext sind Informationen, die der Benutzer, Ihre Client-Anwendung oder eine Lambda-Funktion einem Bot zur Erfüllung einer Absicht zur Verfügung stellt. Der Konversationskontext umfasst vom Benutzer bereitgestellte Slot-Daten, von der Client-Anwendung festgelegte Anforderungsattribute und Sitzungsattribute, die von der Client-Anwendung und den Lambda-Funktionen erstellt werden.

Themen

- [Den Absichtskontext für Ihren Lex V2-Bot festlegen](#)
- [Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot](#)
- [Sitzungsattribute für Ihren Lex V2-Bot festlegen](#)
- [Einstellung von Anforderungsattributen für Ihren Lex V2-Bot](#)
- [Einstellung des Sitzungs-Timeouts](#)
- [Austausch von Informationen zwischen Absichten mit Ihrem Lex V2-Bot](#)
- [Einstellen komplexer Attribute in Ihrem Lex V2-Bot](#)

Den Absichtskontext für Ihren Lex V2-Bot festlegen

Sie können Absichtserklärungen basierend auf dem Kontext auslösen. Ein Kontext ist eine Statusvariable, die einer Absicht zugeordnet werden kann, wenn Sie einen Bot definieren. Sie konfigurieren die Kontexte für eine Absicht, wenn Sie die Absicht mithilfe der Konsole oder mithilfe der [CreateIntent](#) Operation erstellen. Sie können den Kontext nur im Gebietsschema Englisch (US) (en-US) verwenden.

Es gibt zwei Arten von Beziehungen für Kontexte: Ausgabekontexte und Eingabekontexte. Ein Ausgabekontext wird aktiv, wenn eine zugehörige Absicht erfüllt ist. In der Antwort der [RecognizeUtterance](#) Operation [RecognizeText](#) oder wird ein Ausgabekontext an Ihre Anwendung zurückgegeben, und er wird für die aktuelle Sitzung festgelegt. Nachdem ein Kontext aktiviert wurde, bleibt er für die Anzahl der Runden oder das Zeitlimit aktiv, das bei der Definition des Kontexts konfiguriert wurde.

Ein Eingabekontext legt Bedingungen fest, unter denen eine Absicht erkannt werden kann. Eine Absicht kann während einer Konversation nur erkannt werden, wenn alle ihre Eingabekontexte aktiv sind. Eine Absicht ohne Eingabekontexte kann immer anerkannt werden.

verwaltet automatisch den Lebenszyklus von Kontexten, die durch die Erfüllung von Absichten mit Ausgabekontexten aktiviert werden. Sie können aktive Kontexte auch in einem Aufruf der [RecognizeUtterance](#) Operation [RecognizeText](#) oder festlegen.

Sie können den Kontext einer Konversation auch mithilfe der Lambda-Funktion für die Absicht festlegen. Der Ausgabekontext von wird an das Eingabeereignis der Lambda-Funktion gesendet. Die Lambda-Funktion kann in ihrer Antwort Kontexte senden. Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion in deinen Bot integrieren](#).

Nehmen wir zum Beispiel an, Sie möchten einen Mietwagen buchen, der so konfiguriert ist, dass er einen Ausgabekontext namens „book_car_filled“ zurückgibt. Wenn die Absicht erfüllt ist, wird die Ausgabekontextvariable „book_car_filled“ festgelegt. Da es sich bei „book_car_fulfilled“ um einen aktiven Kontext handelt, wird eine Absicht, bei der der Kontext „book_car_filled“ als Eingabekontext festgelegt ist, nun bei der Erkennung berücksichtigt, sofern eine Benutzeräußerung als Versuch erkannt wird, diese Absicht auszulösen. Sie können dies für Absichten verwenden, die erst nach der Buchung eines Autos Sinn ergeben, z. B. das Versenden einer Quittung per E-Mail oder das Ändern einer Reservierung.

Ausgabekontext der Absichten für Ihren Lex V2-Bot

aktiviert die Ausgabekontexte einer Absicht, wenn die Absicht erfüllt ist. Mithilfe des Ausgabekontextes können Sie steuern, welche Absichten zur Weiterverfolgung der aktuellen Absicht in Frage kommen.

Jeder Kontext hat eine Liste von Parametern, die in der Sitzung verwaltet werden. Die Parameter sind die Slot-Werte für die erfüllte Absicht. Sie können diese Parameter verwenden, um Slot-Werte für andere Zwecke vorab auszufüllen. Weitere Informationen finden Sie unter [Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot](#)

Sie konfigurieren den Ausgabekontext, wenn Sie eine Absicht mit der Konsole oder mit der [CreateIntent](#) Operation erstellen. Sie können eine Absicht mit mehr als einem Ausgabekontext konfigurieren. Wenn die Absicht erfüllt ist, werden alle Ausgabekontexte aktiviert und in der [RecognizeUtterance](#) Antwort [RecognizeText](#) oder zurückgegeben.

Wenn Sie einen Ausgabekontext definieren, definieren Sie auch seine Gültigkeitsdauer, also die Dauer oder Anzahl der Runden, ab denen der Kontext in Antworten enthalten ist. Ein Turn ist eine Anfrage von Ihrer Anwendung an. Sobald die Anzahl der Runden oder die Zeit abgelaufen ist, ist der Kontext nicht mehr aktiv.

Ihre Anwendung kann den Ausgabekontext nach Bedarf verwenden. Ihre Anwendung kann den Ausgabekontext beispielsweise für folgende Zwecke verwenden:

- Ändern Sie das Verhalten der Anwendung auf der Grundlage des Kontextes. Beispielsweise könnte eine Reiseanwendung für den Kontext „book_car_filled“ eine andere Aktion als für „rental_hotel_filled“ haben.
- Gibt den Ausgabekontext als Eingabekontext für die nächste Äußerung zurück. Wenn die Äußerung als Versuch erkannt wird, eine Absicht hervorzurufen, verwendet sie den Kontext, um

die Absichten, die zurückgegeben werden können, auf solche mit dem angegebenen Kontext zu beschränken.

Geben Sie den Kontext der Absichten für Ihren Lex V2-Bot ein

Sie legen einen Eingabekontext fest, um die Punkte in der Konversation zu begrenzen, an denen die Absicht erkannt wird. Absichten ohne Eingabekontext können immer erkannt werden.

Sie legen die Eingabekontexte fest, auf die eine Absicht reagiert, indem Sie die Konsole oder den `CreateIntent` Vorgang verwenden. Eine Absicht kann mehr als einen Eingabekontext haben.

Bei einer Absicht mit mehr als einem Eingabekontext müssen alle Kontexte aktiv sein, um die Absicht auszulösen. Sie können einen Eingabekontext festlegen, wenn Sie die [PutSession](#) Operation [RecognizeTextRecognizeUtterance](#), oder aufrufen.

Sie können die Slots so konfigurieren, dass sie Standardwerte aus dem aktuell aktiven Kontext übernehmen. Standardwerte werden verwendet, wenn eine neue Absicht erkannt, aber kein Slot-Wert empfangen wird. Sie geben den Kontextnamen und den Slot-Namen im Formular an `#context-name.parameter-name`, wenn Sie den Slot definieren. Weitere Informationen finden Sie unter [Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot](#).

Verwenden von Standard-Slot-Werten in Intents für Ihren Lex V2-Bot

Wenn Sie einen Standardwert verwenden, geben Sie eine Quelle für einen Slot-Wert an, der für neue Absichten gefüllt werden soll, wenn durch die Benutzereingabe kein Slot bereitgestellt wird. Bei dieser Quelle kann es sich um vorherige Dialogfelder, Anfragen oder Sitzungsattribute oder um einen festen Wert handeln, den Sie bei der Erstellung festlegen.

Sie können Folgendes als Quelle für Ihre Standardwerte verwenden.

- Vorheriger Dialog (Kontexte) — `#context-name.parameter-name`
- Sitzungsattribute — `[Attributname]`
- Attribute anfordern — `<attribute-name>`
- Fester Wert — Jeder Wert, der nicht mit dem vorherigen Wert übereinstimmt

Wenn Sie die [CreateIntent](#) Operation verwenden, um Slots zu einer Absicht hinzuzufügen, können Sie eine Liste mit Standardwerten hinzufügen. Standardwerte werden in der Reihenfolge verwendet, in

der sie aufgelistet sind. Nehmen wir zum Beispiel an, Sie haben eine Absicht mit einem Slot mit der folgenden Definition:

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

Wenn die Absicht erkannt wird, wird der Wert des Slots mit dem Namen reservation-start-date "" auf einen der folgenden Werte gesetzt.

1. Wenn der Kontext book-car-fulfilled "" aktiv ist, wird der Wert des Parameters „StartDate“ als Standardwert verwendet.
2. Wenn der Kontext book-car-fulfilled "" nicht aktiv ist oder wenn der Parameter „StartDate“ nicht festgelegt ist, wird der Wert des Sitzungsattributs reservationStartDate "" als Standardwert verwendet.
3. Wenn keiner der ersten beiden Standardwerte verwendet wird, hat der Slot keinen Standardwert und ruft wie gewohnt einen Wert ab.

Wenn ein Standardwert für den Slot verwendet wird, wird der Slot nicht ausgelöst, auch wenn er erforderlich ist.

Sitzungsattribute für Ihren Lex V2-Bot festlegen

Sitzungsattribute enthalten anwendungsspezifische Informationen, die während einer Sitzung zwischen einem Bot und einer Client-Anwendung ausgetauscht werden. übergibt Sitzungsattribute an

alle für einen Bot konfigurierten Lambda-Funktionen. Wenn eine Lambda-Funktion Sitzungsattribute hinzufügt oder aktualisiert, gibt sie die neuen Informationen zurück an die Client-Anwendung.

Verwenden Sie Sitzungsattribute in Ihren Lambda-Funktionen, um einen Bot zu initialisieren und Eingabeaufforderungen und Antwortkarten anzupassen. Zum Beispiel:

- **Initialisierung** — Bei einem Bot zur Pizzabestellung übergibt die Client-Anwendung den Standort des Benutzers als Sitzungsattribut beim ersten Aufruf der Operation oder [RecognizeText](#)/[RecognizeUtterance](#). Beispiel, "Location": "111 Maple Street". Die Lambda-Funktion verwendet diese Informationen, um die nächstgelegene Pizzeria zu finden, in der die Bestellung aufgegeben werden kann.
- **Ansagen Personalize** — Konfigurieren Sie Eingabeaufforderungen und Antwortkarten so, dass sie sich auf Sitzungsattribute beziehen. Zum Beispiel: „Hey [FirstName], welche Toppings hättest du gern?“ Wenn Sie den Vornamen des Benutzers als Sitzungsattribut (`{"FirstName": "Vivian"}`) übergeben, ersetzt Amazon Lex den Platzhalter durch den Namen. Es sendet dann eine personalisierte Aufforderung an den Benutzer: „Hey Vivian, welche Beläge möchtest du?“

Sitzungsattribute bleiben für die Dauer der Sitzung erhalten. speichert diese bis zum Ende der Sitzung in einem verschlüsselten Datenspeicher. Der Client kann Sitzungsattribute in einer Anfrage erstellen, indem er entweder die [RecognizeUtterance](#) Operation [RecognizeText](#) oder aufruft, wobei das `sessionAttributes` Feld auf einen Wert gesetzt ist. Eine Lambda-Funktion kann in einer Antwort ein Sitzungsattribut erstellen. Nachdem der Client oder eine Lambda-Funktion ein Sitzungsattribut erstellt hat, wird der gespeicherte Attributwert immer dann verwendet, wenn die Client-Anwendung kein `sessionAttribute` Feld in eine Anfrage einbezieht.

Angenommen, Sie haben zwei Sitzungsattribute `{"x": "1", "y": "2"}`. Wenn der Client die `RecognizeUtterance` Operation `RecognizeText` oder aufruft, ohne das `sessionAttributes` Feld anzugeben, ruft er die Lambda-Funktion mit den gespeicherten Sitzungsattributen (`{"x": "1", "y": "2"}`) auf. Wenn die Lambda-Funktion keine Sitzungsattribute zurückgibt, werden die gespeicherten Sitzungsattribute an die Client-Anwendung zurückgegeben.

Wenn entweder die Client-Anwendung oder eine Lambda-Funktion Sitzungsattribute übergibt, werden die gespeicherten Sitzungsattribute aktualisiert. Wird ein bestehender Wert wie `{"x": "2"}` übergeben, wird der gespeicherte Wert aktualisiert. Wenn Sie einen neuen Satz Sitzungsattribute übergeben, wie z. B. `{"z": "3"}`, werden die vorhandenen Werte entfernt, und nur der neue Wert wird beibehalten. Wird eine leere Zuordnung, `{}`, übergeben, werden die gespeicherten Werte gelöscht.

Um Sitzungsattribute an zu senden, erstellen Sie eine string-to-string Zuordnung der Attribute. Das folgende Beispiel zeigt, wie Sitzungsattribute zugeordnet werden:

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

Für den RecognizeText Vorgang fügen Sie die Map mithilfe des sessionAttributes Felds der sessionState Struktur wie folgt in den Hauptteil der Anfrage ein:

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

Für die RecognizeUtterance Operation kodieren Sie die Map mit Base64 und senden sie dann als Teil des x-amz-lex-session-state Headers.

Wenn Sie binäre oder strukturierte Daten in einem Sitzungsattribut übermitteln, müssen Sie die Daten zunächst in eine einfache Zeichenfolge transformieren. Weitere Informationen finden Sie unter [Einstellen komplexer Attribute in Ihrem Lex V2-Bot](#).

Einstellung von Anforderungsattributen für Ihren Lex V2-Bot

Anforderungsattribute enthalten anforderungsspezifische Informationen und gelten nur für die aktuelle Anforderung. Eine Clientanwendung sendet diese Informationen an . Verwenden Sie Anforderungsattribute zur Weitergabe von Informationen, die nicht während der ganzen Sitzung erhalten bleiben müssen. Sie können eigene Anforderungsattribute erstellen oder vordefinierte verwenden. Zum Senden von Anforderungsattributen nutzen Sie den x-amz-lex-request-attributes-Header in einem [RecognizeUtterance](#) oder das requestAttributes-Feld in einer [RecognizeText](#)-Anforderung. Da Anforderungsattribute nicht wie Sitzungsattribute anforderungsübergreifend erhalten bleiben, werden sie nicht in RecognizeUtterance- oder RecognizeText-Antworten zurückgegeben.

 Note

Nutzen Sie Sitzungsattribute, wenn Sie möchten, dass Informationen anforderungsübergreifend erhalten bleiben.

Einstellung benutzerdefinierter Anforderungsattribute für jede Lex V2-Bot-Anfrage

Ein benutzerdefiniertes Anforderungsattribut besteht aus Daten, die Sie bei jeder Anforderung an Ihren Bot senden. Sie senden die Informationen im `amz-lex-request-attributes`-Header einer `RecognizeUtterance`-Anforderung oder im Feld `requestAttributes` einer `RecognizeText`-Anforderung.

Um Anforderungsattribute an zu senden, erstellen Sie eine string-to-string Zuordnung der Attribute. Das folgende Beispiel zeigt, wie Anforderungsattribute zugewiesen werden:

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

Bei der `PostText`-Operation fügen Sie die Zuordnung wie folgt mittels des Felds `requestAttributes` in den Text der Anforderung ein:

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

Bei der `PostContent`-Operation kodieren Sie die Zuordnung mit base64 und senden diese als `x-amz-lex-request-attributes`-Header.

Wenn Sie binäre oder strukturierte Daten in einem Anforderungsattribut übermitteln, müssen Sie die Daten zunächst in eine einfache Zeichenfolge transformieren. Weitere Informationen finden Sie unter [Einstellen komplexer Attribute in Ihrem Lex V2-Bot](#).

Amazon Lex V2 bietet vordefinierte Anforderungsattribute zur Verwaltung der Art und Weise, wie Informationen verarbeitet werden, die an Ihren Bot gesendet werden. Die Attribute bleiben nicht für

die gesamte Sitzung erhalten, daher müssen Sie die vordefinierten Attribute in jede Anforderung mit einschließen. Alle vordefinierten Attribute befinden sich im Namespace `x-amz-lex:`.

Zusätzlich zu den folgenden vordefinierten Attributen bietet Amazon Lex vordefinierte Attribute für Messaging-Plattformen. Eine Liste dieser Attribute finden Sie unter Bereitstellen eines Amazon Lex Lex-Bot auf einer Messaging-Plattform.

Einrichten des Antworttyps

Wenn Sie über zwei Client-Anwendungen mit unterschiedlichen Funktionen verfügen, müssen Sie das Format der Mitteilungen in einer Antwort möglicherweise einschränken. Beispielsweise möchten Sie die an einen Web-Client gesendeten Mitteilungen auf Klartext beschränken, einen mobilen Client jedoch befähigen, sowohl Klartext als auch Speech Synthesis Markup Language (SSML) zu verwenden. Verwenden Sie das Anforderungsattribut `x-amz-lex: accept-content-types`, um das Format der von den PostText Operationen PostContent und zurückgegebenen Nachrichten festzulegen.

Sie können das Attribut auf eine beliebige Kombination der folgenden Mitteilungstypen festlegen:

- `PlainText`— Die Nachricht enthält einfachen UTF-8-Text.
- `SSML`— Die Nachricht enthält Text, der für die Sprachausgabe formatiert ist.
- `CustomPayload`— Die Nachricht enthält ein benutzerdefiniertes Format, das Sie für Ihren Kunden erstellt haben. Sie können die Nutzlast definieren, um den Anforderungen Ihrer Anwendung zu entsprechen.

Amazon Lex V2 gibt nur Nachrichten mit dem angegebenen Typ im Nachrichtenfeld der Antwort zurück. Sie können mehr als einen Wert festlegen, indem Sie die einzelnen Werte durch Komma trennen. Wenn Sie Mitteilungsgruppen verwenden, muss jede Mitteilungsgruppe mindestens eine Mitteilung des angegebenen Typs enthalten. Andernfalls erhalten Sie den Fehler `NoUsableMessageException`. Weitere Informationen finden Sie unter Nachrichtengruppen.

Einstellung vordefinierter Anforderungsattribute in Ihrem Lex V2-Bot

Amazon Lex V2 bietet vordefinierte Anforderungsattribute zur Verwaltung der Art und Weise, wie Informationen verarbeitet werden, die an Ihren Bot gesendet werden. Die Attribute bleiben nicht für die gesamte Sitzung erhalten, daher müssen Sie die vordefinierten Attribute in jede Anforderung mit einschließen. Alle vordefinierten Attribute befinden sich im Namespace `x-amz-lex:`.

Deaktivieren von Intent-Switches in Ihrem Lex V2-Bot

Verwenden Sie das Anforderungsattribut, um zu steuern, ob Benutzer während der Bestätigung von Absichten oder der Slot-Erkennung zwischen den Absichten wechseln können. `x-amz-lex:intent-switch` Wenn dieses Attribut auf `DISABLE` gesetzt ist, verhindert es, dass Benutzer eine andere Absicht auslösen, während sie gerade dabei sind, den aktuellen Absichtsfluss abzuschließen.

Wenn ein Benutzer beispielsweise gerade einen Flug bucht und zur Eingabe von Flugdaten aufgefordert wird, werden Äußerungen wie „Wetter überprüfen“ oder „Hotel buchen“ — die normalerweise andere Absichten auslösen können — ignoriert, sodass sichergestellt wird, dass sich die Konversation weiterhin auf den aktuellen Buchungsprozess konzentriert.

Einstellung des Sitzungs-Timeouts

Amazon Lex speichert Kontextinformationen — Slot-Daten und Sitzungsattribute — bis eine Konversationssitzung endet. Um zu steuern, wie lange eine Sitzung für einen Bot dauert, legen Sie einen Zeitüberschreitungswert für die Sitzung fest. Standardmäßig dauert eine Sitzung 5 Minuten, Sie können aber auch eine Dauer zwischen 0 und 1 440 Minuten (24 Stunden) angeben.

Angenommen, Sie erstellen einen `ShoeOrdering`-Bot zur Unterstützung von Absichten wie `OrderShoes` und `GetOrderStatus`. Wenn Amazon Lex feststellt, dass der Benutzer beabsichtigt, Schuhe zu bestellen, fragt es nach Slot-Daten. Sie fragt beispielsweise nach Schuhgröße, Farbe, Marke etc. Wenn der Benutzer einige der Slot-Daten angibt, den Schuhkauf aber nicht abschließt, merkt sich Amazon Lex alle Slot-Daten und Sitzungsattribute für die gesamte Sitzung. Wenn der Benutzer vor Ablauf zur Sitzung zurückkehrt, kann er die verbleibenden Slot-Daten angeben und den Kauf abschließen.

Sie geben die Sitzungs-Zeitüberschreitung in der -Konsole an, wenn Sie einen Bot erstellen. Mit der AWS-Befehlszeilenschnittstelle (AWS-CLI) oder API legen Sie das Timeout fest, wenn Sie einen Bot mit dem [CreateBot](#)-Vorgang erstellen, indem Sie das Feld [IdleSession TTL](#) in Seconds festlegen.

Austausch von Informationen zwischen Absichten mit Ihrem Lex V2-Bot

unterstützt den Informationsaustausch zwischen Absichten. Verwenden Sie Ausgabekontexte oder Sitzungsattribute, um Inhalte zwischen Absichten auszutauschen.

Um Ausgabekontexte zu verwenden, definieren Sie einen Ausgabekontext, wenn Sie eine Absicht erstellen oder aktualisieren. Wenn die Absicht erfüllt ist, enthalten Antworten von Amazon Lex V2

den Kontext und die Slot-Werte aus der Absicht als Kontextparameter. Sie können diese Parameter als Standardwerte in nachfolgenden Absichten oder in Ihrem Anwendungscode oder in Lambda-Funktionen verwenden.

Um Sitzungsattribute zu verwenden, legen Sie die Attribute in Ihrem Lambda- oder Anwendungscode fest. Angenommen, ein Benutzer des Bots `ShoeOrdering` bestellt Schuhe. Der Bot schaltet sich in die Konversation mit dem Benutzer ein und erfasst Slot-Daten wie Schuhgröße, Farbe und Marke. Wenn der Benutzer eine Bestellung aufgibt, legt die Lambda-Funktion, die die Bestellung erfüllt, das `orderNumber` Sitzungsattribut fest, das die Bestellnummer enthält. Der Benutzer verwendet die Absicht `GetOrderStatus`, um den Status der Bestellung zu erhalten. Der Bot kann den Benutzer nach Slot-Daten fragen, wie beispielsweise Bestellnummer oder -datum. Wenn der Bot die notwendigen Informationen hat, gibt er den Status der Bestellung zurück.

Wenn Sie der Meinung sind, dass Ihre Benutzer möglicherweise Absichten während der Sitzung ändern, können Sie Ihren Bot so konfigurieren, dass der Status der letzten Bestellung wiedergegeben wird. Anstatt den Benutzer erneut nach Bestellinformationen zu fragen, nutzen Sie das Sitzungsattribut `orderNumber`, um Informationen absichtsübergreifend zu teilen und die Absicht `GetOrderStatus` zu erfüllen. Der Bot führt dies durch, indem er den Status der letzten vom Benutzer aufgegebenen Bestellung zurückgibt.

Einstellen komplexer Attribute in Ihrem Lex V2-Bot

Sitzungs- und Anforderungsattribute sind string-to-string Zuordnungen von Attributen und Werten. In vielen Fällen können Sie mit der Zeichenfolgen-Zuordnung Attributwerte zwischen Ihrer Clientanwendung und einem Bot übertragen. In einigen Fällen müssen Sie jedoch möglicherweise binäre Daten oder eine komplexe Struktur übertragen, die schwer in eine Zeichenfolgen-Zuordnung konvertiert werden kann. Das folgende JSON-Objekt stellt beispielsweise ein Array der drei beliebtesten Städte in den USA dar:

```
{
  "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"
      }
    }
  ]
}
```

Dieses Datenarray lässt sich nicht gut in eine string-to-string Map übertragen. In diesem Fall können Sie ein Objekt in eine einfache Zeichenfolge transformieren, sodass Sie sie mit den Operationen [RecognizeText](#) und [RecognizeUtterance](#) an Ihren Bot senden können.

Wenn Sie beispielsweise verwenden, können Sie die `JSON.stringify` Operation verwenden JavaScript, um ein Objekt in JSON zu konvertieren, und die `JSON.parse` Operation, um JSON-Text in ein JavaScript Objekt zu konvertieren:

```
// 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);
```

Um Attribute mit der `RecognizeUtterance` Operation zu senden, müssen Sie die Attribute base64-kodieren, bevor Sie sie dem Anforderungsheader hinzufügen, wie im folgenden JavaScript Code gezeigt:

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

Sie können binäre Daten an die Operationen `RecognizeText` und `RecognizeUtterance` senden, indem Sie die Daten zunächst in eine mit base64 kodierte Zeichenfolge konvertieren und diese Zeichenfolge dann als Wert in den Sitzungsattributen übermitteln:

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

Grundlegendes zu Amazon Lex V2-Bot-Sitzungen

Wenn ein Benutzer eine Konversation mit Ihrem Bot beginnt, erstellt Amazon Lex V2 eine Sitzung. Die zwischen Ihrer Anwendung und Amazon Lex V2 ausgetauschten Informationen bilden den Sitzungsstatus für die Konversation. Wenn Sie eine Anfrage stellen, wird die Sitzung durch eine von Ihnen angegebene Kennung identifiziert. Weitere Informationen zur Sitzungs-ID finden Sie in dem `sessionId` Feld in der [RecognizeUtterance](#) Operation [RecognizeText](#)or.

Sie können den zwischen Ihrer Anwendung und Ihrem Bot gesendeten Sitzungsstatus ändern. Beispielsweise können Sie Sitzungsattribute erstellen und ändern, die benutzerdefinierte Informationen zur Session enthalten. Außerdem können Sie den Gesprächsablauf ändern, indem Sie den Dialogkontext für die Interpretation der nächste Äußerung festlegen.

Es gibt drei Möglichkeiten, den Sitzungsstatus zu aktualisieren.

- Übergeben Sie die Sitzungsinformationen inline als Teil eines Aufrufs an die `RecognizeUtterance` Operation `RecognizeText` oder.
- Verwenden Sie eine Lambda-Funktion mit der `RecognizeUtterance` Operation `RecognizeText` oder, die nach jeder Runde der Konversation aufgerufen wird. Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion in deinen Bot integrieren](#). Die andere Möglichkeit besteht darin, die Amazon Lex V2-Runtime-API in Ihrer Anwendung zu verwenden, um Änderungen am Sitzungsstatus vorzunehmen.
- Verwenden Sie Operationen, mit denen Sie Sitzungsinformationen für eine Konversation mit Ihrem Bot verwalten können. Hierzu gehören die Operation [PutSession](#), die Operation [GetSession](#) und die Operation [DeleteSession](#). Sie können mit diesen Operationen Informationen über den Status der Sitzung Ihres Benutzers mit Ihrem Bot anfordern und eine differenzierte Kontrolle über den Status erlangen.

Verwenden Sie die Operation `GetSession`, wenn Sie den aktuellen Status der Sitzung anfordern möchten. Der Vorgang gibt den aktuellen Status der Sitzung zurück, einschließlich des Status des Dialogs mit Ihrem Benutzer, aller festgelegten Sitzungsattribute und Slot-Werte für die aktuelle Absicht sowie aller anderen Absichten, die Amazon Lex V2 als mögliche Absichten identifiziert hat, die der Äußerung des Benutzers entsprechen.

Die Operation `PutSession` ermöglicht es Ihnen, den aktuellen Sitzungsstatus direkt zu bearbeiten. Sie können die Sitzung festlegen, einschließlich der Art der Dialogaktion, die der Bot als Nächstes ausführt, und der Nachrichten, die Amazon Lex V2 an den Benutzer sendet. Dadurch haben Sie Kontrolle über den Gesprächsablauf mit dem Bot. Stellen Sie das `type` Dialogaktionsfeld `Delegate` auf ein, damit Amazon Lex V2 die nächste Aktion für den Bot bestimmt.

Sie können mit der Operation `PutSession` eine neue Sitzung mit einem Bot erstellen und die Absicht festlegen, mit der der Bot beginnen soll. Sie können mit der Operation `PutSession` auch von einer Absicht zu einer anderen wechseln. Wenn Sie eine Sitzung erstellen oder die Absicht ändern, können Sie auch den Sitzungsstatus, wie z. B. Slot-Werte und Sitzungsattribute, festlegen. Wenn die neue Absicht abgeschlossen ist, haben Sie die Möglichkeit, die vorherige Absicht neu zu starten.

Die Antwort von der Operation `PutSession` enthält die gleichen Informationen wie die von der Operation `RecognizeUtterance`. Sie können diese Informationen, genauso wie die Antwort von der Operation `RecognizeUtterance`, verwenden, um vom Benutzer die nächste Teilinformation anzufordern.

Sie können mit der Operation `DeleteSession` eine vorhandene Sitzung entfernen und mit einer neuen Sitzung ganz von vorne beginnen. Wenn Sie beispielsweise Ihren Bot testen, können Sie mit der Operation `DeleteSession` Testsitzungen von Ihrem Bot entfernen.

Die Sitzungsvorgänge funktionieren mit Ihren Fulfillment-Lambda-Funktionen. Wenn Ihre Lambda-Funktion beispielsweise `Failed` als Erfüllungsstatus zurückkehrt, können Sie die `PutSession` Operation verwenden, um den Aktionstyp des Dialogs auf `close` und `fulfillmentState` auf festzulegen, um den Erfüllungsschritt erneut `ReadyForFulfillment` zu versuchen.

Es folgen einige Aufgaben, die Sie mit den Sitzungsoperationen ausführen können:

- Veranlassen des Bots zum Starten einer Konversation, anstatt auf den Benutzer zu warten.
- Wechseln von Absichten während einer Konversation.
- Zurückkehren zur einer vorherigen Absicht.
- Starten oder Neustarten einer Konversation während der Interaktion.
- Validieren von Slot-Werten und Veranlassen des Bots, für ungültige Werte neue Werte anzufordern.

Jede dieser Aufgaben wird im Folgenden beschrieben.

Eine neue Sitzung starten

Wenn Sie möchten, dass der Bot die Konversation mit Ihrem Benutzer startet, können Sie dazu Operation `PutSession` verwenden.

- Erstellen Sie eine Begrüßungsabsicht ohne Slots und eine abschließende Nachricht, die den Benutzer auffordert, eine Absicht zu nennen. Beispiel: „Was möchten Sie bestellen? Sie können Folgendes sagen: 'Ein Getränk bestellen' oder 'Eine Pizza bestellen'.“
- Aufrufen der `PutSession`-Operation. Legen Sie als Absichtsnamen den Namen Ihrer Begrüßungsabsicht und als Dialogaktion `Delegate` fest.
- Amazon Lex antwortet mit der Aufforderung von `Your Welcome Intent`, die Konversation mit Ihrem Benutzer zu beginnen.

Absicht wechseln

Sie können mit der Operation `PutSession` von einer Absicht zu einer anderen wechseln. Sie können mit ihr auch zu einer vorherigen Absicht zurückwechseln. Sie können mit der Operation `PutSession` Sitzungsattribute oder Slot-Werte für die neue Absicht einstellen.

- Aufrufen der `PutSession`-Operation. Legen Sie als Absichtsnamen den Namen der neuen Absicht und als Dialogaktion `Delegate` fest. Sie können auch alle für die neue Absicht erforderlichen Slot-Werte oder Sitzungsattribute festlegen.
- Amazon Lex beginnt eine Konversation mit dem Benutzer, der die neue Absicht verwendet.

Wiederaufnahme einer vorherigen Absicht

Um eine vorherige Absicht wieder aufzunehmen, verwenden Sie die `GetSession` Operation, um den Status der Absicht abzurufen, die erforderliche Interaktion durchzuführen und dann die `PutSession` Operation zu verwenden, um die Absicht auf den vorherigen Dialogstatus zu setzen.

- Aufrufen der `GetSession`-Operation. Speichern Sie den Status der Absicht.
- Führen Sie eine weitere Interaktion durch, z. B. die Erfüllung einer anderen Absicht.
- Rufen Sie die `PutSession` Operation auf, indem Sie die Informationen verwenden, die für die vorherige Absicht gespeichert wurden. Dadurch gelangt der Benutzer wieder zur vorherigen Absicht an der gleichen Stelle im Gespräch.

In einigen Fällen kann es erforderlich sein, dass die Konversation Ihres Benutzers mit Ihrem Bot fortgesetzt wird. Angenommen, Sie haben einen Kundenservice-Bot erstellt. Ihre Anwendung stellt fest, dass der Benutzer mit einem Kundendienstmitarbeiter sprechen muss. Nach der Unterhaltung mit dem Benutzer kann der Mitarbeiter das Gespräch mit den vom ihm erfassten Informationen wieder an den Bot weiterleiten.

Um eine Sitzung fortzusetzen, verwenden Sie Schritte ähnlich wie diese:

- Ihre Anwendung stellt fest, dass der Benutzer mit einem Kundendienstmitarbeiter sprechen muss.
- Fordern Sie mit der Operation `GetSession` den aktuellen Dialogstatus der Absicht an.
- Der Kundendienstmitarbeiter spricht mit dem Benutzer und löst das Problem.
- Legen Sie mit der Operation `PutSession` den Dialogstatus der Absicht fest. Hierzu gehören möglicherweise Festlegen von Slot-Werten, Einstellen von Sitzungsattributen oder Ändern der Absicht.
- Der Bot setzt die Konversation mit dem Benutzer fort.

Slot-Werte werden validiert

Sie können Antworten an Ihren Bot mit Ihrer Client-Anwendung validieren. Wenn die Antwort nicht gültig ist, können Sie mit der Operation `PutSession` eine neue Antwort von Ihrem Benutzer anfordern. Angenommen, Ihr Bot zur Aufnahme von Blumenbestellungen kann nur Rosen, Tulpen und Lilien verkaufen. Wenn der Benutzer Nelken bestellt, kann Ihre Anwendung wie folgt vorgehen:

- Untersuchen des Slot-Wertes, der von der Antwort `PostText` oder `PostContent` zurückgegeben wird.
- Wenn der Slot-Wert nicht gültig ist, Aufrufen der Operation `PutSession`. Ihre Anwendung sollte den Slot-Wert löschen, das Feld `slotToElicit` festlegen und den Wert `dialogAction.type` auf `elicitSlot` einstellen. Optional können Sie die `messageFormat` Felder `message` und festlegen, wenn Sie die Nachricht ändern möchten, die Amazon Lex verwendet, um den Slot-Wert zu ermitteln.

Eine AWS Lambda Funktion in deinen Bot integrieren

Mit [AWS Lambda](#) Funktionen können Sie das Verhalten Ihres Amazon Lex V2-Bots durch benutzerdefinierte Funktionen, die Sie definieren, erweitern und besser steuern. Amazon Lex V2 verwendet eine Lambda-Funktion pro Bot-Alias pro Sprache statt einer Lambda-Funktion für jede Absicht. Bevor Sie beginnen, legen Sie fest, aus welchen Feldern im [Eingabeereignis](#) Sie Informationen ziehen möchten und welche Felder in der [Antwort](#) Sie bearbeiten und von Ihrer Lambda-Funktion zurückgeben möchten.

Gehen Sie wie folgt vor, um eine Lambda-Funktion in Ihren Amazon Lex V2-Bot zu integrieren:

1. [Erstellen Sie eine AWS Lambda Funktion](#) in der Programmiersprache Ihrer Wahl und schreiben Sie Ihr Skript.
2. Stellen Sie sicher, dass die Funktion eine Struktur zurückgibt, die dem [Antwortformat](#) entspricht.
3. Stellen Sie die Lambda-Funktion bereit.
4. Ordnen Sie die Lambda-Funktion einem Amazon Lex V2-Bot-Alias für die [Konsolen](#) - oder [API-Operationen](#) zu.
5. Wählen Sie die Konversationsphasen aus, in denen Sie Ihre Lambda-Funktion mit den [Konsolen](#) - oder [API-Operationen](#) aufrufen möchten.
6. Erstellen Sie Ihren Amazon Lex V2-Bot und testen Sie, ob die Lambda-Funktion wie vorgesehen funktioniert. [Debuggen](#) Sie Ihre Funktion mit Hilfe von Amazon CloudWatch.

Themen

- [AWS Lambda Eingabeereignisformat für Lex V2](#)
- [AWS Lambda Antwortformat für Lex V2](#)
- [Gemeinsame Strukturen in einer AWS Lambda Funktion für](#)
- [Eine AWS Lambda Funktion für Ihren Bot erstellen](#)
- [Debuggen einer Lambda-Funktion mithilfe von CloudWatch Logs](#)

AWS Lambda Eingabeereignisformat für Lex V2

Der erste Schritt bei der Integration einer Lambda-Funktion in Ihren Amazon Lex V2-Bot besteht darin, die Felder im Amazon Lex V2-Ereignis zu verstehen und die Informationen aus diesen Feldern zu ermitteln, die Sie beim Schreiben Ihres Skripts verwenden möchten. Das folgende JSON-Objekt

zeigt das allgemeine Format eines Amazon Lex V2-Ereignisses, das an eine Lambda-Funktion übergeben wird:

 Note

Das Eingabeformat kann sich ändern, ohne dass eine entsprechende Änderung an der vorgenommen wird `messageVersion`. Ihr Code sollte keinen Fehler auslösen, wenn neue Felder vorhanden sind.

```
{
  "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 Absicht 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 Absicht for details about the structure
    },
    "prompt": {
        "attempt": string
    }
},
"requestAttributes": {
    string: string,
    ...
},
"sessionState": {
    // see Status der Sitzung for details about the structure
},
"transcriptions": [
    {
        "transcription": string,
        "transcriptionConfidence": number,
        "resolvedContext": {
            "intent": string
        },
        "resolvedSlots": {
            slot name: {
                // see Slots for details about the structure
            },
            ...
        }
    },
    ...
]
}

```

Jedes Feld im Eingabeereignis wird unten beschrieben:

messageVersion

Die Version der Nachricht, die das Format der Ereignisdaten identifiziert, die in die Lambda-Funktion eingehen, und das erwartete Format der Antwort von einer Lambda-Funktion.

Note

Sie konfigurieren diesen Wert, wenn Sie eine Absicht definieren. In der aktuellen Implementierung unterstützt Amazon Lex V2 nur Nachrichtenversion 1.0. Daher nimmt die Konsole 1.0 als Standardwert an und zeigt die Mitteilungsversion nicht.

invocationSource

Der Code-Hook, der die Lambda-Funktion aufgerufen hat. Die folgenden Werte sind möglich:

DialogCodeHook— Amazon Lex V2 hat die Lambda-Funktion nach Eingabe des Benutzers aufgerufen.

FulfillmentCodeHook— Amazon Lex V2 hat die Lambda-Funktion aufgerufen, nachdem alle erforderlichen Slots gefüllt wurden und die Absicht zur Ausführung bereit ist.

Eingabemodus

Der Modus der Benutzeräußerung. Die möglichen Werte lauten wie folgt:

DTMF— Der Benutzer gibt die Äußerung über eine Touch-Tone-Tastatur ein (Dual Tone Multi-Frequency).

Speech— Der Benutzer sprach die Äußerung.

Text— Der Benutzer hat die Äußerung eingegeben.

responseContentType

Der Modus, in dem der Bot auf den Benutzer reagiert. `text/plain; charset=utf-8` gibt an, dass die letzte Äußerung geschrieben wurde, während ein Wert, der mit `audio` beginnt, angibt, dass die letzte Äußerung gesprochen wurde.

sessionId

Die alphanumerische Sitzungs-ID, die für die Konversation verwendet wird.

inputTranscript

Eine Transkription der Benutzereingabe.

- Bei der Texteingabe ist dies der Text, den der Benutzer eingegeben hat. Bei DTMF-Eingaben ist dies der Schlüssel, den der Benutzer eingegeben hat.
- Bei der Spracheingabe ist dies der Text, in den Amazon Lex V2 die Benutzeräußerung konvertiert, um eine Absicht auszulösen oder einen Slot zu füllen.

InvocationLabel

Ein Wert, der die Antwort angibt, die die Lambda-Funktion aufgerufen hat. Sie können Aufrufbezeichnungen für die erste Antwort, die Slots und die Bestätigungsantwort festlegen.

Bot

Informationen über den Bot, der die Anfrage bearbeitet hat, bestehend aus den folgenden Feldern:

- **id** — Die Kennung, die dem Bot zugewiesen wurde, als Sie ihn erstellt haben. Sie können die Bot-ID in der Amazon Lex V2-Konsole auf der Seite mit den Bot-Einstellungen sehen.
- **Name** — Der Name, den Sie dem Bot bei der Erstellung gegeben haben.
- **LocaleID** — Die Kennung des Gebietsschemas, das Sie für Ihren Bot verwendet haben. Eine Liste der Gebietsschemas finden Sie unter [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#)
- **Version** — Die Version des Bots, der die Anfrage bearbeitet hat.
- **aliasID** — Die Kennung, die dem Bot-Alias zugewiesen wurde, als Sie ihn erstellt haben. Sie können die Bot-Alias-ID in der Amazon Lex V2-Konsole auf der Seite Aliase sehen. Wenn Sie die Alias-ID nicht in der Liste sehen können, wählen Sie das Zahnradsymbol oben rechts und aktivieren Sie die Alias-ID.
- **AliasName** — Der Name, den Sie dem Bot-Alias gegeben haben.

Interpretationen

Eine Liste mit Informationen über Absichten, die Amazon Lex V2 für mögliche Übereinstimmungen mit der Äußerung des Benutzers hält. Jedes Element ist eine Struktur, die Informationen über die Übereinstimmung der Äußerung mit einer Absicht im folgenden Format bereitstellt:

```
{
  "intent": {
    // see Absicht 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
    }
  }
}
```

Die Felder innerhalb der Struktur lauten wie folgt:

- **Absicht** — Eine Struktur, die Informationen über die Absicht enthält. Einzelheiten [Absicht](#) zur Struktur finden Sie unter.
- **NLUConfidence** — Ein Wert, der angibt, wie sicher Amazon Lex V2 ist, dass die Absicht mit der Absicht des Benutzers übereinstimmt.
- **sentimentResponse** — Eine Analyse der Stimmung in der Antwort, die die folgenden Felder enthält:
 - **Stimmung** — Gibt an, ob die Stimmung der Äußerung, oder ist. POSITIVE NEGATIVE NEUTRAL MIXED
 - **SentimentScore** — Eine Struktur, die jedes Gefühl einer Zahl zuordnet, die angibt, wie sicher Amazon Lex V2 ist, dass die Äußerung dieses Gefühl vermittelt.
- **InterpretationSource** — Gibt an, ob ein Slot von Amazon Bedrock oder Amazon Bedrock gelöst wurde.

proposedNextState

Wenn die Lambda-Funktion das `dialogAction` von `sessionState` auf `setDelegate`, wird dieses Feld angezeigt und zeigt den Vorschlag von Amazon Lex V2 für den nächsten Schritt in der Konversation. Andernfalls hängt der nächste Status von den Einstellungen ab, die Sie in der Antwort Ihrer Lambda-Funktion zurückgeben. Diese Struktur ist nur vorhanden, wenn die beiden folgenden Aussagen zutreffen:

1. Der invocationSource Wert ist DialogCodeHook
2. Der vorhergesagte type von dialogAction istElicitSlot.

Sie können diese Informationen verwenden, um sie runtimeHints an der richtigen Stelle in der Konversation hinzuzufügen. [Verbesserung der Erkennung von Slot-Werten mit Laufzeithinweisen in der Konversation](#)Weitere Informationen finden Sie unter. proposedNextStateist eine Struktur, die die folgenden Felder enthält:

Die Struktur von proposedNextState ist wie folgt:

```
"proposedNextState": {
  "dialogAction": {
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
  },
  "intent": {
    // see Absicht for details about the structure
  },
  "prompt": {
    "attempt": string
  }
}
```

- DialogAction — Enthält Informationen zum nächsten Schritt, den Amazon Lex V2 vorschlägt. Die Felder in der Struktur lauten wie folgt:
 - slotToElicit— Der Slot, der als nächstes ausgewählt werden soll, wie von Amazon Lex V2 vorgeschlagen. Dieses Feld wird nur angezeigt, wenn dies der Fall isttype. ElicitSlot
 - type — Der nächste Schritt in der Konversation, wie von Amazon Lex V2 vorgeschlagen. Die folgenden Werte sind möglich:

Delegate— Amazon Lex V2 bestimmt die nächste Aktion.

ElicitIntent— Die nächste Aktion besteht darin, dem Benutzer eine Absicht zu entlocken.

ElicitSlot— Die nächste Aktion besteht darin, dem Benutzer einen Slot-Wert zu entlocken.

Close— Beendet den Prozess zur Erfüllung der Absicht und gibt an, dass der Benutzer nicht antworten wird.

ConfirmIntent— Die nächste Aktion besteht darin, den Benutzer zu fragen, ob die Slots korrekt sind und ob die Absicht zur Erfüllung bereit ist.

- **Absicht** — Die Absicht, die der Bot festgestellt hat und die der Benutzer zu erfüllen versucht. Einzelheiten [Absicht](#) zur Struktur finden Sie unter.
- **prompt** — Eine Struktur, die das Feld `attempt` enthält, das einem Wert zugeordnet ist, der angibt, wie oft Amazon Lex V2 den Benutzer zur Eingabe des nächsten Slots aufgefordert hat. Die möglichen Werte gelten `Initial` für den ersten Versuch und `Retry1`, `Retry2`, `Retry3`, `Retry4`, und `Retry5` für nachfolgende Versuche.

requestAttributes

Eine Struktur, die anforderungsspezifische Attribute enthält, die der Client in der Anfrage sendet. Verwenden Sie Anforderungsattribute zur Weitergabe von Informationen, die nicht während der ganzen Sitzung erhalten bleiben müssen. Wenn keine Anforderungsattribute vorhanden sind, ist der Wert Null. Weitere Informationen finden Sie unter [Einstellung von Anforderungsattributen für Ihren Lex V2-Bot](#).

SessionState

Der aktuelle Status der Konversation zwischen dem Benutzer und Ihrem Amazon Lex V2-Bot. Einzelheiten [Status der Sitzung](#) zur Struktur finden Sie unter.

Abschriften

Eine Liste von Transkriptionen, die Amazon Lex V2 für mögliche Übereinstimmungen mit der Äußerung des Benutzers hält. Weitere Informationen finden Sie unter [Verwenden Sie Konfidenzwerte für die Sprachtranskription, um die Konversation mit Ihrem Lex V2-Bot zu verbessern](#). Jedes Element ist ein Objekt mit dem folgenden Format, das Informationen über eine mögliche Transkription enthält:

```
{
  "transcription": string,
  "transcriptionConfidence": number,
  "resolvedContext": {
    "intent": string
  },
  "resolvedSlots": {
    slot name: {
      // see Slots for details about the structure
    }
  }
}
```

```

    },
    ...
  }
}

```

Die Felder werden im Folgenden beschrieben.

- **Transkription** — Eine Transkription, die Amazon Lex V2 als mögliche Übereinstimmung mit der Audioäußerung des Benutzers betrachtet.
- **TranscriptionConfidence** — Ein Wert, der angibt, wie sicher Amazon Lex V2 ist, dass die Absicht mit der Absicht des Benutzers übereinstimmt.
- **ResolvedContext** — Eine Struktur, die das Feld enthält, das der Absicht zugeordnet ist `intent`, auf die sich die Äußerung bezieht.
- **ResolvedSlots** — Eine Struktur, deren Schlüssel die Namen der einzelnen Slots sind, die durch die Äußerung aufgelöst werden. Jeder Slot-Name ist einer Struktur zugeordnet, die Informationen über diesen Slot enthält. Einzelheiten [Slots](#) zur Struktur finden Sie unter.

AWS Lambda Antwortformat für Lex V2

Der zweite Schritt bei der Integration einer Lambda-Funktion in Ihren Amazon Lex V2-Bot besteht darin, die Felder in der Lambda-Funktionsantwort zu verstehen und zu bestimmen, welche Parameter Sie manipulieren möchten. Das folgende JSON-Objekt zeigt das allgemeine Format einer Lambda-Antwort, die an Amazon Lex V2 zurückgegeben wird:

```

{
  "sessionState": {
    // see Status der Sitzung 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,
  ...
}
}

```

Jedes Feld in der Antwort wird unten beschrieben:

Sitzungsstatus

Der Status der Konversation zwischen dem Benutzer und Ihrem Amazon Lex V2-Bot, den Sie zurückgeben möchten. Einzelheiten [Status der Sitzung](#) zur Struktur finden Sie unter. Dieses Feld ist immer erforderlich.

messages

Eine Liste von Nachrichten, die Amazon Lex V2 für die nächste Runde der Konversation an den Kunden zurücksendet. Wenn `contentType` Sie, oder angeben `PlainTextCustomPayload`, schreiben Sie die Nachricht SSML, die Sie an den Kunden zurücksenden möchten, in das `content` Feld. Falls das von `contentType` Ihnen angegebene ist `ImageResponseCard`, geben Sie die Details der Karte in das `imageResponseCard` Feld ein. Wenn Sie keine Nachrichten bereitstellen, verwendet Amazon Lex V2 die entsprechende Nachricht, die bei der Erstellung des Bots definiert wurde.

Das `messages` Feld ist erforderlich, wenn `dialogAction.type` es `ElicitIntent` oder `isConfirmIntent`.

Jedes Element in der Liste ist eine Struktur im folgenden Format, die Informationen über eine Nachricht enthält, die an den Benutzer zurückgegeben werden soll. Ein Beispiel:

```

{
  "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
  "content": string,
  "imageResponseCard": {

```

```

    "title": string,
    "subtitle": string,
    "imageUrl": string,
    "buttons": [
        {
            "text": string,
            "value": string
        },
        ...
    ]
}

```

Im Folgenden finden Sie eine Beschreibung für jedes Feld:

- **contentType** — Der Typ der zu verwendenden Nachricht.

CustomPayload— Eine Antwortzeichenfolge, die Sie so anpassen können, dass sie Daten oder Metadaten für Ihre Anwendung enthält.

ImageResponseCard— Ein Bild mit Schaltflächen, die der Kunde auswählen kann. Weitere Informationen finden Sie unter [ImageResponseCard](#).

PlainText— Eine einfache Textzeichenfolge.

SSML— Eine Zeichenfolge, die die Speech Synthesis Markup Language enthält, um die Audioantwort anzupassen.

- **Inhalt** — Die Nachricht, die an den Benutzer gesendet werden soll. Verwenden Sie dieses Feld, wenn der Nachrichtentyp **PlainTextCustomPayload**, oder **istSSML**.
- **imageResponseCard**— Enthält die Definition der Antwortkarte, die dem Benutzer angezeigt werden soll. Verwenden Sie dieses Feld, wenn der Nachrichtentyp **istImageResponseCard**. Ordnet einer Struktur zu, die die folgenden Felder enthält:
 - **Titel** — Der Titel der Antwortkarte.
 - **Untertitel** — Die Aufforderung an den Benutzer, eine Schaltfläche auszuwählen.
 - **imageUrl** — Ein Link zu einem Bild für die Karte.
 - **buttons** — Eine Liste von Strukturen, die Informationen über eine Schaltfläche enthält. Jede Struktur enthält ein **text** Feld mit dem anzuzeigenden Text und ein **value** Feld mit dem Wert, der an Amazon Lex V2 gesendet werden soll, wenn der Kunde diese Schaltfläche auswählt. Sie können bis zu drei Schaltflächen hinzufügen.

requestAttributes

Eine Struktur, die anforderungsspezifische Attribute für die Antwort an den Kunden enthält. Weitere Informationen finden Sie unter [Einstellung von Anforderungsattributen für Ihren Lex V2-Bot](#). Dies ist ein optionales Feld.

Erforderliche Felder in der Antwort

Die Lambda-Antwort muss mindestens ein `sessionState` Objekt enthalten. Geben Sie darin ein `dialogAction` Objekt an und geben Sie das `type` Feld an. Je `type` nachdem `dialogAction`, welche Felder Sie angeben, gibt es möglicherweise weitere Pflichtfelder für die Lambda-Antwort. Diese Anforderungen werden zusammen mit einigen wenigen Anwendungsbeispielen wie folgt beschrieben:

Delegierter

Delegate lässt Amazon Lex V2 den nächsten Schritt bestimmen. Es sind keine weiteren Felder erforderlich.

```
{
  "sessionState": {
    "dialogAction": {
      "type": "Delegate"
    }
  }
}
```

ElicitIntent

ElicitIntent fordert den Kunden auf, eine Absicht zu äußern. Sie müssen mindestens eine Nachricht in das `messages` Feld eingeben, um eine Absicht zu erkennen.

```
{
  "sessionState": {
    "dialogAction": {
      "type": "ElicitIntent"
    },
  },
  "messages": [
    {
      "contentType": PlainText,
      "content": "How can I help you?"
    }
  ]
}
```

```
]
}
```

ElicitSlot

ElicitSlotfordert den Kunden auf, einen Slot-Wert anzugeben. Sie müssen den Namen des Slots in das slotToElicit Feld im dialogAction Objekt aufnehmen. Sie müssen auch den name von intent in das sessionState Objekt aufnehmen.

```
{
  "sessionState": {
    "dialogAction": {
      "slotToElicit": "OriginCity",
      "type": "ElicitSlot"
    },
    "intent": {
      "name": "BookFlight"
    }
  }
}
```

ConfirmIntent

ConfirmIntentbestätigt die Slot-Werte des Kunden und ob die Absicht bereit ist, erfüllt zu werden. Sie müssen das name sessionState Objekt und das intent noch slots zu bestätigende Objekt angeben. Sie müssen außerdem mindestens eine Nachricht in das messages Feld einfügen, um den Benutzer zur Bestätigung der Slot-Werte aufzufordern. Ihre Nachricht sollte mit „Ja“ oder „Nein“ beantwortet werden. Wenn der Benutzer mit „Ja“ antwortet, setzt Amazon Lex V2 confirmationState die Absicht aufConfirmed. Wenn der Benutzer mit „Nein“ antwortet, setzt Amazon Lex V2 confirmationState die Absicht aufDenied.

```
{
  "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?"
    }
]
}

```

Schließen

Schließen beendet den Erfüllungsprozess der Absicht und gibt an, dass keine weiteren Antworten vom Benutzer erwartet werden. Sie müssen das `name` und `state` von `intent` in das `sessionState` Objekt einschließen. Die kompatiblen Absichtszustände sind `Failed`, `Fulfilled`, und `InProgress`.

```

"sessionState": {
  "dialogAction": {
    "type": "Close"
  },
  "intent": {
    "name": "BookFlight",
    "state": "Failed | Fulfilled | InProgress"
  }
}

```

Gemeinsame Strukturen in einer AWS Lambda Funktion für

Innerhalb der Lambda-Antwort gibt es eine Reihe von Strukturen, die sich wiederholen. Einzelheiten zu diesen gemeinsamen Strukturen finden Sie in diesem Abschnitt.

Absicht

```

"intent": {
  "confirmationState": "Confirmed | Denied | None",
  "name": string,
  "slots": {
    // see Slots for details about the structure
  },
  "state": "Failed | Fulfilled | FulfillmentInProgress | InProgress |
ReadyForFulfillment | Waiting",
  "kendraResponse": {
    // Only present when intent is KendraSearchIntent. For details, see
    // https://docs.aws.amazon.com/kendra/latest/dg/API_Query.html#API_Query_ResponseSyntax
  }
}

```

Das intent Feld ist einem Objekt mit den folgenden Feldern zugeordnet:

Bestätigungsstatus

Gibt an, ob der Benutzer die Zeitfenster für die Absicht bestätigt hat und ob die Absicht zur Ausführung bereit ist. Die folgenden Werte sind möglich:

Confirmed— Der Benutzer bestätigt, dass die Slot-Werte korrekt sind.

Denied— Der Benutzer gibt an, dass die Slot-Werte falsch sind.

None— Der Benutzer hat die Bestätigungsphase noch nicht erreicht.

Name

Der Name der Absicht.

slots

Informationen über die Slots, die zur Erfüllung der Absicht erforderlich sind. Einzelheiten [Slots](#) zur Struktur finden Sie unter.

state

Gibt den Erfüllungsstatus für die Absicht an. Die folgenden Werte sind möglich:

Failed— Der Bot konnte die Absicht nicht erfüllen.

Fulfilled— Der Bot hat die Erfüllung der Absicht abgeschlossen.

FulfillmentInProgress— Der Bot ist gerade dabei, die Absicht zu erfüllen.

InProgress— Der Bot ist gerade dabei, die Slot-Werte zu ermitteln, die zur Erfüllung der Absicht erforderlich sind.

ReadyForFulfillment— Der Bot hat alle Slot-Werte für die Absicht ermittelt und ist bereit, die Absicht zu erfüllen.

Waiting— Der Bot wartet auf eine Antwort des Benutzers (beschränkt auf Streaming-Konversationen).

kendraResponse

Enthält Informationen zu den Ergebnissen der Kendra-Suchabfrage. Dieses Feld wird nur angezeigt, wenn die Absicht a `KendraSearchIntent` ist. Weitere Informationen finden Sie [in der Antwortsyntax im Query-API-Aufruf für Kendra](#).

Slots

Das `slots` Feld existiert innerhalb einer `intent` Struktur und ist einer Struktur zugeordnet, deren Schlüssel die Namen der Slots für diese Absicht sind. Wenn es sich bei dem Slot nicht um einen

Slot mit mehreren Werten handelt ([Verwendung mehrerer Werte in einem Slot](#) weitere Informationen finden Sie unter), wird er einer Struktur mit dem folgenden Format zugeordnet. Beachten Sie, dass der istshape. Scalar

```
{
  slot name: {
    "shape": "Scalar",
    "value": {
      "originalValue": string,
      "interpretedValue": string,
      "resolvedValues": [
        string,
        ...
      ]
    }
  }
}
```

Wenn es sich bei dem Slot um einen Slot mit mehreren Werten handelt, enthält das Objekt, dem er zugeordnet ist, ein weiteres Feld namensvalues, das einer Liste von Strukturen zugeordnet ist, von denen jede Informationen über einen Slot enthält, aus dem der Slot mit mehreren Werten besteht. Das Format der einzelnen Objekte in der Liste entspricht dem Format des Objekts, dem ein regulärer Slot zugeordnet ist. Beachten Sie, dass das shape istList, aber das shape der Komponenten, die sich darunter befinden, values istScalar.

```
{
  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,
                ...
            ]
        }
    },
    ...
]
}

```

Die Felder im Slot-Objekt werden im Folgenden beschrieben:

shape

Die Form des Schlitzes. Dieser Wert wird List verwendet, wenn der Steckplatz mehrere Werte enthält ([Verwendung mehrerer Werte in einem Slot](#) weitere Informationen finden Sie unter) und Scalar andernfalls.

Wert

Ein Objekt, das Informationen über den Wert, den der Benutzer für einen Slot angegeben hat, und dessen Interpretation im folgenden Format enthält:

```

{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
        ...
    ]
}

```

Die Felder werden im Folgenden beschrieben.

- **originalValue** — Der Teil der Antwort des Benutzers auf die Slot-Abfrage, der bestimmt, ob er für den Slot-Wert relevant ist.
- **interpretedValue** — Der Wert, der anhand der Benutzereingabe für den Slot bestimmt wird.
- **ResolvedValues** — Eine Liste von Werten, die die möglichen Auflösungen für die Benutzereingabe bestimmt.

values

Eine Liste von Objekten, die Informationen über die Slots enthält, aus denen der Slot mit mehreren Werten besteht. Das Format der einzelnen Objekte entspricht dem eines normalen Slots mit den oben beschriebenen **value** Feldern **shape** und **values** erscheint nur, wenn der Slot aus mehreren Werten besteht ([Verwendung mehrerer Werte in einem Slot](#) weitere Informationen finden Sie unter). Das folgende JSON-Objekt zeigt zwei Komponenten-Slots:

```
"values": [  
  {  
    "shape": "Scalar",  
    "value": {  
      "originalValue": string,  
      "interpretedValue": string,  
      "resolvedValues": [  
        string,  
        ...  
      ]  
    }  
  },  
  {  
    "shape": "Scalar",  
    "value": {  
      "originalValue": string,  
      "interpretedValue": string,  
      "resolvedValues": [  
        string,  
        ...  
      ]  
    }  
  },  
  ...  
]
```


]

Status der Sitzung

Das `sessionState` Feld ist einem Objekt zugeordnet, das Informationen über den Status der Konversation mit dem Benutzer enthält. Die tatsächlichen Felder, die im Objekt angezeigt werden, hängen von der Art der Dialogaktion ab. Informationen zu [Erforderliche Felder in der Antwort](#) den erforderlichen Feldern in einer Lambda-Antwort finden Sie unter. Das Format des `sessionState` Objekts ist wie folgt:

```
"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 Absicht for details about the structure
  },
  "originatingRequestId": string
}

```

Die Felder werden im Folgenden beschrieben.

Aktive Kontexte

Eine Liste von Objekten, die Informationen über einen Kontext enthalten, den ein Benutzer in einer Sitzung verwendet. Verwenden Sie Kontexte, um die Erkennung von Absichten zu erleichtern und zu kontrollieren. Weitere Informationen zu Kontexten finden Sie unter [Den Absichtskontext für Ihren Lex V2-Bot festlegen](#). Jedes Objekt ist wie folgt formatiert:

```

{
  "name": string,
  "contextAttributes": {
    string: string
  },
  "timeToLive": {
    "timeToLiveInSeconds": number,
    "turnsToLive": number
  }
}

```

Die Felder werden im Folgenden beschrieben.

- **name** — Der Name des Kontextes.
- **contextAttributes** — Ein Objekt, das die Namen der Attribute für den Kontext und die Werte enthält, denen sie zugeordnet sind.
- **timeToLive** — Ein Objekt, das angibt, wie lange der Kontext aktiv bleibt. Dieses Objekt kann eines oder beide der folgenden Felder enthalten:
 - **timeToLiveInSeconds** — Die Anzahl der Sekunden, für die der Kontext aktiv bleibt.

- `turnsToLive`— Die Anzahl der Runden, in denen der Kontext aktiv bleibt.

sessionAttributes

Eine Karte von key/value Paaren, die sitzungsspezifische Kontextinformationen darstellen. Weitere Informationen finden Sie unter [Sitzungsattribute für Ihren Lex V2-Bot festlegen](#). Das Objekt ist wie folgt formatiert:

```
{  
  string: string,  
  ...  
}
```

RuntimeHints

Bietet Hinweise zu den Ausdrücken, die ein Kunde wahrscheinlich für einen Slot verwenden wird, um die Audioerkennung zu verbessern. Die Werte, die Sie in den Hinweisen angeben, verbessern die Audioerkennung dieser Werte im Vergleich zu ähnlich klingenden Wörtern. Das Format des `runtimeHints` Objekts ist wie folgt:

```
{  
  "slotHints": {  
    intent name: {  
      slot name: {  
        "runtimeHintValues": [  
          {  
            "phrase": string  
          },  
          ...  
        ]  
      },  
      ...  
    },  
    ...  
  }  
}
```

Das `slotHints` Feld ist einem Objekt zugeordnet, dessen Felder die Namen der Absichten im Bot sind. Jeder Absichtsname ist einem Objekt zugeordnet, dessen Felder die Namen der Slots für diese Absicht sind. Jeder Slot-Name ist einer Struktur mit einem einzigen Feld

zugeordnet runtimeHintValues, bei dem es sich um eine Liste von Objekten handelt. Jedes Objekt enthält ein phrase Feld, das einem Hinweis zugeordnet ist.

dialogAction

Bestimmt die nächste Aktion, die Amazon Lex V2 ausführen soll. Das Format des Objekts ist wie folgt:

```
{
  "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
  "slotToElicit": string,
  "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

Die Felder werden im Folgenden beschrieben.

- slotElicitationStyle— Legt fest, wie Amazon Lex V2 die vom Benutzer eingegebenen Audioeingaben interpretiert, falls der Wert type von dialogAction ist ElicitSlot. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#). Die folgenden Werte sind möglich:

Default— Amazon Lex V2 interpretiert die Audioeingabe standardmäßig, um einen Slot zu füllen.

SpellByLetter— Amazon Lex V2 wartet auf die Schreibweise des Slot-Werts durch den Benutzer.

SpellByWord— Amazon Lex V2 überwacht die Schreibweise des Slot-Werts durch den Benutzer und verwendet dabei Wörter, die jedem Buchstaben zugeordnet sind (z. B. „a wie in Apple“).

- slotToElicit— Definiert den Slot, der dem Benutzer abgefragt werden soll, ob der type Wert von ist dialogAction ElicitSlot
- type — Definiert die Aktion, die der Bot ausführen soll. Die folgenden Werte sind möglich:

Delegate— Überlässt Amazon Lex V2 den nächsten Schritt.

ElicitIntent— Fordert den Kunden auf, eine Absicht zu äußern.

ConfirmIntent— Bestätigt die Slot-Werte des Kunden und ob die Absicht zur Erfüllung bereit ist.

ElicitSlot— Fordert den Kunden auf, einen Slot-Wert für eine Absicht anzugeben.

Close— Beendet den Prozess zur Erfüllung der Absicht.

Absicht

Sehen Sie [Absicht](#) sich die Struktur des `intent` Feldes an.

`originatingRequestId`

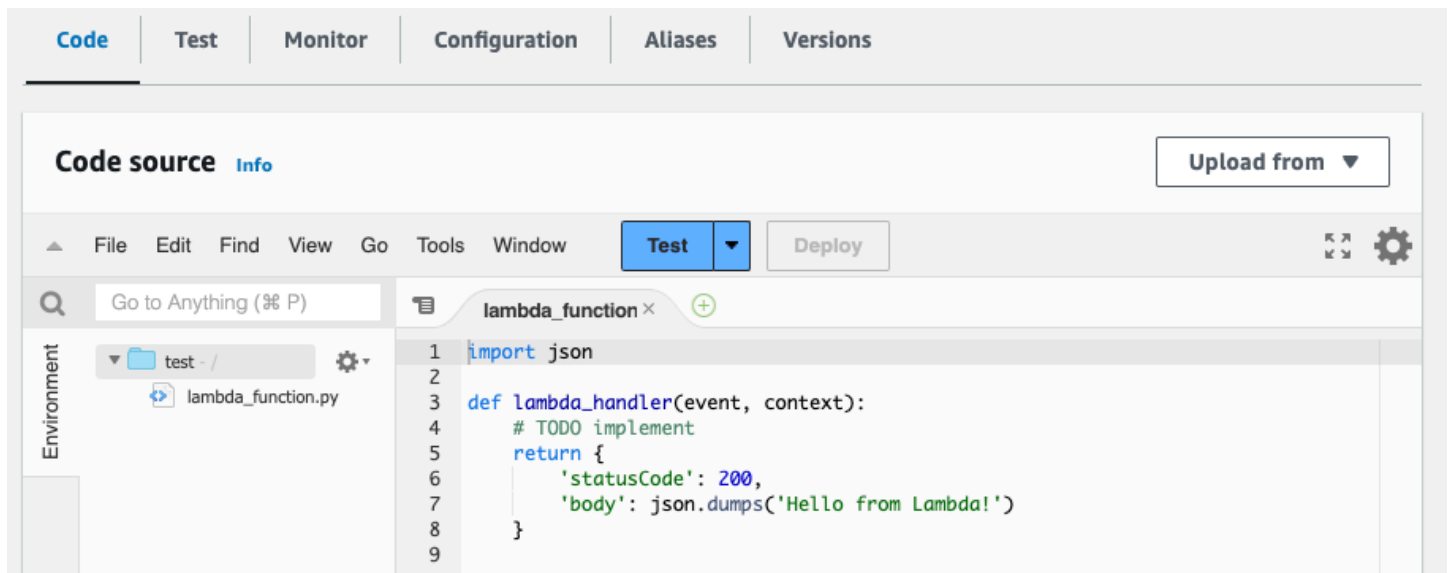
Eine eindeutige Kennung für die Anfrage. Dieses Feld ist für die Lambda-Antwort optional.

Eine AWS Lambda Funktion für Ihren Bot erstellen

Um eine Lambda-Funktion für Ihren Amazon Lex V2-Bot zu erstellen, greifen Sie AWS Lambda von Ihrem aus zu AWS Management Console und erstellen Sie eine neue Funktion. Weitere Informationen zu finden Sie im [AWS Lambda Entwicklerhandbuch](#). AWS Lambda

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter <https://console.aws.amazon.com/lambda/>.
2. Wählen Sie in der linken Seitenleiste Funktionen aus.
3. Wählen Sie Funktion erstellen aus.
4. Sie können Autor von Grund auf neu auswählen, um mit minimalem Code zu beginnen, einen Blueprint verwenden, um Beispielcode für allgemeine Anwendungsfälle aus einer Liste auszuwählen, oder Container-Image, um ein Container-Image auszuwählen, das für Ihre Funktion bereitgestellt werden soll. Wenn Sie Author von Grund auf neu auswählen, fahren Sie mit den folgenden Schritten fort:
 - a. Geben Sie Ihrer Funktion einen aussagekräftigen Funktionsnamen, um zu beschreiben, was sie tut.
 - b. Wählen Sie im Drop-down-Menü unter Runtime eine Sprache aus, in der Sie Ihre Funktion schreiben möchten.
 - c. Wählen Sie eine Befehlssatzarchitektur für Ihre Funktion aus.
 - d. Standardmäßig erstellt Lambda eine Rolle mit Basisberechtigungen. Um eine vorhandene Rolle zu verwenden oder eine Rolle mithilfe von AWS Richtlinienvorlagen zu erstellen, erweitern Sie das Menü Standardausführungsrolle ändern und wählen Sie eine Option aus.
 - e. Erweitern Sie das Menü Erweiterte Einstellungen, um weitere Optionen zu konfigurieren.
5. Wählen Sie Funktion erstellen aus.

Die folgende Abbildung zeigt, was Sie sehen, wenn Sie eine neue Funktion von Grund auf neu erstellen:



Die Lambda-Handler-Funktion unterscheidet sich je nach verwendeter Sprache. Sie benötigt mindestens ein event JSON-Objekt als Argument. Sie können die Felder in der Datei sehen [event](#), die Amazon Lex V2 zur Verfügung stellt, unter [AWS Lambda Eingabeereignisformat für Lex V2](#). Ändern Sie die Handler-Funktion so, dass letztendlich ein response JSON-Objekt zurückgegeben wird, das dem unter beschriebenen Format entspricht [AWS Lambda Antwortformat für Lex V2](#).

- Wenn Sie mit dem Schreiben Ihrer Funktion fertig sind, wählen Sie Deploy aus, damit die Funktion verwendet werden kann.

Denken Sie daran, dass Sie jeden Bot-Alias mit höchstens einer Lambda-Funktion verknüpfen können. Sie können jedoch innerhalb des Lambda-Codes so viele Funktionen definieren, wie Sie für Ihren Bot benötigen, und diese Funktionen in der Lambda-Handler-Funktion aufrufen. Während beispielsweise alle Intents in demselben Bot-Alias dieselbe Lambda-Funktion aufrufen müssen, können Sie eine Router-Funktion erstellen, die für jede Absicht eine separate Funktion aktiviert. Im Folgenden finden Sie ein Beispiel für eine Router-Funktion, die Sie für Ihre Anwendung verwenden oder ändern können:

```

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
```

Wann sollten AWS Lambda Funktionen in Bot-Konversationen verwendet werden

Sie können Lambda-Funktionen an den folgenden Stellen in einer Konversation mit einem Benutzer verwenden:

- In der ersten Antwort, nachdem die Absicht erkannt wurde. Zum Beispiel, nachdem der Benutzer gesagt hat, dass er eine Pizza bestellen möchte.
- Nachdem dem Benutzer ein Slot-Wert abgefragt wurde. Zum Beispiel, nachdem der Benutzer dem Bot die Größe der Pizza mitgeteilt hat, die er bestellen möchte.
- Zwischen jedem erneuten Versuch, einen Slot auszulösen. Zum Beispiel, wenn der Kunde keine anerkannte Pizzagröße verwendet.
- Bei der Bestätigung einer Absicht. Zum Beispiel bei der Bestätigung einer Pizzabestellung.
- Um eine Absicht zu erfüllen. Zum Beispiel, um eine Pizza zu bestellen.
- Nach Erfüllung der Absicht und bevor Ihr Bot die Konversation beendet. Zum Beispiel, um zu einer Absicht zu wechseln, ein Getränk zu bestellen.

Themen

- [Hängen Sie über die Konsole eine AWS Lambda Funktion an einen Bot an](#)
- [Hängen Sie mithilfe von API-Operationen eine AWS Lambda Funktion an einen Bot an](#)

Hängen Sie über die Konsole eine AWS Lambda Funktion an einen Bot an

Sie müssen zuerst eine Lambda-Funktion an Ihren Bot-Alias anhängen, bevor Sie ihn aufrufen können. Sie können jedem Bot-Alias nur eine Lambda-Funktion zuordnen. Führen Sie diese Schritte aus, um die Lambda-Funktion mithilfe der AWS-Konsole anzuhängen.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie im linken Seitenbereich Bots aus und wählen Sie aus der Liste der Bots den Namen des Bots aus, an den Sie eine Lambda-Funktion anhängen möchten.
3. Wählen Sie auf der linken Seite im Menü „Bereitstellung“ die Option Aliase aus.
4. Wählen Sie aus der Liste der Aliase den Namen des Alias aus, an den Sie eine Lambda-Funktion anhängen möchten.
5. Wählen Sie im Bereich Sprachen die Sprache aus, die eine Lambda-Funktion verwenden soll. Wählen Sie Sprachen im Alias verwalten aus, um eine Sprache hinzuzufügen, falls sie nicht im Panel vorhanden ist.
6. Wählen Sie im Dropdownmenü Quelle den Namen der Lambda-Funktion aus, die Sie anhängen möchten.
7. Wählen Sie im Dropdownmenü Version oder Alias der Lambda-Funktion die Version oder den Alias der Lambda-Funktion aus, die Sie verwenden möchten. Wählen Sie dann Save (Speichern) aus. Dieselbe Lambda-Funktion wird für alle Zwecke in einer vom Bot unterstützten Sprache verwendet.

Die Absicht festlegen, eine Lambda-Funktion über die Konsole aufzurufen

1. Nachdem Sie einen Bot ausgewählt haben, wählen Sie im Menü auf der linken Seite unter der Sprache des Bots, für den Sie die Lambda-Funktion aufrufen möchten, die Option Absichten aus.
2. Wählen Sie die Absicht aus, in der Sie die Lambda-Funktion aufrufen möchten, um den Absichtseditor zu öffnen.
3. Es gibt zwei Möglichkeiten, den Lambda-Code-Hook einzustellen:
 1. Um die Lambda-Funktion nach jedem Schritt der Konversation aufzurufen, scrollen Sie zum Abschnitt Code-Hooks unten im Intent-Editor und aktivieren Sie das Kontrollkästchen Lambda-Funktion für Initialisierung und Validierung verwenden, wie in der folgenden Abbildung dargestellt:

▼ Code hooks - optional [Info](#)☒ **Use a Lambda function for initialization and validation**

Lambda function is executed at every turn of the conversation. You can use this function to initialize values or validate user input.

2. Verwenden Sie alternativ den Dialogcode-Hook-Abschnitt in den Konversationsphasen, in denen Sie die Lambda-Funktion aufrufen möchten. Der Abschnitt mit dem Dialog-Code-Hook sieht wie folgt aus:

Dialog code hook [Info](#) **Active**

You can enable Lambda functions to manage initialize the conversation.

▼ Lambda dialog code hook

Invoke Lambda function: No

☐ **Invoke Lambda function****Advanced options**

Configure success, failure, and timeout responses for dialog code hook.

Es gibt zwei Möglichkeiten, zu steuern, wie Amazon Lex V2 den Code-Hook für eine Antwort aufruft:

- Klicken Sie auf die Schaltfläche Aktiv, um sie als aktiv oder inaktiv zu markieren. Wenn ein Code-Hook aktiv ist, ruft Amazon Lex V2 den Code-Hook auf. Wenn der Code-Hook inaktiv ist, führt Amazon Lex V2 den Code-Hook nicht aus.
- Erweitern Sie den Abschnitt Lambda-Dialogcode-Hook und aktivieren Sie das Kontrollkästchen Lambda-Funktion aufrufen, um ihn als aktiviert oder deaktiviert zu markieren. Sie können einen Code-Hook nur aktivieren oder deaktivieren, wenn er als aktiv markiert ist. Wenn er als aktiviert markiert ist, wird der Code-Hook normal ausgeführt. Wenn er deaktiviert ist, wird der Code-Hook nicht aufgerufen und Amazon Lex V2 verhält sich so, als ob der Code-Hook erfolgreich zurückgegeben wurde. Um Antworten zu konfigurieren, nachdem der Dialog-Code-Hook erfolgreich war, fehlschlägt oder das Timeout abgelaufen ist, wählen Sie Erweiterte Optionen

Der Lambda-Code-Hook kann in den folgenden Konversationsphasen aufgerufen werden:

- Um die Funktion als erste Antwort aufzurufen, scrollen Sie zum Abschnitt Erste Antwort, erweitern Sie den Pfeil neben Antwort, um die Anfrage des Benutzers zu bestätigen, und wählen Sie Erweiterte Optionen aus. Suchen Sie unten im sich öffnenden Menü nach dem Abschnitt „Dialog-Code-Hook“.
- Um die Funktion nach der Slot-Abfrage aufzurufen, scrollen Sie zum Abschnitt Steckplätze, erweitern Sie den Pfeil neben der entsprechenden Eingabeaufforderung für den Slot und wählen Sie Erweiterte Optionen. Suchen Sie den Abschnitt mit dem Dialog-Code-Hook am unteren Rand des Menüs, das sich öffnet, direkt über den Standardwerten.

Sie können die Funktion auch nach jedem Aufruf aufrufen. Erweitern Sie dazu im Abschnitt Slot-Eingabeaufforderungen die Option Bot ruft Informationen ab, wählen Sie Weitere Aufforderungsoptionen aus und aktivieren Sie das Kontrollkästchen neben Lambda-Code-Hook nach jedem Aufruf aufrufen.

- Um die Funktion zur Bestätigung der Absicht aufzurufen, scrollen Sie zum Abschnitt Bestätigung, erweitern Sie den Pfeil neben Aufforderungen zur Bestätigung der Absicht und wählen Sie Erweiterte Optionen aus. Suchen Sie unten im sich öffnenden Menü nach dem Abschnitt mit dem Dialog-Code-Hook.
 - Scrollen Sie zum Abschnitt Fulfillment, um die Funktion für die Erfüllung von Absichten aufzurufen. Klicken Sie auf die Schaltfläche Aktiv, um den Code-Hook auf aktiv zu setzen. Erweitern Sie den Pfeil neben Bei erfolgreichem Versand und wählen Sie Erweiterte Optionen aus. Aktivieren Sie im Abschnitt Fulfillment Lambda Code Hook das Kontrollkästchen neben Eine Lambda-Funktion für die Auftragsabwicklung verwenden, um den Code-Hook auf aktiviert zu setzen.
4. Nachdem Sie die Konversationsphasen festgelegt haben, in denen die Lambda-Funktion aufgerufen werden soll, erstellen Sie den Bot erneut, um die Funktion zu testen.

Hängen Sie mithilfe von API-Operationen eine AWS Lambda Funktion an einen Bot an

Sie müssen zuerst eine Lambda-Funktion an Ihren Bot-Alias anhängen, bevor Sie ihn aufrufen können. Sie können jedem Bot-Alias nur eine Lambda-Funktion zuordnen. Führen Sie diese Schritte aus, um die Lambda-Funktion mithilfe von API-Operationen anzuhängen.

Wenn Sie einen neuen Bot-Alias erstellen, verwenden Sie den [CreateBotAlias](#) Vorgang, um eine Lambda-Funktion anzuhängen. Verwenden Sie die [UpdateBotAlias](#) Operation, um eine Lambda-

Funktion an einen vorhandenen Bot-Alias anzuhängen. Ändern Sie das `botAliasLocaleSettings` Feld so, dass es die richtigen Einstellungen enthält:

```
{
  "botAliasLocaleSettings" : {
    locale: {
      "codeHookSpecification": {
        "lambdaCodeHook": {
          "codeHookInterfaceVersion": "1.0",
          "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
        }
      },
      "enabled": true
    },
    ...
  }
}
```

1. Das `botAliasLocaleSettings` Feld ist einem Objekt zugeordnet, dessen Schlüssel die Gebietsschemas sind, an die Sie die Lambda-Funktion anhängen möchten. Eine Liste der unterstützten Gebietsschemas und der Codes, bei denen es sich um gültige Schlüssel handelt, finden Sie unter: [Unterstützte Sprachen und Gebietsschemas](#)
2. Um die Funktion `lambdaARN` für a Lambda zu finden, öffnen Sie die AWS Lambda Konsole zu <https://console.aws.amazon.com/lambda/Hause>, wählen Sie in der linken Seitenleiste Funktionen und wählen Sie die Funktion aus, die dem Bot-Alias zugeordnet werden soll. Auf der rechten Seite der Funktionsübersicht finden Sie `lambdaARN` unter Funktion ARN. Sie sollte eine Region, eine Konto-ID und den Namen der Funktion enthalten.
3. Damit Amazon Lex V2 die Lambda-Funktion für den Alias aufrufen kann, setzen Sie das `enabled` Feld auf `true`

Festlegung der Absicht, eine Lambda-Funktion mithilfe von API-Operationen aufzurufen

Um den Lambda-Funktionsaufruf während einer Absicht einzurichten, verwenden Sie die [CreateIntent](#)Operation, wenn Sie eine neue Absicht erstellen, oder die [UpdateIntent](#)Operation, wenn Sie die Funktion in einer vorhandenen Absicht aufrufen. Die Felder, die den Lambda-Funktionsaufruf in den Intent-Operationen steuern `dialogCodeHook`, sind, `initialResponseSetting` `intentConfirmationSetting`, und `fulfillmentCodeHook`

Wenn Sie die Funktion während des Auslösens eines Slots aufrufen, verwenden Sie die [CreateSlot](#)Operation, wenn Sie einen neuen Slot erstellen, oder die [UpdateSlot](#)Operation, um die Funktion in einem vorhandenen Slot aufzurufen. Das Feld, das den Lambda-Funktionsaufruf in den Slot-Operationen steuert, ist das `slotCaptureSetting` des `valueElicitationSetting` Objekts.

1. Um den Lambda-Dialog-Code-Hook so einzustellen, dass er nach jeder Runde der Konversation ausgeführt wird, setzen Sie das `enabled` Feld des folgenden [DialogCodeHookSettings](#) Objekts im `dialogCodeHook` Feld auf `true`:

```
"dialogCodeHook": {  
  "enabled": boolean  
}
```

2. Alternativ können Sie den Lambda-Dialogcode-Hook so einrichten, dass er nur an bestimmten Stellen in den Konversationen ausgeführt wird, indem Sie das `codeHook` and/or `elicitationCodeHook` Feld innerhalb der Strukturen ändern, die den Konversationsphasen entsprechen, in denen Sie die Funktion aufrufen möchten. Um den Lambda-Dialogcode-Hook für die Erfüllung von Absichten zu verwenden, verwenden Sie das `fulfillmentCodeHook` Feld in der [UpdateIntent](#)Operation [CreateIntent](#). Die Strukturen und Verwendungszwecke dieser drei Arten von Code-Hooks lauten wie folgt:

CodeHook

Das `codeHook` Feld definiert die Einstellungen für den Code-Hook, der in einer bestimmten Phase der Konversation ausgeführt werden soll. Es ist ein [DialogCodeHookInvocationSetting](#) Objekt mit der folgenden Struktur:

```
"codeHook": {  
  "active": boolean,  
  "enableCodeHookInvocation": boolean,  
  "invocationLabel": string,  
  "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,  
}
```

- Ändern Sie das `active` Feld auf `true` für Amazon Lex V2, um den Code-Hook an diesem Punkt in der Konversation aufzurufen.

- Ändern Sie das `enableCodeHookInvocation` Feld auf `true` für Amazon Lex V2, damit der Code-Hook normal ausgeführt werden kann. Wenn Sie es markieren `false`, verhält sich Amazon Lex V2 so, als ob der Code-Hook erfolgreich zurückgegeben wurde.
- Das `invocationLabel` gibt den Dialogschritt an, von dem aus der Code-Hook aufgerufen wird.
- Verwenden Sie das `postCodeHookSpecification` Feld, um die Aktionen und Meldungen anzugeben, die nach einem erfolgreichen, fehlgeschlagenen oder nach einem Timeout des Code-Hooks ausgeführt werden.

`elicitationCodeHook`

Das `elicitationCodeHook` Feld definiert die Einstellungen für den Code-Hook, der ausgeführt werden soll, falls ein oder mehrere Slots erneut abgerufen werden müssen. Dieses Szenario kann eintreten, wenn die Slot-Abfrage fehlschlägt oder die Bestätigung der Absicht verweigert wird. Das `elicitationCodeHook` Feld ist ein [ElicitationCodeHookInvocationSetting](#) Objekt mit der folgenden Struktur:

```
"elicitationCodeHook": {  
  "enableCodeHookInvocation": boolean,  
  "invocationLabel": string  
}
```

- Ändern Sie das `enableCodeHookInvocation` Feld auf `true` für Amazon Lex V2, damit der Code-Hook normal ausgeführt werden kann. Wenn Sie es markieren `false`, verhält sich Amazon Lex V2 so, als ob der Code-Hook erfolgreich zurückgegeben wurde.
- Das `invocationLabel` gibt den Dialogschritt an, von dem aus der Code-Hook aufgerufen wird.

`fulfillmentCodeHook`

Das `fulfillmentCodeHook` Feld definiert die Einstellungen für den Code-Hook, der ausgeführt werden soll, um die Absicht zu erfüllen. Es ist dem folgenden [FulfillmentCodeHookSettings](#) Objekt zugeordnet:

```
"fulfillmentCodeHook": {  
  "active": boolean,  
  "enabled": boolean,  
  "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,  
  "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
```

```
}
```

- Ändern Sie das `active` Feld auf `true` für Amazon Lex V2, um den Code-Hook an diesem Punkt in der Konversation aufzurufen.
- Ändern Sie das `enabled` Feld auf `true` für Amazon Lex V2, damit der Code-Hook normal ausgeführt werden kann. Wenn Sie es markieren `false`, verhält sich Amazon Lex V2 so, als ob der Code-Hook erfolgreich zurückgegeben wurde.
- Verwenden Sie das `fulfillmentUpdatesSpecification` Feld, um die Nachrichten anzugeben, die den Benutzer während der Erfüllung der Absicht auf den neuesten Stand bringen sollen, und den damit verbundenen Zeitpunkt.
- Verwenden Sie das `postFulfillmentStatusSpecification` Feld, um die Meldungen und Aktionen anzugeben, die nach einem erfolgreichen, fehlgeschlagenen oder nach einem Timeout des Code-Hooks ausgeführt werden.

Sie können den Lambda-Code-Hook an den folgenden Stellen in einer Konversation aufrufen, indem Sie die `enabled` Felder `active` und `enableCodeHookInvocation` auf `true` setzen:

Während der ersten Antwort

Um die Lambda-Funktion in der ersten Antwort aufzurufen, nachdem die Absicht erkannt wurde, verwenden Sie die `codeHook` Struktur im `initialResponse` Feld der Operation [CreateIntentor UpdateIntent](#). Das `initialResponse` Feld ist dem folgenden [InitialResponseSetting](#) Objekt zugeordnet:

```
"initialResponse": {
  "codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
  },
  "initialResponse": FulfillmentUpdatesSpecification object,
  "nextStep": PostFulfillmentStatusSpecification object,
  "conditional": ConditionalSpecification object
}
```

Nach der Slot-Excitation oder während der Slot-Re-Excitation

Um die Lambda-Funktion aufzurufen, nachdem ein Slot-Wert abgerufen wurde, verwenden Sie das `slotCaptureSetting` Feld innerhalb des Felds der `valueElicitation` OR-Operation.

[CreateSlotUpdateSlot](#) Das `slotCaptureSetting` Feld ist dem folgenden Objekt zugeordnet: [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
}
```

- Verwenden Sie das Feld, um die Lambda-Funktion aufzurufen, nachdem die Slot-Ermittlung erfolgreich war. `codeHook`
- Verwenden Sie das Feld, um die Lambda-Funktion aufzurufen, nachdem die Slot-Erfassung fehlgeschlagen ist und Amazon Lex V2 versucht, die Slot-Abfrage erneut zu versuchen. `elicitationCodeHook`

Nach Bestätigung oder Ablehnung der Absicht

Verwenden Sie das `intentConfirmationSetting` Feld der Operation [CreateIntentorUpdateIntent](#), um die Lambda-Funktion aufzurufen, wenn Sie eine Absicht bestätigen. Das `intentConfirmation` Feld ist dem folgenden [IntentConfirmationSetting](#) Objekt zugeordnet:

```
"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
}

```

- Verwenden Sie das Feld, um die Lambda-Funktion aufzurufen, nachdem der Benutzer die Absicht und ihre Slots bestätigt hat. `codeHook`
- Verwenden Sie das Feld, um die Lambda-Funktion aufzurufen, nachdem der Benutzer die Bestätigung der Absicht verweigert hat und Amazon Lex V2 versucht, die Slot-Abfrage erneut zu versuchen. `elicitationCodeHook`

Während der Absichtserfüllung

Um die Lambda-Funktion aufzurufen, um eine Absicht zu erfüllen, verwenden Sie das `fulfillmentCodeHook` Feld in der Operation [CreateIntent](#) oder [UpdateIntent](#). Das `fulfillmentCodeHook` Feld ist dem folgenden [FulfillmentCodeHookSettings](#) Objekt zugeordnet:

```

{
  "active": boolean,
  "enabled": boolean,
  "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
  "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}

```


3. Nachdem Sie die Konversationsphasen festgelegt haben, in denen die Lambda-Funktion aufgerufen werden soll, verwenden Sie den BuildBotLocale Vorgang, um den Bot neu zu erstellen, um die Funktion zu testen.

Debuggen einer Lambda-Funktion mithilfe von CloudWatch Logs

[Amazon CloudWatch Logs](#) ist ein Tool zur Nachverfolgung von API-Aufrufen und Metriken, mit dem Sie Ihre Lambda-Funktionen debuggen können. Wenn Sie Ihren Bot in der Konsole oder mit API-Aufrufen testen, CloudWatch protokolliert er jeden Schritt der Konversation. Wenn Sie in Ihrem Lambda-Code eine Druckfunktion verwenden, CloudWatch wird diese ebenfalls angezeigt.

So zeigen Sie CloudWatch Logs für Ihre Lambda-Funktion an

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie in der linken Seitenleiste im Menü Protokolle die Option Protokollgruppen aus.
3. Wählen Sie Ihre Lambda-Funktionsprotokollgruppe aus, die das folgende Format `/aws/lambda/function-name` haben sollte.
4. Die Liste der Log-Streams enthält ein Protokoll für jede Sitzung mit einem Bot. Wählen Sie einen Protokollstream aus, um ihn anzuzeigen.
5. Wählen Sie in der Liste der Protokollereignisse den Rechtspfeil neben dem Zeitstempel aus, um die Details für dieses Ereignis zu erweitern. Alles, was Sie aus Ihrem Lambda-Code drucken, wird als Protokollereignis angezeigt. Verwenden Sie diese Informationen, um Ihren Code zu debuggen.
6. Denken Sie nach dem Debuggen Ihres Codes daran, die Lambda-Funktion bereitzustellen und, falls Sie die Konsole verwenden, das Testfenster neu zu laden, bevor Sie das Verhalten des Bots erneut testen.

Anpassen von Bot-Interaktionen mit Benutzern in Lex V2

Erfahren Sie mehr über die folgenden Funktionen, mit denen Sie Bot-Interaktionen mit Ihren Benutzern anpassen können, indem Sie deren Standardverhalten erweitern und anpassen:

Themen

- [Analysieren Sie die Stimmung der Benutzeräußerungen in Konversationen mit Ihrem Bot](#)
- [Verwendung von Konfidenzwerten zur Verbesserung der Konversationsgenauigkeit](#)
- [Anpassen von Sprachtranskriptionen für die Verwendung mit Ihrem Lex V2-Bot](#)

Analysieren Sie die Stimmung der Benutzeräußerungen in Konversationen mit Ihrem Bot

Sie können die Stimmungsanalyse verwenden, um die Stimmungen einer Benutzeräußerung zu ermitteln. Mit den Stimmungsinformationen können Sie den Konversationsfluss verwalten oder eine Analyse nach dem Anruf durchführen. Wenn die Benutzerstimmung beispielsweise negativ ist, können Sie einen Workflow erstellen, um eine Konversation an einen menschlichen Agenten zu übergeben.

Amazon Lex lässt sich in Amazon Comprehend integrieren, um die Stimmung der Benutzer zu erkennen. Die Antwort von Amazon Comprehend gibt an, ob die allgemeine Stimmung des Textes positiv, neutral, negativ oder gemischt ist. Die Antwort enthält die wahrscheinlichste Stimmung für die Äußerung des Benutzers und die Punktzahl für jede der Stimmungskategorien. Die Punktzahl stellt die Wahrscheinlichkeit dar, dass die Stimmung korrekt erkannt wurde.

Sie aktivieren die Stimmungsanalyse für einen Bot mithilfe der Konsole oder mithilfe der Amazon Lex Lex-API. Sie aktivieren die Stimmungsanalyse für einen Alias für den Bot. Auf der Amazon Lex Lex-Konsole:

1. Wählen Sie einen Alias.
2. Wählen Sie unter Details die Option Bearbeiten aus.
3. Wählen Sie Stimmungsanalyse aktivieren, um die Stimmungsanalyse ein- oder auszuschalten.
4. Wählen Sie Confirm (Bestätigen), um Ihre Änderungen zu speichern.

Wenn Sie die API verwenden, rufen Sie den [CreateBotAlias](#)-Vorgang mit auf `true` eingestelltem `detectSentiment`-Feld auf.

Wenn die Stimmungsanalyse aktiviert ist, gibt die Antwort der [RecognizeUtterance](#) Operationen [RecognizeText](#) und ein Feld zurück, das `sentimentResponse` in der `interpretations` Struktur mit anderen Metadaten aufgerufen wird. Das Feld `sentimentResponse` verfügt über die zwei Felder `sentiment` und `sentimentScore`, die das Ergebnis der Stimmungsanalyse enthalten. Wenn Sie eine Lambda-Funktion verwenden, ist das `sentimentResponse` Feld in den an Ihre Funktion gesendeten Ereignisdaten enthalten.

Der folgende Code ist ein Beispiel für das Feld `sentimentResponse`, das als Teil der Antwort `RecognizeUtterance` `RecognizeText` oder zurückgegeben wird.

```
sentimentResponse {
  "sentimentScore": {
    "mixed": 0.030585512690246105,
    "positive": 0.94992071056365967,
    "neutral": 0.0141543131828308,
    "negative": 0.00893945890665054
  },
  "sentiment": "POSITIVE"
}
```

Amazon Lex ruft in Ihrem Namen Amazon Comprehend an, um die Stimmung in jeder vom Bot verarbeiteten Äußerung zu ermitteln. Durch die Aktivierung der Stimmungsanalyse stimmen Sie den Servicebedingungen und -vereinbarungen für Amazon Comprehend zu. [Weitere Informationen zu den Preisen für Amazon Comprehend finden Sie unter Amazon Comprehend Pricing.](#)

Weitere Informationen zur Funktionsweise der Amazon Comprehend-Stimmungsanalyse finden Sie unter [Ermitteln der Stimmung](#) im Amazon Comprehend Developer Guide.

Verwendung von Konfidenzwerten zur Verbesserung der Konversationsgenauigkeit

Amazon Lex V2 verwendet zwei Schritte, um zu ermitteln, was ein Benutzer sagt. Die erste Methode, die automatische Spracherkennung (ASR), erstellt eine Abschrift der Audioäußerung des Benutzers. Die zweite Methode, Natural Language Understanding (NLU), bestimmt die Bedeutung der Äußerung des Benutzers, um die Absicht des Benutzers oder den Wert von Slots zu erkennen.

Standardmäßig gibt Amazon Lex V2 das wahrscheinlichste Ergebnis von ASR und NLU zurück. Manchmal kann es für Amazon Lex V2 schwierig sein, das wahrscheinlichste Ergebnis zu ermitteln. In diesem Fall werden mehrere mögliche Ergebnisse zusammen mit einem Konfidenzwert zurückgegeben, der angibt, wie wahrscheinlich es ist, dass das Ergebnis korrekt ist. Ein Konfidenzwert ist eine von Amazon Lex V2 abgegebene Bewertung, die das relative Vertrauen zeigt, das Amazon in das Ergebnis hat. Die Konfidenzwerte reichen von 0,0 bis 1,0.

Sie können Ihr Fachwissen zusammen mit dem Konfidenzwert nutzen, um die korrekte Interpretation des ASR- oder NLU-Ergebnisses zu ermitteln.

Der ASR- oder Transkriptionsvertrauenswert ist eine Bewertung dafür, wie sicher Amazon Lex V2 ist, dass eine bestimmte Transkription korrekt ist. Der NLU- oder Intent-Konfidenzwert ist eine Bewertung dafür, wie sicher Amazon Lex V2 ist, dass die in der obersten Transkription angegebene Absicht korrekt ist. Verwenden Sie den Vertrauenswert, der am besten zu Ihrer Anwendung passt.

Themen

- [Verwendung von Intent-Konfidenzwerten zur Verbesserung der Absichtsauswahl mit Lex V2](#)
- [Verwenden Sie Konfidenzwerte für die Sprachtranskription, um die Konversation mit Ihrem Lex V2-Bot zu verbessern](#)

Verwendung von Intent-Konfidenzwerten zur Verbesserung der Absichtsauswahl mit Lex V2

Wenn ein Benutzer eine Äußerung macht, verwendet Amazon Lex V2 Natural Language Understanding (NLU), um die Anfrage des Benutzers zu verstehen und die richtige Absicht zurückzugeben. Standardmäßig gibt Amazon Lex V2 die wahrscheinlichste Absicht zurück, die von Ihrem Bot definiert wurde.

In einigen Fällen kann es für Amazon Lex V2 schwierig sein, die wahrscheinlichste Absicht zu ermitteln. Beispielsweise könnte der Benutzer eine mehrdeutige Äußerung machen, oder es könnte zwei ähnliche Absichten geben. Um die richtige Absicht zu ermitteln, können Sie Ihr Fachwissen mit den NLU-Konfidenzwerten in einer Liste von Interpretationen kombinieren. Ein Konfidenzwert ist eine von Amazon Lex V2 abgegebene Bewertung, die zeigt, wie sicher es ist, dass eine Absicht die richtige Absicht ist.

Um den Unterschied zwischen zwei Absichten innerhalb einer Interpretation zu ermitteln, können Sie deren Konfidenzwerte vergleichen. Wenn beispielsweise eine Absicht einen Konfidenzwert von 0,95 und eine andere einen Wert von 0,65 hat, ist die erste Absicht wahrscheinlich richtig. Wenn

jedoch eine Absicht einen Wert von 0,75 und eine andere einen Wert von 0,72 hat, besteht eine Mehrdeutigkeit zwischen den beiden Absichten, sodass Sie anhand von Domänenwissen in Ihrer Anwendung möglicherweise unterscheiden können.

Sie können Konfidenzwerte auch verwenden, um Testanwendungen zu erstellen, mit denen ermittelt wird, ob Änderungen an den Äußerungen einer Absicht das Verhalten des Bots beeinflussen. Sie können beispielsweise anhand einer Reihe von Äußerungen die Konfidenzwerte für die Absichten eines Bots ermitteln und die Absichten dann mit neuen Äußerungen aktualisieren. Anschließend können Sie die Konfidenzwerte überprüfen, um festzustellen, ob eine Verbesserung eingetreten ist.

Bei den Konfidenzwerten, die Amazon Lex V2 zurückgibt, handelt es sich um Vergleichswerte. Sie sollten sich nicht auf sie als absolute Punktzahl verlassen. Die Werte können sich aufgrund von Verbesserungen an Amazon Lex V2 ändern.

Amazon Lex V2 gibt in jeder Antwort die wahrscheinlichste Absicht und bis zu 4 alternative Absichten mit den zugehörigen Bewertungen in der `interpretations` Struktur zurück. Der folgende JSON-Code zeigt die `interpretations` Struktur in der Antwort auf den [RecognizeText](#) Vorgang:

```
"interpretations": [
  {
    "intent": {
      "confirmationState": "string",
      "name": "string",
      "slots": {
        "string" : {
          "value": {
            "interpretedValue": "string",
            "originalValue": "string",
            "resolvedValues": [ "string" ]
          }
        }
      },
      "state": "string"
    },
    "nluConfidence": number
  }
]
```

AMAZON.FallbackIntent

Amazon Lex V2 kehrt in zwei Situationen `AMAZON.FallbackIntent` als Top-Intent zurück:

1. Wenn die Konfidenzwerte aller möglichen Absichten unter dem Konfidenzschwellenwert liegen. Sie können den Standardschwellenwert verwenden oder einen eigenen Schwellenwert festlegen. Wenn Sie das `AMAZON.KendraSearchIntent` konfiguriert haben, gibt Amazon Lex V2 es in dieser Situation ebenfalls zurück.
2. Wenn die Interpretationssicherheit für höher `AMAZON.FallbackIntent` ist als die Interpretationssicherheit aller anderen Absichten.

Beachten Sie, dass Amazon Lex V2 keinen Konfidenzwert für `AMAZON.FallbackIntent` anzeigt.

Einstellung und Änderung des Konfidenzschwellenwerts

Der Konfidenzschwellenwert muss eine Zahl zwischen 0,00 und 1,00 sein. Sie können den Schwellenwert für jede Sprache in Ihrem Bot auf folgende Weise festlegen:

Verwenden der Amazon Lex V2-Konsole

- Um den Schwellenwert festzulegen, wenn Sie Ihrem Bot mit „Sprache hinzufügen“ eine Sprache hinzufügen, können Sie den gewünschten Wert in das Fenster „Schwellenwert für die Vertrauensbewertung“ eingeben.
- Um den Schwellenwert zu aktualisieren, können Sie im Bereich mit den Sprachdetails in einer Sprache für Ihren Bot die Option Bearbeiten auswählen. Geben Sie dann Ihren gewünschten Wert in das Fenster mit dem Schwellenwert für die Vertrauensbewertung ein.

Verwenden von API-Operationen

- Um den Schwellenwert festzulegen, legen Sie den `nluIntentConfidenceThreshold` Parameter der [CreateBotLocale](#) Operation fest.
- Um den Konfidenzschwellenwert zu aktualisieren, legen Sie den `nluIntentConfidenceThreshold` Parameter der [UpdateBotLocale](#) Operation fest.

Sitzungsverwaltung

Um die Absicht zu ändern, die Amazon Lex V2 in einer Konversation mit dem Benutzer verwendet, können Sie die Antwort aus Ihrer Dialogcode-Hook-Lambda-Funktion verwenden, oder Sie können die Sitzungsverwaltung APIs in Ihrer benutzerdefinierten Anwendung verwenden.

Verwenden einer Lambda-Funktion mit Ihrem Lex V2-Bot

Wenn Sie eine Lambda-Funktion verwenden, ruft Amazon Lex V2 sie mit einer JSON-Struktur auf, die die Eingabe für die Funktion enthält. Die JSON-Struktur enthält ein Feld `currentIntent`, das die Absicht enthält, die Amazon Lex V2 als wahrscheinlichste Absicht für die Äußerung des Benutzers identifiziert hat. Die JSON-Struktur umfasst auch ein `alternativeIntents` Feld, das bis zu vier zusätzliche Absichten enthält, die der Absicht des Benutzers entsprechen können. Jede Absicht enthält ein Feld `intentConfidenceScore`, das den Konfidenzwert enthält, den Amazon Lex V2 der Absicht zugewiesen hat.

Um eine alternative Absicht zu verwenden, geben Sie sie in der Dialogaktion `ConfirmIntent` oder in der `ElicitSlot` Dialogaktion in Ihrer Lambda-Funktion an.

Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion in deinen Bot integrieren](#).

Verwenden der Session Management API mit Ihrem Lex V2-Bot

Um eine andere Absicht als die aktuelle Absicht zu verwenden, verwenden Sie die [PutSession](#) Operation. Wenn Sie beispielsweise entscheiden, dass die erste Alternative der von Amazon Lex V2 ausgewählten Absicht vorzuziehen ist, können Sie den `PutSession` Vorgang verwenden, um die Absichten so zu ändern, dass die nächste Absicht, mit der der Benutzer interagiert, die von Ihnen gewählte ist.

Weitere Informationen finden Sie unter [Grundlegendes zu Amazon Lex V2-Bot-Sitzungen](#).

Verwenden Sie Konfidenzwerte für die Sprachtranskription, um die Konversation mit Ihrem Lex V2-Bot zu verbessern

Wenn ein Benutzer eine Sprachäußerung macht, verwendet Amazon Lex V2 automatische Spracherkennung (ASR), um die Anfrage des Benutzers zu transkribieren, bevor sie interpretiert wird. Standardmäßig verwendet Amazon Lex V2 die wahrscheinlichste Transkription des Audiomaterials für die Interpretation.

In einigen Fällen kann es mehr als eine mögliche Transkription des Audios geben. Beispielsweise könnte ein Benutzer eine Äußerung mit einem mehrdeutigen Ton machen, z. B. „Mein Name ist John“, die als „Mein Name ist Juan“ verstanden werden könnte. In diesem Fall können Sie Techniken zur Begriffsklärung verwenden oder Ihr Fachwissen mit dem Transkriptionsvertrauenswert kombinieren, um festzustellen, welche Transkription in einer Liste von Transkriptionen die richtige ist.

Amazon Lex V2 umfasst die oberste Transkription und bis zu zwei alternative Transkriptionen für Benutzereingaben in der Anfrage an Ihre Lambda-Code-Hook-Funktion. Jede Transkription enthält einen Vertrauenswert, dass es sich um die richtige Transkription handelt. Jede Transkription enthält auch alle Slot-Werte, die aus der Benutzereingabe abgeleitet wurden.

Sie können die Konfidenzwerte zweier Transkriptionen vergleichen, um festzustellen, ob zwischen ihnen Unklarheiten bestehen. Wenn beispielsweise eine Transkription einen Konfidenzwert von 0,95 und die andere einen Konfidenzwert von 0,65 hat, ist die erste Transkription wahrscheinlich korrekt und die Mehrdeutigkeit zwischen ihnen ist gering. Wenn die beiden Transkriptionen Konfidenzwerte von 0,75 und 0,72 haben, ist die Ambiguität zwischen ihnen hoch. Möglicherweise können Sie anhand Ihres Fachwissens zwischen ihnen unterscheiden.

Wenn die abgeleiteten Slot-Werte in zwei Transkripten mit einem Konfidenzwert von 0,75 und 0,72 beispielsweise „John“ und „Juan“ lauten, können Sie die Benutzer in Ihrer Datenbank nach der Existenz dieser Namen abfragen und eine der Transkriptionen entfernen. Wenn „John“ kein Benutzer in Ihrer Datenbank ist und „Juan“ schon, können Sie den Dialogcode-Hook verwenden, um den abgeleiteten Slot-Wert für den Vornamen in „Juan“ zu ändern.

Bei den Konfidenzwerten, die Amazon Lex V2 zurückgibt, handelt es sich um Vergleichswerte. Verlassen Sie sich nicht auf sie als absolute Punktzahl. Die Werte können sich aufgrund von Verbesserungen an Amazon Lex V2 ändern.

Zuverlässigkeitswerte für Audiotranskriptionen sind nur in den Sprachen Englisch (GB) (en_GB) und Englisch (US) (en_US) verfügbar. Konfidenzwerte werden nur für 8-kHz-Audioeingänge unterstützt. Für die Audioeingabe aus dem [Testfenster](#) auf der Amazon Lex V2-Konsole werden keine Werte für die Transkriptionssicherheit bereitgestellt, da diese 16-kHz-Audioeingabe verwendet.

Note

Bevor Sie die Konfidenzwerte für die Audiotranskription mit einem vorhandenen Bot verwenden können, müssen Sie den Bot zunächst neu erstellen. Bestehende Versionen eines Bots unterstützen keine Transkriptionsvertrauenswerte. Sie müssen eine neue Version des Bots erstellen, um sie verwenden zu können.

Sie können Konfidenzwerte für mehrere Entwurfsmuster für Konversationen verwenden:

- Wenn der höchste Konfidenzwert aufgrund einer lauten Umgebung oder einer schlechten Signalqualität unter einen Schwellenwert fällt, können Sie den Benutzer mit derselben Frage auffordern, Audio in besserer Qualität aufzunehmen.
- Wenn mehrere Transkriptionen ähnliche Konfidenzwerte für Slot-Werte wie „John“ und „Juan“ aufweisen, können Sie die Werte mit einer bereits vorhandenen Datenbank vergleichen, um Eingaben zu vermeiden, oder Sie können den Benutzer auffordern, einen der beiden Werte auszuwählen. Beispiel: „Sagen Sie 1 für John oder 2 für Juan“.
- Wenn Ihre Geschäftslogik eine Absichtsumschaltung auf der Grundlage bestimmter Schlüsselwörter in einem alternativen Transkript mit einem Konfidenzwert nahe dem obersten Transkript erfordert, können Sie die Absicht mithilfe Ihrer Dialogcode-Hook-Lambda-Funktion oder mithilfe von Sitzungsverwaltungsoperationen ändern. Weitere Informationen finden Sie unter [Sitzungsverwaltung](#).

Amazon Lex V2 sendet die folgende JSON-Struktur mit bis zu drei Transkriptionen für die Benutzereingabe in Ihre Lambda-Code-Hook-Funktion:

```
"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"
            ]
        }
    }
]

```

Die JSON-Struktur enthält den Transkriptionstext, die Absicht, die für die Äußerung aufgelöst wurde, und Werte für alle in der Äußerung erkannten Slots. Für Textbenutzereingaben enthalten die Transkriptionen ein einzelnes Transkript mit einem Konfidenzwert von 1,0.

Der Inhalt der Transkripte hängt von der Wendung der Konversation und der erkannten Absicht ab.

Für die erste Runde, die Absichtserkennung, bestimmt Amazon Lex V2 die drei wichtigsten Transkriptionen. Für die oberste Transkription werden die Absicht und alle abgeleiteten Slot-Werte in der Transkription zurückgegeben.

Bei aufeinanderfolgenden Runden (Slot-Excitation) hängen die Ergebnisse wie folgt von der abgeleiteten Absicht für jede der Transkriptionen ab.

- Wenn die abgeleitete Absicht für das oberste Transkript dieselbe ist wie für die vorherige Runde und alle anderen Transkripte dieselbe Absicht haben, dann
 - Alle Transkripte enthalten abgeleitete Slot-Werte.
- Wenn sich die abgeleitete Absicht für das oberste Transkript von der vorherigen Runde unterscheidet und alle anderen Transkripte dieselbe Absicht haben, dann

- Das oberste Transkript enthält die abgeleiteten Slot-Werte für die neue Absicht.
 - Andere Transkripte enthalten die Werte für die vorherige Absicht und die abgeleiteten Slot-Werte für die vorherige Absicht.
-
- Wenn sich die abgeleitete Absicht für das oberste Protokoll von der vorherigen Runde unterscheidet, ein Transkript mit der vorherigen Absicht identisch ist und ein Transkript eine andere Absicht darstellt, dann
 - Das oberste Transkript enthält die neue abgeleitete Absicht und alle abgeleiteten Slot-Werte in der Äußerung.
 - Das Transkript, das die vorherige abgeleitete Absicht enthält, enthält abgeleitete Slot-Werte für diese Absicht.
 - Das Transkript mit der anderen Absicht hat keinen Namen für die abgeleitete Absicht und keine abgeleiteten Slot-Werte.
-
- Wenn sich die abgeleitete Absicht für das oberste Protokoll von der vorherigen Runde unterscheidet und alle anderen Abschriften unterschiedliche Absichten haben, dann
 - Das oberste Transkript enthält die neue abgeleitete Absicht und alle abgeleiteten Slot-Werte in der Äußerung.
 - Andere Transkripte enthalten keine abgeleiteten Absichten und keine abgeleiteten Slot-Werte.
-
- Wenn die abgeleitete Absicht für die ersten beiden Transkripte dieselbe ist und sich von der vorherigen Runde unterscheidet und die dritte Abschrift eine andere Absicht hat, dann
 - Die beiden obersten Transkripte enthalten die neue abgeleitete Absicht und alle abgeleiteten Slot-Werte in der Äußerung.
 - Das dritte Transkript hat keinen Namen für die Absicht und keine aufgelösten Slot-Werte.

Sitzungsverwaltung

Um die Absicht zu ändern, die Amazon Lex V2 in einer Konversation mit dem Benutzer verwendet, verwenden Sie die Antwort Ihrer Dialogcode-Hook-Lambda-Funktion. Oder Sie können die Sitzungsverwaltung APIs in Ihrer benutzerdefinierten Anwendung verwenden.

Verwenden einer Lambda-Funktion mit Ihrem Lex V2-Bot

Wenn Sie eine Lambda-Funktion verwenden, ruft Amazon Lex V2 sie mit einer JSON-Struktur auf, die die Eingabe für die Funktion enthält. Die JSON-Struktur enthält ein Feld `namenstranscriptions`, das die möglichen Transkriptionen enthält, die Amazon Lex V2 für die Äußerung bestimmt hat. Das `transcriptions` Feld enthält ein bis drei mögliche Transkriptionen mit jeweils einem Konfidenzwert.

Um die Absicht aus einer alternativen Transkription zu verwenden, geben Sie sie in der Dialogaktion `ConfirmIntent` oder in der `ElicitSlot` Dialogaktion in Ihrer Lambda-Funktion an. Um einen Slot-Wert aus einer alternativen Transkription zu verwenden, legen Sie den Wert in das `intent` Feld in Ihrer Lambda-Funktionsantwort fest. Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion in deinen Bot integrieren](#).

Beispielcode mit Lambda mit Lex V2

Das folgende Codebeispiel ist eine Python-Lambda-Funktion, die Audiotranskriptionen verwendet, um das Konversationserlebnis für den Benutzer zu verbessern.

Um den Beispielcode verwenden zu können, benötigen Sie:

- Ein Bot mit einer Sprache, entweder Englisch (GB) (`en_GB`) oder Englisch (US) (`en_US`).
- Eine Absicht, `OrderBirthStone` Stellen Sie sicher, dass die Option Lambda-Funktion für Initialisierung und Validierung verwenden im Abschnitt Code-Hooks der Absichtsdefinition ausgewählt ist.
- Die Absicht sollte zwei Slots haben, "BirthMonth" und „Name“, beide vom Typ. `AMAZON.AlphaNumeric`
- Ein Alias mit definierter Lambda-Funktion. Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion für Ihren Bot erstellen](#).

```
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)

```

Verwenden Sie die Sitzungsverwaltungs-API, um einen anderen Intent- oder Slot-Wert auszuwählen

Um eine andere Absicht als die aktuelle Absicht zu verwenden, verwenden Sie die [PutSession](#) Operation. Wenn Sie beispielsweise entscheiden, dass die erste Alternative der von Amazon Lex V2 ausgewählten Absicht vorzuziehen ist, können Sie den PutSession Vorgang verwenden, um Absichten zu ändern. Auf diese Weise ist die nächste Absicht, mit der der Benutzer interagiert, die, die Sie ausgewählt haben.

Sie können den PutSession Vorgang auch verwenden, um den Slot-Wert in der intent Struktur zu ändern, um einen Wert aus einer alternativen Transkription zu verwenden.

Weitere Informationen finden Sie unter [Grundlegendes zu Amazon Lex V2-Bot-Sitzungen](#).

Anpassen von Sprachtranskriptionen für die Verwendung mit Ihrem Lex V2-Bot

Das Standardverhalten Ihres Bots kann manchmal zu ungenauen Sprachtranskriptionen führen. Die folgenden Funktionen helfen Ihrem Bot dabei, Wörter oder Namen zu erkennen, die weniger häufig vorkommen oder leicht zu verwechseln sind.

Themen

- [Verbesserung der Spracherkennung mit einem benutzerdefinierten Wortschatz](#)
- [Verbesserung der Erkennung von Slot-Werten mit Laufzeithinweisen in der Konversation](#)
- [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#)

Verbesserung der Spracherkennung mit einem benutzerdefinierten Wortschatz

Sie können Amazon Lex V2 weitere Informationen zur Verarbeitung von Audiokonversationen mit einem Bot geben, indem Sie ein benutzerdefiniertes Vokabular in einer bestimmten Sprache erstellen. Ein benutzerdefiniertes Vokabular ist eine Liste bestimmter Ausdrücke, die Amazon Lex V2 in der Audioeingabe erkennen soll. Dies sind im Allgemeinen Eigennamen oder domänenspezifische Wörter, die Amazon Lex V2 nicht erkennt.

Nehmen wir zum Beispiel an, Sie haben einen Bot für technischen Support. Sie können einem benutzerdefinierten Vokabular „Backup“ hinzufügen, damit der Bot das Audio korrekt als „Backup“ transkribieren kann, auch wenn das Audio wie „zusammenpacken“ klingt. Ein benutzerdefiniertes Vokabular kann auch dabei helfen, seltene Wörter im Audio wie „Zahlungsfähigkeit“ für Finanzdienstleistungen oder Eigennamen wie „Cognito“ oder „Monitron“ zu erkennen.

Grundlagen des benutzerdefinierten Wortschatzes

- Ein benutzerdefiniertes Vokabular basiert auf der Transkription von Audioeingaben an einen Bot. Sie müssen Musteräußerungen angeben, um eine Absicht oder einen Slot-Wert zu erkennen.
- Ein benutzerdefiniertes Vokabular ist für eine bestimmte Sprache einzigartig. Sie müssen benutzerdefinierte Vokabulare für jede Sprache unabhängig konfigurieren. Benutzerdefinierte Vokabeln werden für die folgenden Sprachen unterstützt: Englisch (UK), Englisch (USA), Koreanisch (Korea), Italienisch (Italien), Deutsch (Deutschland), Französisch (Kanada), Französisch (Frankreich), Spanisch (Spanien), Spanisch (USA), Japanisch (Japan), Mandarin (VR China), Schwedisch (Schweden), Golfarabisch (Vereinigte Arabische Emirate), Hindi (Indien), Niederländisch (Niederlande), Norwegisch (Norwegen), Polnisch (Polen), Finnisch (Finnland), Portugiesisch (Portugal), Portugiesisch (Brasilien), Katalanisch (Spanien).
- Benutzerdefinierte Vokabulare sind mit [Contact-Center-Integrationen](#) verfügbar, die von Amazon Lex V2 unterstützt werden. Das [Testfenster](#) in der Amazon Lex V2-Konsole unterstützt benutzerdefinierte Vokabulare für alle Amazon Lex V2-Bots, die am oder nach dem 31. Juli 2022

erstellt wurden. Wenn Sie Probleme mit benutzerdefinierten Vokabularen im Testfenster haben, erstellen Sie den Bot neu und versuchen Sie es erneut.

Amazon Lex V2 verwendet benutzerdefinierte Vokabeln, um sowohl Absichten als auch Slots zu ermitteln. Dieselbe benutzerdefinierte Vokabeldatei wird für Absichten und Slots verwendet. Sie können die Funktion für benutzerdefiniertes Vokabular für einen Slot selektiv deaktivieren, wenn Sie einen Slot-Typ hinzufügen.

Eine Absicht wecken — Sie können ein benutzerdefiniertes Vokabular erstellen, um eine Absicht zu wecken. Diese Phrasen werden zur Transkription verwendet, wenn Ihr Bot die Absicht des Benutzers bestimmt. Wenn Sie beispielsweise den Ausdruck „Backup“ in Ihrem benutzerdefinierten Wortschatz konfiguriert haben, transkribiert Amazon Lex V2 die Benutzereingabe in „Können Sie bitte meine Fotos sichern?“ —auch wenn der Ton wie „Kannst du bitte meine Fotos zusammenpacken“ klingt. Sie können den Verstärkungsgrad für jede Phrase angeben, indem Sie eine Gewichtung von 0, 1, 2 oder 3 konfigurieren. Sie können auch eine alternative Darstellung für den Ausdruck in der letzten Ausgabe von Rede und Text angeben, indem Sie ein `displayAs` Feld hinzufügen.

Die benutzerdefinierten Vokabeln, die zur Verbesserung der Transkription bei der Absichtsermittlung verwendet werden, wirken sich nicht auf die Transkriptionen während der Erfassung von Zeitfenstern aus. Weitere Informationen zum Erstellen eines benutzerdefinierten Vokabulars für das Ermitteln von Absichten finden Sie unter [Erstellen Sie ein benutzerdefiniertes Vokabular zum Ermitteln von Absichten und Zeitabschnitten](#).

Ermitteln von benutzerdefinierten Zeitfenstern — Sie können ein benutzerdefiniertes Vokabular verwenden, um die Slot-Erkennung bei Audiokonversationen zu verbessern. Um die Fähigkeit Ihres Amazon Lex V2-Bots zu verbessern, Slot-Werte zu erkennen, erstellen Sie einen benutzerdefinierten Slot und fügen Sie die Slot-Werte zum benutzerdefinierten Slot hinzu. Wählen Sie dann Slot-Werte als benutzerdefiniertes Vokabular verwenden. Zu den Slot-Werten gehören beispielsweise Produktnamen, Kataloge oder Eigennamen. Sie sollten in benutzerdefinierten Vokabularen keine gebräuchlichen Wörter oder Ausdrücke wie „Ja“ und „Nein“ verwenden.

Nachdem die Slot-Werte hinzugefügt wurden, werden diese Werte verwendet, um die Slot-Erkennung zu verbessern, wenn der Bot Eingaben für den benutzerdefinierten Slot erwartet. Diese Werte werden nicht für die Transkription verwendet, wenn eine Absicht ausgelöst wird. Weitere Informationen finden Sie unter [Slot-Typen hinzufügen](#).

Bewährte Methoden für die Erstellung eines benutzerdefinierten Wortschatzes

Eine Absicht erwecken

- Benutzerdefinierte Vokabulare sind am besten zur Ausrichtung auf bestimmte Wörter oder Ausdrücke geeignet. Fügen Sie einem benutzerdefinierten Wortschatz nur Wörter hinzu, wenn sie von Amazon Lex V2 nicht ohne Weiteres erkannt werden.
- Entscheiden Sie, wie viel Gewicht einem Wort beigemessen werden soll, je nachdem, wie oft das Wort in der Transkription nicht erkannt wird und wie selten das Wort in der Eingabe vorkommt. Schwer auszusprechende Wörter erfordern eine höhere Gewichtung.
- Verwenden Sie ein repräsentatives Testset, um festzustellen, ob ein Gewicht angemessen ist. Sie können ein Audio-Testset sammeln, indem Sie die Audioprotokollierung in den Konversationsprotokollen aktivieren.
- Vermeiden Sie es, kurze Wörter wie „an“, „es“, „bis“, „ja“, „nein“ in einem benutzerdefinierten Wortschatz zu verwenden.

Einen benutzerdefinierten Slot ausloten

- Fügen Sie dem benutzerdefinierten Slot-Typ die Werte hinzu, von denen Sie erwarten, dass sie erkannt werden. Fügen Sie alle möglichen Slot-Werte für den benutzerdefinierten Slot-Typ hinzu, unabhängig davon, wie häufig oder selten der Slot-Wert ist.
- Aktivieren Sie die Option nur, wenn der benutzerdefinierte Slot-Typ eine Liste von Katalogwerten oder Entitäten wie Produktnamen oder Investmentfonds enthält.
- Deaktivieren Sie die Option, wenn der Slot-Typ verwendet wird, um allgemeine Ausdrücke wie „Ja“, „Nein“, „Ich weiß nicht“, „vielleicht“ oder allgemeine Wörter wie „eins“, „zwei“, „drei“ zu erfassen.
- Beschränken Sie die Anzahl der Slot-Werte und Synonyme auf 500 oder weniger, um eine optimale Leistung zu erzielen.

Geben Sie Akronyme oder andere Wörter, deren Buchstaben einzeln ausgesprochen werden sollen, als einzelne Buchstaben ein, die durch einen Punkt und ein Leerzeichen getrennt sind. Verwenden Sie keine einzelnen Buchstaben, es sei denn, sie sind Teil eines Ausdrucks, wie „J. P. Morgan“ oder „A. W. S.“ Sie können Groß- oder Kleinbuchstaben verwenden, um ein Akronym zu definieren.

Erstellen Sie ein benutzerdefiniertes Vokabular zum Ermitteln von Absichten und Zeitabschnitten

Sie können die Amazon Lex V2-Konsole verwenden, um ein benutzerdefiniertes Vokabular zu erstellen und zu verwalten, oder Sie können Amazon Lex V2-API-Operationen verwenden. Es gibt zwei Möglichkeiten, ein benutzerdefiniertes Vokabular über die Konsole zu erstellen:

Konsole

Benutzerdefiniertes Vokabular in die Konsole importieren:

1. Öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>
2. Wählen Sie aus der Liste der Bots den Bot aus, dem Sie das benutzerdefinierte Vokabular hinzufügen möchten.
3. Wählen Sie auf der Bot-Detailseite im Abschnitt Sprachen hinzufügen die Option Sprachen anzeigen aus.
4. Wählen Sie aus der Liste der Sprachen die Sprache aus, zu der Sie das benutzerdefinierte Vokabular hinzufügen möchten.

Erstellen Sie direkt über die Konsole ein neues benutzerdefiniertes Vokabular:

1. Klicken Sie auf der Seite mit den Sprachdetails im Bereich Benutzerdefinierter Wortschatz auf Erstellen. Dadurch wird ein Bearbeitungsfenster geöffnet, in dem kein benutzerdefiniertes Vokabular vorhanden ist.
2. Fügen Sie nach Bedarf Eingaben für Wortgruppe DisplayAs, und Gewichtung hinzu. Sie können zusätzliche Inline-Änderungen an hinzugefügten Elementen vornehmen, indem Sie deren Felder aktualisieren oder sie aus der Liste löschen.
3. Klicken Sie auf Speichern. Bitte beachte: Das neue benutzerdefinierte Vokabular wird erst in deinem Bot gespeichert, wenn du auf Speichern klickst.
4. Sie können auf dieser Seite weitere Inline-Änderungen vornehmen und auf Speichern klicken, wenn Sie fertig sind.
5. Auf dieser Seite können Sie auch eine benutzerdefinierte Vokabeldatei aus dem Drop-down-Menü oben rechts importieren, exportieren und löschen.

API

Verwenden Sie die **ListCustomVocabularyItems** API, um die benutzerdefinierten Vokabeleinträge anzuzeigen:

1. Verwenden Sie den ListCustomVocabularyItems Vorgang, um die benutzerdefinierten Vokabeleinträge anzuzeigen. Der Text der Anfrage wird wie folgt aussehen:

```
{
```

```
"maxResults": number,  
"nextToken": "string"  
}
```

2. Bitte beachten Sie, dass `maxResults` `nextToken` es sich bei den Feldern um optionale Felder für den Anfragetext handelt.
3. Die Antwort der `ListCustomVocabularyItems` Operation sieht wie folgt aus:

```
{  
  "botId": "string",  
  "botVersion": "string",  
  "localeId": "string",  
  "customVocabularyItems": [  
    {  
      "itemId": "string",  
      "phrase": "string",  
      "weight": number,  
      "displayAs": "string"  
    }  
  ]  
}
```

Verwenden Sie die **BatchCreateCustomVocabularyItem** API, um neue benutzerdefinierte Vokabeleinträge zu erstellen:

1. Wenn für das Gebietsschema Ihres Bots noch kein benutzerdefiniertes Vokabular erstellt wurde, folgen Sie bitte den Schritten, um ein benutzerdefiniertes Vokabular [StartImport](#) zu erstellen.
2. Nachdem das benutzerdefinierte Vokabular erstellt wurde, verwenden Sie den `BatchCreateCustomVocabularyItem` Vorgang, um neue benutzerdefinierte Vokabeleinträge zu erstellen. Der Text der Anfrage wird wie folgt aussehen:

```
{  
  "customVocabularyItemList": [  
    {  
      "phrase": "string",  
      "weight": number,  
      "displayAs": "string"  
    }  
  ]  
}
```

3. Bitte beachten Sie, dass `weight` `displayAs` es sich bei den Feldern um optionale Felder für den Anfragetext handelt.
4. Die Antwort von `BatchCreateCustomVocabularyItem` wird wie folgt aussehen:

```
{
  "botId": "string",
  "botVersion": "string",
  "localeId": "string",
  "errors": [
    {
      "itemId": "string",
      "errorMessage": "string",
      "errorCode": "string"
    }
  ],
  "resources": [
    {
      "itemId": "string",
      "phrase": "string",
      "weight": number,
      "displayAs": "string"
    }
  ]
}
```

5. Da es sich um einen Batch-Vorgang handelt, schlägt die Anforderung nicht fehl, wenn eines der Elemente nicht erstellt werden kann. Die Fehlerliste enthält Informationen darüber, warum der Vorgang für diesen bestimmten Eintrag fehlgeschlagen ist. Die Ressourcenliste wird alle Einträge enthalten, die erfolgreich erstellt wurden.
6. Denn `BatchCreateCustomVocabularyItem` Sie können mit folgenden Fehlern rechnen:
 - `RESOURCE_DOES_NOT_EXIST`: Das benutzerdefinierte Vokabular ist nicht vorhanden. Folgen Sie den Schritten zum Erstellen eines benutzerdefinierten Vokabulars, bevor Sie diesen Vorgang aufrufen.
 - `DUPLICATE_INPUT`: Die Liste der Eingaben enthält doppelte Phrasen.
 - `RESOURCE_ALREADY_EXISTS`: Die angegebene Phrase für den Eintrag ist bereits in Ihrem benutzerdefinierten Wortschatz vorhanden.
 - `INTERNAL_SERVER_FAILURE`: Bei der Bearbeitung Ihrer Anfrage ist ein Fehler im Backend aufgetreten. Dies kann auf einen Serviceausfall oder ein anderes Problem hinweisen.

Verwenden Sie die **BatchDeleteCustomVocabularyItem** API, um bestehende benutzerdefinierte Vokabeleinträge zu löschen:

1. Wenn für das Gebietsschema Ihres Bots noch kein benutzerdefiniertes Vokabular erstellt wurde, folgen Sie bitte den Schritten unter Verwenden Sie das, um ein benutzerdefiniertes Vokabular [StartImport](#) zu erstellen, um eines zu erstellen.
2. Nachdem das benutzerdefinierte Vokabular erstellt wurde, verwenden Sie den **BatchDeleteCustomVocabularyItem** Vorgang, um bestehende benutzerdefinierte Vokabeleinträge zu löschen. Der Text der Anfrage wird wie folgt aussehen:

```
{
  "customVocabularyItemList": [
    {
      "itemId": "string"
    }
  ]
}
```

3. Die Antwort von **BatchDeleteCustomVocabularyItem** wird wie folgt aussehen:

```
{
  "botId": "string",
  "botVersion": "string",
  "localeId": "string",
  "errors": [
    {
      "itemId": "string",
      "errorMessage": "string",
      "errorCode": "string"
    }
  ],
  "resources": [
    {
      "itemId": "string",
      "phrase": "string",
      "weight": number,
      "displayAs": "string"
    }
  ]
}
```


4. Da es sich um einen Batch-Vorgang handelt, schlägt die Anfrage nicht fehl, wenn eines der Elemente nicht gelöscht werden kann. Die Fehlerliste enthält Informationen darüber, warum der Vorgang für diesen bestimmten Eintrag fehlgeschlagen ist. Die Ressourcenliste wird alle Einträge enthalten, die erfolgreich gelöscht wurden.
5. Denn `BatchDeleteCustomVocabularyItem` Sie können mit folgenden Fehlern rechnen:
 - `RESOURCE_DOES_NOT_EXIST`: Der benutzerdefinierte Vokabeleintrag, den Sie löschen möchten, existiert nicht.
 - `INTERNAL_SERVER_FAILURE`: Bei der Bearbeitung Ihrer Anfrage ist ein Fehler im Backend aufgetreten. Dies kann auf einen Serviceausfall oder ein anderes Problem hinweisen.

Verwenden Sie die **`BatchUpdateCustomVocabularyItem`** API, um bestehende benutzerdefinierte Vokabeleinträge zu aktualisieren:

1. Wenn für das Gebietsschema Ihres Bots noch kein benutzerdefiniertes Vokabular erstellt wurde, folgen Sie bitte den Schritten unter Verwenden Sie das, um ein benutzerdefiniertes Vokabular [StartImport](#) zu erstellen, um ein benutzerdefiniertes Vokabular zu erstellen.
2. Nachdem das benutzerdefinierte Vokabular erstellt wurde, verwenden Sie den `BatchUpdateCustomVocabularyItem` Vorgang, um bestehende benutzerdefinierte Vokabeleinträge zu aktualisieren. Der Text der Anfrage wird wie folgt aussehen:

```
{
  "customVocabularyItemList": [
    {
      "itemId": "string",
      "phrase": "string",
      "weight": number,
      "displayAs": "string"
    }
  ]
}
```

3. Bitte beachten Sie, dass `weight` `displayAs` es sich bei den Feldern um optionale Felder für den Anfragetext handelt.
4. Die Antwort von `BatchUpdateCustomVocabularyItem` wird wie folgt aussehen:

```
{
  "botId": "string",
```

```
"botVersion": "string",
"localeId": "string",
"errors": [
  {
    "itemId": "string",
    "errorMessage": "string",
    "errorCode": "string"
  }
],
"resources": [
  {
    "itemId": "string",
    "phrase": "string",
    "weight": number,
    "displayAs": "string"
  }
]
}
```

5. Da es sich um einen Batch-Vorgang handelt, schlägt die Anfrage nicht fehl, wenn eines der Elemente nicht gelöscht werden kann. Die Fehlerliste enthält Informationen darüber, warum der Vorgang für diesen bestimmten Eintrag fehlgeschlagen ist. Die Ressourcenliste wird alle Einträge enthalten, die erfolgreich aktualisiert wurden.
6. Denn `BatchUpdateCustomVocabularyItem` Sie können mit folgenden Fehlern rechnen:
 - `RESOURCE_DOES_NOT_EXIST`: Der benutzerdefinierte Vokabeleintrag, den Sie aktualisieren möchten, ist nicht vorhanden.
 - `DUPLICATE_INPUT`: Die Liste der Eingaben enthält doppelte ItemIDs.
 - `RESOURCE_ALREADY_EXISTS`: Die angegebene Phrase für den Eintrag ist bereits in Ihrem benutzerdefinierten Wortschatz vorhanden.
 - `INTERNAL_SERVER_FAILURE`: Bei der Bearbeitung Ihrer Anfrage ist ein Fehler im Backend aufgetreten. Dies kann auf einen Serviceausfall oder ein anderes Problem hinweisen.

Eine benutzerdefinierte Vokabeldatei erstellen

Eine benutzerdefinierte Vokabeldatei ist eine durch Tabulatoren getrennte Liste von Werten, die den zu erkennenden Satz, eine Gewichtung zur Erhöhung der Bedeutung und ein `displayAs` Feld, das den Ausdruck im Sprachprotokoll ersetzt, enthält. Phrasen mit einem höheren Boost-Wert werden eher verwendet, wenn sie in der Audioeingabe erscheinen.

Die benutzerdefinierte Vokabeldatei muss benannt **CustomVocabulary.tsv** und in einer ZIP-Datei komprimiert werden, bevor sie importiert werden kann. Die ZIP-Datei muss weniger als 300 MB groß sein. Die maximale Anzahl von Ausdrücken in einem benutzerdefinierten Wortschatz beträgt 500.

- Satz 1—4 Wörter, die erkannt werden sollten. Trennen Sie die Wörter in der Phrase durch Leerzeichen. Die Datei darf keine doppelten Ausdrücke enthalten. Das Textfeld ist erforderlich.
- Gewicht — Der Grad, in dem die Phrasenerkennung verbessert wird. Der Wert ist eine Ganzzahl 0, 1, 2 oder 3. Wenn Sie kein Gewicht angeben, ist der Standardwert 1. Entscheiden Sie sich für die Gewichtung, je nachdem, wie oft das Wort in der Transkription nicht erkannt wird und wie selten das Wort in der Eingabe vorkommt. Die Gewichtung 0 bedeutet, dass kein Boosting angewendet wird und der Eintrag nur für die Durchführung von Ersetzungen anhand des `displayAs` Felds verwendet wird.
- DisplayAs — Definiert, wie Ihre Phrase in Ihrer Transkriptionsausgabe aussehen soll. Dies ist ein optionales Feld im benutzerdefinierten Wortschatz.

Die benutzerdefinierte Vokabeldatei muss eine Kopfzeile mit den Überschriften „Phrase“, „Gewicht“ und „DisplayAs“ enthalten. Die Überschriften können in beliebiger Reihenfolge sein, müssen aber der obigen Nomenklatur entsprechen.

Das folgende Beispiel ist eine benutzerdefinierte Vokabeldatei. Das Tabulatorzeichen, das zur Trennung von Wortgruppe, Gewichtung und DisplayAs erforderlich ist, wird durch den Text „[TAB]“ dargestellt. Wenn Sie dieses Beispiel verwenden, ersetzen Sie den Text durch ein Tabulatorzeichen.

```
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
```

Verbesserung der Erkennung von Slot-Werten mit Laufzeithinweisen in der Konversation

Mit Hinweisen zur Laufzeit können Sie Amazon Lex V2 je nach Kontext eine Reihe von Slot-Werten zuweisen, um eine bessere Erkennung bei Audiokonversationen und eine bessere Slot-Auflösung zu erreichen. Sie können Runtime-Hinweise verwenden, um zur Laufzeit eine Liste von Phrasen bereitzustellen, die als Kandidaten für die Auflösung eines Slot-Werts in Frage kommen.

Wenn beispielsweise ein Benutzer, der mit einem Flugreservierungs-Bot interagiert, häufig nach San Francisco, Jakarta, Seoul und Moskau reist, können Sie bei der Suche nach dem Ziel Laufzeithinweise mit einer Liste dieser vier Städte konfigurieren, um die Wiedererkennung häufig bereister Städte zu verbessern.

Laufzeithinweise sind nur in den Sprachen Englisch (USA) und Englisch (Großbritannien) verfügbar. Sie können mit den folgenden Slot-Typen verwendet werden:

- Benutzerdefinierte Slot-Typen
- Amazon.city
- Amazon.Land
- AMAZON. FirstName
- AMAZON. LastName
- Bundesstaat Amazonas
- AMAZON. StreetName

Grundlagen der Runtime-Hinweise

- Runtime-Hinweise werden nur verwendet, wenn einem Benutzer ein Slot-Wert abgefragt wird.
- Wenn Sie Runtime-Hinweise verwenden, werden die Werte der Hinweise ähnlichen Werten vorgezogen. Beispielsweise können Sie für einen Bot zur Essensbestellung eine Liste von Menüelementen als Laufzeithinweise festlegen und gleichzeitig für Lebensmittel in einem benutzerdefinierten Slot veranlassen, „Filet“ einem ähnlich klingenden „Kerl“ vorzuziehen.
- Wenn sich die Benutzereingabe von den in den Runtime-Hinweisen angegebenen Werten unterscheidet, wird die ursprüngliche Benutzereingabe für den Slot verwendet.
- Bei benutzerdefinierten Slot-Typen werden die als Runtime-Hinweise angegebenen Werte zur Auflösung des Slots verwendet, auch wenn sie bei der Bot-Erstellung nicht Teil des benutzerdefinierten Slots sind.
- Laufzeithinweise werden nur für 8-kHz-Audioeingänge unterstützt. Sie sind mit [Contact-Center-Integrationen](#) erhältlich, die von Amazon Lex V2 unterstützt werden. Für die Audioeingabe aus dem [Testfenster](#) auf der Amazon Lex V2-Konsole werden keine Laufzeithinweise bereitgestellt, da sie einen 16-kHz-Audioeingang verwendet.

 Note

Bevor Sie Runtime-Hinweise mit einem vorhandenen Bot verwenden können, müssen Sie den Bot zunächst neu erstellen. Bestehende Versionen eines Bots unterstützen keine Runtime-Hinweise. Sie müssen eine neue Version des Bots erstellen, um sie verwenden zu können.

Mithilfe der [StartConversation](#) Operation, oder können Sie Laufzeithinweise an [PutSession](#) Amazon Lex V2 senden. [RecognizeTextRecognizeUtterance](#) Sie können Laufzeithinweise auch mithilfe einer Lambda-Funktion hinzufügen.

Sie können zu Beginn einer Konversation Laufzeithinweise senden, um die Hinweise für jeden im Bot verwendeten Slot zu konfigurieren, oder Sie können Hinweise als Teil des Sitzungsstatus während einer Konversation senden. Das `runtimeHints` Attribut ordnet den Hinweisen für diesen Slot einen Slot zu.

Sobald Sie einen Laufzeithinweis an Amazon Lex V2 gesendet haben, bleiben sie für jede Runde der Konversation bestehen, bis die Sitzung endet. Wenn Sie eine `runtimeHints` Nullstruktur senden, werden die vorhandenen Hinweise verwendet. Sie können die Hinweise wie folgt ändern:

- Eine neue `runtimeHints` Struktur an den Bot senden. Der Inhalt der neuen Struktur ersetzt die vorhandenen.
- Senden einer leeren `runtimeHints` Struktur an den Bot. Dadurch werden die Runtime-Hinweise für den Bot gelöscht.

Slot-Werte im Kontext hinzufügen

Fügen Sie Kontext für Ihren Bot hinzu, indem Sie erwartete Slot-Werte als Laufzeithinweise angeben, wenn Ihre Anwendung Informationen über die nächste wahrscheinliche Äußerung des Benutzers hat. Fügen Sie Ihrem Bot einen Lambda-Dialog-Code-Hook hinzu ([Eine AWS Lambda Funktion in deinen Bot integrieren](#) weitere Informationen finden Sie unter) und verwenden Sie das `proposedNextState` Feld in, [AWS Lambda Eingabeereignisformat für Lex V2](#) um die Runtime-Hinweise zu ermitteln, die Sie einbeziehen sollten, um die Konversation mit dem Benutzer zu verbessern.

In einer Banking-App können Sie beispielsweise eine Liste mit Kontonamen für einen bestimmten Benutzer erstellen und diese Liste dann verwenden, um das Konto zu ermitteln, auf das der Benutzer zugreifen möchte.

Senden Sie zu Beginn der Konversation Laufzeithinweise, wenn Sie Kontext haben, damit Ihr Bot Benutzereingaben interpretieren kann. Wenn Sie beispielsweise die Telefonnummer des Benutzers kennen, können Sie anhand dieser Informationen nach dem Benutzer suchen, sodass Sie die `StartConversation Operation PutSession` oder verwenden können, um Hinweise auf Vor- und Nachnamen an den Bot weiterzuleiten, wenn Sie den Namen des Benutzers zur Überprüfung seiner Anmeldeinformationen erfragen möchten.

Während einer Konversation können Sie Informationen aus einem Slot-Wert sammeln, die bei einem anderen Slot-Wert hilfreich sein können. Wenn Sie beispielsweise in einer Autopflege-App die Kontonummer des Benutzers haben, können Sie nach den Autos suchen, die der Kunde besitzt, und sie als Hinweise an einen anderen Automaten weitergeben.

Geben Sie Akronyme oder andere Wörter, deren Buchstaben einzeln ausgesprochen werden sollen, als einzelne Buchstaben ein, die durch einen Punkt und ein Leerzeichen getrennt sind. Verwenden Sie keine einzelnen Buchstaben, es sei denn, sie sind Teil eines Ausdrucks, wie „J. P. Morgan“ oder „A.W.S“. Sie können Groß- oder Kleinbuchstaben verwenden, um ein Akronym zu definieren.

Hinweise zu einem Slot hinzufügen

Um Laufzeithinweise zu einem Slot hinzuzufügen, verwenden Sie die `runtimeHints` Struktur, die Teil der `sessionState` Struktur ist. Das Folgende ist ein Beispiel für die `runtimeHints` Struktur. Es enthält Hinweise für zwei Slots, "FirstName" und "LastName" für die Absicht `MakeAppointment` "".

```
{
  "sessionState": {
    "intent": {},
    "activeContexts": [],
    "dialogAction": {},
    "originatingRequestId": {},
    "sessionAttributes": {},
    "runtimeHints": {
      "slotHints": {
        "MakeAppointment": {
          "FirstName": {
            "runtimeHintValues": [
              {
                "phrase": "John"
```

```
{
    "firstName": "Mary",
    "lastName": {
        "runtimeHintValues": [
            {
                "phrase": "Stiles"
            },
            {
                "phrase": "Major"
            }
        ]
    }
}
```

Sie können auch eine Lambda-Funktion verwenden, um während einer Konversation Laufzeithinweise hinzuzufügen. Um Laufzeithinweise hinzuzufügen, fügen Sie die `runtimeHints` Struktur zum Sitzungsstatus der Antwort hinzu, die Ihre Lambda-Funktion an Amazon Lex V2 sendet. Weitere Informationen finden Sie unter [AWS Lambda Antwortformat für Lex V2](#).

Sie müssen slotName in der Anfrage ein gültiges intentName und angeben, andernfalls gibt Amazon Lex V2 einen Laufzeitfehler zurück.

Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation

Amazon Lex V2 bietet integrierte Steckplätze zur Erfassung benutzerspezifischer Informationen wie Vorname, Nachname, E-Mail-Adresse oder alphanumerische Identifikatoren. Sie können den AMAZON.LastName Slot beispielsweise verwenden, um Nachnamen wie „Jackson“ oder „Garcia“ zu erfassen. Amazon Lex V2 kann jedoch mit Nachnamen verwechselt werden, die schwer auszusprechen sind oder in einem bestimmten Land nicht üblich sind, wie z. B. „Xiulan“. Um solche Namen zu erfassen, können Sie den Benutzer bitten, ihn buchstabenweise oder wortweise einzugeben.

Amazon Lex V2 bietet drei Slot-Excitation-Stile, die Sie verwenden können. Wenn Sie einen Slot-Excitation-Stil festlegen, ändert sich dadurch die Art und Weise, wie Amazon Lex V2 die Benutzereingaben interpretiert.

Buchstabenweise — Mit diesem Stil können Sie den Bot anweisen, auf Rechtschreibweisen zu achten, anstatt auf die gesamte Phrase. Um beispielsweise einen Nachnamen wie „Xiulan“ zu erfassen, kannst du den Benutzer anweisen, seinen Nachnamen buchstabenweise buchstabieren zu lassen. Der Bot erfasst die Schreibweise und löst die Buchstaben zu einem Wort auf. Wenn der Benutzer beispielsweise „x i u l a n“ sagt, erfasst der Bot den Nachnamen als „xiulan“.

Buchstabieren nach Wort — In Sprachgesprächen, insbesondere am Telefon, gibt es einige Buchstaben wie „t“, „b“, „p“, die sich ähnlich anhören. Wenn die Erfassung alphanumerischer Werte oder die Schreibweise von Namen zu einem falschen Wert führt, können Sie den Benutzer auffordern, zusammen mit dem Buchstaben ein identifizierendes Wort einzugeben. Wenn die Sprachantwort auf eine Anfrage nach einer Buchungs-ID beispielsweise „abp123“ lautet, erkennt Ihr Bot stattdessen möglicherweise stattdessen den Ausdruck „ab b 123“. Wenn dies ein falscher Wert ist, können Sie den Benutzer bitten, die Eingabe als „a wie in Alpha b wie in Junge P wie in Peter eins zwei drei“ einzugeben. Der Bot löst die Eingabe auf „abp123“ auf.

Bei der Verwendung von Wort für Wort können Sie die folgenden Formate verwenden:

- „wie in“ (a wie bei Apple)
- „für“ (a für Apfel)
- „wie“ (ein wie ein Apfel)

Standard — Dies ist der natürliche Stil der Slot-Capture, bei der die Aussprache von Wörtern verwendet wird. So können beispielsweise Namen wie „John Stiles“ auf natürliche Weise erfasst werden. Wenn kein Slot-Elicitation-Stil angegeben ist, verwendet der Bot den Standardstil. Für die Slot-Typen `AMAZON.AlphaNumeric` und `AMAZON.UKPostal Code` unterstützt der Standardstil die buchstabenweise Eingabe.

Wenn der Name „Xiulan“ mit einer Mischung aus Buchstaben und Wörtern gesprochen wird, z. B. „x wie in X-ray i u l a s in lion a n“, muss der Stil für die Darstellung des Schlitzes auf `style` gesetzt werden. `spell-by-word` Der `spell-by-letter` Stil erkennt es nicht.

Für ein besseres Benutzererlebnis sollten Sie eine Sprachschnittstelle erstellen, die Slot-Werte mit einem natürlichen Konversationsstil erfasst. Bei Eingaben, die mit dem natürlichen Stil nicht korrekt

erfasst wurden, können Sie den Benutzer erneut auffordern und den Stil der Slot-Eingabe auf oder setzen. `spell-by-letter` `spell-by-word`

Sie können `spell-by-word` und `spell-by-letter` Styles für die folgenden Slot-Typen in den Sprachen Englisch (USA), Englisch (Großbritannien) und Englisch (Australien) verwenden:

- [AMAZON.AlphaNumeric](#)
- [AMAZON.EmailAddress](#)
- [AMAZON.FirstName](#)
- [AMAZON.LastName](#)
- [AMAZON.UKPostalCode](#)
- [Benutzerdefinierte Slot-Typen](#)

Rechtschreibung aktivieren

Sie aktivieren `spell-by-letter` und zur `spell-by-word` Laufzeit, wenn Sie dem Benutzer Slots entlocken. Sie können den Rechtschreibstil mit der Operation [PutSession](#), [RecognizeTextRecognizeUtterance](#), oder [StartConversation](#) festlegen. Sie können auch eine Lambda-Funktion aktivieren `spell-by-letter` und `spell-by-word` verwenden.

Sie legen den Rechtschreibstil mithilfe des `dialogAction` Felds in der `sessionState` Anforderung einer der oben genannten API-Operationen oder bei der Konfiguration der Lambda-Antwort fest ([AWS Lambda Antwortformat für Lex V2](#) weitere Informationen finden Sie unter). Sie können den Stil nur festlegen, wenn der Aktionstyp des Dialogs ist `ElicitSlot` und wenn es sich bei dem zu ermittelnden Slot um einen der unterstützten Slot-Typen handelt.

Der folgende JSON-Code zeigt das `dialogAction` Feld, das für die Verwendung des `spell-by-word` Stils festgelegt ist:

```
"dialogAction": {
  "slotElicitationStyle": "SpellByWord",
  "slotToElicit": "BookingId",
  "type": "ElicitSlot"
}
```

Das Feld `slotElicitationStyle` kann auf `SpellByLetter`, `SpellByWord` oder `Default` gesetzt werden. Wenn Sie keinen Wert angeben, wird der Wert auf `Default` gesetzt.

 Note

Sie können Stile nicht über spell-by-word die Konsole aktivieren spell-by-letter oder aufrufen.

Beispielcode mit Lambda und Lex V2

Das Ändern des Rechtschreibstils wird normalerweise durchgeführt, wenn der erste Versuch, einen Slot-Wert aufzulösen, nicht funktioniert hat. Das folgende Codebeispiel ist eine Python-Lambda-Funktion, die den spell-by-word Stil beim zweiten Versuch verwendet, einen Slot aufzulösen.

Um den Beispielcode verwenden zu können, benötigen Sie:

- Ein Bot mit einer Sprache, Englisch (GB) (en_GB).
- Eine Absicht "CheckAccount" mit einer Beispielaussage: „Ich möchte mein Konto überprüfen“. Stellen Sie sicher, dass im Abschnitt Code-Hooks der Absichtsdefinition die Option Lambda-Funktion für Initialisierung und Validierung verwenden ausgewählt ist.
- Die Absicht sollte einen Slot, "PostalCode,, des AMAZON.UKPostalCode integrierten Typs haben.
- Ein Alias mit definierter Lambda-Funktion. Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion für Ihren Bot erstellen](#).

```
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-elicite 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': 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx'
        },
        '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
```

Überwachung der Bot-Leistung in Lex V2

Die Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer Amazon Lex V2-Chatbots aufrechtzuerhalten. Dieses Thema beschreibt die Verwendung von Konversationsprotokollen zur Überwachung von Konversationen zwischen Ihren Benutzern und Ihren Chatbots, die Verwendung von Meinungsstatistiken, um zu ermitteln, welche Äußerungen Ihre Bots erkennen und übersehen, und wie Sie Amazon CloudWatch Logs verwenden und Amazon Lex AWS CloudTrail V2 überwachen können. Außerdem werden die Laufzeit- und Kanalzuordnungsmetriken von Amazon Lex V2 beschrieben.

Verwenden Sie diese Tools und Metriken, um zu verstehen, welche Richtungen und Maßnahmen Sie ergreifen können, um die Leistung Ihrer Bots zu verbessern.

Themen

- [Messung der Geschäftsleistung mit Analytics](#)
- [Konversationsprotokolle für Ihre Lex V2-Bots aktivieren](#)
- [Fehler mit Fehlerprotokollen in Lex V2 protokollieren](#)
- [Überwachung der Betriebskennzahlen in Lex V2](#)
- [Bewertung der Leistung von Lex V2-Bots mit der Test Workbench](#)

Messung der Geschäftsleistung mit Analytics

Mit Analytics können Sie die Leistung Ihres Bots anhand von Kennzahlen bewerten, die sich auf die Erfolgs- und Misserfolgsraten der Interaktionen Ihrer Bots mit Kunden beziehen. Sie können auch Muster von Konversationsabläufen zwischen Ihrem Bot und Kunden visualisieren. Analytics optimiert Ihre Erfahrung, indem es diese Kennzahlen in Grafiken und Diagrammen zusammenfasst. Analytics bietet Tools, mit denen Sie Ergebnisse filtern können, um Probleme und Probleme im Zusammenhang mit Absichten, Terminen, Äußerungen und Konversationen zu identifizieren. Sie können diese Daten verwenden, um Ihren Bot zu iterieren und zu verbessern, um ein besseres Kundenerlebnis zu schaffen.

Note

Damit ein Benutzer auf Analytics zugreifen kann, muss seiner IAM-Rolle entweder die [Von AWS verwaltete Richtlinie: AmazonLexFullAccess](#) oder eine benutzerdefinierte Richtlinie,

die Analytics-API-Berechtigungen enthält, zugewiesen werden. Einzelheiten [Verwaltung von Zugriffsberechtigungen für Analysen](#) zum Umgang mit Benutzerberechtigungen mit einer benutzerdefinierten Richtlinie finden Sie unter. Wenn der an die IAM-Rolle eines Kunden angehängt [Von AWS verwaltete Richtlinie: AmazonLexReadOnly](#) ist, werden in einer Fehlermeldung die fehlenden Berechtigungen angezeigt, die Sie der IAM-Rolle des Benutzers hinzufügen müssen, damit dieser auf die Analytics-Dashboards zugreifen kann.

Um auf Analytics zuzugreifen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie im Navigationsbereich unter Bots den Bot aus, den Sie in Analytics anzeigen möchten.
3. Wählen Sie unter Analytics den Abschnitt aus, den Sie anzeigen möchten.

Themen

- [Die wichtigsten Definitionen](#)
- [Filtern von Ergebnissen](#)
- [Überblick: Eine Zusammenfassung Ihrer Bot-Leistung](#)
- [Konversations-Dashboard: eine Zusammenfassung Ihrer Bot-Konversationen](#)
- [Leistungs-Dashboard: eine Zusammenfassung der Kennzahlen zu Absicht und Äußerung Ihres Bots](#)
- [Für Analysen verwenden APIs](#)
- [Verwaltung von Zugriffsberechtigungen für Analysen](#)

Die wichtigsten Definitionen

Dieses Thema enthält wichtige Definitionen, die Ihnen bei der Interpretation Ihrer Bot-Analysen helfen. Diese Definitionen beziehen sich auf die Leistung Ihres Bots in vier Kontexten: Intents, Slots, Conversations und Utterances. Die folgenden Felder sind für viele Leistungskennzahlen relevant:

- Das [stateFeld des Intent](#) Objekts.
- Das [typeFeld des dialogAction Objekts](#) innerhalb des [SessionState](#) Objekts.

Absichten

Amazon Lex V2 kategorisiert Absichten auf folgende Weise:

- Erfolg — Der Bot hat die Absicht erfolgreich erfüllt. Eine der folgenden Situationen ist wahr:
 - Die Absicht state ist ReadyForFulfillment und das type Von dialogAction ist Close.
 - Die Absicht state ist Fulfilled und das type Von dialogAction ist Close.
- Fehlgeschlagen — Der Bot hat die Absicht nicht erfüllt. Der Status der Absicht. Eine der folgenden Situationen ist wahr:
 - Die Absicht state ist Failed und type der Zweck dialogAction ist Close (z. B. hat der Benutzer die Bestätigungsaufforderung abgelehnt).
 - Der Bot wechselt zu, AMAZON.FallbackIntent bevor die Absicht abgeschlossen ist.
- Gewechselt — Der Bot erkennt eine andere Absicht und wechselt stattdessen zu dieser Absicht, bevor die ursprüngliche Absicht als erfolgreich oder gescheitert eingestuft wird.
- Abgesagt — Der Kunde reagiert erst, wenn die Absicht als erfolgreich oder gescheitert eingestuft wurde.

Slots

Amazon Lex V2 kategorisiert Steckplätze auf folgende Weise:

- Erfolgreich — Der Bot hat den Slot gefüllt und ist erfolgreich zu einem anderen Slot oder zum Bestätigungsschritt übergegangen.
- Fehlgeschlagen — Der Bot konnte den Slot nicht füllen, auch nachdem die maximale Anzahl von Wiederholungen erreicht wurde.
- Abgebrochen — Der Kunde reagiert nicht oder wechselt zu einer anderen Absicht, bevor der Slot als erfolgreich oder gescheitert eingestuft wird.

Konversationen

Wenn ein Kunde Amazon Lex V2 zur Laufzeit aufruft, stellt er eine bereit [sessionId](#) und Amazon Lex V2 generiert eine [originatingRequestId](#). Wenn der Kunde nicht innerhalb des Sitzungs-Timeouts ([idleSessionTTLInSeconds](#)) reagiert, das Sie für den Bot festgelegt haben, läuft die Sitzung ab. Wenn ein Kunde mit derselben Sitzung zur Sitzung zurückkehrt [sessionId](#), generiert Amazon Lex V2 eine neue [originatingRequestId](#).

Für Analysen ist eine Konversation eine einzigartige Kombination aus a `sessionId` und `anoriginatingRequestId`. Amazon Lex V2 kategorisiert Konversationen auf folgende Weise:

- Erfolg — Die endgültige Absicht in der Konversation wird als erfolgreich eingestuft.
- Fehlgeschlagen — Die letzte Absicht in der Konversation ist gescheitert. Die Konversation schlägt auch fehl, wenn Amazon Lex V2 standardmäßig auf den [AMAZON.FallbackIntent](#) Wert eingestellt ist.
- Abgebrochen — Der Kunde antwortet erst, wenn die Konversation als erfolgreich oder fehlgeschlagen eingestuft wurde.

Äußerungen

Amazon Lex V2 kategorisiert Äußerungen auf folgende Weise:

- Erkannt — Amazon Lex V2 erkennt die Äußerung als Versuch, eine für einen Bot konfigurierte Absicht aufzurufen.
- Verpasst — Amazon Lex V2 erkennt die Äußerung nicht.

Filtern von Ergebnissen

Oben auf jeder Seite können Sie die Ergebnisse für Ihre Bot-Analysen filtern.

Sie können nach den folgenden Parametern filtern:

- Zeit — Sie können Ergebnisse nach einem relativen oder absoluten Zeitraum filtern. Wenn Sie eine Start- und Endzeit auswählen, ruft Amazon Lex V2 Konversationen ab, die nach der Startzeit begonnen und vor der Endzeit beendet wurden.
- Relativer Bereich — Wählen Sie 1d, um die Ergebnisse des letzten Tages, 1w für die letzte Woche oder 1m für den letzten Monat zu sehen.

Für weitere Optionen wählen Sie Benutzerdefiniert und wählen Sie im Menü Relativer Bereich eine Dauer aus. Wenn Sie mehr Kontrolle über die Dauer haben möchten, wählen Sie Benutzerdefinierter Bereich aus, geben Sie eine Zahl in das Feld Dauer ein und wählen Sie im Dropdownmenü eine Zeiteinheit aus.

- Absoluter Bereich — Wählen Sie Benutzerdefiniert und wählen Sie das Menü Absoluter Bereich, um nach Konversationen innerhalb eines von Ihnen angegebenen Zeitraums zu filtern. Sie

können ein Start- und Enddatum im Kalender auswählen oder es im YYYY/MM/DD Format eingeben.

 Note

Für den Analysezeitraum gelten die folgenden Einschränkungen:

- Das Startdatum muss innerhalb der letzten 365 Tage nach dem aktuellen Datum liegen.
- Das Enddatum darf nicht mehr als 1 Monat nach dem Startdatum liegen.

- Bot-Filter — Um nach Gebietsschema, Alias und Version Ihres Bots zu filtern, wählen Sie die Dropdownmenüs mit den Bezeichnungen Alle Gebietsschemas, Alle Aliase und Alle Versionen aus.
- Modalität — Wähle das Zahnradsymbol und dann das Drop-down-Menü Modalität, um auszuwählen, ob Ergebnisse für Sprache oder Text angezeigt werden sollen.
- Kanal — Wählen Sie das Zahnradsymbol und anschließend das Dropdownmenü Kanal aus, um den Kanal auszuwählen, für den Sie Ergebnisse anzeigen möchten. Weitere Informationen zur Kanalintegration finden Sie unter [Integration eines Amazon Lex V2-Bots in eine Messaging-Plattform](#) und [Amazon Connect Connect-Kontaktzentren](#)

Überblick: Eine Zusammenfassung Ihrer Bot-Leistung

Auf der Übersichtsseite wird die Leistung Ihres Bots bei Konversationen, der Erkennung von Äußerungen und der Nutzung von Absichten zusammengefasst. Die Übersicht besteht aus den folgenden Abschnitten:

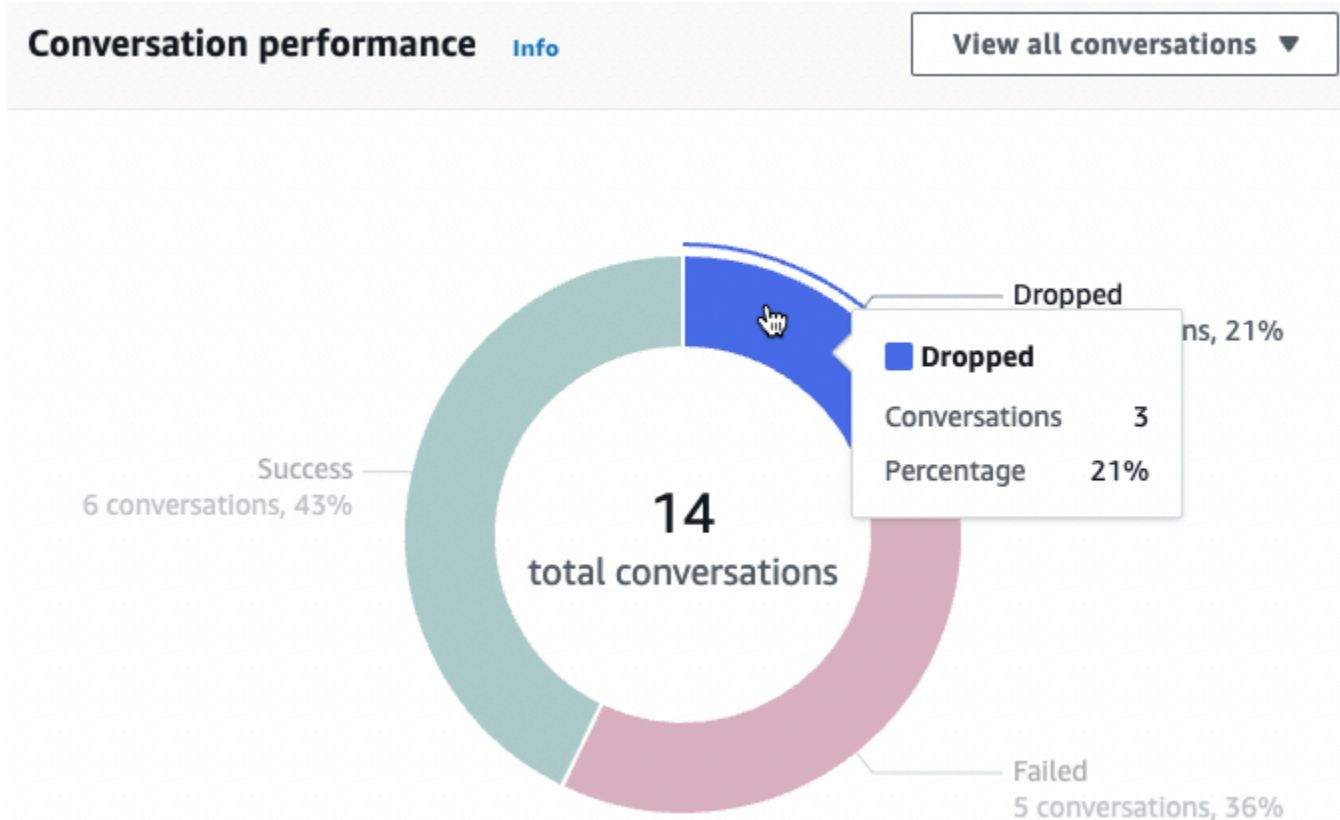
- [Leistung der Konversation](#)
- [Erkennungsrate von Äußerungen](#)
- [Leistungsverlauf der Konversation](#)
- [Die 5 am häufigsten verwendeten Absichten](#)
- [Die 5 am häufigsten fehlgeschlagenen Absichten](#)

Leistung der Konversation

Verwenden Sie dieses Diagramm, um die Anzahl und den Prozentsatz der Konversationen nachzuverfolgen, die als erfolgreich, gescheitert und abgebrochen eingestuft wurden. Um auf eine Liste von Konversationen zuzugreifen, wählen Sie Alle Konversationen anzeigen aus, um ein

Dropdownmenü einzublenden. Sie können wählen, ob Sie eine Liste aller Benutzerkonversationen mit dem Bot anzeigen oder nach Konversationen mit einem bestimmten Ergebnis filtern möchten (erfolgreich, fehlgeschlagen oder gelöscht). Über diese Links gelangen Sie zum Unterabschnitt Konversationen des Konversations-Dashboards. Weitere Informationen finden Sie unter [Konversationen](#).

Um ein Feld mit der Anzahl und dem Prozentsatz der Konversationen mit diesem Ergebnis einzublenden, bewegen Sie den Mauszeiger über ein Segment des Diagramms, wie in der folgenden Abbildung.

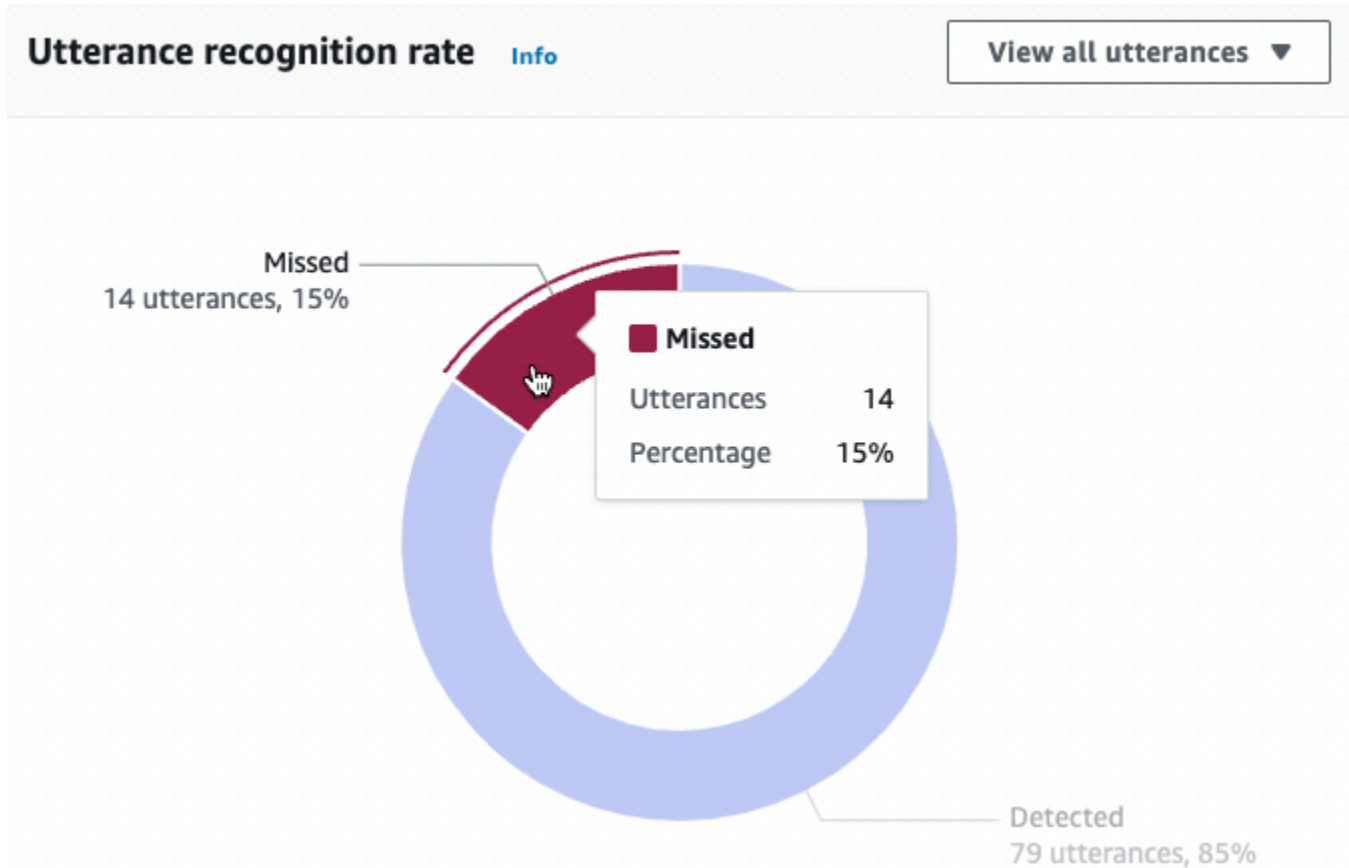


Erkennungsrate von Äußerungen

Verwenden Sie dieses Diagramm, um die Anzahl und den Prozentsatz der Äußerungen zu verfolgen, die von Ihrem Bot erkannt und übersehen wurden. Um auf eine Liste von Äußerungen zuzugreifen, wählen Sie Äußerungen anzeigen aus, um ein Drop-down-Menü einzublenden. Sie können wählen, ob Sie eine Liste aller Benutzeräußerungen anzeigen oder nach Äußerungen mit einem bestimmten Ergebnis (übersehen oder erkannt) filtern möchten. Über diese Links gelangen Sie zum Unterabschnitt zur Erkennung von Äußerungen im Leistungs-Dashboard. Weitere

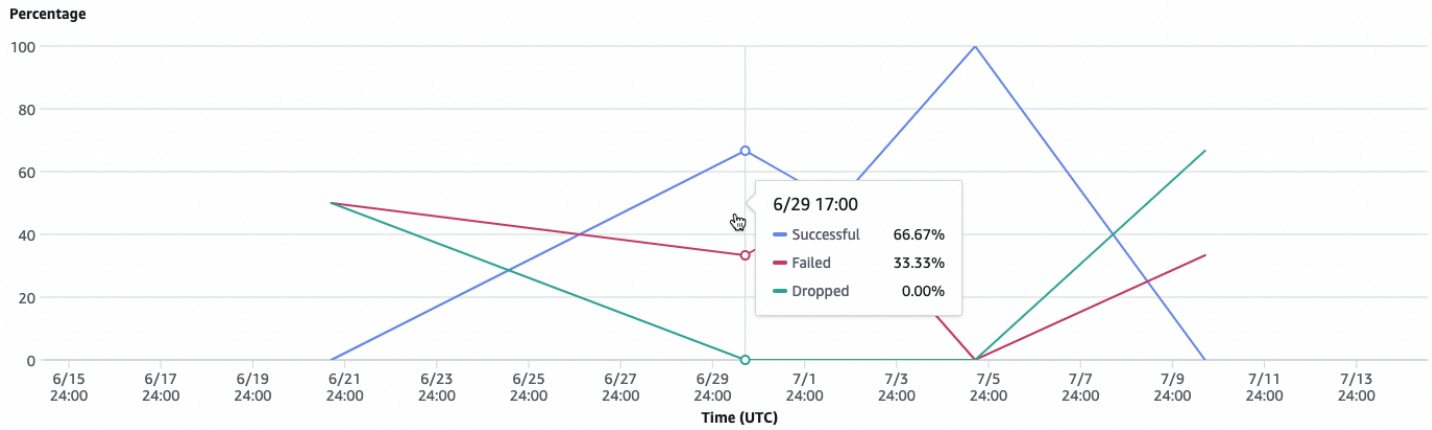
Informationen finden Sie unter Äußerungen anzeigen, zu denen Sie navigieren können. [Erkennung von Äußerungen](#)

Um ein Feld mit der Anzahl und dem Prozentsatz der Äußerungen einzublenden, bewegen Sie den Mauszeiger über ein Segment des Diagramms, wie in der folgenden Abbildung dargestellt.



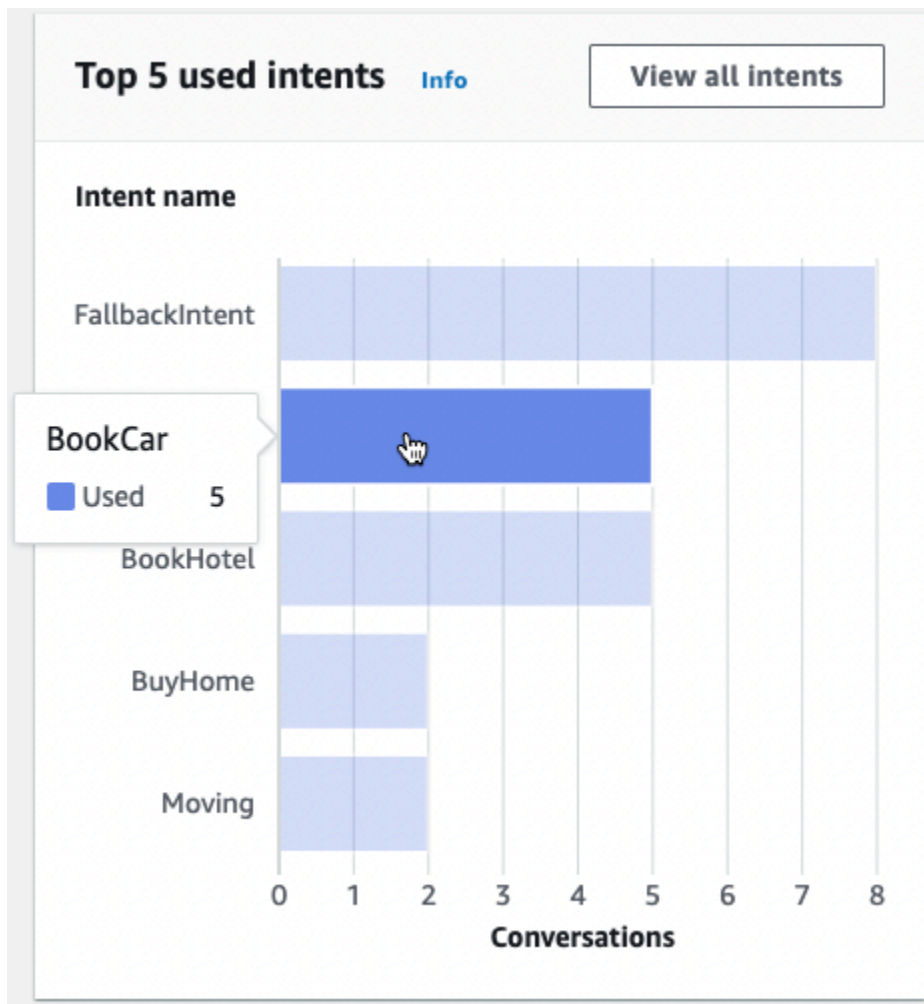
Leistungsverlauf der Konversation

Verwenden Sie dieses Diagramm, um den Prozentsatz der Konversationen zu verfolgen, die über den Zeitraum, den Sie in den Filtern festgelegt haben, als erfolgreich, fehlgeschlagen und abgebrochen eingestuft wurden. Um den Prozentsatz der Konversationen mit einem bestimmten Ergebnis in einem bestimmten Zeitraum zu sehen, bewegen Sie den Mauszeiger über dieses Intervall, wie in der folgenden Abbildung.

Conversation performance history [Info](#)

Die 5 am häufigsten verwendeten Absichten

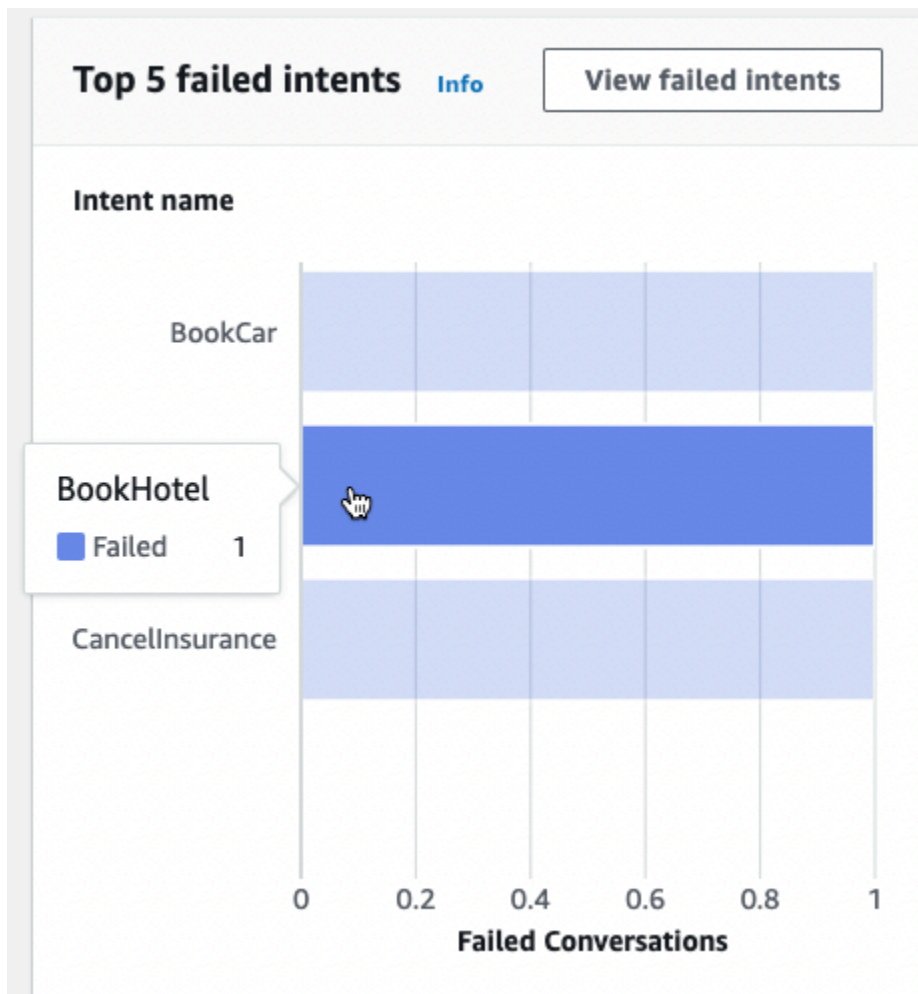
Verwenden Sie dieses Diagramm, um die fünf wichtigsten Absichten zu identifizieren, die Kunden mit Ihrem Bot verwendet haben. Bewegen Sie den Mauszeiger über einen Balken, um zu sehen, wie oft Ihr Bot diese Absicht erkannt hat, wie in der folgenden Abbildung.



Wählen Sie Alle Absichten anzeigen aus, um zum Unterabschnitt Intents Performance des Performance-Dashboards zu gelangen. Dort können Sie sich Kennzahlen zur Leistung Ihres Bots bei der Erfüllung von Intents ansehen. Weitere Informationen finden Sie unter [Leistung der Absichten](#).

Die 5 am häufigsten fehlgeschlagenen Absichten

Identifizieren Sie anhand dieses Diagramms die fünf wichtigsten Absichten, die Ihr Bot nicht erfüllt hat (die Definition einer fehlgeschlagenen Absicht finden Sie unter [Absichten](#)). Bewegen Sie den Mauszeiger über einen Balken, um zu sehen, wie oft Ihr Bot diese Absicht nicht erfüllt hat, wie in der folgenden Abbildung dargestellt.



Wählen Sie Fehlgeschlagene Absichten anzeigen aus, um zum Unterabschnitt Intents Performance des Performance-Dashboards zu gelangen. Dort können Sie sich Kennzahlen zu den Absichten anzeigen lassen, die Ihr Bot nicht erfüllt hat. Weitere Informationen finden Sie unter [Leistung der Absichten](#).

Konversations-Dashboard: eine Zusammenfassung Ihrer Bot-Konversationen

Das Konversations-Dashboard visualisiert Messwerte für Kundengespräche (die Definition einer Konversation finden [Konversationen](#) Sie unter) mit Ihrem Bot.

Die Zusammenfassung enthält die folgenden Informationen zu Benutzerkonversationen mit Ihrem Bot. Die Zahlen werden auf der Grundlage der Filtereinstellungen berechnet.

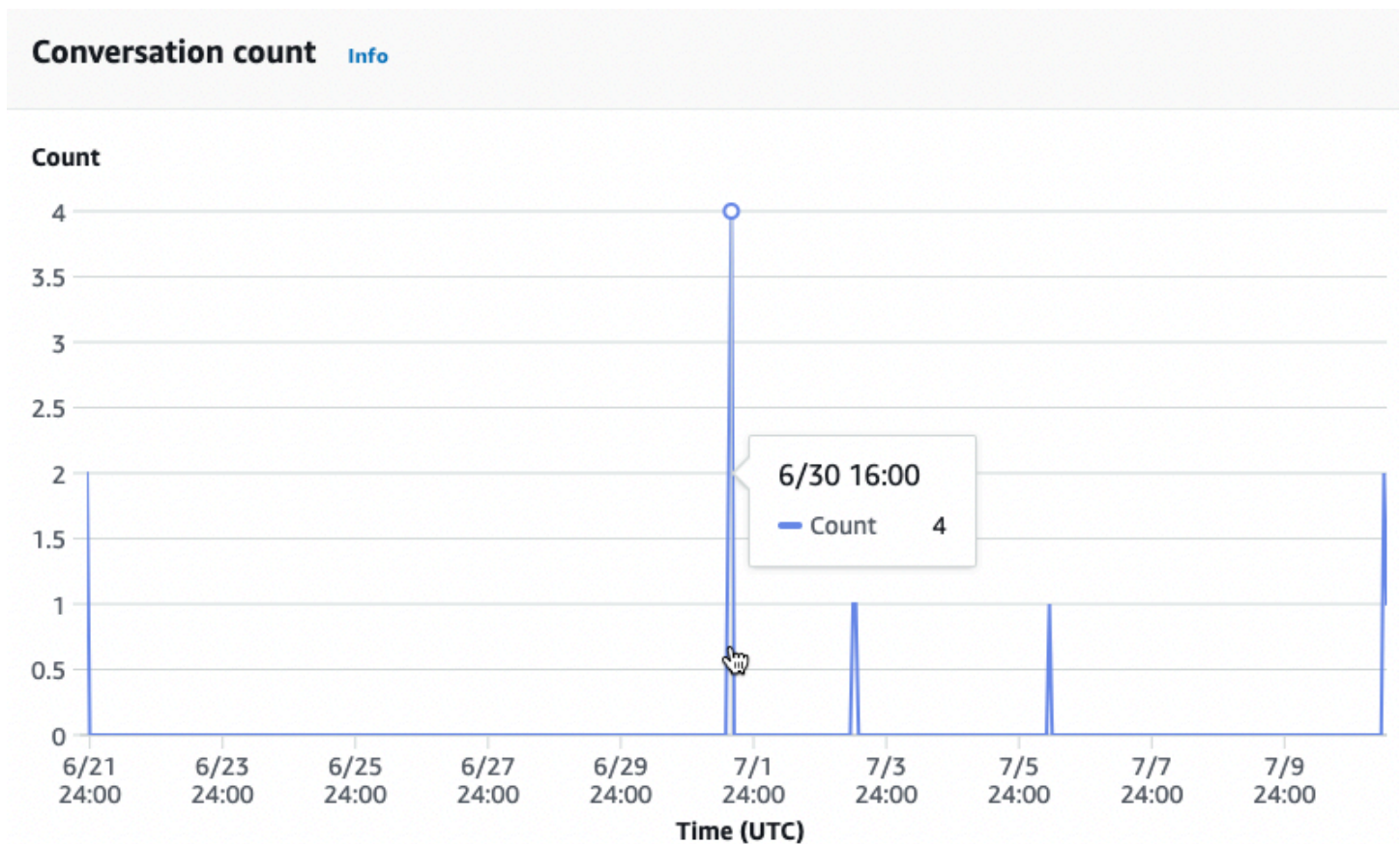
- Konversationen insgesamt — Die Gesamtzahl der Konversationen mit dem Bot.

- **Durchschnittliche Gesprächsdauer** — Die durchschnittliche Dauer der Benutzerkonversationen mit dem Bot in Minuten und Sekunden. Das Format ist mm:ss.
- **Durchschnittliche Anzahl an Runden pro Konversation** — Die durchschnittliche Anzahl von Runden, die eine Konversation dauert.

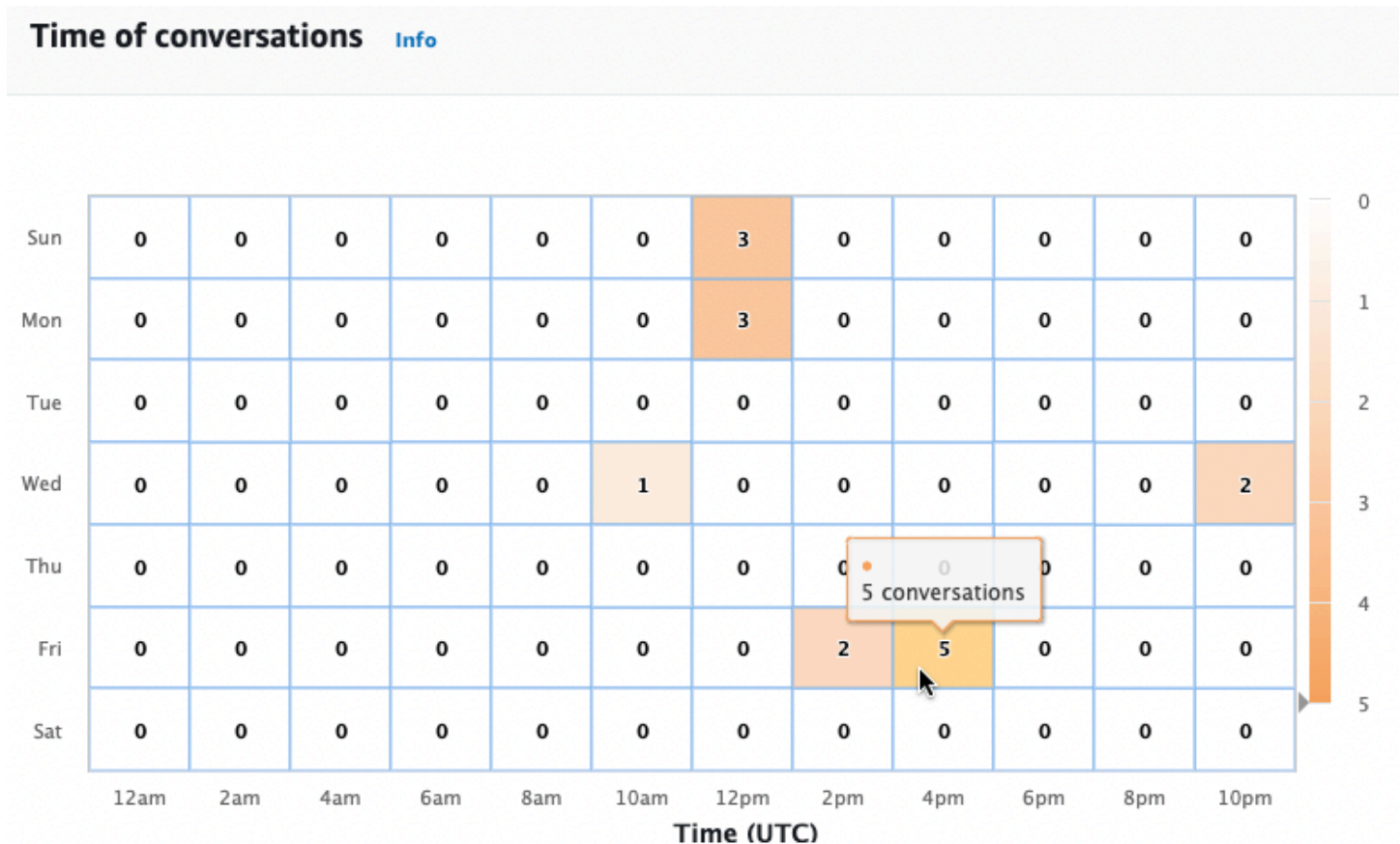
Die Abschnitte Anzahl der Konversationen und Anzahl der Nachrichten enthalten jeweils ein Diagramm, das die Anzahl der Konversationen bzw. Nachrichten über den Zeitraum anzeigt, den Sie in Ihren Filtern angeben. Zeigen Sie mit der Maus auf ein Zeitsegment, um die Anzahl der Konversationen oder Nachrichten in diesem Segment zu sehen. Die Größe des Zeitsegments hängt vom angegebenen Zeitraum ab:

- **Weniger als 1 Woche** — Die Anzahl wird für jede Stunde angezeigt.
- **1 Woche oder länger** — Die Anzahl wird für jeden Tag angezeigt.

In der folgenden Abbildung sehen Sie ein Beispiel für das Verhalten beim Schweben.



Im Abschnitt Zeit der Konversationen wird die Anzahl der Konversationen zwischen Ihrem Bot und Kunden in jedem Zwei-Stunden-Intervall an jedem Wochentag innerhalb des von Ihnen in den Filtern angegebenen Zeitraums angezeigt. Dunklere Felder geben an, zu welchen Zeiten mehr Konversationen stattgefunden haben. Zeigen Sie mit der Maus auf ein Feld, um die Anzahl der Konversationen in den 2 Stunden ab diesem Zeitfenster anzuzeigen. Die Aktion in der folgenden Abbildung zeigt beispielsweise die Anzahl der Konversationen, die zwischen 16:00 Uhr und 18:00 Uhr UTC stattfanden.



Das Konversations-Dashboard enthält zwei Tools: Konversationsflüsse und Konversationen. Greifen Sie auf ein Tool zu, indem Sie es im linken Navigationsbereich unter Konversations-Dashboard auswählen.

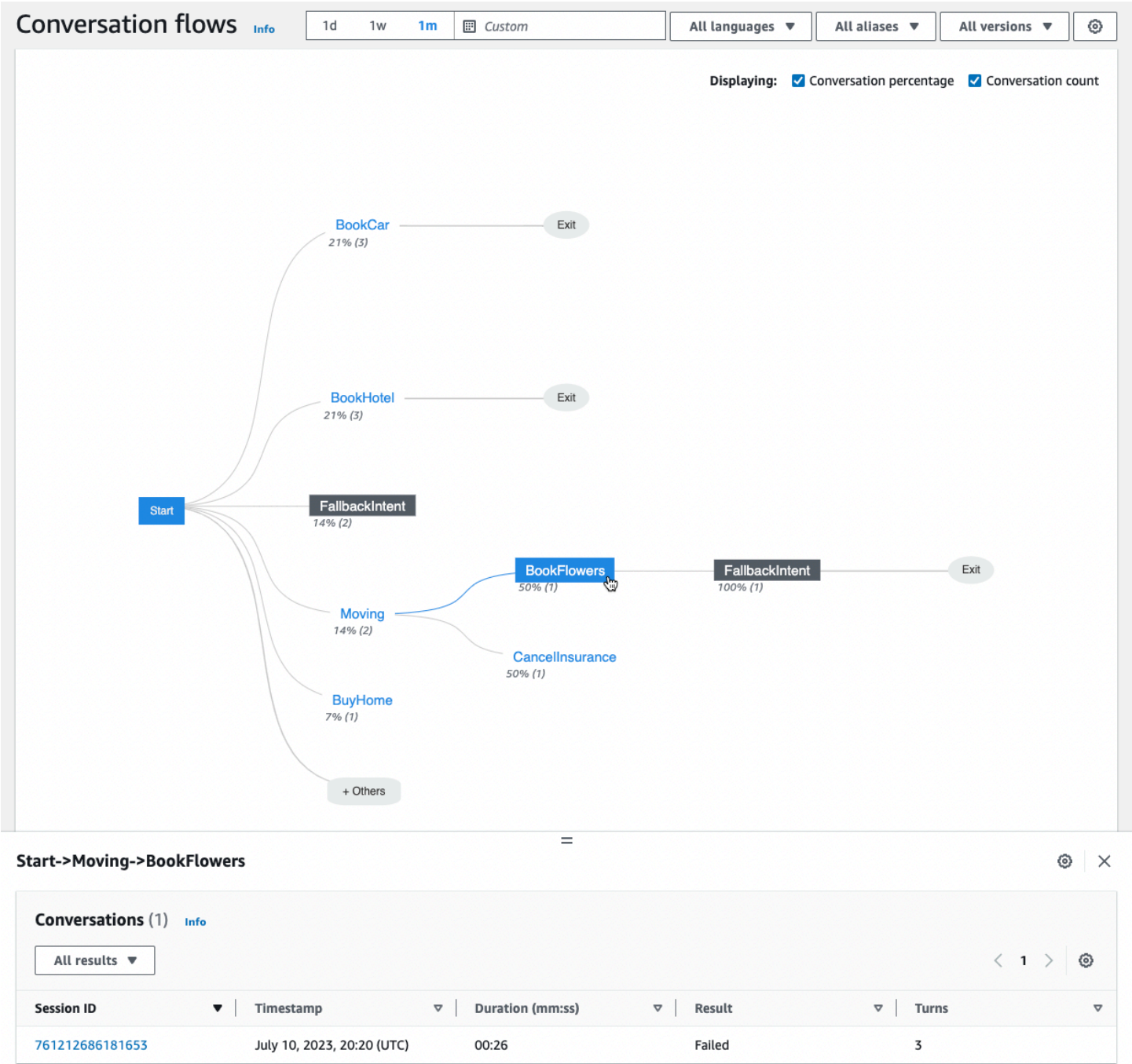
Konversationsabläufe

Verwenden Sie Konversationsabläufe, um die Reihenfolge der Absichten zu visualisieren, die Kunden in Gesprächen mit Ihrem Bot annehmen. Unter jeder Absicht befinden sich der Prozentsatz und die Anzahl der Konversationen, die zu diesem Zeitpunkt in der Konversation zu dieser Absicht aufgerufen haben. Sie können den Prozentsatz und die Anzahl ausschalten, indem Sie oben die Optionen Konversationsprozentsatz und Anzahl Konversationen auswählen. Standardmäßig werden die fünf

häufigsten Absichten zu diesem Zeitpunkt in der Konversation in absteigender Reihenfolge ihrer Häufigkeit angezeigt. Wählen Sie + Andere, um alle Absichten anzuzeigen.

Wählen Sie eine Absicht aus, um zu einer neuen Spalte mit Verzweigungen zu gelangen, in der eine Liste der an diesem Punkt der Konversation getroffenen Absichten angezeigt wird, sortiert nach absteigender Häufigkeit.

Wenn Sie einen Knoten im Konversationsablauf auswählen, können Sie das Fenster darunter erweitern, sodass eine Liste der Konversationen angezeigt wird, die dieser Reihenfolge gefolgt sind. Wählen Sie die Sitzungs-ID, die einer Konversation entspricht, um Details zu dieser Konversation anzuzeigen. Die folgende Abbildung zeigt einen Konversationsablauf und unten ein erweitertes Konversationsfenster.



Konversationen

Das Conversations-Tool zeigt eine Liste von Konversationen für Ihren Bot an. Sie können eine Spalte auswählen, um nach dieser Spalte in aufsteigender oder absteigender Reihenfolge zu sortieren.

Um die Konversationen nach Ergebnis zu filtern, wählen Sie Alle Ergebnisse und anschließend Erfolgreich, Fehlgeschlagen oder Abgebrochen aus.

Um die Konversationen nach Dauer zu filtern

1. Wählen Sie die Suchleiste mit der Bezeichnung Konversationen nach Dauer filtern aus
2. Definieren Sie den Filter auf eine der folgenden Arten:
 - Verwenden Sie die vordefinierten Optionen.
 - a. Wählen Sie Dauer aus.
 - b. Wählen Sie zwischen den Operatoren = (gleich), > (größer als) und < (kleiner als).
 - c. Wählen Sie eine Zeitdauer aus.
 - Geben Sie eine Eingabe im Format „Dauer {Operator} {Zahl} Sekunde“ ein **Duration > 30 sec**. Geben Sie beispielsweise ein, um nach allen Konversationen zu suchen, die länger als 30 Sekunden dauern. Geben Sie die Zeitdauer in Sekunden an.

Wählen Sie die Sitzungs-ID einer Konversation aus, um detaillierte Informationen über die Sitzung anzuzeigen, einschließlich Metadaten, Verwendung der Absicht und ein Protokoll.

Note

Da es sich bei einer Konversation um eine eindeutige Kombination aus einem `sessionId` und `handeltoriginatingRequestId`, `sessionId` kann dieselbe Konversation in der Tabelle mehrfach vorkommen.

Der Abschnitt „Details“ enthält die folgenden Metadaten:

- Zeitstempel — Gibt das Datum und die Startzeit der Konversation an. Die Uhrzeit ist im Format hh:mm:ss.
- Dauer — Gibt im Format mm:ss an, wie lange die Konversation gedauert hat. Die Dauer beinhaltet nicht die Dauer des Sitzungs-Timeouts (). `idleSessionTTLInSeconds`
- Ergebnis — Gibt an, ob die Konversation als erfolgreich, fehlgeschlagen oder abgebrochen eingestuft wurde. [Konversationen](#) Weitere Informationen zu diesen Ergebnissen finden Sie unter.
- Modus — Gibt an, ob es sich bei der Konversation um `SpeechText`, oder DTMF (Tastatureingaben) handelte. Eine Konversation, die aus mehreren Modi besteht, ist `Multimode`
- Kanal — Gibt den Kanal an, auf dem die Konversation stattgefunden hat, falls zutreffend. Siehe [Integration eines Amazon Lex V2-Bots in eine Messaging-Plattform](#).
- Sprache — Gibt die Sprache des Bots an.

Die Absichten, die der Bot in der Konversation hervorgerufen hat, sind unter „Details“ aufgeführt. Wählen Sie Gehe zu Absicht, um im Absichtseditor zu dieser Absicht zu gelangen. Wählen Sie „An Abschrift ausrichten“, um das Transkript automatisch zur ersten Instanz zu verschieben, in der der Bot die Absicht ausgelöst hat.

Klicken Sie auf den Rechtspfeil neben dem Namen der Absicht, um Details zu den Slots anzuzeigen, die für die Absicht ausgelöst wurden, einschließlich der Namen der Slots, des Werts, den der Bot für jeden Slot abgerufen hat, und der Anzahl der Versuche, die einzelnen Slots auszulösen.

Mit dem Transkript können Sie die Äußerungen der Konversation und das Verhalten Ihres Bots bei der Erfassung von Absichten und Slots überprüfen. Benutzeräußerungen werden auf der linken Seite und Bot-Äußerungen auf der rechten Seite angezeigt. Verwenden Sie die Suchleiste mit der Aufschrift „Transkripte in dieser Sitzung filtern“, um Text im Protokoll zu finden. Neben „Angezeigt“: Unter jeder Konversationsrunde werden drei Informationen angezeigt, von denen Sie auswählen können, ob sie angezeigt werden sollen oder nicht:

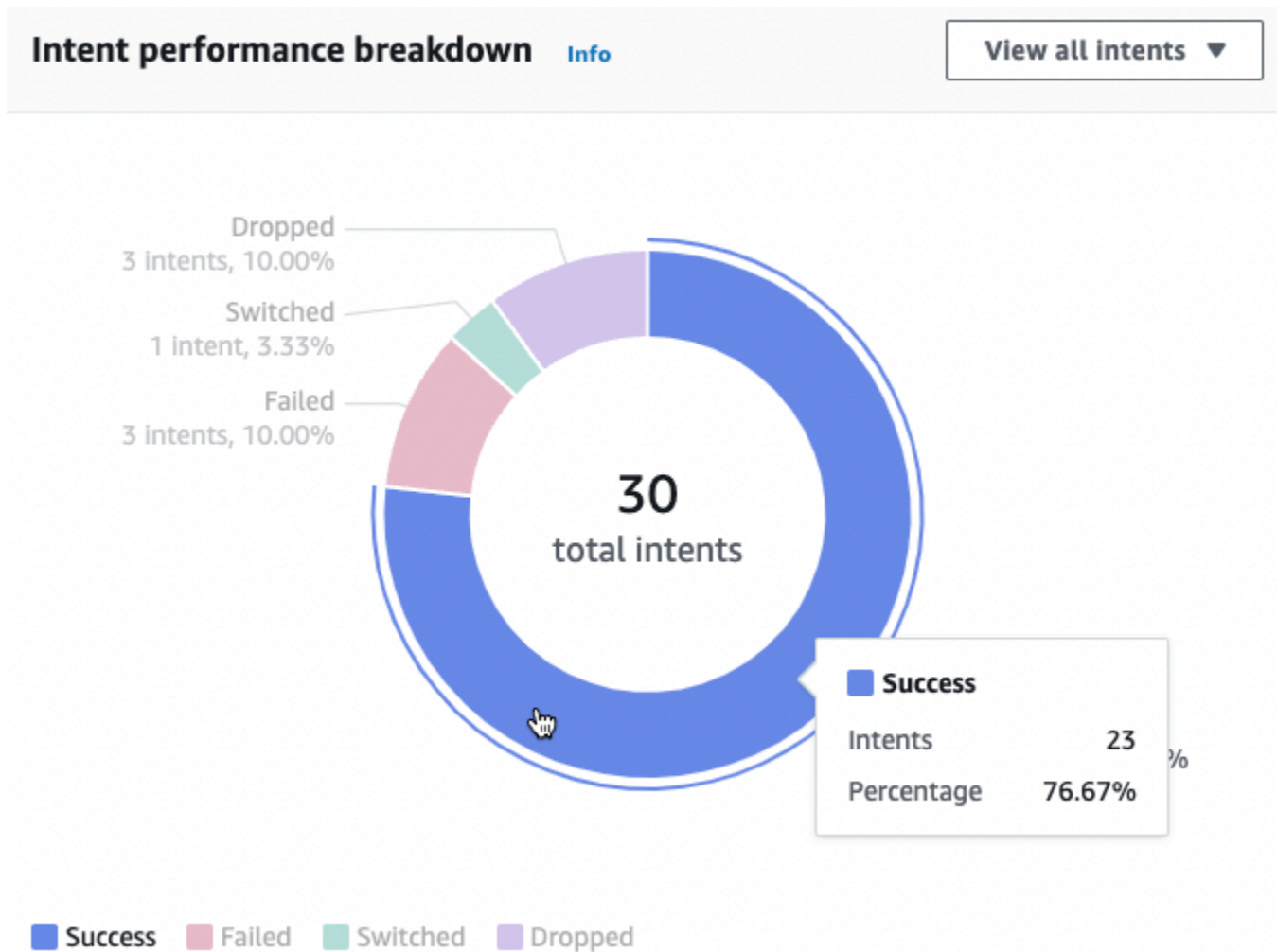
- Zeitstempel — Gibt den Zeitpunkt der Äußerung an.
- Status der Absicht — Gibt die Absicht an, die der Bot während einer Äußerung hervorruft, und gegebenenfalls das Ergebnis der Absicht. Die folgenden Absichtsstatus sind möglich:
 - Aufgerufene Absicht: *intent name* — Der Bot hat eine Absicht identifiziert, auf die sich der Kunde beruft.
 - Geänderte Absicht: *intent name* — Der Bot hat aufgrund der Äußerung zu einer anderen Absicht gewechselt.
 - *intent name*: Erfolg — Der Bot hat die Absicht erfüllt.
- Slot-Status — Gibt gegebenenfalls den Slot an, den der Bot während einer Äußerung auslöst, und den Wert, den der Kunde bietet.

Leistungs-Dashboard: eine Zusammenfassung der Kennzahlen zu Absicht und Äußerung Ihres Bots

Im Leistungs-Dashboard können Sie sich Details zur Leistung der Absichtserfüllung und der Erkennung von Äußerungen durch Ihren Bot anzeigen lassen.

Im Abschnitt Aufschlüsselung der Leistung von Absichten wird die Gesamtzahl der Aufrufe einer Absicht durch Ihren Bot angezeigt. Außerdem wird aufgeschlüsselt, wie oft und wie oft die Absichten als erfolgreich, fehlgeschlagen, gelöscht und umgestellt wurden. Eine Erläuterung dieser Definitionen finden Sie unter [Absichten](#). Zeigen Sie mit der Maus auf ein Segment des Diagramms, um ein Feld

mit der Anzahl und dem Prozentsatz der Konversationen mit diesem Ergebnis anzuzeigen, wie in der folgenden Abbildung.



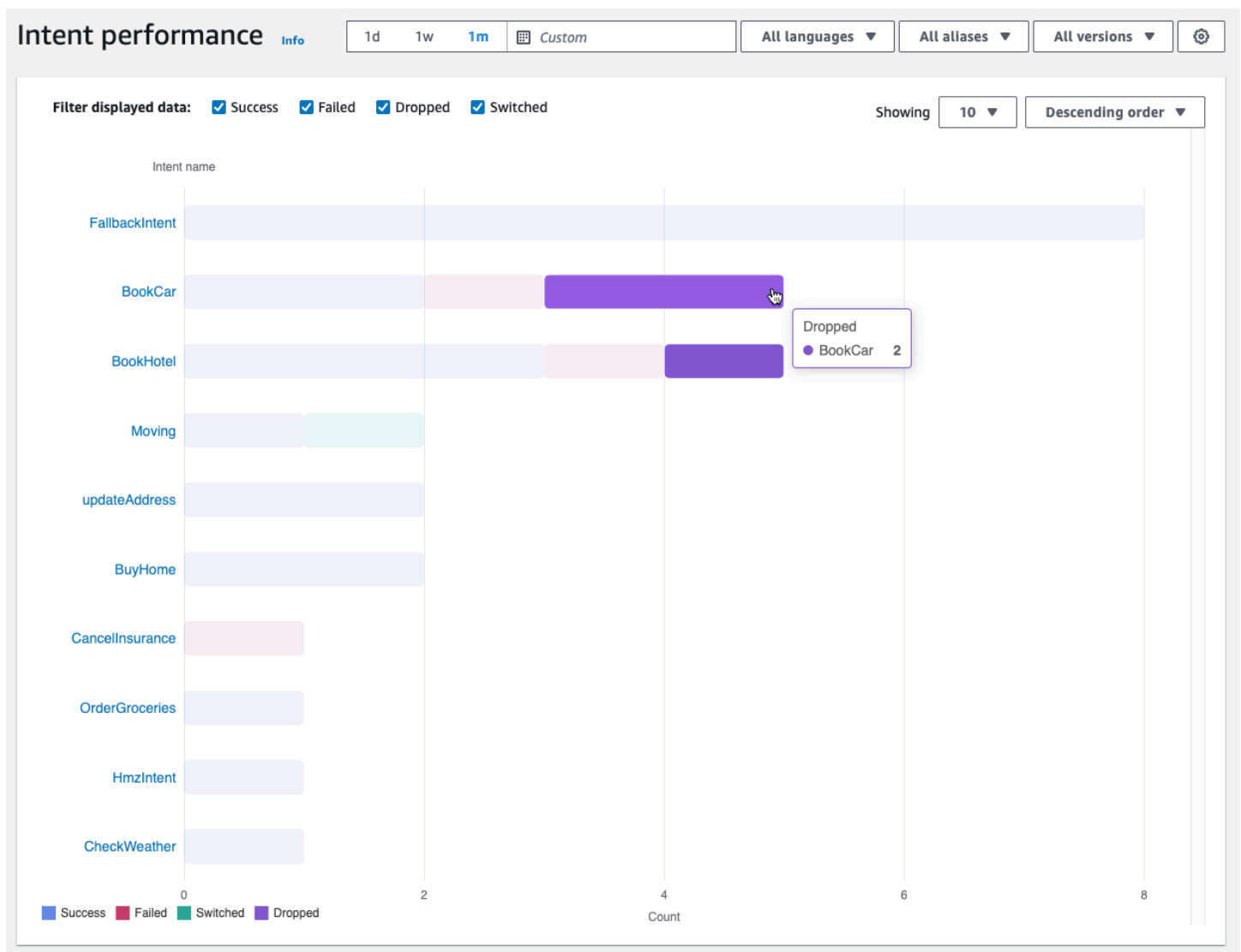
Wählen Sie Alle Absichten anzeigen aus, um ein Dropdownmenü anzuzeigen, in dem Sie eine Liste der vom Bot ausgelösten Absichten anzeigen können. Du kannst auch auswählen, ob Absichten mit einem bestimmten Ergebnis angezeigt werden sollen (erfolgreich, fehlgeschlagen, gelöscht oder geändert). Über diese Links gelangen Sie zum Unterabschnitt Intent Performance des Performance-Dashboards. Weitere Informationen finden Sie unter [Leistung der Absichten](#).

Im Abschnitt zur Erkennung von Äußerungen wird die Anzahl der übersehenen und erkannten Äußerungen zusammengefasst. Wählen Sie Details anzeigen aus, um zu einer Liste von Äußerungen für den Bot zu navigieren. Wählen Sie die Zahl unter Verpasste Äußerungen aus, um eine Liste der verpassten Äußerungen zu sehen, und die Zahl unter Entdeckte Äußerungen, um eine Liste der erkannten Äußerungen für den Bot zu sehen. Weitere Informationen finden Sie unter [Erkennung von Äußerungen](#).

Wählen Sie im Leistungs-Dashboard in der linken Seitenleiste die Optionen Leistung der Absichten und Erkennung von Äußerungen aus, um Details zu Absichten und Äußerungen in Ihrem Bot anzuzeigen.

Leistung der Absichten

Dieses Dashboard fasst die Leistung der mit Ihrem Bot verwendeten Absichten in absteigender Reihenfolge der Häufigkeit zusammen. Die Leiste neben jeder Absicht visualisiert, wie oft die Absicht als erfolgreich eingestuft, gescheitert, verworfen und geändert wurde. Eine [Absichten](#) Erläuterung dieser Definitionen finden Sie unter. Bewegen Sie den Mauszeiger über ein Segment der Leiste, um die Anzahl der Konversationen zu sehen, die diese Absicht mit diesem Ergebnis verwenden, wie in der folgenden Abbildung dargestellt:



 Note

Das Dashboard zeigt die 1.000 besten Ergebnisse für eine Reihe von Filtereinstellungen. Um zielgerichtetere Ergebnisse zu erhalten, konfigurieren Sie detaillierte Filtereinstellungen.

Am oberen Rand des Diagramms können Sie die Absichtsstatus, die Sie anzeigen möchten, mithilfe der Kontrollkästchen Erfolgreich, Fehlgeschlagen, Verworfen und Vertauscht ein- und ausschalten.

Wählen Sie die Dropdownmenüs rechts neben Wird angezeigt, um die Anzahl der anzuzeigenden Absichten einzustellen und festzulegen, ob Absichten in aufsteigender oder absteigender Reihenfolge der Häufigkeit angezeigt werden sollen.

Wählen Sie einen Absichtsnamen aus, um zu einer Seite zu gelangen, auf der drei Diagramme angezeigt werden: Aufschlüsselung der Intent-Leistung, Slot-Leistung und Intent-Switches.

Im Abschnitt Aufschlüsselung der Leistung der Absicht wird angezeigt, wie oft der Bot die Absicht insgesamt verwendet hat, und es wird aufgeschlüsselt, wie oft und wie oft die Absichtserfüllung als erfolgreich eingestuft, fehlgeschlagen, abgebrochen und gewechselt wurde. Eine [Absichten](#) Erläuterung dieser Definitionen finden Sie unter. Zeigen Sie mit der Maus auf ein Segment des Diagramms, um zu sehen, wie oft und in Prozent die Erfüllung der Absicht zu diesem Ergebnis geführt hat.

Im Bereich Slot-Performance werden Kennzahlen für die Slots angezeigt, die zur aktuellen Absicht gehören. Um nach einer Spalte zu sortieren, wählen Sie diese Spalte einmal aus, um sie in aufsteigender Reihenfolge zu sortieren, und zweimal, um sie in absteigender Reihenfolge zu sortieren. Sie können die Suchleiste verwenden, um einen bestimmten Slot zu finden, oder die Seitenzahlen-Schaltflächen verwenden, um durch die Slots zu navigieren.

 Note

Das Dashboard zeigt die 1.000 besten Ergebnisse für eine Reihe von Filtereinstellungen. Um zielgerichtetere Ergebnisse zu erhalten, konfigurieren Sie detaillierte Filtereinstellungen.

Im Abschnitt Intent Switches werden die Instanzen aufgeführt, in denen der Bot von der aktuellen Absicht zu einer anderen gewechselt ist, mit den folgenden Informationen:

- Phase — Die Phase der Konversation, in der der Bot die Absicht geändert hat.

- Absicht wurde geändert — Die Absicht, auf die der Bot die aktuelle Absicht umgestellt hat.
- Anzahl der Sitzungen — Die Anzahl der Sessions, in denen die Kombination Stage und Intent, zu der gewechselt wurde, stattgefunden haben.

Note

Das Dashboard zeigt die 1.000 besten Ergebnisse für eine Reihe von Filtereinstellungen. Um zielgerichtetere Ergebnisse zu erhalten, konfigurieren Sie detaillierte Filtereinstellungen.

Erkennung von Äußerungen

Auf dieser Seite sind alle Äußerungen aufgeführt, die von Ihrem Bot übersehen und erkannt wurden. Außerdem finden Sie Tools, mit denen Sie Ihren Absichten Beispieläußerungen hinzufügen können, um Ihren Bot zu trainieren. Eine Erläuterung dieser Definitionen finden [Äußerungen](#) Sie unter. Verwenden Sie die Tabs oben, um zwischen einer Liste der verpassten Äußerungen und der Liste der erkannten Äußerungen zu wechseln.

Note

Das Dashboard zeigt die 1.000 besten Ergebnisse für eine Reihe von Filtereinstellungen. Um zielgerichtetere Ergebnisse zu erhalten, konfigurieren Sie detaillierte Filtereinstellungen.

So fügst du einer Absicht Äußerungen hinzu:

1. Aktivieren Sie das Kontrollkästchen neben den Äußerungen, die Sie als Beispieläußerungen für eine Absicht hinzufügen möchten.
2. Wählen Sie Zur Absicht hinzufügen und wählen Sie im Dropdownmenü unter Absicht die Absicht aus, zu der Sie die Äußerungen hinzufügen möchten.
3. Wählen Sie Hinzufügen aus.

Für Analysen verwenden APIs

In diesem Abschnitt werden die API-Operationen beschrieben, mit denen Sie Analysen für einen Bot abrufen.

 Note

Um das [ListUtteranceMetrics](#) und verwenden zu können [ListUtteranceAnalyticsData](#), muss Ihre IAM-Rolle über die erforderlichen Berechtigungen für die Ausführung des [ListAggregatedUtterances](#) Vorgangs verfügen, der Zugriff auf Analysen im Zusammenhang mit Äußerungen bietet. Einzelheiten und die IAM-Richtlinie, die [Meinungsstatistiken aus Lex V2-Konversationen anzeigen](#) für die IAM-Rolle gilt, finden Sie unter.

- Die folgenden API-Operationen rufen zusammenfassende Metriken für einen Bot ab:
 - [ListSessionMetrics](#)
 - [ListIntentMetrics](#)
 - [ListIntentStageMetrics](#)
 - [ListUtteranceMetrics](#)
- Die folgenden API-Operationen rufen eine Liste von Metadaten für Sitzungen und Äußerungen ab:
 - [ListSessionAnalyticsData](#)
 - [ListUtteranceAnalyticsData](#)
- Bei diesem [ListIntentPaths](#) Vorgang werden Kennzahlen über die Reihenfolge der Absichten abgerufen, die Kunden in Gesprächen mit einem Bot verfolgen.

Filtern von Ergebnissen

Bei den Analytics-API-Anfragen müssen Sie den Wert und angeben. `startTime` `endTime` Die API gibt Sitzungen, Absichten, Absichtsphasen oder Äußerungen zurück, die nach dem begannen `startTime` und vor dem endeten. `endTime`

`filters` ist ein optionales Feld in den Analytics-API-Anfragen. Es wird einer Liste von [AnalyticsSessionFilter](#), [AnalyticsIntentFilter](#) [AnalyticsIntentStageFilter](#), oder [AnalyticsUtteranceFilter](#) Objekten zugeordnet. Verwenden Sie in jedem Objekt die Felder, um einen Ausdruck zu erstellen, nach dem gefiltert werden soll. Wenn Sie der Liste beispielsweise den folgenden Filter hinzufügen, sucht der Bot nach Konversationen, die länger als 30 Sekunden dauern.

```
{
  "name": "Duration",
  "operator": "GT",
  "value": "30 sec",
```

```
}
```

Metriken für einen Bot werden abgerufen

Verwenden Sie die *ListUtteranceMetrics* Operationen *ListSessionMetrics*, *ListIntentMetrics*, *ListIntentStageMetrics*, und, um zusammenfassende Kennzahlen für Sitzungen, Absichten, Absichtsphasen und Äußerungen abzurufen.

Füllen Sie für diese Operationen die folgenden Pflichtfelder aus:

- Geben Sie ein `startTime` und ein `endTime`, um einen Zeitraum zu definieren, für den Sie Ergebnisse abrufen möchten.
- Geben Sie die Metriken, die Sie berechnen möchten `metrics`, eine Liste von [AnalyticsSessionMetric](#), [AnalyticsIntentMetric](#), [AnalyticsIntentStageMetric](#), oder [AnalyticsUtteranceMetric](#) Objekten an. Verwenden Sie in jedem Objekt das `name` Feld, um die Metrik für die Berechnung anzugeben, das `statistic` Feld, um anzugeben, ob die Max Zahl SumAverage, oder berechnet werden soll, und das `order` Feld, um anzugeben, ob die Ergebnisse in Ascending oder in der Descending Reihenfolge sortiert werden sollen.

Note

`metrics` Sowohl die `binBy` Objekte als auch enthalten ein `order` Feld. Sie können die Sortierung nur `order` in einem der beiden Objekte angeben.

Die übrigen Felder in der Anfrage sind optional. Sie können die Ergebnisse auf folgende Weise filtern und organisieren:

- Ergebnisse filtern — Verwenden Sie das `filters` Feld, um die Ergebnisse zu filtern. Weitere Details finden Sie unter [Filtern von Ergebnissen](#).
- Ergebnisse nach Kategorie gruppieren — Geben Sie das `groupBy` Feld an, eine Liste, die ein einzelnes [AnalyticsSessionResult](#), [AnalyticsIntentResult](#), [AnalyticsIntentStageResult](#), oder [AnalyticsUtteranceResult](#) Objekt enthält. Geben Sie im Objekt das `name` Feld mit der Kategorie an, nach der Sie die Ergebnisse gruppieren möchten.

Wenn Sie in der Anfrage ein `groupBy` Feld angeben, enthält `groupByKeys` das `results` Objekt in der Antwort eine Liste von [AnalyticsSessionGroupByKey](#), [AnalyticsIntentGroupByKey](#),

[AnalyticsIntentStageGroupByKey](#) oder [AnalyticsUtteranceGroupByKey](#) Objekten, jedes mit dem `name`, was Sie in der Anfrage angegeben haben, und einem Mitglied dieser Kategorie im `value` Feld.

- Ergebnisse nach Zeit sortieren — Geben Sie das `binBy` Feld an, eine Liste, die ein einzelnes [AnalyticsBinBySpecification](#) Objekt enthält. Geben Sie im Objekt das `name` Feld an, mit `ConversationStartTime` dem die Ergebnisse nach Beginn der Konversation oder `UtteranceTimestamp` nach dem Zeitpunkt der Äußerung sortiert werden sollen. Geben Sie das Zeitintervall an, nach dem Sie die Ergebnisse im `interval` Feld einteilen möchten, und legen Sie fest, ob die Sortierung `Ascending` oder die `Descending` Reihenfolge im `order` Feld erfolgen soll.

Wenn Sie in der Anforderung ein `binBy` Feld angeben, enthält `binKeys` das `results` Objekt in der Antwort eine Liste von [AnalyticsBinKey](#) Objekten, jedes mit dem `name`, was Sie in der Anfrage angegeben haben, und dem Zeitintervall, das diese Ablage im `value` Feld definiert.

 Note

metrics Sowohl die `binBy` Objekte als auch enthalten ein `order` Feld. Sie können die Sortierung nur `order` in einem der beiden Objekte angeben.

Verwenden Sie die folgenden Felder, um die Anzeige der Antwort zu verwalten:

- Geben Sie eine Zahl zwischen 1 und 1.000 in das `maxResults` Feld ein, um die Anzahl der Ergebnisse zu begrenzen, die in einer einzigen Antwort zurückgegeben werden.
- Wenn die Anzahl der Ergebnisse größer ist als die Zahl, die Sie in dem `maxResults` Feld angeben, enthält die Antwort einen `nextToken`. Stellen Sie die Anfrage erneut, verwenden Sie jedoch diesen Wert im `nextToken` Feld, um den nächsten Ergebnisstapel zurückzugeben.

Wenn Sie verwenden `ListUtteranceMetrics`, können Sie Attribute angeben, die im `attributes` Feld zurückgegeben werden sollen. Dieses Feld ist einer Liste zugeordnet, die ein einzelnes [AnalyticsUtteranceAttribute](#) Objekt enthält. Geben Sie `LastUsedIntent` in dem `name` Feld an, dass die Absicht zurückgegeben werden soll, die Amazon Lex V2 zum Zeitpunkt der Äußerung verwendet.

In der Antwort wird das `results` Feld einer Liste von [AnalyticsSessionResult](#), [AnalyticsIntentResult](#) [AnalyticsIntentStageResult](#), oder [AnalyticsUtteranceResult](#) Objekten zugeordnet. Jedes Objekt enthält ein `metrics` Feld, das den Wert einer zusammenfassenden Statistik für eine

von Ihnen angeforderte Metrik zurückgibt, zusätzlich zu allen Abschnitten oder Gruppen, die mit den von Ihnen angegebenen Methoden erstellt wurden.

Metadaten für Sitzungen und Äußerungen in einem Bot abrufen

Verwenden Sie die [ListUtteranceAnalyticsData](#) Operationen [ListSessionAnalyticsData](#) und, um Metadaten zu einzelnen Sitzungen und Äußerungen abzurufen.

Füllen Sie die erforderlichen `endTime` Felder `startTime` und `end`, um einen Zeitraum zu definieren, für den Sie Ergebnisse abrufen möchten.

Die übrigen Felder in der Anfrage sind optional. So filtern und sortieren Sie Ergebnisse:

- Ergebnisse filtern — Verwenden Sie das `filters` Feld, um die Ergebnisse zu filtern. Weitere Details finden Sie unter [Filtern von Ergebnissen](#).
- Ergebnisse sortieren — Sortiert die Ergebnisse nach dem `sortBy` Feld, das ein [SessionDataSortByUtteranceDataSortBy](#) Oder-Objekt enthält. Geben Sie den Wert an, nach dem Sie im `name` Feld sortieren möchten, und legen Sie fest, ob im `order` Feld sortiert `Ascending` oder in der `Descending` Reihenfolge sortiert werden soll.

Verwenden Sie die folgenden Felder, um die Anzeige der Antwort zu verwalten:

- Geben Sie eine Zahl zwischen 1 und 1.000 in das `maxResults` Feld ein, um die Anzahl der Ergebnisse zu begrenzen, die in einer einzigen Antwort zurückgegeben werden.
- Wenn die Anzahl der Ergebnisse größer ist als die Zahl, die Sie in dem `maxResults` Feld angeben, enthält die Antwort einen `nextToken`. Stellen Sie die Anfrage erneut, verwenden Sie jedoch diesen Wert im `nextToken` Feld, um den nächsten Ergebnisstapel zurückzugeben.

In der Antwort wird das `utterances` Feld `sessions` oder einer Liste von [SessionSpecificationUtteranceSpecification](#) Objekten zugeordnet. Jedes Objekt enthält Metadaten für eine einzelne Sitzung oder Äußerung.

Analysedaten für den Absichtspfad werden abgerufen

Verwenden Sie diesen [ListIntentPaths](#) Vorgang, um Kennzahlen zu einer Reihenfolge von Absichten abzurufen, die Kunden im Gespräch mit einem Bot annehmen.

Füllen Sie für diesen Vorgang die folgenden Pflichtfelder aus:

- Geben Sie ein `startTime` und ein `endTime`, um einen Zeitraum zu definieren, für den Sie Ergebnisse abrufen möchten.
- Geben Sie eine `intentPath`, um eine Reihenfolge der Absichten zu definieren, für die Sie Metriken abrufen möchten. Trennen Sie die Absichten im Pfad durch einen Schrägstrich voneinander. Füllen Sie das `intentPath` Feld beispielsweise mit `aus`, um Details darüber **/BookCar/BookHotel** zu sehen, wie oft Benutzer die `BookHotel` Absichten `BookCar` und in dieser Reihenfolge aufgerufen haben.

Verwenden Sie das optionale `filters` Feld, um die Ergebnisse zu filtern. Weitere Details finden Sie unter [Filtern von Ergebnissen](#).

Meinungsstatistiken aus Lex V2-Konversationen anzeigen

Sie können anhand von Statistiken zu Äußerungen ermitteln, welche Äußerungen Ihre Benutzer an Ihren Bot senden. Sie können sowohl die Äußerungen sehen, die Amazon Lex V2 erfolgreich erkennt, als auch die Äußerungen, die es nicht erkennt. Sie können diese Informationen verwenden, um Ihren Bot zu optimieren.

Wenn Sie beispielsweise feststellen, dass Ihre Benutzer eine Äußerung senden, bei der Amazon Lex V2 fehlt, können Sie die Äußerung zu einer Absicht hinzufügen. Die Entwurfsversion der Absicht wird mit der neuen Äußerung aktualisiert und Sie können sie testen, bevor Sie sie für Ihren Bot bereitstellen.

Eine Äußerung wird erkannt, wenn Amazon Lex V2 die Äußerung als Versuch erkennt, eine für einen Bot konfigurierte Absicht aufzurufen. Eine Äußerung wird übersehen, wenn Amazon Lex V2 die Äußerung nicht erkennt und stattdessen die Äußerung aufruft. `AMAZON.FallbackIntent`

Statistiken zu Äußerungen können mithilfe der API und der `ListUtteranceMetrics` API eingesehen werden. `ListAggregatedUtterance`

Unter den folgenden Bedingungen werden keine Statistiken zu Äußerungen mithilfe der `ListUtteranceMetrics` API generiert:

- Die Einstellung `Child Online Privacy Protection Act` war auf `Ja` gesetzt, als der Bot mit der Konsole erstellt wurde, oder das `childDirected` Feld wurde auf `true` gesetzt, als der Bot mit dem `CreateBot` Vorgang erstellt wurde.

Die `ListUtteranceMetrics` API bietet zusätzliche Funktionen, darunter:

- Weitere Informationen sind verfügbar, z. B. eine Zuordnung der Absicht zu erkannten Äußerungen.
- Mehr Filtermöglichkeiten (einschließlich Kanal und Modus).
- Längerer Aufbewahrungszeitraum (30 Tage).
- Sie können die API auch dann verwenden, wenn Sie die Datenspeicherung deaktiviert haben. Die Konsolenfunktionalität für verpasste und erkannte Äußerungen hängt von der `ListUtteranceMetrics` API ab.

Unter den folgenden Bedingungen werden keine Statistiken zu Äußerungen mithilfe der `ListAggregatedUtterance` API generiert:

- Die Einstellung Child Online Privacy Protection Act war auf Ja gesetzt, als der Bot mit der Konsole erstellt wurde, oder das `childDirected` Feld wurde auf true gesetzt, als der Bot mit dem `CreateBot` Vorgang erstellt wurde.
- Sie verwenden die Slot-Obfuscation mit einem oder mehreren Steckplätzen.
- Sie haben sich von der Teilnahme an der Verbesserung von Amazon Lex abgemeldet.

Die `ListAggregatedUtterance` API bietet Funktionen wie:

- Weniger detaillierte Informationen verfügbar (keine Zuordnung der Absicht für Äußerungen).
- Eingeschränkte Filterfähigkeit (ohne Kanal und Modus).
- Kurzer Aufbewahrungszeitraum (15 Tage).

Anhand von Statistiken zu Äußerungen können Sie sehen, ob eine bestimmte Äußerung erkannt oder übersehen wurde und wann die Äußerung zuletzt in einer Bot-Interaktion verwendet wurde.

Amazon Lex V2 speichert kontinuierlich Äußerungen, während Benutzer mit Ihrem Bot interagieren. Sie können die Statistiken über die Konsole oder den `ListAggregatedUtterances` Vorgang abfragen. Es hat eine Datenspeicherung von 15 Tagen und ist nicht verfügbar, wenn der Benutzer die Datenspeicherung deaktiviert hat. Sie können Äußerungen mithilfe des `DeleteUtterances` Vorgangs oder durch Deaktivierung der Datenspeicherung löschen. Alle Äußerungen werden gelöscht, wenn Sie Ihr Konto schließen. AWS Gespeicherte Äußerungen werden mit einem serververwalteten Schlüssel verschlüsselt.

Wenn Sie eine Bot-Version löschen, sind die Statistiken zu den Äußerungen für die Version bis zu 30 Tage mit `ListUtteranceMetrics` und für die Nutzung bis zu 15 Tage verfügbar.

ListAggregatedUtterances In der Amazon Lex V2-Konsole können Sie keine Statistiken für die gelöschte Version sehen. Um die Statistiken für gelöschte Versionen zu sehen, können Sie beide **ListUtteranceMetrics** Operationen **ListAggregatedUtterances** und verwenden.

Sowohl bei als auch werden Äußerungen nach dem Text der Äußerung aggregiert.

ListAggregatedUtterances **ListUtteranceMetrics** APIs Beispielsweise werden alle Fälle, in denen der Kunde den Ausdruck „Ich möchte eine Pizza bestellen“ verwendet hat, in einer Antwort zu derselben Zeile zusammengefasst. Wenn Sie die [RecognizeUtterance](#) Operation verwenden, ist der verwendete Text das Eingabeprotokoll.

Um das **ListAggregatedUtterances** und zu verwenden **ListUtteranceMetrics** APIs, wenden Sie die folgende Richtlinie auf eine Rolle an.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAggregatedUtterancesPolicy",
      "Effect": "Allow",
      "Action": "lex:ListAggregatedUtterances",
      "Resource": "*"
    }
  ]
}
```

Verwaltung von Zugriffsberechtigungen für Analysen

Um einem Benutzer Zugriff auf Analysen zu gewähren, fügen Sie einer IAM-Rolle eine Richtlinie hinzu, die es der Rolle ermöglicht, die API-Operationen für Analysen aufzurufen. Sie können die [Von AWS verwaltete Richtlinie: AmazonLexFullAccess](#) IAM-Rolle anhängen, um vollen Zugriff auf Amazon Lex Lex-API-Operationen zu gewähren, oder Sie können eine benutzerdefinierte Richtlinie erstellen, die nur Analyseberechtigungen zulässt, und sie einer IAM-Rolle zuordnen.

Um eine benutzerdefinierte Richtlinie zu erstellen, die Berechtigungen für Analysen enthält

1. Wenn Sie zuerst eine IAM-Rolle erstellen müssen, folgen Sie den Schritten unter [Erstellen einer Rolle, um Berechtigungen an einen IAM-Benutzer zu delegieren](#).

2. Folgen Sie den Schritten unter [IAM-Richtlinien erstellen](#), um eine Richtlinie mit dem folgenden JSON-Objekt zu erstellen. Um den Analysezugriff auf bestimmte Bots für die IAM-Rolle zu ermöglichen, fügen Sie dem Resource Feld den ARN jedes Bots hinzu. Ersetzen Sie das *regionaccount-id*, und *BOTID* durch die Werte, die den Bots entsprechen. Sie können den Bezeichner der Anweisung auch durch einen Namen Ihrer Wahl ersetzen. *AnalyticsActions*

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AnalyticsActions",
      "Effect": "Allow",
      "Action": [
        "lex:ListAggregatedUtterances",
        "lex:ListIntentMetrics",
        "lex:ListSessionAnalyticsData",
        "lex:ListIntentPaths",
        "lex:ListIntentStageMetrics",
        "lex:ListSessionMetrics"
      ],
      "Resource": [
        "arn:aws:lex:region:account-id:bot/BOTID"
      ]
    }
  ]
}
```

3. Ordnen Sie die von Ihnen erstellte Richtlinie der Rolle zu, der Sie Analyseberechtigungen erteilen möchten. Folgen Sie dazu den Schritten unter [Hinzufügen und Entfernen von IAM-Identitätsberechtigungen](#).
4. Die Rolle sollte nun über Berechtigungen zum Anzeigen von Analysen für die von Ihnen angegebenen Bots verfügen.

Konversationsprotokolle für Ihre Lex V2-Bots aktivieren

Verwenden Sie Konversationsprotokolle, um Benutzerkonversationen mit Ihrem Bot zu speichern. Überprüfen Sie diese Protokolle, um Probleme bei den Interaktionen Ihres Bots mit Benutzern

zu identifizieren und das Verhalten Ihres Bots anhand dieser Erkenntnisse zu ändern. In diesem Abschnitt wird auch beschrieben, wie Sie Slot-Werte verschleiern können, um die Privatsphäre der Benutzer zu schützen.

Themen

- [Protokollierung von Konversationen mit Konversationsprotokollen in Lex V2](#)
- [Verdecken von Slot-Werten in Konversationsprotokollen von Lex V2](#)
- [Selektive Erfassung von Konversationsprotokollen in Lex V2](#)

Protokollierung von Konversationen mit Konversationsprotokollen in Lex V2

Sie aktivieren Konversationsprotokolle zum Speichern von Bot-Interaktionen. Sie können diese Protokolle verwenden, um die Leistung Ihres Bots zu überprüfen und Probleme mit Konversationen zu beheben. Sie können Text für die [RecognizeText](#)-Operation protokollieren. Sie können sowohl Text als auch Audio für die [RecognizeUtterance](#)-Operation protokollieren. Durch die Aktivierung von Konversationsprotokollen erhalten Sie eine detaillierte Ansicht der Konversationen, die Benutzer mit Ihrem Bot führen.

Beispielsweise hat eine Sitzung mit Ihrem Bot eine Sitzungs-ID. Sie können diese ID verwenden, um das Transkript der Konversation zu erhalten, einschließlich der Äußerungen des Benutzers und der entsprechenden Bot-Antworten. Sie erhalten auch Metadaten wie den Namen der Absicht und Slot-Werte für eine Äußerung.

Note

Sie können keine Konversationsprotokolle mit einem Bot verwenden, der dem Children's Online Privacy Protection Act (COPPA) unterliegt.

Konversationsprotokolle werden für einen Alias konfiguriert. Jeder Alias kann unterschiedliche Einstellungen für seine Text- und Audioprotokolle haben. Sie können Textprotokolle, Audioprotokolle oder beides für jeden Alias aktivieren. Textprotokolle speichern Texteingaben, Transkripte von Audioeingaben und zugehörige Metadaten in CloudWatch Protokollen. Audioprotokolle speichern Audioeingaben in Amazon S3. Sie können die Verschlüsselung von Text- und Audioprotokollen mithilfe von AWS KMS Customer Managed aktivieren CMKs.

Verwenden Sie die Konsole oder die [UpdateBotAlias](#) Operation [CreateBotAlias](#) oder, um die Protokollierung zu konfigurieren. Nachdem Sie die Konversationsprotokolle für einen Alias aktiviert

haben, werden mit der [RecognizeUtterance](#) Operation [RecognizeText](#) oder für diesen Alias die Text- oder Audioäußerungen in der konfigurierten Protokollgruppe oder dem S3-Bucket CloudWatch protokolliert.

Themen

- [IAM-Richtlinien für Konversationsprotokolle](#)
- [Konfiguration von Konversationsprotokollen für Ihren Lex V2-Bot](#)
- [Textprotokolle in Amazon CloudWatch Logs von Lex V2 aus anzeigen](#)
- [Zugreifen auf Audioprotokolle in Amazon S3](#)
- [Den Status des Konversationsprotokolls anhand von CloudWatch Messwerten überwachen](#)

IAM-Richtlinien für Konversationsprotokolle

Abhängig von der Art der Protokollierung, die Sie auswählen, benötigt Amazon Lex V2 die Erlaubnis, Amazon CloudWatch Logs- und Amazon Simple Storage Service (S3) -Buckets zum Speichern Ihrer Protokolle zu verwenden. Sie müssen AWS Identity and Access Management Rollen und Berechtigungen erstellen, damit Amazon Lex V2 auf diese Ressourcen zugreifen kann.

Erstellen einer IAM-Rolle und von Richtlinien für Konversationsprotokolle

Um Konversationsprotokolle zu aktivieren, müssen Sie Schreibberechtigungen für CloudWatch Logs und Amazon S3 erteilen. Wenn Sie die Objektverschlüsselung für Ihre S3-Objekte aktivieren, müssen Sie Zugriffsberechtigungen für die AWS KMS Schlüssel erteilen, die zur Verschlüsselung der Objekte verwendet werden.

Sie können die IAM-Konsole, die IAM-API oder die verwenden, um die Rolle und die Richtlinien AWS Command Line Interface zu erstellen. In diesen Anweisungen werden AWS CLI die Rolle und die Richtlinien erstellt.

Note

Der folgende Code ist für Linux und MacOS formatiert. Ersetzen Sie unter Windows das Linux-Zeilenumbruchzeichen (\n) durch ein Caret-Zeichen (^).

Um eine IAM-Rolle für Konversationsprotokolle zu erstellen

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis mit dem Namen **LexConversationLogsAssumeRolePolicyDocument.json**, fügen Sie den folgenden Code hinzu und speichern Sie es. In diesem Richtliniendokument wird Amazon Lex V2 als vertrauenswürdige Entität zur Rolle hinzugefügt. Auf diese Weise können Sie die Rolle übernehmen, Protokolle an die für Konversationsprotokolle konfigurierten Ressourcen zu übermitteln.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Führen Sie in der den folgenden Befehl aus AWS CLI, um die IAM-Rolle für Konversationsprotokolle zu erstellen.

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

Erstellen Sie als Nächstes eine Richtlinie und fügen Sie sie der Rolle hinzu, die es Amazon Lex V2 ermöglicht, in CloudWatch Protokolle zu schreiben.

Um eine IAM-Richtlinie für die Protokollierung von Konversationstext in Logs zu CloudWatch erstellen

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis namens **LexConversationLogsCloudWatchLogsPolicy.json**, fügen Sie ihm die folgende IAM-Richtlinie hinzu und speichern Sie es.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:region:account-id:log-group:log-group-
name:*"
    }
  ]
}
```

2. Erstellen Sie in der AWS CLI die IAM-Richtlinie, die der Protokollgruppe CloudWatch Logs Schreibberechtigungen gewährt.

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

3. Ordnen Sie die Richtlinie der IAM-Rolle zu, die Sie für Konversationsprotokolle erstellt haben.

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

Wenn Sie Audio in einem S3-Bucket protokollieren, erstellen Sie eine Richtlinie, die es Amazon Lex V2 ermöglicht, in den Bucket zu schreiben.

Um eine IAM-Richtlinie für die Audioprotokollierung in einem S3-Bucket zu erstellen

1. Erstellen Sie ein Dokument im aktuellen Verzeichnis mit dem Namen **LexConversationLogsS3Policy.json**, fügen Sie die folgende Richtlinie hinzu und speichern Sie es.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::bucket-name/*"
    }
  ]
}
```

2. Erstellen Sie in der die IAM-Richtlinie AWS CLI, die Schreibberechtigungen für Ihren S3-Bucket gewährt.

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

3. Fügen Sie die Richtlinie der Rolle an, die Sie für Konversationsprotokolle erstellt haben.

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

Erteilen Sie die Erlaubnis, eine IAM-Rolle zu übergeben

Wenn Sie die Konsole, das oder ein AWS SDK verwenden AWS Command Line Interface, um eine IAM-Rolle für Konversationsprotokolle anzugeben, muss der Benutzer, der die IAM-Rolle für Konversationsprotokolle angibt, berechtigt sein, die Rolle an Amazon Lex V2 weiterzugeben. Damit

der Benutzer die Rolle an Amazon Lex V2 weitergeben kann, müssen Sie dem IAM-Benutzer, der Rolle oder der Gruppe des Benutzers die PassRole entsprechende Berechtigung erteilen.

Die folgende Richtlinie definiert die Berechtigung, die dem Benutzer, der Rolle oder der Gruppe erteilt werden soll. Sie können die `iam:AssociatedResourceArn` und `iam:PassedToService`-Bedingungsschlüssel verwenden, um den Umfang der Berechtigung einzuschränken. Weitere Informationen finden Sie unter [Erteilen von Benutzerberechtigungen zur Übergabe einer Rolle an einen AWS Service](#) und [IAM- und AWS STS Bedingungskontextschlüssel](#) im AWS Identity and Access Management Benutzerhandbuch.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/role-name",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "lexv2.amazonaws.com"
        },
        "StringLike": {
          "iam:AssociatedResourceARN": "arn:aws:lex:region:account-id:bot:bot-name:bot-alias"
        }
      }
    }
  ]
}
```

Konfiguration von Konversationsprotokollen für Ihren Lex V2-Bot

Sie aktivieren und deaktivieren Konversationsprotokolle über die Konsole oder das `conversationLogSettings` Feld der UpdateBotAlias Operation CreateBotAlias oder. Sie können Audioprotokolle, Textprotokolle oder beides aktivieren oder deaktivieren. Die Protokollierung beginnt bei neuen Botsitzungen. Änderungen an den Protokolleinstellungen werden für aktive Sitzungen nicht berücksichtigt.

Um Textprotokolle zu speichern, verwenden Sie eine Amazon CloudWatch Logs-Protokollgruppe in Ihrem AWS Konto. Sie können jede gültige Protokollgruppe verwenden. Die Protokollgruppe muss sich in derselben Region wie der Amazon Lex V2-Bot befinden. Weitere Informationen zum Erstellen einer CloudWatch Logs-Protokollgruppe finden Sie unter [Working with Log Groups and Log Streams](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

Verwenden Sie zum Speichern von Audioprotokollen einen Amazon S3 S3-Bucket in Ihrem AWS Konto. Sie können jeden gültigen S3-Bucket verwenden. Der Bucket muss sich in derselben Region wie der Amazon Lex V2-Bot befinden. Weitere Informationen zum Erstellen eines S3-Buckets finden Sie unter [Erstellen eines Buckets](#) im Amazon Simple Storage Service Getting Started Guide.

Wenn Sie Konversationsprotokolle mit der Konsole verwalten, aktualisiert die Konsole Ihre Servicerolle, sodass sie Zugriff auf die Protokollgruppe und den S3-Bucket hat.

Wenn Sie die Konsole nicht verwenden, müssen Sie eine IAM-Rolle mit Richtlinien bereitstellen, die es Amazon Lex V2 ermöglichen, in die konfigurierte Protokollgruppe oder den konfigurierten Bucket zu schreiben. Wenn Sie mit dem eine serviceverknüpfte Rolle erstellen AWS Command Line Interface, müssen Sie der Rolle mithilfe der `custom-suffix` Option ein benutzerdefiniertes Suffix hinzufügen, wie im folgenden Beispiel gezeigt. Weitere Informationen finden Sie unter [Erstellen einer IAM-Rolle und von Richtlinien für Konversationsprotokolle](#).

```
aws iam create-service-linked-role \
  --aws-service-name lexv2.amazon.aws.com \
  --custom-suffix suffix
```

Die IAM-Rolle, die Sie zum Aktivieren von Konversationsprotokollen verwenden, muss über die entsprechende Berechtigung verfügen. `iam:PassRole` Die folgende Richtlinie sollte der Rolle beigefügt werden:

JSON

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

```
}  
]
```

Konversationsprotokolle aktivieren

So aktivieren Sie Protokolle über die Konsole

1. Öffnen Sie die Amazon Lex V2-Konsole <https://console.aws.amazon.com/lexv2>.
2. Wählen Sie aus der Liste einen Bot aus.
3. Wählen Sie im linken Menü Aliase aus.
4. Wählen Sie in der Liste der Aliase den Alias aus, für den Sie Konversationsprotokolle konfigurieren möchten.
5. Wählen Sie im Abschnitt Konversationsprotokolle die Option Konversationsprotokolle verwalten aus.
6. Für Textprotokolle wählen Sie Aktivieren und geben Sie dann den Namen der Amazon CloudWatch Logs-Protokollgruppe ein.
7. Für Audioprotokolle wählen Sie Aktivieren und geben Sie dann die S3-Bucket-Informationen ein.
8. Optional. Um Audioprotokolle zu verschlüsseln, wählen Sie den AWS KMS Schlüssel aus, der für die Verschlüsselung verwendet werden soll.
9. Wählen Sie Save (Speichern), um Konversationen zu protokollieren. Falls erforderlich, aktualisiert Amazon Lex V2 Ihre Servicerolle mit Berechtigungen für den Zugriff auf die CloudWatch Logs-Protokollgruppe und den ausgewählten S3-Bucket.

Deaktivieren von Konversationsprotokollen in Lex V2

So deaktivieren Sie Protokolle über die Konsole

1. Öffnen Sie die Amazon Lex V2-Konsole <https://console.aws.amazon.com/lexv2>.
2. Wählen Sie aus der Liste einen Bot aus.
3. Wählen Sie im linken Menü Aliase aus.
4. Wählen Sie in der Liste der Aliase den Alias aus, für den Sie Konversationsprotokolle konfigurieren möchten.
5. Wählen Sie im Abschnitt Konversationsprotokolle die Option Konversationsprotokolle verwalten aus.

6. Deaktivieren Sie die Textprotokollierung, die Audioprotokollierung oder beides, um die Protokollierung zu deaktivieren.
7. Wählen Sie Save (Speichern), um die Protokollierung von Konversationen zu beenden.

Textprotokolle in Amazon CloudWatch Logs von Lex V2 aus anzeigen

Amazon Lex V2 speichert Textprotokolle für Ihre Konversationen in Amazon CloudWatch Logs. Verwenden Sie zum Anzeigen der CloudWatch Protokolle die Logs-Konsole oder die API. Weitere Informationen finden Sie unter [Suchprotokolldaten mithilfe von Filtermustern](#) und [CloudWatch Logs Insights-Abfragesyntax](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

So zeigen Sie Protokolle mit der Amazon Lex V2-Konsole an

1. Öffnen Sie die Amazon Lex V2-Konsole <https://console.aws.amazon.com/lexv2>.
2. Wählen Sie aus der Liste einen Bot aus.
3. Wählen Sie im linken Menü Analytics und dann Metriken ausCloudWatch .
4. Sehen Sie sich die Metriken für Ihren Bot auf der Seite mit den CloudWatch Kennzahlen an.

Sie können auch die CloudWatch Konsole oder API verwenden, um Ihre Logeinträge einzusehen. Um die Protokolleinträge zu suchen, navigieren Sie zu der Protokollgruppe, die Sie für den Alias konfiguriert haben. Sie finden das Protokollstream-Präfix für Ihre Protokolle in der Amazon Lex V2-Konsole oder mithilfe des [DescribeBotAlias](#)Vorgangs.

Protokolleinträge für eine Benutzeräußerung befinden sich in mehreren Protokolldatenströmen. Eine Äußerung in der Konversation hat einen Eintrag in einem der Protokolldatenstreams mit dem angegebenen Präfix. Ein Eintrag im Logstream enthält die folgenden Informationen:

Nachrichtenversion

Die Version des Nachrichtenschemas.

Bot

Details über den Bot, mit dem der Kunde interagiert.

messages

Die Antwort, die der Bot an den Benutzer zurückgesendet hat.

Kontext der Äußerung

Informationen zur Verarbeitung dieser Äußerung.

- `runtimeHints`— Laufzeitkontext, der zum Transkribieren und Interpretieren der Benutzereingaben verwendet wird. Weitere Informationen finden Sie unter [Verbesserung der Erkennung von Slot-Werten mit Laufzeithinweisen in der Konversation](#).
- `slotElicitationStyle`— Der Slot-Elicitation-Stil, der zur Interpretation von Benutzereingaben verwendet wird. Weitere Informationen finden Sie unter [Erfassung von Slot-Werten mit Rechtschreibstilen während der Konversation](#).

Sitzungsstatus

Der aktuelle Status der Konversation zwischen dem Benutzer und dem Bot. Weitere Informationen finden Sie unter [Bot-Konversationen verstehen](#).

Interpretationen

Eine Liste von Absichten, von denen Amazon Lex V2 festgestellt hat, dass sie der Äußerung des Benutzers entsprechen könnten. [Verwendung von Konfidenzwerten zur Verbesserung der Konversationsgenauigkeit](#).

Quelle der Interpretation

Gibt an, ob ein Slot von Amazon Bedrock oder Amazon Lex gelöst wurde. Werte: Lex | Bedrock

sessionId

Die Kennung der Benutzersitzung, die die Konversation führt.

inputTranscript

Eine Transkription der Benutzereingabe.

- Bei der Texteingabe ist dies der Text, den der Benutzer eingegeben hat. Bei DTMF-Eingaben ist dies der Schlüssel, den der Benutzer eingegeben hat.
- Bei der Spracheingabe ist dies der Text, in den Amazon Lex V2 die Benutzeräußerung konvertiert, um eine Absicht auszulösen oder einen Slot zu füllen.

rawInputTranscript

Das Rohprotokoll der Benutzereingabe, bevor eine Textverarbeitung angewendet wird. Hinweis: Die Textverarbeitung ist nur für die Gebietsschemas en-US und en-GB verfügbar.

Abschriften

Eine Liste möglicher Transkriptionen der Benutzereingaben. Weitere Informationen finden Sie unter [Verwenden Sie Konfidenzwerte für die Sprachtranskription, um die Konversation mit Ihrem Lex V2-Bot zu verbessern](#).

Rohtranskription

Verwendung von Konfidenzwerten für die Sprachtranskription. Weitere Informationen finden Sie unter [Verwenden Sie Konfidenzwerte für die Sprachtranskription, um die Konversation mit Ihrem Lex V2-Bot zu verbessern](#).

Verpasste Ausdrucksweise

Gibt an, ob Amazon Lex V2 die Äußerung des Benutzers erkennen konnte.

requestId

Amazon Lex V2 hat eine Anforderungs-ID für die Benutzereingabe generiert.

Zeitstempel

Der Zeitstempel der Benutzereingabe.

Entwickler Override

Gibt an, ob der Konversationsablauf mithilfe eines Dialog-Code-Hooks aktualisiert wurde. Weitere Informationen zur Verwendung eines Dialog-Code-Hooks finden Sie unter [Eine AWS Lambda Funktion in deinen Bot integrieren](#).

Eingabemodus

Gibt den Typ der Eingabe an. Kann Audio, DTMF oder Text sein.

requestAttributes

Die Anforderungsattribute, die bei der Verarbeitung der Benutzereingaben verwendet werden.

AudioEigenschaften

Wenn Audiokonversationsprotokolle aktiviert sind und die Benutzereingabe im Audioformat erfolgte, umfasst dies die Gesamtdauer der Audioeingabe, die Dauer der Stimme und die Dauer der Stille im Audio. Es enthält auch einen Link zur Audiodatei.

Bargeln

Gibt an, ob die Benutzereingabe die vorherige Bot-Antwort unterbrochen hat.

Grund der Antwort

Der Grund, warum eine Antwort generiert wurde. Kann einer der folgenden sein:

- `UtteranceResponse`— Antwort auf Benutzereingaben
- `StartTimeout`— vom Server generierte Antwort, wenn der Benutzer keine Eingabe gemacht hat
- `StillWaitingResponse`— vom Server generierte Antwort, wenn der Benutzer den Bot auffordert, zu warten
- `FulfillmentInitiated`— vom Server generierte Antwort, dass die Ausführung bald eingeleitet wird
- `FulfillmentStartedResponse`— vom Server generierte Antwort, dass die Erfüllung begonnen hat
- `FulfillmentUpdateResponse`— regelmäßige vom Server generierte Antwort, während die Erfüllung läuft
- `FulfillmentCompletedResponse`— vom Server generierte Antwort, wenn die Erfüllung abgeschlossen ist.

operationName

Die API, die für die Interaktion mit dem Bot verwendet wurde. Kann einer von `PutSession`, `RecognizeTextRecognizeUtterance`, oder `startConversation` sein.

```
{
  "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"
}
```

Der Inhalt des Protokolleintrags hängt vom Ergebnis einer Transaktion und der Konfiguration des Bots und der Anfrage ab.

- Die Felder `intent`, `slots` und `slotToElicit` werden nicht in einem Eintrag angezeigt, wenn das `missedUtterance`-Feld den Wert `true` hat.
- Das `s3PathForAudio`-Feld wird nicht angezeigt, wenn Audioprotokolle deaktiviert sind oder wenn das `inputDialogMode`-Feld `Text` ist.
- Das `responseCard`-Feld wird nur angezeigt, wenn Sie eine Antwortkarte für den Bot definiert haben.
- Die `requestAttributes`-Karte wird nur angezeigt, wenn Sie in der Anforderung Anforderungsattribute angegeben haben.
- Das `kendraResponse` Feld ist nur vorhanden, wenn der eine Anfrage zur Suche in einem Amazon Kendra Kendra-Index `AMAZON.KendraSearchIntent` stellt.
- Das `developerOverride` Feld ist wahr, wenn in der Lambda-Funktion des Bots eine alternative Absicht angegeben wurde.
- Die `sessionAttributes`-Karte wird nur angezeigt, wenn Sie Sitzungsattribute in der Anforderung angegeben haben.
- Die `sentimentResponse`-Karte wird nur angezeigt, wenn Sie den Bot so konfigurieren, dass er Stimmungswerte zurückgibt.

 Note

Das Eingabeformat kann sich auch ohne entsprechende Änderung der `messageVersion` ändern. Ihr Code sollte keinen Fehler ausgeben, wenn neue Felder vorhanden sind.

Zugreifen auf Audioprotokolle in Amazon S3

Amazon Lex V2 speichert Audioprotokolle für Ihre Konversationen in einem S3-Bucket.

Sie können die Amazon S3 S3-Konsole oder API verwenden, um auf Audioprotokolle zuzugreifen. Sie können das S3-Objektschlüsselpräfix der Audiodateien in der Amazon Lex V2-Konsole oder im `conversationLogSettings` Feld in der `DescribeBotAlias` Betriebsantwort sehen.

Den Status des Konversationsprotokolls anhand von CloudWatch Messwerten überwachen

Verwenden Sie Amazon CloudWatch , um die Lieferkennzahlen Ihrer Konversationsprotokolle zu überwachen. Sie können Alarme für Metriken festlegen, damit Sie Probleme mit der Protokollierung erkennen, wenn sie auftreten sollten.

Amazon Lex V2 bietet vier Metriken im AWS/Lex Namespace für Konversationsprotokolle:

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

Die Erfolgsmetriken zeigen, dass Amazon Lex V2 Ihre Audio- oder Textprotokolle erfolgreich an ihre Ziele geschrieben hat.

Die Fehlermetriken zeigen, dass Amazon Lex V2 keine Audio- oder Textprotokolle an das angegebene Ziel senden konnte. In der Regel ist dies ein Konfigurationsfehler. Wenn Ihre Fehlermetriken über Null liegen, überprüfen Sie Folgendes:

- Stellen Sie sicher, dass Amazon Lex V2 eine vertrauenswürdige Entität für die IAM-Rolle ist.
- Stellen Sie für die Textprotokollierung sicher, dass die Protokollgruppe CloudWatch Logs vorhanden ist. Stellen Sie für die Audioprotokollierung sicher, dass der S3-Bucket vorhanden ist.

- Stellen Sie sicher, dass die IAM-Rolle, die Amazon Lex V2 für den Zugriff auf die CloudWatch Logs-Protokollgruppe oder den S3-Bucket verwendet, über Schreibberechtigungen für die Protokollgruppe oder den Bucket verfügt.
- Stellen Sie sicher, dass der S3-Bucket in derselben Region wie der Amazon Lex V2-Bot existiert und zu Ihrem Konto gehört.

Verdecken von Slot-Werten in Konversationsprotokollen von Lex V2

Amazon Lex V2 ermöglicht es Ihnen, den Inhalt von Slots zu verschleiern oder auszublenden, sodass der Inhalt nicht sichtbar ist. Um vertrauliche Daten zu schützen, die als Slot-Werte erfasst wurden, können Sie die Slot-Verschleierung aktivieren, um diese Werte für die Protokollierung zu maskieren.

Wenn Sie sich dafür entscheiden, Slot-Werte zu verschleiern, ersetzt Amazon Lex V2 den Wert des Slots durch den Namen des Slots in Konversationsprotokollen. Bei einem aufgerufenen `full_name`, Slot würde der Wert des Slots wie folgt verschleiert:

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

Wenn eine Äußerung Klammerzeichen ({}) enthält, maskiert Amazon Lex V2 die Klammern mit zwei umgekehrten Schrägstrichen (\\). Beispielsweise {John Stiles} wird der Text wie folgt verschleiert:

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

Slot-Werte werden in Konversationsprotokollen verschleiert. Die Slot-Werte sind weiterhin in der Antwort der `RecognizeUtterance` Und-Operationen verfügbar, und die Slot-Werte sind für Ihre Lambda-Funktionen zur Validierung und Erfüllung verfügbar. `RecognizeText` Wenn Sie Slot-Werte in Ihren Eingabeaufforderungen oder Antworten verwenden, werden diese Slot-Werte nicht in Gesprächsprotokollen verschleiert.

In der ersten Runde einer Konversation verschleiert Amazon Lex V2 Slot-Werte, wenn es einen Slot und einen Slot-Wert in der Äußerung erkennt. Wenn kein Slot-Wert erkannt wird, verschleiert Amazon Lex V2 die Äußerung nicht.

In der zweiten und späteren Runde weiß Amazon Lex V2, welcher Slot abgerufen werden muss und ob der Slot-Wert verschleiert werden sollte. Wenn Amazon Lex V2 den Slot-Wert erkennt, wird der Wert verschleiert. Wenn Amazon Lex V2 einen Wert nicht erkennt, wird die gesamte Äußerung verschleiert. Alle Slot-Werte in verpassten Äußerungen werden nicht verschleiert.

Amazon Lex V2 verschleiert auch keine Slot-Werte, die Sie in Anforderungs- oder Sitzungsattributen speichern. Wenn Sie Slot-Werte speichern, die als Attribut verschleiert werden sollen, müssen Sie den Wert verschlüsseln oder anderweitig verschleiern.

Amazon Lex V2 verschleiert den Slot-Wert im Audio nicht. Es verschleiert den Slot-Wert in der Audiotranskription.

Sie können mithilfe der Konsole oder mithilfe der Amazon Lex V2-API auswählen, welche Slots verschleiert werden sollen. Wählen Sie in der Konsole in den Einstellungen für einen Steckplatz die Option Slot-Verschleierung aus. Wenn Sie die API verwenden, stellen Sie das `obfuscationSetting` Feld des Steckplatzes auf den `DEFAULT_OBFUSCATION` Zeitpunkt ein, an dem Sie die Operation [CreateSlot](#) oder [UpdateSlot](#) aufrufen.

Selektive Erfassung von Konversationsprotokollen in Lex V2

Bei der selektiven Erfassung von Konversationsprotokollen kann der Benutzer auswählen, wie Konversationsprotokolle mit Text- und Audiodaten aus den Live-Konversationen erfasst werden.

Um die Ausgabe der Funktion zur selektiven Erfassung von Konversationsprotokollen zu aktivieren und zu erfassen, müssen Sie die Funktion in der Amazon Lex V2-Konsole aktivieren und die erforderlichen Sitzungsattribute in den API-Einstellungen aktivieren, um die ausgewählte Ausgabe aus den Protokollen zu erfassen.

Sie können die folgenden Optionen für die selektive Erfassung von Konversationsprotokollen auswählen:

- nur Text
- nur Audio
- Text und Audio

Sie können bestimmte Teile der Konversation aufzeichnen und wählen, ob Audio, Text oder beides für das Konversationsprotokoll aufgezeichnet werden sollen.

 Note

Die selektive Erfassung von Konversationsprotokollen funktioniert nur für Amazon Lex V2.

Themen

- [Verwalten Sie die selektive Erfassung von Konversationsprotokollen](#)
- [Beispiel für die selektive Erfassung von Konversationsprotokollen](#)

Verwalten Sie die selektive Erfassung von Konversationsprotokollen

Mithilfe der Lex-Konsole können Sie die Einstellungen für die selektive Aufzeichnung von Konversationsprotokollen aktivieren und auswählen, für welche Slots Sie die selektive Erfassung von Konversationsprotokollen aktivieren möchten.

Aktivieren Sie die selektive Erfassung von Konversationsprotokollen in der Amazon Lex V2-Konsole:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie in den linken Seitenbereichen Bots aus und wählen Sie den Bot aus, für den Sie die selektive Erfassung von Konversationsprotokollen aktivieren möchten. Verwenden Sie einen vorhandenen Bot oder erstellen Sie einen neuen.
3. Wählen Sie im Bereich Einsatz auf der linken Seite Aliase für Ihren ausgewählten Bot aus.
4. Wähle den Alias deines Bots und dann Konversationsprotokolle verwalten aus.
5. Wähle im Bereich „Konversationsprotokolle verwalten“ für Textprotokolle aus, ob Textprotokolle aktiviert oder deaktiviert werden sollen, indem du das Optionsfeld auswählst. Wenn Sie für Textprotokolle die Option Aktiviert wählen, müssen Sie einen Namen für die Protokollgruppe eingeben oder einen vorhandenen Protokollgruppennamen aus dem Drop-down-Menü auswählen. Aktivieren Sie das Kontrollkästchen für Äußerungen selektiv protokollieren, wenn Sie Textdateien selektiv protokollieren.

 Note

Aktivieren Sie and/or Text-Audioprotokolle, indem Sie in den Einstellungen für Konversationsprotokolle (and/or Textaudio) in den Build-Time-Einstellungen das Kontrollkästchen Äußerungen selektiv protokollieren aktivieren. BotAlias Sie müssen die

CloudWatch Protokollgruppe und den Amazon S3 S3-Bucket konfigurieren, um diese Option auszuwählen.

6. Wählen Sie im Abschnitt Audioprotokolle aus, ob Audioprotokolle aktiviert oder deaktiviert werden sollen, indem Sie das Optionsfeld auswählen. Wenn Sie „Aktiviert“ für Audioprotokolle wählen, müssen Sie den Amazon S3 S3-Bucket-Speicherort und (optional) den KMS-Schlüssel für die Verschlüsselung Ihrer Audiodaten angeben. Aktivieren Sie das Kontrollkästchen für Äußerungen selektiv protokollieren, wenn Sie Audiodateien selektiv protokollieren.

Manage conversation logs

Text logs

Configure text logging in Amazon CloudWatch Logs log groups. Text logging stores text input, transcripts of audio input, and associated metadata.

Text logs

☒ Enabled

☐ Disabled

☒ Selectively log utterances

When activated, only utterances that trigger intents and slots specified in session attributes will be logged. [Learn more](#)

Log group name

[Learn more about CloudWatch logs](#)

[Learn more about CloudWatch logs encryption](#)

Audio logs

Configure audio logging to an S3 bucket. Audio logging stores audio input as recordings.

Audio logs

☒ Enabled

☐ Disabled

☒ Selectively log utterances

When activated, only utterances that trigger intents and slots specified in session attributes will be logged. [Learn more](#)

S3 Bucket

KMS key - optional

[Learn more about Amazon S3](#)

[Learn more about Amazon S3 encryption](#)

7. Wählen Sie in der unteren rechten Ecke des Panels Speichern aus, um Ihre Einstellungen für die selektive Erfassung von Konversationsprotokollen zu speichern.

Aktivieren Sie die selektive Erfassung von Konversationsprotokollen in der Konsole:

1. Gehen Sie zu Intents und wählen Sie den Namen der Absicht, Erste Antwort, Erweiterte Einstellungen, Werte festlegen und Sitzungsattribute aus.
2. Stellen Sie die folgenden Attribute auf basierend auf den Absichten und Slots ein, für die Sie die selektive Erfassung von Konversationsprotokollen aktivieren möchten:
 - `x-amz-lex:enable-audio-logging:intent:slot` = "true"
 - `x-amz-lex:enable-text-logging:intent:slot` = "true"

Initial response advanced options [Info](#)User request acknowledgement [Info](#)

You can provide messages to acknowledge a user's request. You can provide responses, set values, and next steps. You can also branch based on conditions.

▶ Response for acknowledging the user's request

Message: -

▼ Set values

-

Next step in conversation

Invoke dialog code hook

Slot values - *optional*

Add slot values as: {slot} = value

```
{slot} = "value"
{slot} = $.transcriptions[N]...
{slot} = [session attribute]
```

Separate values with a new line.

Session attributes - *optional*

Add session attributes as: [session attribute] = value

```
x-amz-lex:enable-audio-logging:<intent>:<slot> =
"true"
x-amz-lex:enable-text-logging:<intent>:<slot> =
"true"
```

Separate values with a new line.

Next step in conversation

Invoke dialog code hook

[+ Add conditional branching](#)

Dialog code hook [Info](#)

☒ Active

You can enable Lambda functions to manage initialize the conversation.

▶ Lambda dialog code hook

Invoke Lambda function: Yes

Cancel

Update options

Note

Legt fest `x-amz-lex:enable-audio-logging:intent:slot = "true"`, dass Äußerungen erfasst werden, die nur einen bestimmten Punkt in der Konversation enthalten. Die Aktion zur Protokollierung einer Äußerung hängt von der Bewertung

folgender Faktoren ab *intent* : *slot* innerhalb der Äußerung, im Vergleich zu den Ausdrücken des Sitzungsattributs und dem entsprechenden Flag-Wert. Um eine Äußerung zu protokollieren, muss sie von mindestens einem Ausdruck im Sitzungsattribut zugelassen werden, wobei das Kennzeichen „Protokollierung aktivieren“ auf gesetzt ist. `true` Der Wert von *intent* und *slot* kann "*" auch sein. Wenn der and/or Slot-Intent-Wert gleich ist "*", bedeutet das, dass jeder and/or Slot-Intent-Wert von "*" diesem Wert entspricht. Ähnlich wie wird ein neues Sitzungsattribut aufgerufen `x-amz-lex:enable-audio-logging`, das zur Steuerung von Textprotokollen verwendet `x-amz-lex:enable-text-logging` wird.

3. Wählen Sie Aktualisierungsoptionen und erstellen Sie den Bot so, dass er die aktualisierten Einstellungen enthält.

Note

Ihre IAM-Rolle muss über eine Zugriffsberechtigung verfügen, damit Sie Daten in den Amazon S3 S3-Bucket schreiben und die Daten mit einem KMS-Schlüssel verschlüsseln können. Lex aktualisiert Ihre IAM-Rolle mit Lex-Berechtigungen für den Zugriff auf die CloudWatch Logs-Protokollgruppe und den ausgewählten Amazon S3 S3-Bucket.

Richtlinien für die Verwendung der selektiven Erfassung von Konversationsprotokollen:

Sie können die selektive Erfassung von Konversationsprotokollen für and/or Text-Audioprotokolle nur aktivieren, wenn Sie in den Einstellungen für das Konversationsprotokoll die Option and/or Text-Audioprotokolle aktiviert haben. Wenn Sie die selektive Erfassung von Konversationsprotokollen für and/or Text-Audioprotokolle aktivieren, deaktivieren Sie die Protokollierung für alle Aspekte und Abschnitte der Konversation. Um and/or Text-Audioprotokolle für bestimmte Absichten und Zeitpunkte zu generieren, müssen Sie die Sitzungsattribute für die selektive Aufzeichnung von or/and Text-Audio-Konversationsprotokollen für diese Absichten und Slots auf „true“ setzen.

- Wenn die selektive Aufzeichnung von Konversationsprotokollen aktiviert ist und keine Sitzungsattribute mit dem Präfix `x-amz-lex:` vorhanden `enable-audio-logging` sind, wird die Protokollierung standardmäßig für alle Äußerungen deaktiviert. Dieses Szenario gilt auch für `x-amz-lex:enable-text-logging`.
- Äußerungsprotokolle werden ausschließlich für die Segmente der and/or Text-Audiokonversation gespeichert, sofern mindestens ein Ausdruck im Sitzungsattribut dies zulässt.

- Die in den Sitzungsattributen definierten Konfigurationen für die selektive Erfassung von and/or Textaudios in Konversationsprotokollen sind nur wirksam, wenn die selektive Erfassung von Konversationsprotokollen für and/or Textaudio in den Einstellungen für das Konversationsprotokoll innerhalb des Bot-Alias aktiviert ist. Andernfalls werden die Sitzungsattribute ignoriert.
- Wenn die selektive Erfassung von Konversationsprotokollen aktiviert ist, werden alle Slot-Werte in SessionState Interpretationen und Transkriptionen, für die die Protokollierung mithilfe von Sitzungsattributen nicht aktiviert ist, im generierten Textprotokoll verschleiert.
- Die Entscheidung, and/or Audio-Textprotokolle zu erstellen, wird bewertet, indem der vom Bot ausgelöste Slot mit den Sitzungsattributen für die selektive Aufzeichnung von Konversationsprotokollen abgeglichen wird, mit Ausnahme der Runde zur Absichtserkennung, bei der der Benutzer Slot-Werte zusammen mit der Absichtserkennung angeben kann. In einer Runde zur Absichtserkennung werden die in der aktuellen Runde gefüllten Felder mit den Sitzungsattributen für die selektive Erfassung von Konversationsprotokollen abgeglichen.
- Die Slots, die als gefüllt gelten, werden aus dem Sitzungsstatus am Ende des Zuges abgeleitet. Daher wirken sich alle vom Dialog Codehook Lambda an den Slots im Sitzungsstatus vorgenommenen Änderungen auf das Verhalten der selektiven Erfassung von Konversationsprotokollen aus.
- Wenn der Benutzer mehrere Slot-Werte angibt, wird das and/or Text-Audioprotokoll nur dann generiert, wenn die text/audio Sitzungsattribute die Protokollierung aller in diesem Zug gefüllten Slots zulassen.
- Die empfohlene Vorgehensweise besteht darin, das Sitzungsattribut für die selektive Erfassung von Konversationsprotokollen zu Beginn der Sitzung festzulegen und es während der Sitzung nicht zu ändern.
- Wenn Steckplätze vertrauliche Daten enthalten, sollten Sie die Slot-Verschleierung immer aktivieren.

Beispiel für die selektive Erfassung von Konversationsprotokollen

Hier ist ein Beispiel für einen geschäftlichen Anwendungsfall für die selektive Erfassung von Konversationsprotokollen.

Anwendungsfall:

Ein Fintech-Unternehmen verwendet einen Amazon Lex V2-Bot zur Unterstützung seines IVR-Systems, mit dem Benutzer Rechnungszahlungen vornehmen können. Um die Compliance- und Prüfanforderungen zu erfüllen, müssen sie Audioaufzeichnungen der vom Benutzer erteilten

Autorisierungseinwilligungen aufbewahren. Die Aktivierung allgemeiner Audioprotokolle ist jedoch nicht durchführbar, da sie dadurch nicht den Vorschriften entsprechen würden, da es nicht möglich ist, sensible Bereiche wie CardNumber CVV und andere Informationen in den Audioprotokollen zu verschleiern. Stattdessen können sie die selektive Erfassung von Konversationsprotokollen für Audioprotokolle aktivieren und das Sitzungsattribut so festlegen, dass nur Audioprotokolle für Äußerungen erstellt werden, für die eine Autorisierung vorliegt.

BotAlias Einstellungen:

- Textprotokolle aktiviert: wahr
- Selektive Protokollierung für Textprotokolle aktiviert: falsch
- Audioprotokolle aktiviert: wahr
- Selektive Protokollierung von Audioprotokollen aktiviert: true

Sitzungsattribute:

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

Beispiel für ein Gespräch:

- Benutzer (Audioeingang): „Ich möchte meine Rechnung mit der Rechnungsnummer 35 bezahlenXU68.“
- Bot: „Was ist der fällige Betrag in Dollar?“
- Benutzer (Audioeingang): „235.“
- Bot: „Was ist deine Kreditkartennummer?“
- Benutzer (Audioeingang): „9239829722200348“.
- Bot: „Sie zahlen 235 Dollar mit Ihrer Kreditkartennummer, die auf 0348 endet. Bitte sagen Sie 'Ich autorisiere, 235 Dollar zu zahlen'.“
- Benutzer (Audioeingang): „Ich autorisiere, 235 Dollar zu zahlen.“
- Bot: „Ihre Rechnung wurde bezahlt.“

Ausgabe von Konversationsprotokollen:

In diesem Fall werden Textprotokolle für alle Runden erstellt. Audioprotokolle werden jedoch nur für den jeweiligen Zug aufgezeichnet, in dem der AuthorizationConsentSlot innerhalb der PayBillAbsicht ausgelöst wurde, und es werden keine Audioprotokolle für jede andere Runde erstellt.

Fehler mit Fehlerprotokollen in Lex V2 protokollieren

Sie aktivieren Fehlerprotokolle, um Bot-Interaktionen zu speichern. Sie können diese Fehlerprotokolle verwenden, um die Leistung Ihres Bots zu überprüfen und Fehler bei Konversationen zu beheben.

Fehlerprotokolle sind für eine Version konfiguriert. Jede Version kann unterschiedliche Einstellungen für ihre Fehlerprotokolle haben. Textprotokolle speichern Texteingaben in CloudWatch Protokollen. Sie können die Verschlüsselung von Textprotokollen mithilfe von AWS KMS Customer Managed aktivieren CMKs.

IAM-Richtlinien für Fehlerprotokolle

Abhängig von der Art der Protokollierung, die Sie auswählen, benötigt Amazon Lex V2 die Erlaubnis, Amazon CloudWatch Logs- und Amazon Simple Storage Service (S3) -Buckets zum Speichern Ihrer Protokolle zu verwenden. Sie müssen AWS Identity and Access Management Rollen und Berechtigungen erstellen, damit Amazon Lex V2 auf diese Ressourcen zugreifen kann.

Eine IAM-Rolle und Richtlinien für Fehlerprotokolle erstellen

Um Konversationsprotokolle zu aktivieren, müssen Sie Schreibberechtigungen für CloudWatch Logs und Amazon S3 erteilen. Wenn Sie die Objektverschlüsselung für Ihre S3-Objekte aktivieren, müssen Sie Zugriffsberechtigungen für die AWS KMS Schlüssel erteilen, die zur Verschlüsselung der Objekte verwendet werden.

Sie können die IAM-Konsole, die IAM-API oder die verwenden, um die Rolle und die Richtlinien AWS Command Line Interface zu erstellen. In diesen Anweisungen werden AWS CLI die Rolle und die Richtlinien erstellt.

Um eine IAM-Rolle für Fehlerprotokolle zu erstellen

Die IAM-Rolle, die Sie zum Aktivieren von Konversationsprotokollen verwenden, muss über die Berechtigung `iam:verfügen`. PassRole Die folgende Richtlinie sollte der Rolle beigefügt werden:

JSON

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



```
"Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "arn:aws:iam::account:role/role"
}
]
```

Fehlerprotokolle in Lex V2 aktivieren

So aktivieren Sie Fehlerprotokolle mit der Amazon Lex V2-Konsole:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Bot aus, den Sie für Fehlerprotokolle aktivieren möchten.
3. Wählen Sie im linken Menü Version.
4. Wählen Sie in der Versionsliste die Version aus, für die Sie Fehlerprotokolle konfigurieren möchten.
5. Wählen Sie im Abschnitt Versionsdetails die Option Aktivieren aus.
6. Wählen Sie Save (Speichern), um Konversationen zu protokollieren. Falls erforderlich, aktualisiert Amazon Lex V2 Ihre Servicerolle mit Berechtigungen für den Zugriff auf die Protokollgruppe CloudWatch Logs.

Fehlerprotokolle in Lex V2 deaktivieren

So deaktivieren Sie Fehlerprotokolle mit der Amazon Lex V2-Konsole:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Liste der Bots den Bot aus, den Sie für Fehlerprotokolle aktivieren möchten.
3. Wählen Sie im linken Menü Version.
4. Wählen Sie in der Versionsliste die Version aus, für die Sie Fehlerprotokolle konfigurieren möchten.
5. Wählen Sie im Abschnitt Versionsdetails die Option Deaktivieren aus.
6. Wählen Sie Save (Speichern), um die Protokollierung von Konversationen zu beenden.

Überwachung der Betriebskennzahlen in Lex V2

Amazon CloudWatch und AWS CloudTrail sind zwei AWS Dienste, die in Amazon Lex V2 integriert sind, um Ihnen zu helfen, Benutzerinteraktionen mit Ihrem Bot zu überwachen. Verwenden Sie diese Dienste, um Aktionen aufzuzeichnen, Daten nahezu in Echtzeit zu senden und Benachrichtigungen und automatisierte Aktionen einzurichten, wenn die Kriterien erfüllt sind.

Themen

- [Messung betrieblicher Kennzahlen mit Amazon CloudWatch](#)
- [Ereignisse anzeigen mit AWS CloudTrail for Lex V2](#)

Messung betrieblicher Kennzahlen mit Amazon CloudWatch

Sie können Amazon Lex V2 mithilfe von Amazon Lex überwatchen CloudWatch, das Rohdaten sammelt und zu lesbaren Metriken verarbeitet, die nahezu in Echtzeit verfügbar sind. Diese Statistiken werden 15 Monate gespeichert, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt werden. Sie können auch Alarmer einrichten, die auf bestimmte Grenzwerte achten und Benachrichtigungen senden oder Aktivitäten auslösen, wenn diese Grenzwerte erreicht werden. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

Der Amazon Lex V2-Service meldet die folgenden Metriken im AWS/Lex Namespace.

Metrik	Beschreibung
AssistedSlotResolutionModelAccessDeniedErrorCount	Wie oft Amazon Lex V2 der Zugriff auf Amazon Bedrock verweigert wurde Gültige Abmessungen für die StartConversation Operationen RecognizeUtterance und: • BotId, BotAliasId LocaleId, Betrieb, InputMode, ModelType, Modell • BotId, BotVersion LocaleId, Betrieb, InputMode, ModelType, Modell Gültige Abmessungen für RecognizeText : • BotId, BotAliasId LocaleId, Betrieb, ModelType, Modell

Metrik	Beschreibung
	• BotId BotVersion, LocaleId, Betrieb ModelType, Modell Einheit: Anzahl
AssistedSlotResolutionModelInvocationCount	Die Häufigkeit, mit der Amazon Bedrock aufgerufen wurde. Gültige Abmessungen für die Operationen RecognizeUtterance und StartConversation : • BotId, BotAliasId LocaleId, Betrieb, InputMode, ModelType, Modell • BotId, BotVersion LocaleId, Betrieb, InputMode, ModelType, Modell Gültige Abmessungen für RecognizeText : • BotId, BotAliasId LocaleId, Betrieb, ModelType, Modell • BotId BotVersion, LocaleId, Betrieb ModelType, Modell Einheit: Anzahl
AssistedSlotResolutionModelSystemErrorCount	Die Häufigkeit, mit der ein 5xx beim Aufrufen von Amazon Bedrock aufgetreten ist. Gültige Abmessungen für die Operationen RecognizeUtterance und StartConversation : • BotId, BotAliasId LocaleId, Betrieb, InputMode, ModelType, Modell • BotId, BotVersion LocaleId, Betrieb, InputMode, ModelType, Modell Gültige Abmessungen für RecognizeText : • BotId, BotAliasId LocaleId, Betrieb, ModelType, Modell • BotId BotVersion, LocaleId, Betrieb ModelType, Modell Einheit: Anzahl

Metrik	Beschreibung
AssistedSlotResolutionModelThrottlingErrorCount	Die Häufigkeit, mit der Amazon Lex von Amazon Bedrock gedrosselt wurde. Gültige Abmessungen für die Operationen <code>StartConversation</code> und <code>RecognizeUtterance</code> : • BotId, BotAliasId LocaleId, Betrieb, InputMode, ModelType, Modell • BotId, BotVersion LocaleId, Betrieb, InputMode, ModelType, Modell Gültige Abmessungen für <code>RecognizeText</code> : • BotId, BotAliasId LocaleId, Betrieb, ModelType, Modell • BotId BotVersion, LocaleId, Betrieb ModelType, Modell Einheit: Anzahl
AssistedSlotResolutionResolvedSlotCount	Die Häufigkeit, mit der Amazon Bedrock einen Slot-Wert zurückgegeben hat. Gültige Abmessungen für die <code>StartConversation</code> Operationen <code>RecognizeUtterance</code> und: • BotId, BotAliasId LocaleId, Betrieb, InputMode, ModelType, Modell • BotId, BotVersion LocaleId, Betrieb, InputMode, ModelType, Modell Gültige Abmessungen für <code>RecognizeText</code> : • BotId, BotAliasId LocaleId, Betrieb, ModelType, Modell • BotId BotVersion, LocaleId, Betrieb ModelType, Modell Einheit: Anzahl

Metrik	Beschreibung
KendraIndexAccessError	Die Häufigkeit, mit der Amazon Lex V2 nicht auf Ihren Amazon Kendra Kendra-Index zugreifen konnte. • Betrieb,, BotId, BotAliasId LocaleId Einheit: Anzahl
KendraLatency	Die Zeit, die Amazon Kendra benötigt, um auf eine Anfrage von zu antworten. AMAZON.KendraSearchIntent Gültige Dimensionen: • Betrieb,, BotId, BotVersion LocaleId • Betrieb, BotId, BotAliasId, LocaleId Einheit: Millisekunden
KendraSuccess	Die Häufigkeit, mit der Amazon Lex V2 nicht auf Ihren Amazon Kendra Kendra-Index zugreifen konnte. Gültige Dimensionen: • Betrieb, BotId, BotVersion LocaleId • Betrieb, BotId, BotAliasId, LocaleId Einheit: Anzahl
KendraSystemErrors	Die Häufigkeit, mit der Amazon Lex V2 den Amazon Kendra Kendra-Index nicht abfragen konnte. Gültige Dimensionen: • Vorgang, BotId, BotAliasId, InputMode LocaleId Einheit: Anzahl

Metrik	Beschreibung
KendraThrottledEvents	Die Häufigkeit, mit der Amazon Kendra Anfragen von gedrosselt hat. <code>AMAZON.KendraSearchIntent</code> Gültige Dimensionen: Betrieb,, BotId, BotAliasId InputMode LocaleId Einheit: Anzahl
RuntimeConcurrency	Die Anzahl gleichzeitiger Verbindungen im angegebenen Zeitraum. <code>RuntimeConcurrency</code> wird als <code>StatisticSet</code> gemeldet. Gültige Abmessungen für die <code>StartConversation</code> Operationen <code>RecognizeUtterance</code> oder: - Operation BotId, BotVersion, InputMode, LocaleId - Betrieb, BotId, BotAliasId, InputMode, LocaleId Gültige Abmessungen für andere Operationen: - Betrieb BotId, BotVersion, LocaleId - Betrieb, BotId, BotAliasId, LocaleId Einheit: Anzahl
RuntimeInvalidLambdaResponses	Die Anzahl der ungültigen AWS Lambda Antworten im angegebenen Zeitraum. Gültige Dimensionen: Vorgang BotId, BotAliasId, InputMode, LocaleId Einheit: Anzahl

Metrik	Beschreibung
RuntimeLambdaErrors	Die Anzahl der Lambda-Laufzeitfehler im angegebenen Zeitraum. Gültige Dimensionen: • Vorgang, BotId, BotAliasId, InputMode LocaleId Einheit: Anzahl
RuntimePollyErrors	Die Anzahl der ungültigen Amazon Polly Polly-Antworten im angegebenen Zeitraum. Gültige Dimensionen: • Betrieb, BotId, BotAliasId, InputMode LocaleId Einheit: Anzahl
RuntimeRequestCount	Die Anzahl der Laufzeitanforderungen im angegebenen Zeitraum. Gültige Dimensionen für die StartConversation Operationen RecognizeUtterance und: • Operation BotId, BotVersion, InputMode, LocaleId • Betrieb, BotId, BotAliasId, InputMode, LocaleId Gültige Abmessungen für andere Operationen: • Betrieb BotId, BotVersion, LocaleId • Betrieb, BotId, BotAliasId, LocaleId Einheit: Anzahl

Metrik	Beschreibung
RuntimeRequestLength	Gesamtlänge einer Konversation mit einem Amazon Lex V2-Bot. Gilt nur für den StartConversation Vorgang. Gültige Dimensionen: • BotAliasId, BotId LocaleId, Betrieb • BotId, BotAliasId LocaleId, Betrieb Einheit: Millisekunden
RuntimeSuccessfulRequestLatency	Die Latenz für erfolgreiche Anfragen zwischen dem Zeitpunkt, an dem die Anfrage gestellt wurde, und der Rückgabe der Antwort. Gültige Abmessungen für die StartConversation Operationen RecognizeUtterance und: • Operation BotId, BotVersion, InputMode, LocaleId • Betrieb, BotId, BotAliasId, InputMode, LocaleId Gültige Abmessungen für andere Operationen: • Betrieb BotId, BotVersion, LocaleId • Betrieb, BotId, BotAliasId, LocaleId Einheit: Millisekunden

 Important

Diese Metrik ist RuntimeSuccessfulRequestLatency und nicht. RuntimeSuccessfulRequestLatency

Metrik	Beschreibung
RuntimeSystemErrors	Die Anzahl der Systemfehler im angegebenen Zeitraum. Der Antwortcodebereich für einen Systemfehler lautet 500 bis 599. Gültige Abmessungen für die Operationen und <code>RecognizeUtterance</code> : <code>StartConversation</code> • Operation <code>BotId</code>, <code>BotVersion</code>, <code>InputMode</code>, <code>LocaleId</code> • Betrieb, <code>BotId</code>, <code>BotAliasId</code>, <code>InputMode</code>, <code>LocaleId</code> Gültige Abmessungen für andere Operationen: • Betrieb <code>BotId</code>, <code>BotVersion</code>, <code>LocaleId</code> • Betrieb, <code>BotId</code>, <code>BotAliasId</code>, <code>LocaleId</code> Einheit: Anzahl

Metrik	Beschreibung
RuntimeThrottledEvents	Die Anzahl der gedrosselten Ereignisse. Amazon Lex V2 drosselt ein Ereignis, wenn mehr Anfragen eingehen als das für Ihr Konto festgelegte Limit an Transaktionen pro Sekunde. Wenn der Grenzwert für Ihr Konto häufig überschritten wird, können Sie eine Erweiterung des Limits beantragen. Informationen zur Beantragung einer Erhöhung finden Sie unter AWS-Servicebeschränkungen. Gültige Abmessungen für die StartConversation Operationen RecognizeUtterance und: • Operation BotId, BotVersion, InputMode, LocaleId • Betrieb, BotId, BotAliasId, InputMode, LocaleId Gültige Abmessungen für andere Operationen: • Betrieb BotId, BotVersion, LocaleId • Betrieb, BotId, BotAliasId, LocaleId Einheit: Anzahl

Metrik	Beschreibung
RuntimeUserErrors	Die Anzahl der Benutzerfehler im angegebenen Zeitraum. Der Antwortcode-Bereich für einen Benutzerfehler ist 400 bis 499. Gültige Abmessungen für die StartConversation Operationen RecognizeUtterance und: • Operation BotId, BotVersion, InputMode, LocaleId • Betrieb, BotId, BotAliasId, InputMode, LocaleId Gültige Abmessungen für andere Operationen: • Betrieb BotId, BotVersion, LocaleId • Betrieb, BotId, BotAliasId, LocaleId Einheit: Anzahl

Die folgenden Dimensionen werden für die Amazon Lex V2-Metriken unterstützt.

Dimension	Beschreibung
Operation	Der Name des Amazon Lex V2-Vorgangs —RecognizeText, —RecognizeUtterance, StartConversation, GetSession, DeleteSession —PutSession, —
BotId	Die alphanumerische eindeutige Kennung für den Bot.
BotAliasId	Die alphanumerische eindeutige Kennung für den Bot-Alias.
BotVersion	Die numerische Version des Bots.
InputMode	Die Art der Eingabe für den Bot — Sprache, Text oder DTMF.
LocaleId	Die Kennung des Gebietsschemas des Bots, z. B. en-US oder fr-CA.

Dimension	Beschreibung
Model	Gibt die Modell-ID des großen Sprachmodells von Amazon Bedrock an.
ModelType	Gibt den Typ des großen Sprachmodells an, das von Amazon Bedrock aufgerufen wurde.

Ereignisse anzeigen mit AWS CloudTrail for Lex V2

Amazon Lex V2 ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Amazon Lex V2 ausgeführt wurden. CloudTrail erfasst API-Aufrufe für Amazon Lex V2 als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Amazon Lex V2-Konsole und Code-Aufrufe der Amazon Lex V2-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon Lex V2. Auch wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole in Event history (Ereignisverlauf) anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Amazon Lex V2 gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im [AWS CloudTrail Benutzerhandbuch](#).

Informationen zu Amazon Lex V2 in CloudTrail

CloudTrail ist für Ihr AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Amazon Lex V2 eine Aktivität auftritt, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter [Ereignisse mit dem CloudTrail Ereignisverlauf anzeigen](#).

Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Amazon Lex V2, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS -Services konfigurieren, um die in den CloudTrail -Protokollen

erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie hier:

- [Übersicht zum Erstellen eines Trails](#)
- [CloudTrail unterstützte Dienste und Integrationen](#)
- [Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail](#)
- [Empfangen von CloudTrail Protokolldateien aus mehreren Regionen](#) und [Empfangen von CloudTrail Protokolldateien von mehreren Konten](#)

Amazon Lex V2 unterstützt die Protokollierung für alle in [Model Building API V2](#) aufgeführten Aktionen.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management IAM-Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter [CloudTrail -Element userIdentity](#).

Grundlegendes zu Amazon Lex V2-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die [CreateBotAlias](#)Aktion demonstriert.

```
{
  "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",

```

```

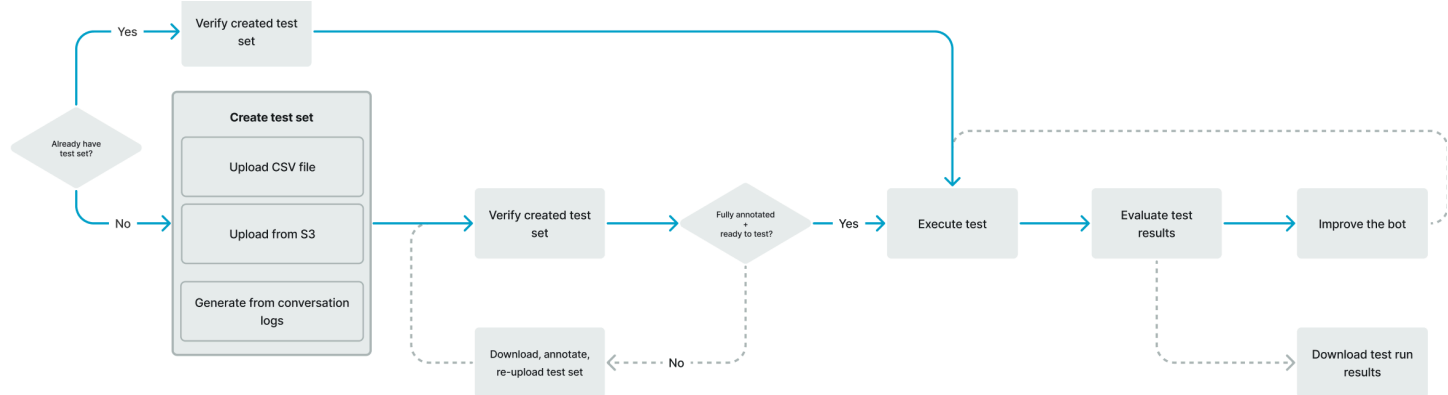
    "botId": "bot ID",
    "botVersion": "1",
    "creationDateTime": creation timestamp
  },
  "requestID": "unique request ID",
  "eventID": "unique event ID",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}

```

Bewertung der Leistung von Lex V2-Bots mit der Test Workbench

Um die Leistung Ihrer Bots zu verbessern, können Sie die Leistung Ihrer Bots im großen Maßstab bewerten. Die Ergebnisse Ihrer Testauswertung werden in einfachen Tabellen und Diagrammen angezeigt.

Sie können die Test Workbench verwenden, um Referenztestsätze zu erstellen, die vorhandene Transkriptionsdaten verwenden. Sie können Bots testen, um die Leistung vor der Bereitstellung zu bewerten, und sich die Aufschlüsselung der Testergebnisse in großem Umfang ansehen.



Benutzer können die Test Workbench verwenden, um die Basisleistung für Bots zu ermitteln. Dies umfasst die Absicht und die Leistung des Zeitfensters für Äußerungen, die in Form von Einzeleingaben oder Konversationen erfolgen. Sobald ein Testset erfolgreich geladen wurde, können Sie es mit Ihren vorhandenen Vorproduktions- oder Produktions-Bots ausführen. Die Test Workbench hilft Ihnen dabei, Möglichkeiten für eine bessere Ausfüllung von Slots und die Klassifizierung von Absichten zu identifizieren.

Themen

- [Generieren Sie ein Testset für Test Workbench](#)
- [Testsätze verwalten](#)
- [Führen Sie einen Test aus](#)
- [Abdeckung der Testsätze in Test Workbench](#)
- [Testergebnisse anzeigen](#)
- [Details zu den Testergebnissen in Test Workbench](#)

Generieren Sie ein Testset für Test Workbench

Sie können ein Testset erstellen, um die Leistung Ihres Bots zu bewerten. Generieren Sie einen Testsatz, indem Sie einen Testsatz hochladen, der in einem CSV-Dateiformat vorliegt, oder indem Sie einen Testsatz aus [Konversationsprotokollen](#) generieren. Das Testset kann Audio- oder Texteingaben enthalten.

Creation method

☒ Generate a baseline test set

Automatically generate test set from your bot design or conversation log.



☐ Upload a file to this test set

Upload test set in CSV format or ingest from your selected S3 bucket.



▼ How it works



Step 1. Generate a baseline test set

A CSV file will be generated based on your existing data



Step 2. Review and annotate

Download and evaluate the test set file to make any necessary annotations.



Step 3. Update the test set

Upload an annotated test set file and you'll be ready for testing.

Baseline test set creation

☐ Generate from bot configuration

Automatically generate test set from your bot using sample utterances mapped to the intents and slots.

☒ Generate from conversation logs

Automatically generate test set from your bot using conversation logs

Bot name

-- Select a bot -- ▼

Bot alias

-- Select an alias -- ▼

Language

-- Select a language -- ▼

Time range

 Filter by a date and time range

IAM role [Info](#)

Amazon Lex requires permissions to access your conversation logs.

☒ Create an IAM role

Your role grants Amazon Lex permission to access other AWS services on your behalf.

[Learn more about the permissions policy attached to this role.](#) [↗](#)

☐ Use an existing IAM role

Wenn ein Testsatz Validierungsfehler verursacht, entfernen Sie den Testsatz und ersetzen Sie ihn durch eine andere Liste von Testsatzdaten, oder bearbeiten Sie die Daten in der CSV-Datei mit einem Tabellenkalkulationsprogramm.

Um ein Testset zu erstellen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie in der linken Seitenleiste Test Workbench aus.
3. Wählen Sie in den Optionen unter Test-Workbench die Option Testsätze aus.
4. Wählen Sie in der Konsole die Schaltfläche Testsatz erstellen aus.
5. Geben Sie in den Details einen Namen für das Testset und optional eine Beschreibung ein.
6. Wählen Sie Basis-Testsatz generieren aus.
7. Wählen Sie Aus Konversationsprotokollen generieren aus.
8. Wählen Sie Bot-Name, Bot-Alias und Sprache aus den Drop-down-Menüs aus.
9. Wenn Sie einen Basistest anhand eines Konversationsprotokolls generieren, wählen Sie bei Bedarf Zeitraum und IAM-Rolle aus. Sie können eine Rolle mit den grundlegenden Amazon Lex V2-Berechtigungen erstellen oder eine vorhandene Rolle verwenden.
10. Wählen Sie eine Modalität von Audio oder Text für das Testset, das Sie erstellen. HINWEIS: Die Test Workbench kann Textdateien bis zu 50.000 und bis zu 5 Stunden Audio importieren.
11. Wählen Sie einen Amazon S3 S3-Standort aus, um Ihre Testergebnisse zu speichern, und fügen Sie einen optionalen KMS-Schlüssel hinzu, um Ausgabetranskripte zu verschlüsseln.
12. Wählen Sie Erstellen aus.

Um einen vorhandenen Testsatz in einem CSV-Dateiformat hochzuladen oder den Testsatz zu aktualisieren:

1. Wählen Sie in der linken Seitenleiste die Option Test Workbench aus.
2. Wählen Sie in den Optionen unter Test-Workbench die Option Testsätze aus.
3. Wählen Sie auf der Konsole eine Datei in dieses Testset hochladen aus.
4. Wählen Sie Aus Amazon Amazon S3 S3-Bucket hochladen oder Von Ihrem Computer hochladen. HINWEIS: Sie können eine aus einer Vorlage erstellte CSV-Datei hochladen. Klicken Sie auf CSV-Vorlage, um eine ZIP-Datei herunterzuladen, die die Vorlagen enthält.
5. Wählen Sie „Eine Rolle mit grundlegenden Amazon Lex Lex-Berechtigungen erstellen“ oder „Eine bestehende Rolle verwenden“ für die Rolle ARN aus.
6. Wählen Sie eine Modalität von Audio oder Text für das Testset, das Sie erstellen. HINWEIS: Die Test Workbench kann Textdateien bis zu 50.000 und bis zu 5 Stunden Audio importieren.

7. Wählen Sie einen Amazon S3 S3-Standort aus, um Ihre Testergebnisse zu speichern, und fügen Sie einen optionalen KMS-Schlüssel hinzu, um Ausgabetranskripte zu verschlüsseln.
8. Wählen Sie Erstellen aus.

Wenn der Vorgang erfolgreich ist, wird in der Bestätigungsnachricht angezeigt, dass der Testsatz testbereit ist, und der Status wird Bereit zum Testen angezeigt.

Tipps für die Erstellung eines erfolgreichen Testsets

- Sie können eine IAM-Rolle für die Test Workbench in der Konsole erstellen oder Ihre IAM-Rolle konfigurieren. [step-by-step](#) Weitere Informationen finden Sie unter [Erstellen einer IAM-Rolle für die Test Workbench](#).
- Bevor Sie einen Test ausführen, überprüfen Sie den Testsatz und die Bot-Definition mithilfe der Schaltfläche Diskrepanz validieren auf etwaige Inkonsistenzen. Wenn die im Testsatz verwendeten Konventionen für Absicht und Slot-Benennung mit denen des Bots übereinstimmen, fahren Sie mit der Ausführung des Tests fort. Wenn Anomalien festgestellt werden, überarbeiten Sie den Testsatz, aktualisieren Sie den Testsatz und wählen Sie Diskrepanz validieren aus. Wiederholen Sie diese Sequenz erneut, bis keine Inkonsistenzen mehr festgestellt werden, und führen Sie dann den Test aus.
- Die Test Workbench kann mit verschiedenen Slot-Werteformaten in der Spalte Expected Output Slot testen. Für jeden integrierten Slot können Sie den in der Benutzereingabe angegebenen Wert wählen (z. B. Datum = morgen) oder seinen absoluten aufgelösten Wert angeben (z. B. Datum = 2023-03-21). [Weitere Informationen zu integrierten Steckplätzen und ihren absoluten Werten finden Sie unter Integrierte Steckplätze.](#)
- Aus Gründen der Konsistenz und Lesbarkeit der Spalten mit dem erwarteten Ausgabeslot sollten Sie die Konvention "SlotName = SlotValue" (z. B. AppointmentType = bereinigen) mit einem Leerzeichen vor und nach dem Gleichheitszeichen einhalten.
- Wenn der Bot zusammengesetzte Steckplätze enthält, definieren Sie unter Expected Output Slot Unterslots für den Slot-Namen, getrennt durch einen Punkt (zum Beispiel „Car.Color“). Andere Syntax und Interpunktion funktionieren nicht.
- Wenn der Bot Slots mit mehreren Werten enthält, geben Sie im Feld Expected Output Slot mehrere Slot-Werte an, die durch ein Komma getrennt sind (" FlowerType = Rosen, Lilien"). Andere Syntax und Interpunktion funktionieren nicht.
- Stellen Sie sicher, dass der Testsatz aus gültigen Konversationsprotokollen erstellt wurde.
- Der Wert slot:slot befindet sich im CSV-Format in derselben Spalte nach den Intent-Spalten.

- DTMF-Eingaben von einem Benutzer, der an der Reihe ist, werden als erwartete Transkription interpretiert und enthalten keinen Amazon S3 S3-Standort.

Erstellen eines Testfalls innerhalb eines Testsatzes mithilfe von Test Workbench

Die Ergebnisse von Test Workbench hängen von der Bot-Definition und dem entsprechenden Testsatz ab. Sie können einen Testsatz mit den Informationen aus der Bot-Definition generieren, um Bereiche zu ermitteln, die verbessert werden müssen. Erstellen Sie einen Testdatensatz mit Beispielen, von denen Sie vermuten (oder wissen), dass es für den Bot schwierig sein wird, sie richtig zu interpretieren, wenn Sie das aktuelle Bot-Design und Ihr Wissen über Ihre Kundengespräche berücksichtigen.

Überprüfen Sie regelmäßig Ihre Absichten auf der Grundlage der Erkenntnisse aus Ihrem Produktions-Bot. Füge die Beispieläußerungen und Slot-Werte des Bots hinzu und passe sie an. Erwägen Sie, die Slot-Auflösung zu verbessern, indem Sie die verfügbaren Optionen verwenden, z. B. Hinweise zur Laufzeit. Das Design und die Entwicklung Ihres Bots sind ein iterativer Prozess, der ein kontinuierlicher Zyklus ist.

Hier sind einige weitere Tipps zur Optimierung Ihres Testsets:

- Wählen Sie die häufigsten Anwendungsfälle mit häufig verwendeten Absichten und Slots im Testset aus.
- Erkunden Sie verschiedene Möglichkeiten, wie ein Kunde auf Ihre Absichten und Slots verweisen könnte. Dies kann Benutzereingaben in Form von Aussagen, Fragen und Befehlen beinhalten, deren Länge von minimal bis umfangreich variiert.
- Schließen Sie Benutzereingaben mit einer unterschiedlichen Anzahl von Steckplätzen ein.
- Fügen Sie häufig verwendete Synonyme oder Abkürzungen für benutzerdefinierte Slot-Werte hinzu, die von Ihrem Bot unterstützt werden (z. B. „Root Canal“, „Canal“ oder „RC“).
- Fügen Sie Varianten der integrierten Slot-Werte hinzu (z. B. „morgen“, „so schnell wie möglich“ oder „am nächsten Tag“).
- Untersuchen Sie die Robustheit des Bots für die gesprochene Modalität, indem Sie Benutzereingaben sammeln, die falsch interpretiert werden können (z. B. „Tinte“, „Knöchel“ oder „Anker“).

Erstellen eines Testsatzes aus einer CSV-Datei für Test Workbench

Sie können einen Testsatz aus der CSV-Dateivorlage erstellen, die in der Amazon Lex V2-Konsole bereitgestellt wird, indem Sie die Werte direkt mit einem CSV-Tabellenkalkulationseditor eingeben. Das Testset ist eine Datei mit kommagetrennten Werten (CSV), die aus Äußerungen einzelner Benutzer und Konversationen mit mehreren Runden besteht, die in den folgenden Spalten aufgezeichnet wurden:

- **Zeile #** — Diese Spalte ist ein inkrementeller Zähler, der die Gesamtzahl der zu testenden Zeilen erfasst.
- **Konversation #** — In dieser Spalte wird die Anzahl der Runden in einer Konversation erfasst. Bei einzelnen Eingaben kann diese Spalte leer gelassen und mit „-“ oder „N/A“ gefüllt werden. Bei Konversationen wird jeder Runde innerhalb einer Konversation dieselbe Konversationsnummer zugewiesen.
- **Quelle** — Diese Spalte ist auf „Benutzer“ oder „Agent“ gesetzt. Bei Einzeleingaben wird sie immer auf „Benutzer“ gesetzt.
- **Eingabe** — Diese Spalte enthält die Benutzeräußerung oder die Bot-Eingabeaufforderungen.
- **Erwartete Ausgabeabsicht** — In dieser Spalte wird die in der Eingabe erfüllte Absicht erfasst.
- **Absicht Expected Output Slot 1** — In dieser Spalte wird der erste Slot erfasst, der in der Benutzereingabe ausgelöst wurde. Das Testset sollte für jeden Slot in der Benutzereingabe eine Spalte mit dem Namen Expected Output Slot X enthalten.

Beispiel für einen Testsatz mit einzelnen Eingaben:

Zeile #	Konversation #	Quelle	Eingabe	Erwartete Ausgabeabsicht	Erwarteter Ausgangsslotplatz 1	Erwarteter Ausgangsslotplatz 2
1		Benutzer	buchen Sie morgen einen Reinigungstermin	MakeAppointment	AppointmentType = Reinigung	Datum = morgen

Zeile #	Konversation #	Quelle	Eingabe	Erwartete Ausgabeabsicht	Erwarteter Ausgangss teckplatz 1	Erwarteter Ausgangss teckplatz 2
2	N/A	Benutzer	buchen Sie einen Reinigungstermin am 15. April	MakeAppointment	AppointmentType = Reinigung	Datum = 15.04.23
3	N/A	Benutzer	buchen Sie einen Termin für den ersten Dezember	MakeAppointment	Datum = Erster Dezember	
4	N/A	Benutzer	einen Reinigungstermin buchen	MakeAppointment	AppointmentType = Reinigung	
1		Benutzer	Können Sie mir helfen, einen Termin zu buchen?	MakeAppointment		

Beispiel für ein Testset mit Konversationen

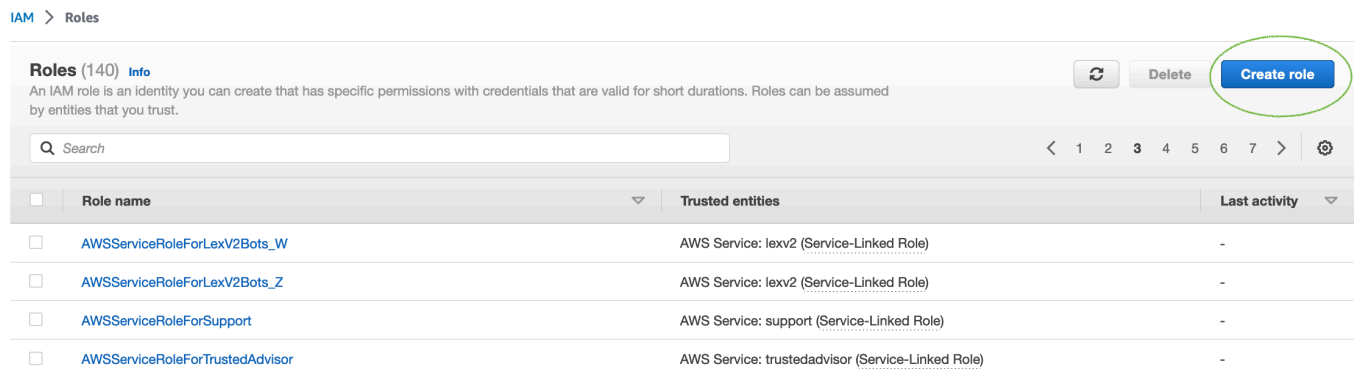
Zeile #	Konversation #	Quelle	Eingabe	Erwartete Ausgabeabsicht	Erwarteter Ausgangsteckplatz 1	Erwarteter Ausgangsteckplatz 2	Erwarteter Ausgangsteckplatz 3
1	1	Benutzer	einen Termin buchen	MakeAppointment			
2	1	Kundendienstmitarbeiter	Welche Art von Termin möchten Sie vereinbaren?	MakeAppointment			
3	1	Benutzer	Reinigung	MakeAppointment	AppointmentType = Reinigung		
4	1	Kundendienstmitarbeiter	Wann sollte ich Ihren Termin vereinbaren?	MakeAppointment			
5	1	Benutzer	tomorrow	MakeAppointment		Datum = morgen	
6	2	Benutzer	buchen Sie noch heute einen	MakeAppointment	AppointmentType = Wurzelkanal	Datum = heute	

Zeile #	Konversation #	Quelle	Eingabe	Erwartete Ausgabeabsicht	Erwarteter Ausgangsteckplatz 1	Erwarteter Ausgangsteckplatz 2	Erwarteter Ausgangsteckplatz 3
			Wurzelkanaltermin				
7	2	Kundendienstmitarbeiter	Zu welcher Zeit sollte ich Ihren Termin vereinbaren?	MakeAppointment			
8	2	Benutzer	elf Uhr	MakeAppointment			Zeit = elf Uhr

Erstellen Sie eine IAM-Rolle für die Test Workbench

Um eine IAM-Rolle für die Test Workbench zu erstellen

1. Folgen Sie den Schritten unter Einen IAM-Benutzer [erstellen, um einen IAM-Benutzer](#) zu erstellen, der für den Zugriff auf die Test-Workbench-Konsole verwendet werden kann.
2. Wählen Sie die Schaltfläche Create role.



3. Wählen Sie die Option für Benutzerdefinierte Vertrauensrichtlinie aus.

IAM > Roles > Create role

Step 1

Select trusted entity

Step 2

Add permissions

Step 3

Name, review, and create

Select trusted entity [Info](#)

Trusted entity type

☐ AWS service

Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ Web identity

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☒ SAML 2.0 federation

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☒ Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

```

1 - {
2   "Version": "2012-10-17",
3   "Statement": [
4     {

```

Edit statement

Statement1

Remove

1 Add actions for STS

4. Geben Sie unten die Vertrauensrichtlinie ein und klicken Sie auf Weiter.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

5. Wählen Sie die Schaltfläche Richtlinie erstellen aus.
6. In Ihrem Browser wird eine neue Registerkarte geöffnet, in der Sie die unten stehende Richtlinie eingeben und auf Weiter: Tags klicken können.

JSON

```

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

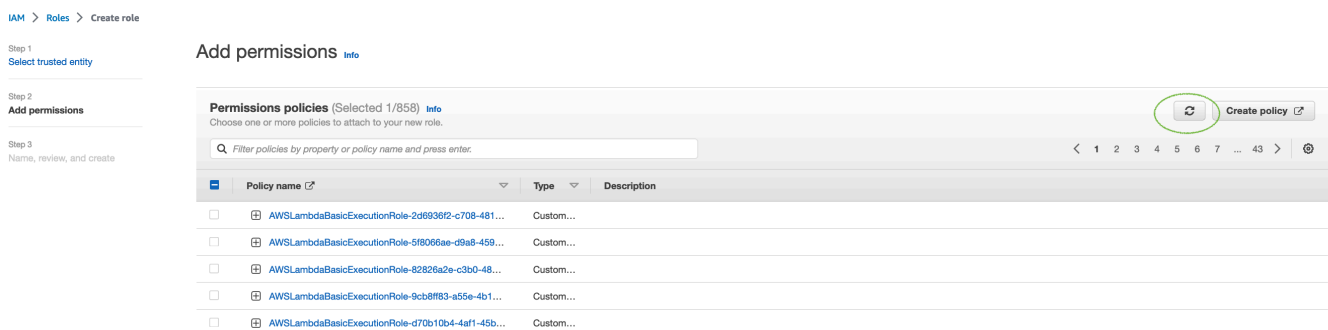
```

```

        "Action": [
            "s3:*"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:FilterLogEvents"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "lex:*"
        ],
        "Resource": "*"
    }
]
}

```

7. Geben Sie einen Richtliniennamen ein, zum Beispiel LexTestWorkbenchPolicy ", und klicken Sie dann auf Richtlinie erstellen.
8. Kehren Sie zur vorherigen Registerkarte in Ihrem Browser zurück und aktualisieren Sie die Liste der Richtlinien, indem Sie auf die Schaltfläche Aktualisieren klicken, wie unten gezeigt.



9. Suchen Sie in der Liste der Richtlinien, indem Sie den Richtliniennamen eingeben, den Sie im 6. Schritt verwendet haben, und die Richtlinie auswählen.
10. Wählen Sie die Schaltfläche Weiter.
11. Geben Sie den Rollennamen ein und klicken Sie dann auf die Schaltfläche Rolle erstellen.
12. Wählen Sie Ihre neue IAM-Rolle aus, wenn Sie in der Amazon Lex V2-Konsole für Test Workbench dazu aufgefordert werden.

Erstellen Sie eine IAM-Rolle für die Test Workbench — Erweiterte Funktionen

Einrichtung von Berechtigungen für die IAM-Rolle Test Workbench

Dieser Abschnitt enthält mehrere Beispiele für identitätsbasierte Richtlinien für AWS Identity and Access Management (IAM) zur Implementierung von Zugriffskontrollen mit geringsten Rechten für Test Workbench-Berechtigungen.

1. Richtlinie für Test Workbench zum Lesen von Audiodateien in S3 — Diese Richtlinie ermöglicht es Test Workbench, Audiodateien zu lesen, die in den Testsätzen verwendet werden. Die folgende Richtlinie sollte entsprechend geändert werden, um sie *S3Path* zu aktualisieren *S3BucketName* und sie auf einen Amazon S3 S3-Speicherort der Audiodateien in einem Testsatz zu verweisen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TestWorkbenchS3AudioFilesReadOnly",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::S3BucketName/S3Path/*"
      ]
    }
  ]
}
```

2. Richtlinie für Test Workbench zum Lesen und Schreiben von Testsätzen und Ergebnissen in einen Amazon S3 S3-Bucket — Diese Richtlinie ermöglicht es Test Workbench, die Eingaben und Ergebnisse der Testsätze zu speichern. Die folgende Richtlinie sollte geändert werden, um sie auf den Amazon S3 S3-Bucket *S3BucketName* zu aktualisieren, in dem Testsatzdaten gespeichert werden. Test Workbench speichert diese Daten ausschließlich in Ihrem Amazon S3 S3-Bucket und nicht in der Lex Service-Infrastruktur. Aus diesem Grund benötigt Test Workbench Zugriff auf Ihren Amazon S3 S3-Bucket, um ordnungsgemäß zu funktionieren.

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. Richtlinie für das Lesen von CloudWatch Protokollen durch Test Workbench — Diese Richtlinie ermöglicht es Test Workbench, Testsätze aus Lex Conversation-Textprotokollen zu generieren, die in Amazon Logs gespeichert sind. CloudWatch Die folgende Richtlinie sollte geändert werden, um sie zu aktualisieren, *. Region AwsAccountId LogGroupName*

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TestWorkbenchLogsReadOnly",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:Region:AwsAccountId:log-group:LogGroupName:"
      ]
    }
  ]
}

```

4. Richtlinie für den Aufruf von Lex Runtime durch Test Workbench — Diese Richtlinie ermöglicht es Test Workbench, einen Testsatz gegen Lex-Bots auszuführen. Die folgende Richtlinie sollte geändert werden, um *Region*, *AwsAccountId* zu aktualisieren. *BotId* Da Test Workbench jeden Bot in Ihrer Lex-Umgebung testen kann, können Sie die Ressource durch „arn:aws:lex:*Region* ::bot-alias/**AwsAccountId*“ ersetzen, um Test Workbench Zugriff auf alle Amazon Lex V2-Bots in einem Konto zu gewähren.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TestWorkbenchLexRuntime",
      "Effect": "Allow",
      "Action": [
        "lex:RecognizeText",
        "lex:RecognizeUtterance",
        "lex:StartConversation"
      ],
      "Resource": [
        "arn:aws:lex:Region:AwsAccountId:bot-alias/BotId/*"
      ]
    }
  ]
}
```

5. (Optional) Richtlinie für Test Workbench zum Verschlüsseln und Entschlüsseln von Testsatzdaten — Wenn Test Workbench so konfiguriert ist, dass Testset-Eingaben und -Ergebnisse mithilfe eines vom Kunden verwalteten KMS-Schlüssels in Amazon S3 S3-Buckets gespeichert werden, benötigt Test Workbench sowohl Verschlüsselungs- als auch Entschlüsselungsberechtigungen für den KMS-Schlüssel. Die folgende Richtlinie sollte zur Aktualisierung geändert werden *Region*, und *KmsKeyId* wo *KmsKeyId* ist die ID des vom Kunden *AwsAccountId* verwalteten KMS-Schlüssels?

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TestWorkbenchKmsEncryption",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
    }
  ]
}
```

```

    "Resource": [
      "arn:aws:kms:Region:AwsAccountId:key/KmsKeyId"
    ],
    "Condition": {
      "StringLike": {
        "kms:ViaService": [
          "s3.*.amazonaws.com"
        ]
      }
    }
  }
}

```

6. (Optional) Richtlinie für Test Workbench zum Entschlüsseln von Audiodateien — Wenn Audiodateien mithilfe eines vom Kunden verwalteten KMS-Schlüssels im S3-Bucket gespeichert werden, benötigt Test Workbench die Entschlüsselungsberechtigung für die KMS-Schlüssel. Die folgende Richtlinie sollte zur Aktualisierung geändert werden *Region**AwsAccountId*, und *KmsKeyId* wo *KmsKeyId* ist die ID des vom Kunden verwalteten KMS-Schlüssels, der zum Verschlüsseln der Audiodateien verwendet wird.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TestWorkbenchKmsDecryption",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:Region:AwsAccountId:key/KmsKeyId"
      ],
      "Condition": {
        "StringLike": {
          "kms:ViaService": [
            "s3.*.amazonaws.com"
          ]
        }
      }
    }
  ]
}

```

```
}  
]  
}
```

Testsätze verwalten

Sie können Testsätze im Testset-Fenster herunterladen, aktualisieren und löschen. Oder Sie können die Liste der verfügbaren Testsätze verwenden, um Ihre Testsatzdatei zu bearbeiten oder manuell mit Anmerkungen zu versehen. Laden Sie es dann erneut hoch, um die Validierung aufgrund von Fehlern oder anderen Eingabeproblemen erneut zu versuchen.

So laden Sie die Testsatzdatei aus dem Testsatzdatensatz herunter:

1. Wählen Sie den Namen des Testsatzes aus der Liste der Testsätze aus.
2. Klicken Sie im Fenster mit den Testdatensätzen auf der rechten Seite des Bildschirms im Bereich Testeingaben auf die Schaltfläche Herunterladen.
3. Wenn oben im Fenster Details zu Validierungsfehlern bezüglich des Testsatzes angezeigt werden, klicken Sie auf die Schaltfläche Herunterladen. Die Datei wird in Ihrem Download-Ordner gespeichert. Sie können die Validierungsfehler im Testsatz anhand der Fehlermeldungen in der CSV-Datei des Testsatzes beheben. Suchen Sie nach dem im Validierungsschritt identifizierten Fehler, korrigieren Sie die Zeile oder entfernen Sie sie und laden Sie die Datei hoch, um den Validierungsschritt erneut zu versuchen.
4. Wenn Sie das Testset erfolgreich heruntergeladen haben, wird eine grüne Bannermeldung angezeigt.

So laden Sie ein Testset aus der Liste der Testsätze herunter:

1. Wählen Sie in der Liste der Testsätze das Optionsfeld neben dem Testset-Element aus, das Sie herunterladen möchten.
2. Wählen Sie im Aktionsmenü oben rechts die Option Herunterladen aus.
3. Eine grüne Bannernachricht zeigt an, ob Sie das Testset erfolgreich heruntergeladen haben. Die Datei wird in Ihrem Download-Ordner gespeichert.

Von Test Workbench unterstützte Testset-Spalten

Im Folgenden finden Sie die vollständige Liste der von Test Workbench unterstützten Testset-Spalten sowie Anweisungen zu deren Verwendung mit Amazon Lex V2.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Line Number	Text und Audio	Anzahl	Nein	Dies ist eine Benutzerspalte, die von ignoriert wird. Es soll einem Testsatzautor helfen, die Testsatzzeilen zu sortieren und zu filtern. „Zeile #“ kann als alternativer Spaltenname verwendet werden.
Nummer der Konversation	Text und Audio	Anzahl	Nein	In dieser Spalte können Sie Zeilen in einer Konversation zusammenfügen. „Konversation #“ kann als alternativer Spaltenname verwendet werden.
Quelle	Text und Audio	Enum („Benutzer“, „Agent“)	Nein	Der Wert in dieser Spalte gibt an, ob die

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				Zeile für einen Benutzer oder einen Agenten bestimmt ist. „Konversationsteilnehmer“ kann als alternativer Spaltenname verwendet werden.
Eingabe	Text	String	Nein	Diese Spalte wird verwendet, um das Testset „Transkript für Text“ hinzuzufügen. • Die Texteingabe sollte in Benutzerzeilen verwendet werden. • Die Agenteneingabeaufforderung sollte in Agentenzeilen verwendet werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Erwartete Transkription	Audio	String	Nein	Diese Spalte wird verwendet , um das Transkript für den Audiotest satz hinzuzufügen. • Die erwartete Transkription der Audiodate i sollte in Benutzerzeilen mit Audioeingabe verwendet werden. • Die DTMF-Eingabe kann in Benutzerzeilen mit DTMF-Eingabe verwendet werden. • Die Agenten-Eingabeaufforderung sollte in Agentenzeilen verwendet werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
S3-Audios tandort	Audio	String	Nein	Diese Spalte wird verwendet , um den Speicherort der Audiodate i hinzuzufügen und gilt nur für Audio-Testsets. • Der S3-Pfad sollte in den Benutzerzeilen mit der Audioeingabe verwendet werden. • Dieses Feld sollte in Benutzerzeilen mit DTMF-Eingabe leer gelassen werden. • Dieses Feld sollte in den Agentenzeilen leer gelassen werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Kontext-Tag eingeben	Text und Audio	String	Ja	Diese Spalte wird verwendet, um den Namen eines Eingabekontextes anzugeben, der bei der Eingabe für Lex verwendet wird, während die Zeile im Testsatz ausgeführt wird. • Dies bezieht sich auf den Eingabekontext in Setting Intent Context for your Lex V2 Bot. • Beachten Sie, dass Test Workbench nur den Namen des Kontextes unterstützt. Die Parameter in einem Kontext werden nicht unterstützt.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				• Es können mehrere Spalten mit Namen wie „Eingabekontext-Tag 1“, „Eingabekontext-Tag 2“ usw. verwendet werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Attribut anfordern	Text und Audio	String	Ja	Diese Spalte wird verwendet, um ein Anforderungsattribut bereitzustellen, das als Eingabe für Lex verwendet wird, während die Zeile im Testsatz ausgeführt wird. - Der Wert in einer Spalte sollte im Format <code><request-attribute-name> = <request-attribute-value></code> angegeben werden. Aus Gründen der besseren Lesbarkeit können Leerzeichen um '=' herum hinzugefügt werden. - Beispiele:

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				• request-attribute-foo = Dies ist eine Scheinantwort • request-attribute-foo = 'das ist eine „Scheinantwort“' • request-attribute-foo = „Das ist eine 'Scheinantwort'“ • Es können mehrere Spalten mit Namen wie „Anforderungsattribut 1“, „Anforderungsattribut 2“ usw. verwendet werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Sitzungsattribut	Text und Audio	String	Ja	Diese Spalte wird verwendet, um ein Sitzungsattribut bereitzustellen, das als Eingabe für Lex verwendet wird, während die Zeile im Testsatz ausgeführt wird. - Der Wert in einer Spalte sollte im Format <code><session-attribute-name> = <session-attribute-value></code> angegeben werden. Aus Gründen der besseren Lesbarkeit können Leerzeichen um '=' herum hinzugefügt werden. - Beispiele: <code>session-attribute-foo =</code>

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				Dies ist eine Scheinantwort • session-attribute-foo = 'das ist eine „Scheinantwort“' • session-attribute-foo = „Das ist eine 'Scheinantwort'“ • Es können mehrere Spalten mit Namen wie „Sitzungsattribut 1“, „Sitzungsattribut 2“ usw. verwendet werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
RunTime Hinweis	Text und Audio	String	Ja	Diese Spalte wird verwendet , um einen Runtime-Hinweis für einen Slot innerhalb einer Absicht bereitzustellen, der als Eingabe für Lex verwendet wird, während die Zeile im Testsatz ausgeführt wird. Im Folgenden finden Sie Beispiele: • Der Wert in einer Spalte sollte im Format <code>`angegeben werden<intent-name>.<slot -name><slot-value>=`</code>. Aus Gründen der Lesbarkeit können Leerzeichen um <code>'='</code> hinzugefügt werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				- Beispiele: - IntentNameFoo. slotNameFoo = ein Dummy-Wert - IntentNameFoo. slotNameFoo = 'a „Dummy-Wert“ - IntentNameFoo. slotNameFoo = „ein 'Dummy-Wert“ - Test Workbench unterstützt keine zusammengesetzten Steckplätze für Laufzeithinweise. - Es können mehrere Spalten mit Namen wie

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				„RunTime Hinweis 1“, „RunTime Hinweis 2“ usw. verwendet werden.
Hereinplatzen	Audio	Boolesch	Nein	In dieser Spalte wird angegeben , ob Test Workbench beim Senden einer Audiodatei an Lex Runtime für die Zeile im Testsatz einspringen soll. • Gilt nur für Audio-Testsätze für die Streaming-API. • Diese Spalte wird ignoriert , wenn ein Testsatz im Nicht-Streaming-API-Modus ausgeführt wird.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Erwartete Ausgabeabsicht	Text und Audio	String	Nein	Diese Spalte wird verwendet, um den Namen einer Absicht anzugeben, die in der Ausgabe von Lex für die Zeile im Testsatz erwartet wird.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Erwarteter Ausgabeslot	Text und Audio	String	Ja	Diese Spalte wird verwendet , um einen Slot-Wert bereitzustellen, der in der Ausgabe von Lex erwartet wird, während die Zeile im Testsatz ausgeführt wird. • <code>< slot-value></code>Der Wert in einer Spalte sollte im Format <code>`<slot-name>='`</code> angegeben werden. Aus Gründen der besseren Lesbarkeit können Leerzeichen um '=' herum hinzugefügt werden. • Beispiele für Steckplätze, bei denen es sich weder um Steckplätze

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				mit mehreren Werten noch um zusammengesetzte Steckplätze handelt: • slotNameFoo = ein Dummy-Wert • slotNameFoo = 'a ein „Dummy-Wert“' • slotNameFoo = „ein 'Dummy-Wert““ • Beispiele für Slots mit mehreren Werten: • slotNameFoo = Wert1, Wert2 • slotNameFoo = Wert1, „Artikel von Foo“

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				• slotNameF oo = Wert1, 'Wert2' • Beispiele für zusammengesetzte Steckplätze, bei denen der Steckplatzname 'Car' und der Name des Untersteckplatzes 'Make' ist: • Automarke = Toyota • Automarke = „Toyota“ • Automarke = 'Toyota' • Es können mehrere Spalten mit Namen wie „Erwarteter Ausgangsteckplatz 1“, „Erwarteter Ausgangsteckplatz 2“ usw.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				verwendet werden.

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
Erwartetes Ausgabe-Kontext-Tag	Text und Audio	String	Ja	Diese Spalte wird verwendet, um den Namen eines Ausgabe-Kontextes anzugeben, der in der Ausgabe von Lex für die Zeile im Testsatz erwartet wird. • Dies bezieht sich auf den Ausgabe-Kontext in Setting Intent Context für Ihren Lex V2-Bot. • Beachten Sie, dass Test Workbench nur den Namen des Kontextes und noch keine Parameter innerhalb des Kontextes unterstützt. • Es können mehrere Spalten mit

Spaltenname	Typ des Testsatzes	Wert-Typ	Support mehrerer Spalten	Beschreibung
				Namen wie „Expected Output Context Tag 1“, „Expected Output Context Tag 2“ usw. verwendet werden.

Fehler bei der Testvalidierung in der Test Workbench anzeigen

Sie können Testsätze korrigieren, die Validierungsfehler melden. Diese Validierungsfehler werden generiert, wenn ein Testsatz noch nicht zum Testen bereit ist. Die Test Workbench kann Ihnen zeigen, welche erforderlichen Spalten in der CSV-Eingabedatei für das Testset keinen Wert im erwarteten Format hatten.

So zeigen Sie Fehler bei der Testvalidierung an:

1. Wählen Sie aus der Liste der Testsätze den Namen des Testsatzes aus, der einen Fehler mit dem Status eines Validierungsfehlers meldet, den Sie anzeigen möchten. Die Namen der Testsätze sind aktive Links, über die Sie zu Details zum Testsatz gelangen.
2. Im Testsatzdatensatz werden oben auf dem Bildschirm Details zu den Validierungsfehlern angezeigt. Wählen Sie „Details anzeigen“, um den Bericht über Validierungsfehler anzuzeigen.
3. Überprüfen Sie im Fenster mit dem Fehlerbericht die Zeilennummer und den Fehlertyp, um zu sehen, wo der Fehler aufgetreten ist. Für eine lange Liste von Fehlern können Sie wählen, ob Sie den Fehlerbericht herunterladen möchten.
4. Vergleichen Sie die in Ihrer CSV-Eingabedatei für das Testset aufgelisteten Fehler mit Ihrer ursprünglichen Testdatei, um etwaige Probleme zu beheben, und laden Sie den Testsatz erneut hoch.

In der folgenden Tabelle sind die Eingabe-CSV-Validierungsfehlermeldungen mit Szenarien aufgeführt.

Szenario	Fehlermeldung	Hinweise
Die Dateigröße des Testsatzes überschreitet	Die Dateigröße des Testsets ist größer als 200 MB. Geben Sie eine kleinere Datei an und versuchen Sie es erneut mit Ihrer Anfrage.	
Der Testsatz überschreitet die maximale Anzahl von Datensätzen	Die Eingabedatei hatte mehr als die unterstützte maximale Anzahl von 200.000.	
Leeren Testsatz hochladen	Der importierte Testsatz ist leer. Geben Sie einen nicht leeren Testsatz an und versuchen Sie es erneut mit Ihrer Anfrage.	
Leerer Name der Spaltenüberschrift	Zeile mit Spaltenüberschriften: In Spalte Nummer 5 wurde ein leerer Spaltenname gefunden.	
Unbekannter Name der Spaltenüberschrift	Zeile mit den Spaltenüberschriften: Der Spaltenname 'Dummy' in Spalte Nummer 2 konnte nicht erkannt werden.	
Doppelter Name der Spaltenüberschrift	Zeile mit den Spaltenüberschriften: Es wurden mehrere Spalten „S3-Audio link“ und „S3-Audiolink“ gefunden, die identisch oder gleichwertig sind. Entferne eine dieser Spalten oder benenne sie um.	

Szenario	Fehlermeldung	Hinweise
Der Spaltenname mit mehreren Werten hat das Limit überschritten	Zeile mit Spaltenüberschrift en: Die Anzahl der Spalten für „Erwarteter Ausgabeslot“ hat die maximal unterstützte Anzahl überschritten: 6. Entfernen Sie einige Spalten für „Erwarteter Ausgabeslot“ und versuchen Sie es erneut.	Die maximale Anzahl unterstützter Spalten für Spalten mit mehreren Werten ist 6.
Die Spaltenüberschrift im Zusammenhang mit Text oder Audio ist nicht vorhanden	Es konnten keine Spalten für Text- oder Audiokonversationen gefunden werden. Verwenden Sie für Textkonversationen die Spalten {'Texteingabe'}. Verwenden Sie für Audiokonversationen die Spalten {'S3-Audiolink', 'Erwartete Transkription'}.	Obligatorische Audiospalten: {'S3-Audiolink', 'Erwartete Transkription'} Obligatorische Textspalten: {'Texteingabe'}
Es gibt sowohl Text- als auch Audiospaltenüberschriften	Es wurden Spalten für Text- und Audiokonversationen gefunden. Sie können entweder die Spalten {'Texteingabe'} für Textkonversationen oder die Spalten {'S3-Audiolink', 'Erwartete Transkription'} für Audiokonversationen verwenden.	Obligatorische Audiospalten: {'S3-Audiolink', 'Erwartete Transkription'} Obligatorische Textspalten: {'Texteingabe'}
Die obligatorische Spalte fehlt	Die obligatorischen Spalten ["Expected Output Intent"] konnten nicht gefunden werden.	Obligatorische Spalten: {"Zeile #", „Quelle“, „Erwartete Ausgabeabsicht“}

Szenario	Fehlermeldung	Hinweise
Es wurden Daten in einer Spalte ohne Überschrift gefunden	Es wurden Daten in Spalte 8 für Zeile 6 gefunden, aber die entsprechende Spalte hatte keine Spaltenüberschrift.	
Für obligatorische Spalten wurden keine Daten gefunden	Zeile=12: Für die obligatorischen Spalten wurden keine Werte gefunden: {"Source", „Expected Output Intent"}	
Doppelte Konversations-ID gefunden	Die Konversationsnummer '19' wurde für die vorherige Konversation in Zeile 39 angezeigt.“ Stellen Sie sicher, dass dieselbe Konversationsnummer nicht für zwei Konversationen angegeben wurde. Sie können dies tun, indem Sie sicherstellen, dass alle Zeilen für eine Konversationsnummer gruppiert sind.	
Ungültige Konversations-ID angegeben	In der Spalte „Konversation #" wurde ein ungültiger Wert 'test_conversation' gefunden. Der Wert für diese Spalte muss entweder numerisch oder N/A (d. h. nicht zutreffend) für eine Benutzerzeile sein.	
Für die Zeilennummer wurde ein nicht numerischer Wert angegeben	Der nicht numerische Wert 'test_line' wurde in der Spalte 'Zeile #' gefunden. Sein Wert muss numerisch sein.	

Szenario	Fehlermeldung	Hinweise
Die Konversation wurde in der Agentenzeile nicht gefunden	Für die Spalte „Konversation #“ wurde kein Wert gefunden. Er muss für eine Agentenzeile angegeben werden.	
In der Agentenzeile wurde eine nicht numerische Konversations-ID gefunden	Der nicht numerische Wert 'test_conversation' wurde in der Spalte 'Konversation #' gefunden. Sein Wert muss für eine Agentenzeile numerisch sein.	
Ungültiger S3-Standort	Ungültiger Wert 'Bucket/Ordner' wurde angegeben. Das gültige Format ist S3://<bucketName>/<keyName>.	
Ungültiger S3-Bucket-Name	Es wurde ein ungültiger S3-Bucket-Name 'test_bucket' angegeben. Überprüfen Sie den Bucket-Namen.	
Der S3-Audiospeicherort ist der Ordner	Der angegebene Audiospeicherort 'S3: //bucket/folder' ist ungültig. Er verweist auf einen S3-Ordner.	
Ungültiger Name der Absicht	In der Absicht 'intent @name' waren ungültige Zeichen enthalten. Überprüfen Sie den Namen der Absicht.	Regex-Prüfung: ^ ([0-9a-Za-Z] [_-]?) +\$
Ungültiger Slot-Name	Im Steckplatz 'Slot @Name' waren ungültige Zeichen vorhanden. Überprüfen Sie den Steckplatznamen.	Regex: ^ ([0-9a-Za-Z] [_-]?) +\$ Es sollte nicht mit einem Punkt (.) beginnen oder enden

Szenario	Fehlermeldung	Hinweise
Der Slot-Wert wurde für den übergeordneten Steckplatz angegeben	Die Slot-Werte wurden sowohl für den Unterslot „Address.City“ als auch für den übergeordneten Slot „Address“ bereitgestellt. Werte sollten nur für den Unterslot angegeben werden.	Der übergeordnete Steckplatz in CST sollte keinen Slot-Wert haben
Ungültiges Zeichen im Kontextnamen	Im Kontextnamen 'context @1 'waren ungültige Zeichen vorhanden. Überprüfen Sie den Kontextnamen.	Regex: ^ ([a-zA-Z] _?) +\$
Ungültiger Schreibstil für Spielautomaten	Es wurde ein ungültiger Wert 'test' angegeben. Stellen Sie sicher, dass sie alle in Großbuchstaben geschrieben sind. Gültige Werte sind ["Standard", "SpellByLetter", "SpellByWord"].	Unterstützte Werte ["Standard", "SpellByLetter", "SpellByWord"]
Der Teilnehmer oder die Quelle muss entweder ein Agent oder ein Benutzer sein	Es wurde ein ungültiger Wert 'Bot' angegeben. Gültige Werte sind ["Agent", "User"].	Unterstützte Aufzählungen: „Agent“, „User“
Die Zeilennummer sollte nicht dezimal sein	Es wurde ein ungültiger Wert '10.1' angegeben. Es sollte eine gültige Zahl ohne Brüche sein.	
Die Gesprächsnummer sollte keine Dezimalzahl sein	Es wurde ein ungültiger Wert '10.1' angegeben. Es sollte eine gültige Zahl ohne Brüche sein.	

Szenario	Fehlermeldung	Hinweise
Die Zeilennummer sollte innerhalb des zulässigen Bereichs liegen	Es wurde ein ungültiger Wert '92233720368547758071' angegeben. Er sollte größer oder gleich 1 und kleiner oder gleich 9223372036854775807 sein.	
Die Barge-In-Spalte akzeptiert nur boolesche Werte	Der ungültige Wert 'test' wurde angegeben. Es sollte ein gültiger boolescher Wert wie 'wahr' oder 'falsch' sein. Alternativ können „Ja“ und „Nein“ verwendet werden.	Mögliche Werte: "True“, „true“, „T“, „Ja“, „ja“, „Y“, „1“, „1.0“, „False“, „falsch“, „F“, „Nein“, „nein“, „N“, „0“, „0.0"
Erwarteter Slot, Sitzungsattribut und Anforderungsattribut sollten durch Gleichheit (=) getrennt werden	Der Wert 'slotName:slotValue' hat kein '='. <key><value>Dieser Wert sollte als Schlüssel-Wert-Paar im Format '=' bereitgestellt werden.	Zum Beispiel: slotName = SlotType
Der erwartete Slot, das Sitzungsattribut und das Anforderungsattribut sollten ein Schlüssel-Wert-Paar enthalten	'=SlotValue' hat keinen Schlüssel vor '='. <key><value>Ein solcher Wert sollte als Schlüssel-Wert-Paar im Format '=' bereitgestellt werden.	Zum Beispiel: slotName = SlotType
Ungültiges Anführungszeichen am Ende	In „Foo's item“ wurde ein falsches Zitat gefunden.“ Es beginnt mit dem Anführungszeichen `“`, endet aber nicht mit demselben Anführungszeichen.	Zum Beispiel: `„Foo's item“, KFC`

Szenario	Fehlermeldung	Hinweise
Ungültiges Zitat in der Mitte	Falsches Zitat in <code>""Foo's"</code> Burger usw. gefunden. Es enthält das Anführungszeichen <code>""</code> in seinem Inhalt. Werte, die einfache Anführungszeichen enthalten, sollten in doppelte Anführungszeichen gesetzt werden und umgekehrt.	Richtig Zum Beispiel: <code>`,Foo's item`, KFC`</code>
Erforderliche Anführungszeichen	<code>`key = Foo's item`</code> enthält einfache oder doppelte Anführungszeichen, wurde aber nicht in Anführungszeichen gesetzt. Werte, die einfache Anführungszeichen enthalten, sollten in doppelte Anführungszeichen gesetzt werden und umgekehrt.	
Doppelter Schlüssel wird in der Spalte wiederholt	Der Schlüssel <code>`key1`</code> wurde in zwei Spalten wiederholt: <code>`Session-Attribut 3`</code> und <code>`Session-Attribut 1`</code> .	
Ungültiges Format im Runtime-Hinweis	Ungültiger Schlüssel <code>`BookFlight.Car.`,</code> wurde für Runtime-Hinweise bereitgestellt. Für Runtime-Hinweise sollte der Schlüssel das Format haben <code><intentName>.<slotName></code> .	Falls <code>.'</code> in der Mitte des Schlüssels stehen muss, können Absichtsname und Steckplatzname nicht aus einem solchen Schlüssel extrahiert werden. Beispiele für eine solche falsche Formatierung: <code>"BookFlight,,", "BookFlight.Auto", "BookFlight.Auto."</code>

Szenario	Fehlermeldung	Hinweise
Ungültiger Absichtsname im Runtime-Hinweisschlüssel	Die ungültige Absicht `intent @name` für Runtime-Hinweise wurde gefunden. Überprüfen Sie den Namen der Absicht.	Regex-Prüfung: <code>^ ([0-9a-Za-Z] [_-]?) +\$</code>
Ungültiger Steckplatzname im Runtime-Hinweisschlüssel	In `Slot @Name` wurde ein ungültiger Steckplatzname für Runtime-Hinweise gefunden. Überprüfen Sie den Steckplatznamen.	Regex: <code>^ ([0-9a-Za-Z] [_-]?) +\$</code> Es sollte nicht mit einem Punkt (.) beginnen oder enden

Löschen Sie einen Testsatz in Test Workbench

Sie können einen Testsatz ganz einfach aus Ihrer Testsatzliste löschen.

Um ein Testset zu löschen:

1. Gehen Sie im Menü auf der linken Seite zur Liste der Testsätze, um die Liste der Testsätze zu sehen.
2. Wählen Sie aus der Liste der Testsätze den Testsatz aus, den Sie löschen möchten.
3. Gehen Sie oben rechts zum Drop-down-Menü Aktionen und wählen Sie Löschen.
4. Eine Meldung bestätigt, dass der Testsatz gelöscht wurde.

Bearbeiten Sie die Details des Testsatzes

Sie können den Namen und die Details eines Testsets in der Liste der Testsätze bearbeiten. Der Name oder die Details können später hinzugefügt oder aktualisiert werden. Sie müssen jedoch Ihr Testset aktualisieren, bevor Sie den Test mit Ihren Bot- oder Transkriptionsdaten ausführen können.

So bearbeiten Sie die Testset-Details:

1. Gehen Sie im Menü auf der linken Seite zur Liste der Testsätze, um die Liste der Testsätze zu sehen.
2. Wählen Sie in der Liste der Testsätze das Kontrollkästchen für den Testsatz aus, den Sie bearbeiten möchten.

3. Gehen Sie oben rechts zum Dropdownmenü Aktionen und wählen Sie Details bearbeiten aus.
4. Eine Meldung bestätigt, dass der Testsatz erfolgreich bearbeitet wurde.

Testsatz aktualisieren

Sie können Elemente aus dem Testsatz aktualisieren, korrigieren, ändern oder löschen, um Ihre Ausgangsergebnisse zu optimieren oder um andere Fehler zu korrigieren, die möglicherweise im Testsatz aufgetreten sind

Sie können einen Testsatz herunterladen und die Validierungsfehler beheben, bevor Sie den korrigierten Testsatz hochladen. Weitere Informationen finden Sie unter [Fehler bei der Testvalidierung anzeigen](#).

Um ein Testset zu aktualisieren:

1. Wählen Sie im Testsatzdatensatz oben rechts die Schaltfläche Testsatz aktualisieren aus.
2. Wählen Sie eine Datei aus, die Sie von Ihrem Amazon S3 S3-Konto hochladen möchten, oder laden Sie eine CSV-Testdatei von Ihrem Computer hoch. HINWEIS: Durch die Aktualisierung eines Testsatzes werden die vorhandenen Daten überschrieben.
3. Wählen Sie die Schaltfläche Aktualisieren.
4. Eine Meldung bestätigt, dass der Testsatz erfolgreich aktualisiert wurde. HINWEIS: Dieser Vorgang kann je nach Komplexität und Größe des Testsatzes einige Minuten dauern.
5. Eine Meldung bestätigt, dass der Testsatz erfolgreich aktualisiert wurde, und der Status zeigt Bereit zum Testen an.

Führen Sie einen Test aus

Um einen Testsatz auszuführen, müssen Sie den entsprechenden Bot auswählen, um den Test mit dem Testsatz auszuführen. Sie können einen Bot aus Ihrem AWS Konto aus dem Drop-down-Menü unter Testset auswählen. Bei diesem Vorgang wird Ihr ausgewählter Bot anhand Ihrer validierten Testdaten getestet, um Leistungskennzahlen anhand der Basisdaten aus dem Testset zu erstellen.

Execute a test [Info](#)

Evaluate the performance of a bot by running it against a test set.

If you are running a test set against a bot alias for the first time, validate its coverage to ensure good test coverage.

Settings

Test set

The test set you want to use to execute this test.

demoTestSet ▼

Bot

The bot you want to use to execute this test.

travelBot ▼

Bot alias

The bot alias you want to use to execute this test.

Default_alias ▼

Language

The bot language you want to use to execute this test.

English (US) ▼

Modality

Define if this test will be text-based or transcribed from audio.

☒ Text

☐ Audio

Endpoint selection

Define whether or not this test will use streaming endpoints.

 Use streaming for test sets with wait&continue

☒ Streaming

☐ Non-streaming

Cancel

Validate coverage

Execute

Um einen Test in der Test Workbench auszuführen

1. Wählen Sie auf der Seite mit den Testdatensätzen die Option Test ausführen aus.
2. Wählen Sie den Testsatz aus, den Sie im Test verwenden möchten.
3. Wählen Sie im Dropdownmenü Bot den Namen des Bots aus, der im Test verwendet werden soll.
4. Wählen Sie gegebenenfalls einen Bot-Alias aus dem Drop-down-Menü Bot-Alias aus.

5. Wählen Sie aus der Sprachauswahl eine Version von Englisch aus.
6. Wählen Sie als Modalitätstyp Text oder Audio aus.
7. Wählen Sie Ihren Amazon S3 S3-Standort. (nur Audio)
8. Wählen Sie Ihre Endpunkt-Auswahl für Ihren Bot. (nur Streaming)
9. Klicken Sie auf die Schaltfläche Abdeckung überprüfen, um zu bestätigen, dass Ihr Test startbereit ist. Wenn im Validierungsschritt Fehler auftreten, überprüfen Sie die vorherigen Parameter und nehmen Sie Korrekturen vor.
10. Wählen Sie Ausführen aus, um den Test auszuführen.
11. Eine Meldung bestätigt, dass der Test erfolgreich ausgeführt wurde.

Abdeckung der Testsätze in Test Workbench

Eine begrenzte Abdeckung von Absichten und Zeitfenstern zwischen dem Testset und dem Bot kann zu erwarteten Leistungskennzahlen führen. Wir empfehlen Ihnen, die Abdeckung des Testsets vor der Ausführung des Tests zu überprüfen.

Test set discrepancies

Download X

Limited coverage of intents and slots between the test set and bot alias will result in a test result with low coverage.

Intents and slots that are found in the test set but not in the bot alias are displayed here.

Intents

Slots

Intent discrepancies (30)

< 1 2 3 4 > ⚙

Intent name	Discrepancy
BookTrip	Found in demoTestSet, but not in Default_alias
BookCruise	Found in demoTestSet, but not in Default_alias
BookCar	Found in demoTestSet, but not in Default_alias
CardServices	Found in demoTestSet, but not in Default_alias
BookPlane	Found in demoTestSet, but not in Default_alias

Um den Umfang der Validierung zu überprüfen

1. Wählen Sie in den Datensätzen des Testsatzes die Schaltfläche „Abdeckung überprüfen“.
2. Die Meldung weist darauf hin, dass die Abdeckung zwischen dem Testset und dem ausgewählten Bot überprüft wird.
3. Sobald der Vorgang abgeschlossen ist, wird in der Meldung angezeigt, dass die Überprüfung der Abdeckung erfolgreich war.
4. Wählen Sie unten im Fenster die Schaltfläche „Details anzeigen“.
5. Sie können sich die Abweichungen der Testsätze in Bezug auf Absichten und Slots anzeigen lassen, indem Sie jeweils die entsprechende Registerkarte auswählen. Sie können diese Daten in ein CSV-Format herunterladen, indem Sie auf die Schaltfläche Herunterladen klicken.
6. Überprüfen Sie die Validierungsergebnisse für Ihre Testsatzdaten, Bot-Absichten und Slots. Identifizieren Sie Probleme und nehmen Sie Änderungen an der Architektur Ihres Bot-Testsets vor, um die Ergebnisse zu verbessern. Laden Sie das bearbeitete Testset und den Bot hoch,

um den Test auszuführen, sobald Sie Änderungen an der CSV-Datei vorgenommen haben.
HINWEIS: Die Überprüfung bezieht sich auf das Testset und nicht auf den Bot. Absichten, die zwar im Bot enthalten sind, aber nicht im Testset enthalten sind, werden nicht abgedeckt.

Testergebnisse anzeigen

Interpretieren Sie die Testergebnisse aus der Test Workbench, um festzustellen, wo die Konversation zwischen Ihrem Bot und dem Kunden möglicherweise fehlschlägt oder dass der Kunde mehrere Versuche unternehmen muss, um die Absicht zu erfüllen.

Indem Sie diese Probleme in Ihren Testergebnissen lokalisieren, können Sie die Leistung Ihres Bots optimieren, indem Sie die Leistung Ihrer Absichten mithilfe verschiedener Trainingsdaten oder Äußerungen verbessern, die besser mit den Echtzeit-Transkriptionswerten des Bots übereinstimmen.

Sie können sich einen detaillierten Überblick über die Absichten und Slots verschaffen, bei denen es zu Leistungsunterschieden kam. Sobald Sie Intents oder Slots mit Diskrepanzen identifiziert haben, können Sie die Äußerungen und den Gesprächsverlauf genauer untersuchen und überprüfen.

Test results (10) Info								Delete	Download
Test runs evaluate a test set against a selected bot alias									
Find test results IDs, bot names				Any status ▾	Any type ▾	< 1 >			
☐	Test result ID	Created time ▾	Status	Test set ▾	Bot name ▾	Language ▾	Test type		
☐	1234567890abcdef0	March 30, 2022 08:55:15 PST	✓ Complete	demoTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef1	March 29, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Text		
☐	1234567890abcdef2	March 28, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef3	March 27, 2022 08:55:15 PST	✗ Error	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef4	March 26, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef5	March 25, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef6	March 24, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef7	March 23, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef8	March 22, 2022 08:55:15 PST	✓ Complete	goodTestSet	travelBot	English (US)	Audio		
☐	1234567890abcdef9	March 21, 2022 08:55:15 PST	⏸ Stopped	goodTestSet	travelBot	English (US)	Audio		

So überprüfen Sie die Testergebnisse:

1. Gehen Sie im Menü auf der linken Seite zur Liste der Testsätze und wählen Sie unter Test Workbench die Option Testergebnisse aus. HINWEIS: Die Testergebnisse zeigen den Status „abgeschlossen“ an, wenn sie erfolgreich waren.
2. Wählen Sie die Testergebnis-ID für die Testergebnisse aus, die Sie überprüfen möchten.

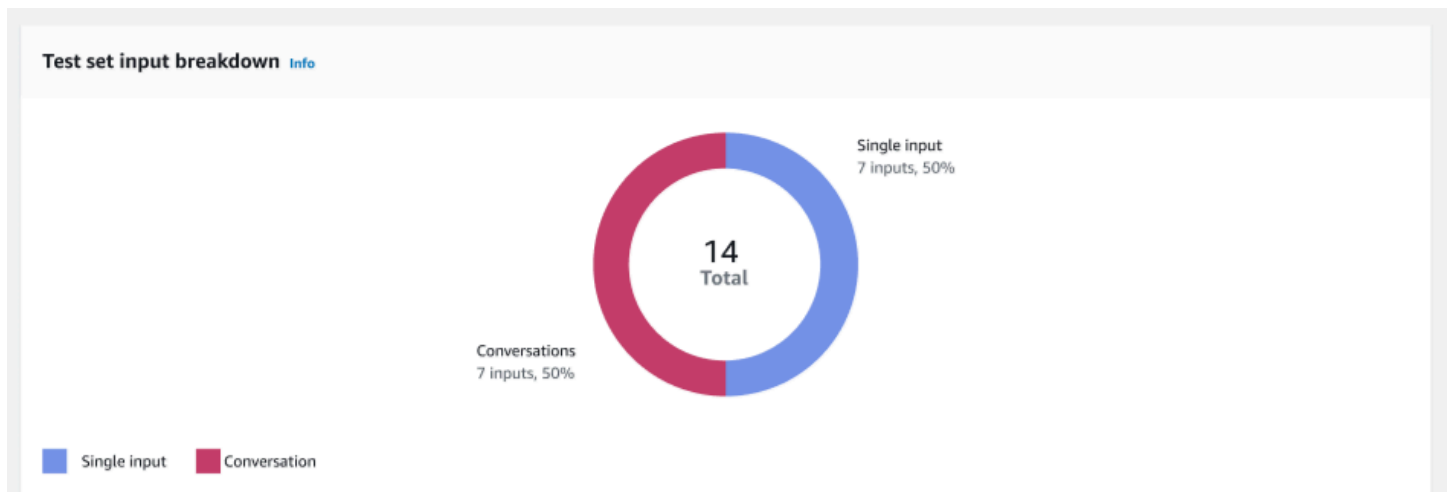
Details zu den Testergebnissen in Test Workbench

Die Testergebnisse zeigen die Details des Testsatzes, die verwendeten Absichten und die verwendeten Steckplätze. Es enthält auch die gesamte Aufschlüsselung der Testset-Eingaben, einschließlich der Gesamtergebnisse, der Konversationsergebnisse, der Absicht und der Slot-Ergebnisse.

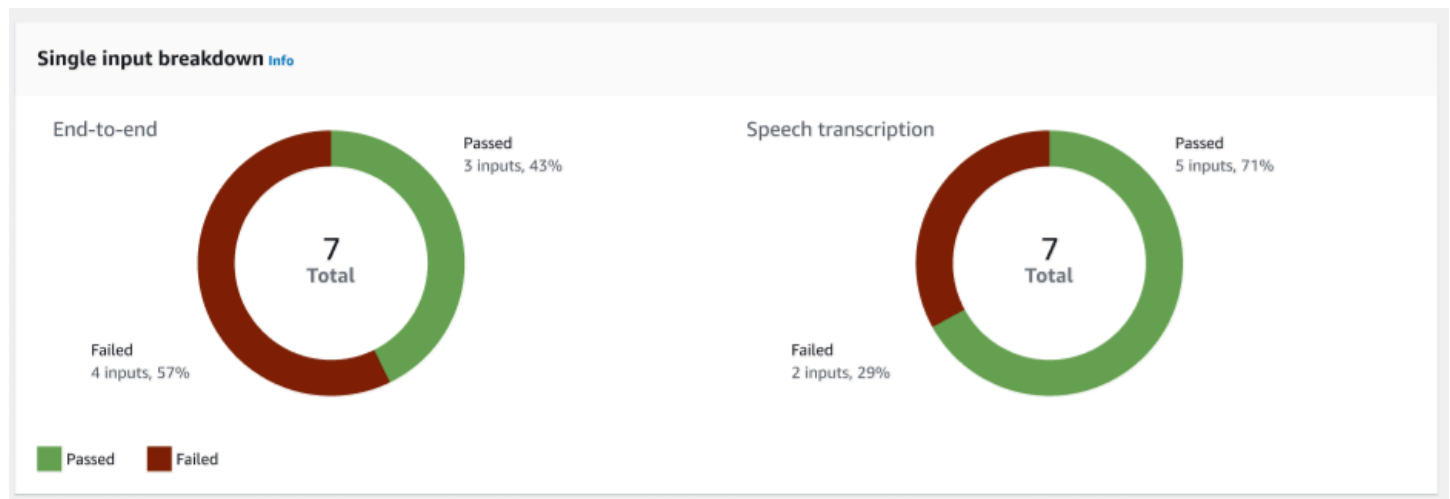
Die Testergebnisse umfassen alle testbezogenen Informationen wie:

- Metadaten der Testdetails
- Ergebnisse insgesamt
- Ergebnisse der Konversation
- Absicht und Slot-Ergebnisse
- Detaillierte Ergebnisse

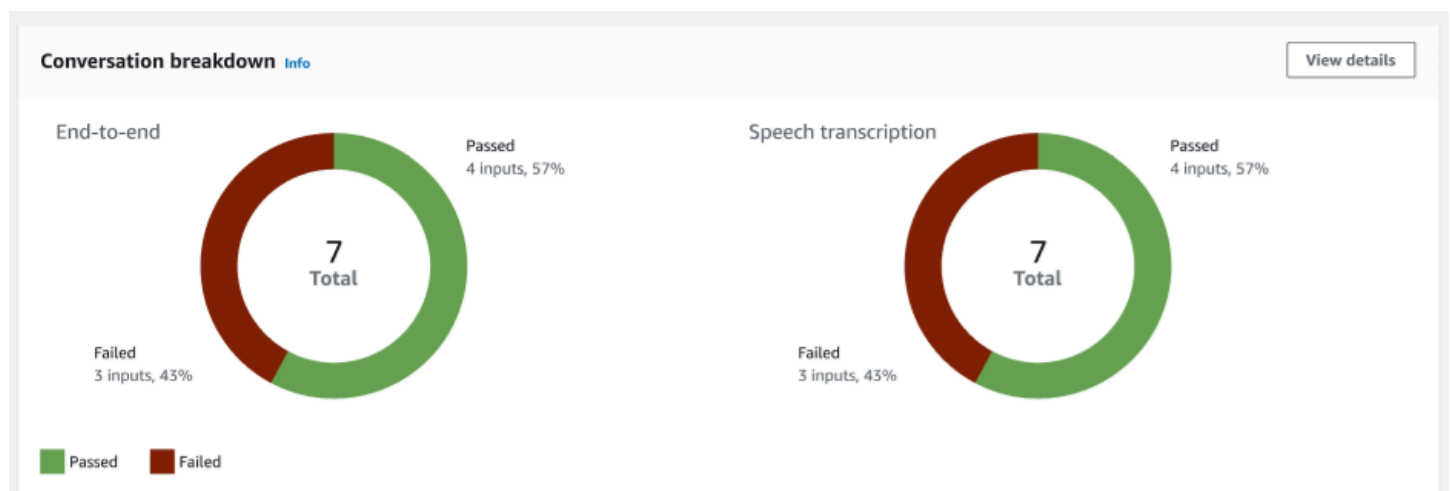
Registerkarte „Gesamtergebnisse“:



Aufschlüsselung der Testset-Eingaben — Dieses Diagramm zeigt die Aufschlüsselung der Anzahl der Konversationen und der einzelnen Eingabeäußerungen im Testset.



Aufschlüsselung nach einzelnen Eingaben — Zeigt zwei Diagramme an, die end-to-end Konversationen und Sprachtranskriptionen enthalten. Die Anzahl der erfolgreichen und fehlgeschlagenen Eingaben ist in jeder Tabelle angegeben. Hinweis: Die Sprachtranskriptionstabelle ist nur für das Audiotest-Set sichtbar.



Aufschlüsselung der Konversation — Zeigt zwei Diagramme an, die end-to-end Konversationen und Sprachtranskriptionen enthalten. Die Anzahl der erfolgreichen und fehlgeschlagenen Eingaben wird in jedem Diagramm angegeben. Hinweis: Die Sprachtranskriptionstabelle ist nur für das Audiotest-Set sichtbar.

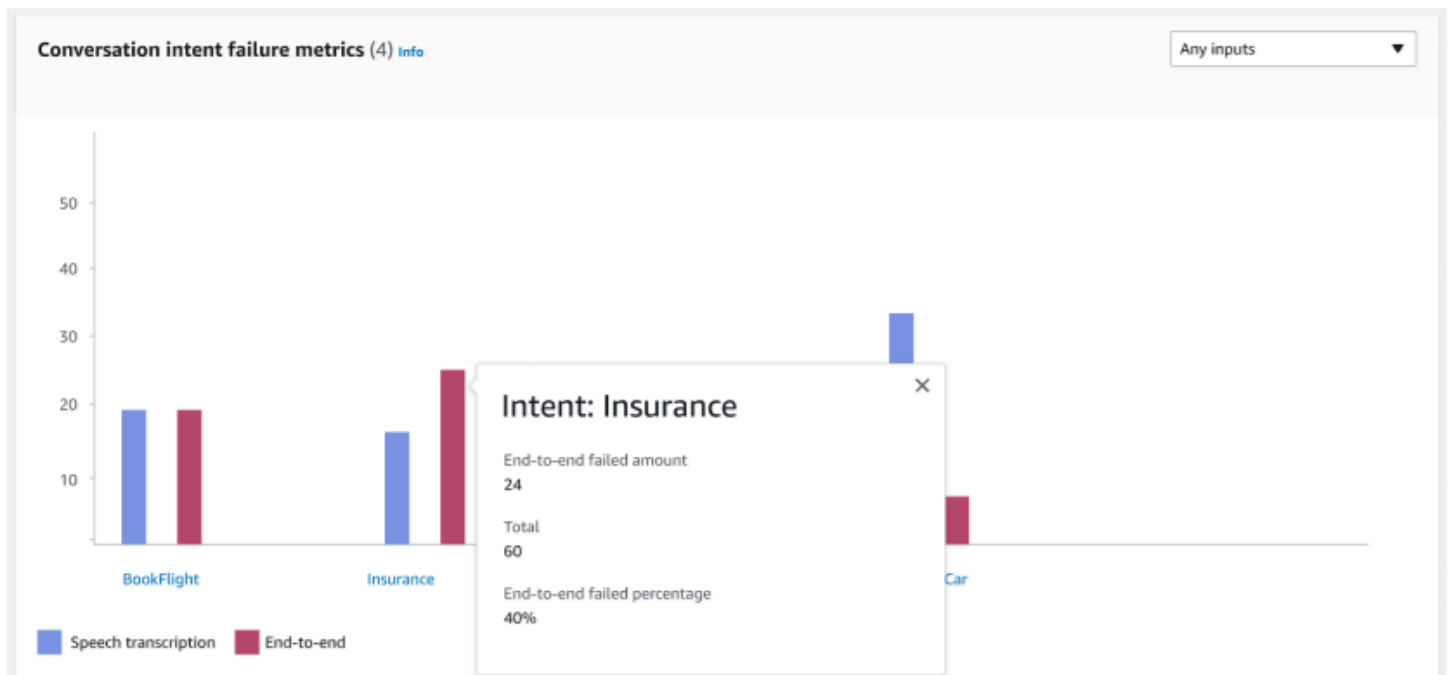
Registerkarte mit Konversationsergebnissen:

Conversation pass rates (5) [Info](#)

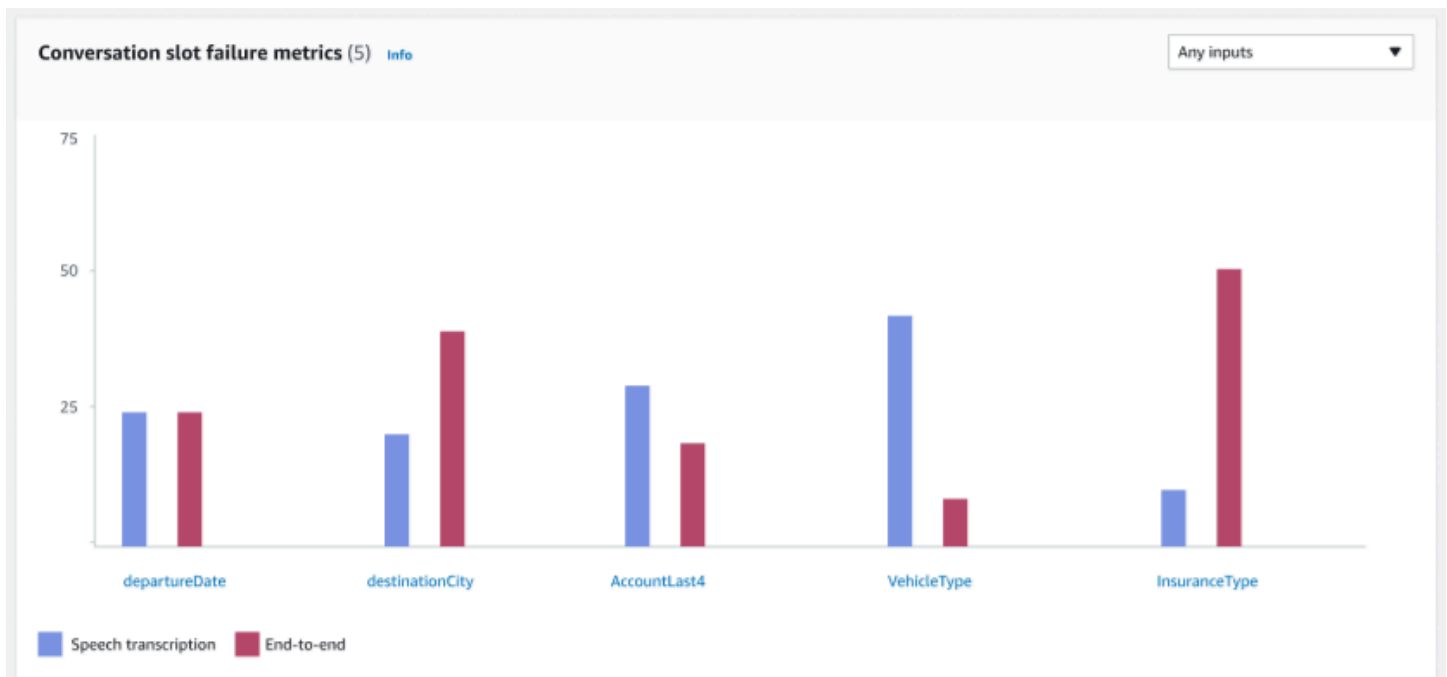
Any outcomes ▾ < 1 2 3 4 > ⚙

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

Erfolgsquoten für Konversationen — Anhand der Tabelle mit den Erfolgsquoten für Konversationen wird ermittelt, welche Absichten und Zeitpunkte in den einzelnen Konversationen im Testset verwendet wurden. Sie können visualisieren, wo die Konversation gescheitert ist, indem Sie überprüfen, welche Absicht oder welcher Slot fehlgeschlagen ist, sowie den Prozentsatz, in dem die Konversation bestanden hat.



Messwerte für fehlgeschlagene Konversationsabsichten — Diese Metrik zeigt die fünf Intentionen mit der schlechtesten Leistung im Testset. In diesem Bereich wird anhand der Konversationsprotokolle oder der Transkription des Bots grafisch dargestellt, wie viel Prozent oder wie viele Intents erfolgreich waren oder nicht. Eine erfolgreiche Absicht bedeutet nicht, dass die gesamte Konversation erfolgreich war. Diese Kennzahlen beziehen sich nur auf den Wert der Absichten, unabhängig davon, welche Absicht davor oder danach kam.



Metriken zum Ausfall von Konversationsslots — Diese Metrik zeigt die fünf Slots mit der schlechtesten Leistung im Testsatz. Zeigt die Erfolgsquote für jeden Slot im Intent an. Das Balkendiagramm zeigt sowohl die Sprachtranskription als auch die end-to-end Konversationen für jeden Slot in der Absicht.

Registerkarte „Absicht“ und „Slot-Ergebnisse“:

Intent recognition metrics (8)

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%

Kennzahlen zur Absichtserkennung — Zeigt in einer Tabelle an, wie viele Absichten erfolgreich erkannt wurden. Zeigt die Erfolgsquote der Sprachtranskription und end-to-end der Konversationen an.

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						

Metriken zur Slot-Auflösung — Zeigt die Absichten und Slots getrennt an sowie die Erfolgs- und Misserfolgsrate jedes Slots für jede Absicht, die in der Konversation oder einzelnen Eingabe verwendet wurde. Zeigt die Erfolgsquote der Sprachtranskription und end-to-end der Konversationen an.

Registerkarte mit detaillierten Ergebnissen:

Detailed results (160)								
								Download
< 1 2 3 4 >								
Line #	Conversation #	S3 Audio link 	Source	Slot spelling 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	-	-

Detaillierte Ergebnisse — Zeigt eine detaillierte Tabelle im Konversationsprotokoll mit den Äußerungen von Benutzern und Agenten sowie der erwarteten Ausgabe und der erwarteten Transkription für jeden Slot an. Sie können diesen Bericht herunterladen, indem Sie auf die Schaltfläche Herunterladen klicken.

In der folgenden Tabelle sind die Fehlermeldungen mit den entsprechenden Szenarien aufgeführt.

Szenario	Fehlermeldung	Aktion
Absicht stimmt nicht überein	Erwartete BookFlight Absicht, aber es war BookHotel Absicht.	Überspringe andere Runden in der Konversation
Slot-Elicitation stimmt nicht überein	Es wurde erwartet, dass das Zeitfenster für das Abflugdatum ausgewählt wurde, aber es war CabinType.	Überspringe andere Runden in der Konversation
Die Slot-Werte stimmen nicht überein	Nichtübereinstimmung zwischen dem erwarteten und dem tatsächlichen Slot-Wert.	Fahren Sie mit anderen Runden in den Konversationen fort
Back-to-back Die Agenten-Eingabeaufforderung fehlt	Es wurde erwartet, dass der Bot in diesem Zug eine Agentenaufforderung zurückgibt, aber sie wurde nicht empfangen.	Überspringe andere Runden in der Konversation
Die Transkription stimmt nicht überein	Die erwartete Transkription stimmte nicht mit der tatsächlichen Transkription überein.	Fahren Sie mit anderen Runden in den Konversationen fort
Optional Slot wurde nicht ausgelöst	Es wird erwartet, dass in der nächsten Runde der CabinType-Slot ausgelöst wird, die aktuelle Absicht wurde jedoch zuvor erfüllt.	Überspringe andere Runden in der Konversation

Szenario	Fehlermeldung	Aktion
Steckplatz wurde nicht erkannt	Der Slot Expected Departure Date wurde in dieser Runde nicht erkannt.	Überspringe andere Runden in der Konversation
Zusätzliche back-to-back Agentenaufforderung	Es wurde erwartet, dass ein Benutzer an der Reihe war, aber es war eine Agentenauff	Überspringe andere Runden in der Konversation

Konversationen an einen Amazon Lex V2-Bot streamen

Sie können die Amazon Lex V2-Streaming-API verwenden, um einen bidirektionalen Stream zwischen einem Amazon Lex V2-Bot und Ihrer Anwendung zu starten. Durch das Starten eines Streams kann der Bot die Konversation zwischen dem Bot und dem Benutzer verwalten. Der Bot reagiert auf Benutzereingaben, ohne dass Sie Code schreiben müssen, um die Antworten des Benutzers zu verarbeiten. Der Bot kann:

- Behandeln Sie Unterbrechungen durch den Benutzer, während er eine Aufforderung abspielt. Weitere Informationen finden Sie unter [So aktivieren, dass Ihr Amazon Lex V2-Bot vom Benutzer unterbrochen wird](#).
- Warten Sie, bis der Benutzer Eingaben macht. Beispielsweise kann der Bot darauf warten, dass der Benutzer Kreditkarteninformationen sammelt. Weitere Informationen finden Sie unter [Den Amazon Lex V2-Bot so aktivieren, dass er während einer Pause darauf wartet, dass der Benutzer weitere Informationen bereitstellt](#).
- Nehmen Sie sowohl Dual-Tone-Multiple-Frequency (DTMF) als auch Audioeingänge im selben Stream auf.
- Behandeln Sie Pausen bei Benutzereingaben besser, als wenn Sie die Konversation von Ihrer Anwendung aus verwalten würden.

Der Amazon Lex V2-Bot reagiert nicht nur auf Daten, die von Ihrer Anwendung gesendet wurden, sondern sendet auch Informationen über den Status der Konversation an Ihre Anwendung. Sie können diese Informationen verwenden, um zu ändern, wie Ihre Anwendung auf Kunden reagiert.

Der Amazon Lex V2-Bot überwacht auch die Verbindung zwischen dem Bot und Ihrer Anwendung. Er kann feststellen, ob bei der Verbindung ein Timeout aufgetreten ist.

Informationen zur Verwendung der API zum Starten eines Streams zu einem Amazon Lex V2-Bot finden Sie unter [Starten eines Konversationsstreams zu einem Amazon Lex V2-Bot](#).

Wenn Sie von Ihrer Anwendung aus zu einem Amazon Lex V2-Bot streamen, können Sie den Bot so konfigurieren, dass er Audio- oder Texteingaben vom Benutzer akzeptiert. Sie können auch wählen, ob der Benutzer als Antwort auf seine Eingabe Audio oder Text erhält.

Wenn Sie den Amazon Lex V2-Bot so konfiguriert haben, dass er Audioeingaben vom Benutzer akzeptiert, kann er keine Texteingabe annehmen. Wenn Sie den Bot so konfiguriert haben, dass

er Texteingaben akzeptiert, kann der Benutzer nur geschriebenen Text verwenden, um mit ihm zu kommunizieren.

Wenn ein Amazon Lex V2-Bot eine Streaming-Audioeingabe entgegennimmt, bestimmt der Bot, wann ein Benutzer zu sprechen beginnt und wann er aufhört zu sprechen. Er verarbeitet alle Pausen oder Unterbrechungen durch den Benutzer. Es kann auch DTMF-Eingaben (Dual-Tone Multifrequency) und Spracheingaben im selben Stream verarbeiten. Dies hilft dem Benutzer, natürlicher mit dem Bot zu interagieren. Sie können Benutzern Willkommensnachrichten und Eingabeaufforderungen präsentieren. Sie können Benutzern auch ermöglichen, diese Nachrichten und Eingabeaufforderungen zu unterbrechen.

Wenn Sie einen bidirektionalen Stream starten, verwendet Amazon Lex V2 das [HTTP/2-Protokoll](#). Ihre Anwendung und der Bot tauschen Daten in einem einzigen Stream als eine Reihe von Ereignissen aus. Ein Ereignis kann eines der folgenden sein:

- Text-, Audio- oder DTMF-Eingabe vom Benutzer.
- Signale von der Anwendung an den Amazon Lex V2-Bot. Dazu gehört der Hinweis, dass die Audiowiedergabe einer Nachricht abgeschlossen wurde oder dass der Benutzer die Verbindung zur Sitzung getrennt hat.

Weitere Informationen über -Ereignisse finden Sie unter [Starten eines Konversationsstreams zu einem Amazon Lex V2-Bot](#). Hinweise zum Kodieren von Ereignissen finden Sie unter [Ereignis-Stream-Kodierung](#)

Themen

- [Starten eines Konversationsstreams zu einem Amazon Lex V2-Bot](#)
- [Ereignis-Stream-Kodierung](#)
- [So aktivieren, dass Ihr Amazon Lex V2-Bot vom Benutzer unterbrochen wird](#)
- [Den Amazon Lex V2-Bot so aktivieren, dass er während einer Pause darauf wartet, dass der Benutzer weitere Informationen bereitstellt](#)
- [Updates zum Fulfillment-Fortschritt für Ihren Lex V2-Bot konfigurieren](#)
- [Konfiguration von Timeouts für die Erfassung von Benutzereingaben mit einem Lex V2-Bot](#)

Starten eines Konversationsstreams zu einem Amazon Lex V2-Bot

Sie verwenden den [StartConversation](#) Vorgang, um einen Stream zwischen dem Benutzer und dem Amazon Lex V2-Bot in Ihrer Anwendung zu starten. Die POST Anfrage von der Anwendung stellt eine Verbindung zwischen Ihrer Anwendung und dem Amazon Lex V2-Bot her. Auf diese Weise können Ihre Anwendung und der Bot über Ereignisse Informationen miteinander austauschen.

Note

Wenn bei der Verwendung `StartConversation` die Audiodauer der Äußerung den konfigurierten Wert für `overshoot-max-length-ms` überschreitet, wird die Audiowiedergabe bei der angegebenen Dauer unterbrochen.

Der `StartConversation` Vorgang wird nur in den folgenden SDKs Fällen unterstützt:

- [AWS SDK für C++](#)
- [AWS SDK für Java V2](#)
- [AWS-SDK für JavaScript Version 3](#)
- [AWS SDK für Ruby V3](#)

Das erste Ereignis, das Ihre Bewerbung an den Amazon Lex V2-Bot senden muss, ist ein [ConfigurationEvent](#). Dieses Ereignis enthält Informationen wie das Format des Antworttyps. Die folgenden Parameter können Sie in einem Konfigurationsereignis verwenden:

- `responseContentType`— Legt fest, ob der Bot auf Benutzereingaben mit Text oder Sprache reagiert.
- `sessionState` — Informationen zur Streaming-Sitzung mit dem Bot, z. B. vorgegebene Absicht oder Dialogstatus.
- `WelcomeMessages` — Gibt die Willkommensnachrichten an, die dem Benutzer zu Beginn seiner Konversation mit einem Bot abgespielt werden. Diese Nachrichten werden abgespielt, bevor der Benutzer etwas eingibt. Um eine Willkommensnachricht zu aktivieren, müssen Sie auch Werte für die `dialogAction` Parameter `sessionState` und angeben.
- `DisablePlayback` — Legt fest, ob der Bot auf einen Hinweis vom Client warten soll, bevor er auf Anrufereingaben wartet. Standardmäßig ist die Wiedergabe aktiviert, der Wert dieses Felds ist also `false`

- `requestAttributes` — Stellt zusätzliche Informationen für die Anfrage bereit.

Informationen zum Angeben von Werten für die vorherigen Parameter finden Sie unter dem [ConfigurationEvent](#)-Datentyp des [StartConversation](#)-Vorgangs.

Jeder Stream zwischen einem Bot und Ihrer Anwendung kann nur ein Konfigurationsereignis haben. Nachdem Ihre Anwendung ein Konfigurationsereignis gesendet hat, kann der Bot zusätzliche Kommunikation von Ihrer Anwendung entgegennehmen.

Wenn Sie angegeben haben, dass Ihr Benutzer Audio für die Kommunikation mit dem Amazon Lex V2-Bot verwendet, kann Ihre Anwendung während dieser Konversation die folgenden Ereignisse an den Bot senden:

- [AudioInputEvent](#) — Enthält einen Audio-Chunk mit einer maximalen Größe von 320 Byte. Ihre Anwendung muss mehrere Audioeingabeereignisse verwenden, um eine Nachricht vom Server an den Bot zu senden. Jedes Audioeingabeereignis im Stream muss dasselbe Audioformat haben.
- [DTMFInputEreignis](#) — Sendet eine DTMF-Eingabe an den Bot. Jeder DTMF-Tastendruck entspricht einem einzelnen Ereignis.
- [PlaybackCompletionEvent](#) — Informiert den Server darüber, dass eine Antwort auf Benutzereingaben für ihn wiedergegeben wurde. Sie müssen ein Ereignis zum Abschluss der Wiedergabe verwenden, wenn Sie eine Audioantwort an den Benutzer senden. Wenn es sich `disablePlayback` bei Ihrem Konfigurationsereignis um ein Ereignis handelt `true`, können Sie diese Funktion nicht verwenden.
- [DisconnectionEvent](#) — Informiert den Bot darüber, dass der Benutzer die Verbindung zur Konversation unterbrochen hat.

Wenn Sie angegeben haben, dass der Benutzer Text verwendet, um mit dem Bot zu kommunizieren, kann Ihre Anwendung während dieser Konversation die folgenden Ereignisse an den Bot senden:

- [TextInputEvent](#) — Text, der von Ihrer Anwendung an den Bot gesendet wird. Ein Texteingabeereignis kann bis zu 512 Zeichen enthalten.
- [PlaybackCompletionEvent](#) — Informiert den Server darüber, dass eine Antwort auf die Benutzereingabe für ihn wiedergegeben wurde. Sie müssen dieses Ereignis verwenden, wenn Sie dem Benutzer Audio wiedergeben. Wenn es sich bei `disablePlayback` Ihrer Konfiguration um ein Ereignis handelt `true`, können Sie diese Funktion nicht verwenden.

- [DisconnectionEvent](#)— Informiert den Bot darüber, dass der Benutzer die Verbindung zur Konversation unterbrochen hat.

Sie müssen jedes Ereignis, das Sie an einen Amazon Lex V2-Bot senden, im richtigen Format codieren. Weitere Informationen finden Sie unter [Ereignis-Stream-Kodierung](#).

Jedes Ereignis hat eine Ereignis-ID. Um Probleme zu beheben, die im Stream auftreten könnten, weisen Sie jedem Eingabeereignis eine eindeutige Ereignis-ID zu. Anschließend können Sie alle Verarbeitungsfehler mit dem Bot beheben.

Amazon Lex V2 verwendet auch Zeitstempel für jedes Ereignis. Sie können diese Zeitstempel zusätzlich zur Ereignis-ID verwenden, um Probleme mit der Netzwerkübertragung zu beheben.

Während der Konversation zwischen dem Benutzer und dem Amazon Lex V2-Bot kann der Bot die folgenden ausgehenden Ereignisse als Antwort an den Benutzer senden:

- [IntentResultEvent](#)— Enthält die Absicht, die Amazon Lex V2 anhand der Benutzeräußerung ermittelt hat. Jedes interne Ergebnisereignis beinhaltet:
 - `InputMode` — Die Art der Benutzeräußerung. Gültige Werte sind `Speech`, `DTMF` oder `Text`.
 - `Interpretationen` — Interpretationen, die Amazon Lex V2 anhand der Benutzeräußerung bestimmt.
 - `requestAttributes` — Wenn Sie die Anforderungsattribute nicht mithilfe einer Lambda-Funktion geändert haben, handelt es sich um dieselben Attribute, die zu Beginn der Konversation übergeben wurden.
 - `sessionId` — Sitzungs-ID, die für die Konversation verwendet wird.
 - `sessionState` — Der Status der Benutzersitzung mit Amazon Lex V2.
- [TranscriptEvent](#)— Wenn der Benutzer eine Eingabe in Ihre Anwendung eingibt, enthält dieses Ereignis die Abschrift der Äußerung des Benutzers gegenüber dem Bot. Ihre Anwendung erhält keine, `TranscriptEvent` wenn keine Benutzereingabe erfolgt.

Der Wert des an Ihre Anwendung gesendeten Transkriptereignisses hängt davon ab, ob Sie Audio (Sprache und DTMF) oder Text als Konversationsmodus angegeben haben:

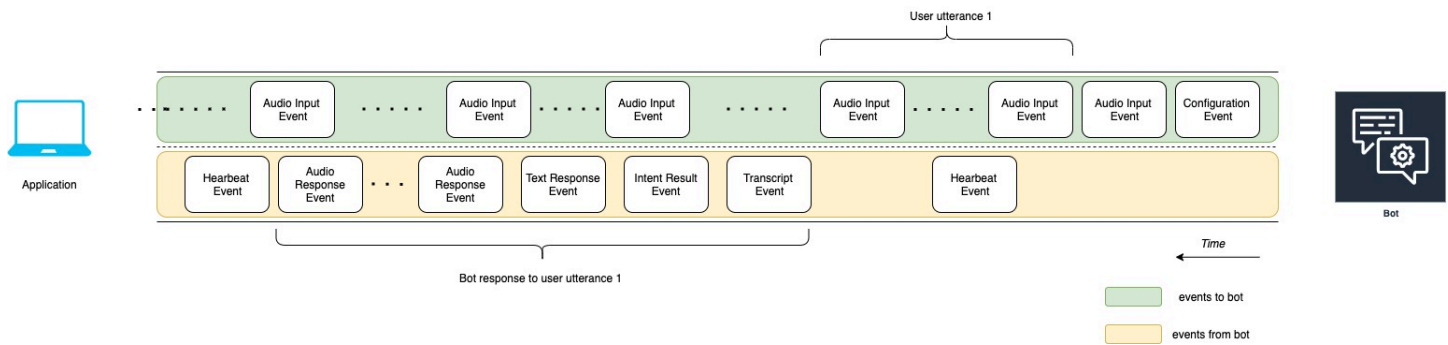
- `Transkript der Spracheingabe` — Wenn der Benutzer mit dem Bot spricht, ist das Transkriptionsereignis die Transkription des Audios des Benutzers. Es ist eine Abschrift der gesamten Sprache von dem Zeitpunkt an, an dem der Benutzer zu sprechen beginnt, bis zu dem Zeitpunkt, an dem er mit dem Sprechen aufhört.

- Abschrift der DTMF-Eingabe — Wenn der Benutzer auf einer Tastatur tippt, enthält das Transkriptionereignis alle Ziffern, die der Benutzer bei der Eingabe gedrückt hat.
- Abschrift der Texteingabe — Wenn der Benutzer eine Texteingabe vornimmt, enthält das Transkriptionereignis den gesamten Text der Benutzereingabe.
- [TextResponseEvent](#)— Enthält die Bot-Antwort im Textformat. Standardmäßig wird eine Textantwort zurückgegeben. Wenn Sie Amazon Lex V2 so konfiguriert haben, dass eine Audioantwort zurückgegeben wird, wird dieser Text verwendet, um eine Audioantwort zu generieren. Jedes Textantwort-Ereignis enthält eine Reihe von Nachrichtenobjekten, die der Bot an den Benutzer zurückgibt.
- [AudioResponseEvent](#)— Enthält die Audioantwort, die aus dem in der generierten Text synthetisiert wurde. `TextResponseEvent` Um Audioantwortereignisse zu empfangen, müssen Sie Amazon Lex V2 so konfigurieren, dass eine Audioantwort bereitgestellt wird. Alle Audioantwort-Ereignisse haben dasselbe Audioformat. Jedes Ereignis enthält Audioblöcke von nicht mehr als 100 Byte. Amazon Lex V2 sendet einen leeren Audioblock, bei dem das `bytes` Feld auf gesetzt ist, `null` um das Ende des Audioantwortereignisses an Ihre Anwendung anzuzeigen.
- [PlaybackInterruptionEvent](#)— Wenn ein Benutzer eine Antwort unterbricht, die der Bot an Ihre Anwendung gesendet hat, löst Amazon Lex V2 dieses Ereignis aus, um die Wiedergabe der Antwort zu stoppen.
- [HeartbeatEvent](#)— Amazon Lex V2 sendet dieses Ereignis regelmäßig zurück, um zu verhindern, dass es bei der Verbindung zwischen Ihrer Anwendung und dem Bot zu einem Timeout kommt.

Zeitliche Abfolge von Ereignissen für eine Audiokonversation bei Verwendung eines Amazon Lex V2-Bots

Die folgenden Diagramme zeigen eine Streaming-Audiokonversation zwischen einem Benutzer und einem Amazon Lex V2-Bot. Die Anwendung streamt kontinuierlich Audio an den Bot, und der Bot sucht anhand des Audios nach Benutzereingaben. In diesem Beispiel verwenden sowohl der Benutzer als auch der Bot Sprache, um zu kommunizieren. Jedes Diagramm entspricht einer Benutzeräußerung und der Reaktion des Bots auf diese Äußerung.

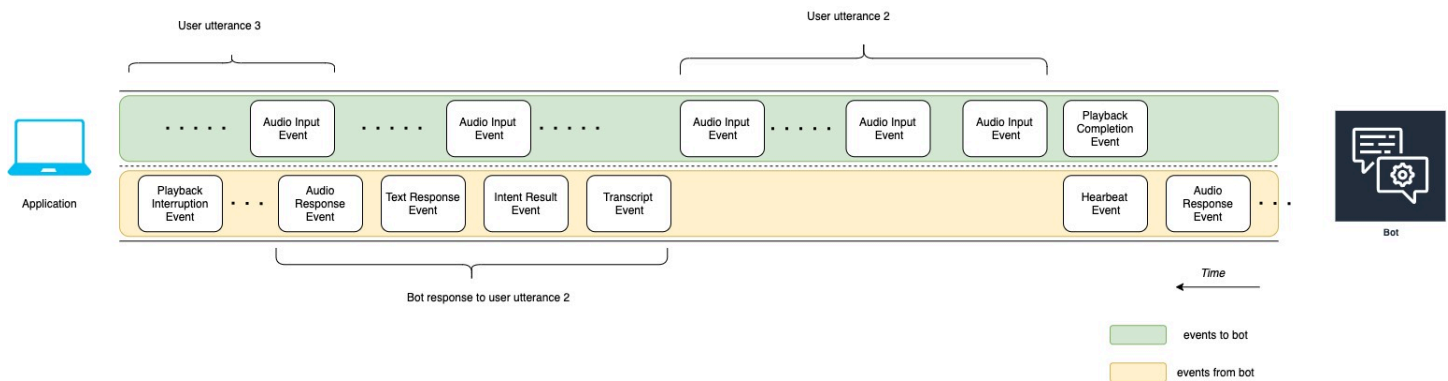
Das folgende Diagramm zeigt den Beginn einer Konversation zwischen der Anwendung und dem Bot. Der Stream beginnt zum Zeitpunkt Null (t_0).



In der folgenden Liste werden die Ereignisse des vorherigen Diagramms beschrieben.

- t0: Die Anwendung sendet ein Konfigurationsereignis an den Bot, um den Stream zu starten.
- t1: Die Anwendung streamt Audiodaten. Diese Daten werden in eine Reihe von Eingabeereignissen der Anwendung aufgeteilt.
- t2: Bei Benutzeräußerung 1 erkennt der Bot ein Audioeingabeereignis, wenn der Benutzer zu sprechen beginnt.
- t2: Während der Benutzer spricht, sendet der Bot ein Heartbeat-Ereignis, um die Verbindung aufrechtzuerhalten. Er sendet diese Ereignisse in regelmäßigen Abständen, um sicherzustellen, dass die Verbindung nicht unterbrochen wird.
- t3: Der Bot erkennt das Ende der Äußerung des Benutzers.
- t4: Der Bot sendet ein Transkriptereignis zurück, das eine Abschrift der Sprache des Benutzers an die Anwendung enthält. Dies ist der Beginn der Bot-Antwort auf Benutzeräußerung 1.
- t5: Der Bot sendet ein Absichtsergebnis, um die Aktion anzugeben, die der Benutzer ausführen möchte.
- t6: Der Bot beginnt, seine Antwort als Text in einem Textantwortereignis bereitzustellen.
- t7: Der Bot sendet eine Reihe von Audioantwortereignissen an die Anwendung, um sie für den Benutzer abzuspielen.
- t8: Der Bot sendet ein weiteres Heartbeat-Ereignis, um die Verbindung zeitweise aufrechtzuerhalten.

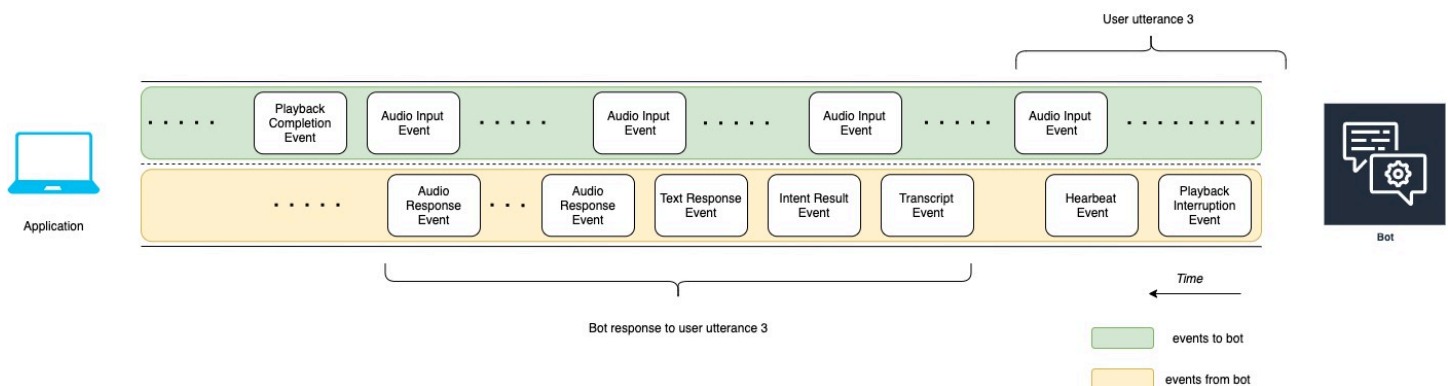
Das folgende Diagramm ist eine Fortsetzung des vorherigen Diagramms. Es zeigt, dass die Anwendung ein Ereignis zum Abschluss der Wiedergabe an den Bot sendet, um anzuzeigen, dass die Audioantwort für den Benutzer nicht mehr abgespielt wurde. Die Anwendung gibt dem Benutzer die Antwort des Bots auf die Benutzeräußerung 1 wieder. Der Benutzer reagiert auf die Bot-Antwort auf Benutzeräußerung 1 mit Benutzeräußerung 2.



In der folgenden Liste werden die Ereignisse des vorherigen Diagramms beschrieben:

- t10: Die Anwendung sendet ein Ereignis zum Abschluss der Wiedergabe, um anzuzeigen, dass die Wiedergabe der Bot-Nachricht an den Benutzer abgeschlossen ist.
- t11: Die Anwendung sendet die Benutzerantwort als Benutzeräußerung 2 an den Bot zurück.
- t12: Bei der Bot-Antwort auf Benutzeräußerung 2 wartet der Bot, bis der Benutzer aufhört zu sprechen, und beginnt dann, eine Audioantwort zu geben.
- t13: Während der Bot die Bot-Antwort auf Benutzeräußerung 2 an die Anwendung sendet, erkennt der Bot den Beginn von Benutzeräußerung 3. Der Bot stoppt die Bot-Antwort auf Benutzeräußerung 2 und sendet ein Ereignis zur Unterbrechung der Wiedergabe.
- t14: Der Bot sendet ein Ereignis zur Unterbrechung der Wiedergabe an die Anwendung, um zu signalisieren, dass der Benutzer die Aufforderung unterbrochen hat.

Das folgende Diagramm zeigt, wie der Bot auf Benutzeräußerung 3 reagiert und dass die Konversation fortgesetzt wird, nachdem der Bot auf die Benutzeräußerung reagiert hat.



Verwenden der API, um eine Streaming-Konversation zu starten

Wenn Sie einen Stream zu einem Amazon Lex V2-Bot starten, führen Sie die folgenden Aufgaben aus:

1. Stellen Sie eine erste Verbindung zum Server her.
2. Konfigurieren Sie die Sicherheitsanmeldedaten und Bot-Details. Zu den Bot-Details gehört, ob der Bot DTMF- und Audioeingaben oder Texteingaben akzeptiert.
3. Ereignisse an den Server senden. Bei diesen Ereignissen handelt es sich um Text- oder Audiodaten des Benutzers.
4. Vom Server gesendete Ereignisse verarbeiten. In diesem Schritt legen Sie fest, ob die Bot-Ausgabe dem Benutzer als Text oder Sprache präsentiert wird.

Die folgenden Codebeispiele initialisieren eine Streaming-Konversation mit einem Amazon Lex V2-Bot und Ihrem lokalen Computer. Sie können den Code an Ihre Bedürfnisse anpassen.

Der folgende Code ist eine Beispielanforderung, mit der die Verbindung AWS SDK für Java zu einem Bot hergestellt und die Bot-Details und Anmeldeinformationen konfiguriert werden.

```
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();
        }
    }
}

```

Der folgende Code ist eine Beispielanforderung, mit der Ereignisse AWS SDK für Java an den Bot gesendet werden. Der Code in diesem Beispiel verwendet das Mikrofon Ihres Computers, um Audioereignisse zu senden.

```

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();
        }
    }
}
```

Der folgende Code ist eine Beispielanforderung, die verwendet wird AWS SDK für Java , um Antworten des Bots zu verarbeiten. Der Code in diesem Beispiel konfiguriert Amazon Lex V2 so, dass es Ihnen eine Audioantwort wiedergibt.

```
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;
}
}

```

Um einen Bot so zu konfigurieren, dass er auf Eingabeereignisse mit Audio reagiert, müssen Sie zuerst Audioereignisse von Amazon Lex V2 abonnieren und dann den Bot so konfigurieren, dass er eine Audioantwort auf die Eingabeereignisse des Benutzers bereitstellt.

Der folgende Code ist ein AWS SDK für Java Beispiel für das Abonnieren von Audio-Events von 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);
}

}

}

}

```

Im folgenden AWS SDK für Java Beispiel wird der Amazon Lex V2-Bot so konfiguriert, dass er eine Audioantwort auf die Eingabeereignisse liefert.

```

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();  
}  
}
```

Ereignis-Stream-Kodierung

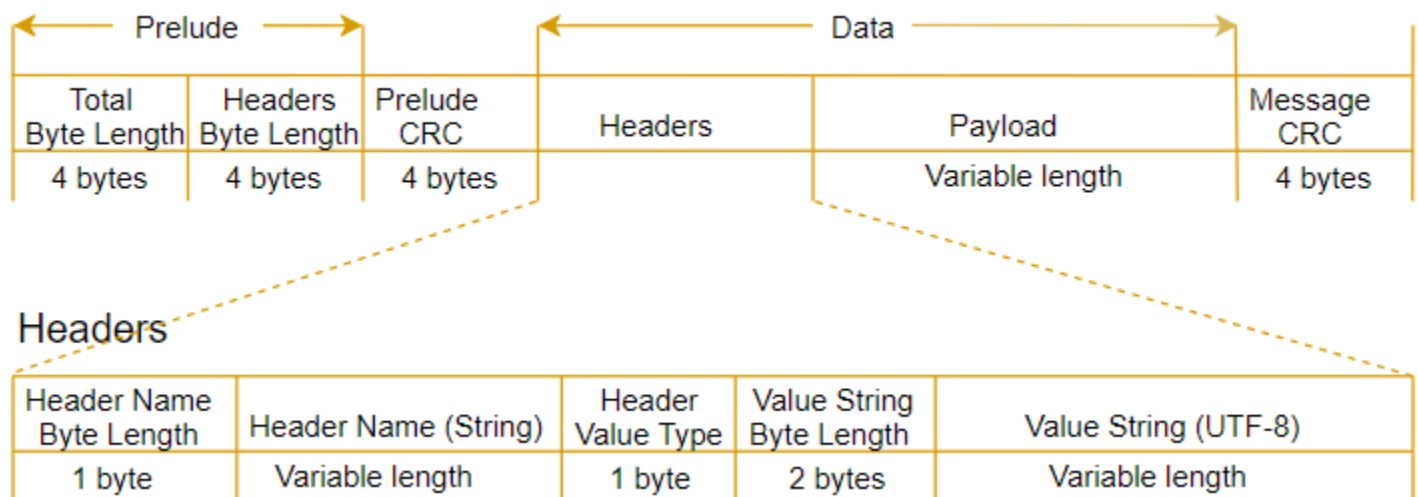
Die Ereignis-Stream-Kodierung bietet bidirektionale Kommunikation anhand von Nachrichten zwischen einem Client und einem Server. Datenframes, die an den Amazon Lex V2-Streaming-Dienst gesendet werden, sind in diesem Format codiert. Die Antwort von Amazon Lex V2 verwendet ebenfalls diese Kodierung.

Jede Nachricht besteht aus zwei Abschnitten: der Einleitung und den Daten. Der Prelude-Abschnitt enthält die Gesamtbytelänge der Nachricht und die kombinierte Bytelänge aller Header. Der Datenbereich enthält die Header und eine Nutzlast.

Jeder Abschnitt endet mit einer 4-Byte-Big-Endian-Ganzzahl-CRC-Prüfsumme. Die CRC-Prüfsumme der Nachricht umfasst den Prälude-Abschnitt und den Datenbereich. Amazon Lex V2 verwendet CRC32 (oft als GZIP bezeichnet CRC32), um beide CRCs zu berechnen. Weitere Informationen CRC32 dazu finden Sie in der [GZIP-Dateiformatspezifikation Version 4.3](#).

Der gesamte Nachrichten-Overhead, einschließlich der Einleitung und beider Prüfsummen, beträgt 16 Bytes.

Das folgende Diagramm zeigt die Komponenten, aus denen eine Nachricht und ein Header bestehen. Pro Nachricht gibt es mehrere Header.



Jede Nachricht enthält die folgenden Komponenten:

- Einleitung: immer eine feste Größe von 8 Bytes (zwei Felder von jeweils 4 Bytes).
 - Die ersten 4 Bytes: die gesamte Byte-Länge. Dies ist die Big-Endian-Ganzzahl-Byte-Länge der gesamten Nachricht, einschließlich des 4-Byte-Längenfelds selbst.
 - Die zweiten 4 Bytes: die Byte-Länge der Header. Dies ist die Big-Endian-Ganzzahl-Byte-Länge des Header-Teils der Nachricht, ohne das Header-Längenfeld selbst.
- Einleitungs-CRC: die 4-Byte-CRC-Prüfsumme des Einleitungsteils der Nachricht, mit Ausnahme der CRC selbst. Das Prelude hat einen vom Nachrichten-CRC getrennten CRC, um sicherzustellen, dass Amazon Lex V2 beschädigte Informationen mit Bytelänge sofort erkennen kann, ohne Fehler wie Pufferüberläufe zu verursachen.
- Header: Metadaten, welche die Nachricht mit Anmerkungen versehen, wie z. B. Nachrichtentyp, Inhaltstyp usw. Nachrichten verfügen über mehrere Header. Header sind Schlüssel-Wert-Paare, bei denen der Schlüssel aus einer UTF-8-Zeichenfolge besteht. Header können in beliebiger Reihenfolge im Header-Teil der Nachricht erscheinen und jeder beliebige Header kann ausschließlich einmal erscheinen. Die erforderlichen Header-Typen finden Sie in den folgenden Abschnitten.
- Payload: Der Audio- oder Textinhalt, der an Amazon Lex gesendet wird.
- Nachrichten-CRC: die 4-Byte-CRC-Prüfsumme vom Beginn der Nachricht bis zum Beginn der Prüfsumme. Das beinhaltet alles in der Nachricht außer dem CRC selbst.

Jeder Header enthält die folgenden Komponenten. Pro Frame gibt es mehrere Header.

- Byte-Länge des Header-Namens: die Byte-Länge des Header-Namens.

- **Header-Name:** der Name des Headers, der den Header-Typ angibt. Gültige Werte finden Sie in den folgenden Frame-Beschreibungen.
- **Header-Werttyp:** eine Aufzählung, die den Header-Werttyp angibt.
- **Byte-Länge der Wertzeichenfolge:** die Byte-Länge der Header-Wertzeichenfolge.
- **Header-Wert:** der Wert der Header-Zeichenfolge. Gültige Werte für dieses Feld sind vom Header-Typ abhängig. Gültige Werte finden Sie in den folgenden Frame-Beschreibungen.

So aktivieren, dass Ihr Amazon Lex V2-Bot vom Benutzer unterbrochen wird

Wenn Sie einen bidirektionalen Audiostream zwischen einem Amazon Lex V2-Bot und Ihrer Anwendung starten, können Sie den Bot so konfigurieren, dass er auf Benutzereingaben wartet, während er eine Aufforderung zurücksendet. Diese Eingabe kann als Interrupt festgelegt werden. Damit kann der Benutzer die Aufforderung unterbrechen, bevor der Bot die Wiedergabe beendet hat. Sie können diese Konfiguration für Situationen verwenden, in denen der Benutzer die Antwort auf eine Frage möglicherweise bereits kennt, z. B. wenn er aufgefordert wird, einen CVV-Code einzugeben.

Ein Bot weiß, wann der Benutzer eine Aufforderung unterbricht, wenn er Benutzereingaben erkennt, bevor Ihre Anwendung ein `PlaybackCompletion` Ereignis zurücksenden kann. Wenn der Benutzer einen Bot unterbricht, sendet der Bot eine `PlaybackInterruptionEvent`.

Standardmäßig kann der Benutzer jede Aufforderung unterbrechen, die der Bot in Ihre Anwendung streamt. Sie können diese Einstellung in der Amazon Lex V2-Konsole ändern.

Sie können ändern, wie ein Benutzer auf eine Aufforderung reagieren kann, indem Sie einen Slot bearbeiten. Ein Slot ist Teil einer Absicht, und es ist das Mittel, mit dem der Benutzer Ihnen die gewünschten Informationen zur Verfügung stellt. In jedem Slot wird der Benutzer aufgefordert, Ihnen diese Informationen zur Verfügung zu stellen. Weitere Informationen zu Spielautomaten finden Sie unter [Kernkonzepte von Amazon Lex V2](#).

Um zu ändern, ob der Benutzer eine Aufforderung unterbrechen kann (Konsole)

1. Melden Sie sich an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole auf der [Amazon Lex V2-Konsole](#).
2. Wählen Sie unter Bots einen Bot aus.

3. Wählen Sie unter Sprache die Sprache des Bots aus.
4. Wählen Sie „Absichten anzeigen“.
5. Wählen Sie die Absicht aus.
6. Wählen Sie für Slots einen Slot aus.
7. Wählen Sie unter Erweiterte Optionen die Option Slot-Eingabeaufforderungen aus.
8. Wählen Sie Weitere Eingabeaufforderungsoptionen.
9. Aktivieren oder deaktivieren Sie Benutzer können die Aufforderung unterbrechen, wenn sie gelesen wird.

Sie können diese Funktionalität testen, indem Sie einen Bot mit zwei Steckplätzen erstellen und angeben, dass Benutzer eine Aufforderung für einen Slot nicht unterbrechen können. Wenn Sie eine unterbrechbare Aufforderung unterbrechen, sendet der Bot ein Ereignis zur Unterbrechung der Wiedergabe. Wenn Sie eine unterbrechungsfreie Verbindung unterbrechen, wird die Aufforderung weiter abgespielt.

Den Amazon Lex V2-Bot so aktivieren, dass er während einer Pause darauf wartet, dass der Benutzer weitere Informationen bereitstellt

Wenn Sie einen bidirektionalen Stream von einem Amazon Lex V2-Bot zu Ihrer Anwendung starten, können Sie den Bot so konfigurieren, dass er darauf wartet, dass der Benutzer zusätzliche Informationen bereitstellt. Unter bestimmten Umständen ist ein Benutzer möglicherweise nicht bereit, auf eine Aufforderung zu antworten. Beispielsweise ist ein Benutzer möglicherweise nicht bereit, seine Kreditkarteninformationen anzugeben, weil sich seine Brieftasche in einem anderen Raum befindet.

Mithilfe des Verhaltens „Warten und fortfahren“ des Amazon Lex V2-Bots können Benutzer Formulierungen wie „Moment mal warten“ sagen, sodass der Bot darauf wartet, dass sie die Informationen finden und bereitstellen. Wenn Sie dieses Verhalten aktivieren, erinnert der Bot den Benutzer regelmäßig daran, die Informationen bereitzustellen. Er sendet keine Transkriptereignisse zurück, da es keine Benutzeräußerungen gibt, die er transkribieren könnte.

Der Amazon Lex V2-Bot verwaltet automatisch eine Streaming-Konversation. Sie müssen keinen zusätzlichen Code schreiben, um diese Funktionalität zu aktivieren. Wenn ein Bot vom Benutzer

aufgefordert wird, zu warten, gilt das `state` von dem Intent ist `Waiting` und das `type` von dem `DialogAction` ist `ElicitSlot`. Sie können diese Informationen verwenden, um Ihre Anwendung an Ihre Bedürfnisse anzupassen. Sie können Ihre Anwendung beispielsweise so konfigurieren, dass Musik abgespielt wird, wenn der Benutzer nach seiner Kreditkarte sucht.

Sie aktivieren das Verhalten „Warten und Fortfahren“ für einen einzelnen Slot. Weitere Informationen zu Spielautomaten finden Sie unter [Kernkonzepte von Amazon Lex V2](#).

So aktivieren Sie „Warten und fortfahren“

1. Melden Sie sich an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole auf der [Amazon Lex V2-Konsole](#).
2. Wählen Sie unter Bots einen Bot aus.
3. Wählen Sie unter Sprache die Sprache des Bots aus.
4. Wählen Sie „Absichten anzeigen“.
5. Wählen Sie die Absicht aus.
6. Wählen Sie unter Slots einen Slot aus.
7. Wählen Sie unter Erweiterte Optionen die Option Warten und fortfahren aus.
8. Geben Sie unter Warten und fortfahren die folgenden Felder an:
 - Antwort, wenn der Benutzer möchte, dass der Bot wartet — So reagiert der Bot, wenn der Benutzer ihn bittet, auf die zusätzlichen Informationen zu warten.
 - Antwort, wenn der Benutzer möchte, dass der Bot weiter wartet — Dies ist die Antwort, die der Bot sendet, um den Benutzer daran zu erinnern, dass er immer noch auf die Informationen wartet. Sie können ändern, wie oft der Bot den Benutzer daran erinnert.
 - Antwort, wenn der Benutzer fortfahren möchte — Dies ist die Antwort des Bots, wenn der Benutzer die angeforderten Informationen hat.

Für jede Bot-Antwort können Sie mehrere Varianten der Antwort angeben, und eine wird dem Benutzer nach dem Zufallsprinzip angezeigt. Sie können auch wählen, ob diese Antworten vom Benutzer unterbrochen werden können.

Um die Funktion „Warten und Fortfahren“ zu testen, konfigurieren Sie Ihren Bot so, dass er auf Benutzereingaben wartet und einen Stream zu einem Amazon Lex V2-Bot startet. Informationen zum Streamen zu einem Bot finden Sie unter [Verwenden der API, um eine Streaming-Konversation zu starten](#).

Möglicherweise müssen Sie die Option „Warten und Antworten fortsetzen“ deaktivieren. Verwenden Sie den Schalter Aktiv, um festzulegen, ob die Antworten Warten und Fortfahren verwendet werden sollen.

Wait and continue

 Active

You can use the responses below to manage a conversation if the user needs to time to provide information requested by the bot. This functionality is available only in streaming conversations.

Note

Wait and continue ist nur in den folgenden englischen Sprachversionen verfügbar: en-AU, en-GB, en-IN, en-US, en-ZA.

Updates zum Fulfillment-Fortschritt für Ihren Lex V2-Bot konfigurieren

Wenn die Fulfillment-Lambda-Funktion für eine Absicht aufgerufen wird, sendet der Bot keine Antwort, bis die Funktion abgeschlossen ist. Wenn die Ausführung der Lambda-Funktion länger als ein paar Sekunden dauert, denkt der Benutzer möglicherweise, dass der Bot nicht reagiert. Um dieses Problem zu lösen, können Sie Ihren Bot so konfigurieren, dass er Updates an den Benutzer sendet, während die Fulfillment-Lambda-Funktion ausgeführt wird, sodass der Benutzer weiß, dass der Bot weiterhin an seiner Anfrage arbeitet.

Wenn Sie einer Absicht Fulfillment-Updates hinzufügen, reagiert der Bot zu Beginn des Fulfillments und in regelmäßigen Abständen, während der Fulfillment-Vorgang läuft. Wenn Sie die Startantwort konfigurieren, können Sie eine Verzögerung angeben, bis der Bot die Antwort sendet. Auf diese Weise können Sie Fälle unterstützen, in denen die Auftragsabwicklung nicht relativ schnell abgeschlossen ist. Wenn Sie eine Antwort auf Updates konfigurieren, geben Sie die Häufigkeit an, mit der die Updates gesendet werden sollen. Sie konfigurieren auch ein Timeout, um die Zeit zu begrenzen, in der die Fulfillment-Funktion ausgeführt werden muss.

Sie können einem Bot auch Antworten nach der Auftragsabwicklung hinzufügen. Auf diese Weise kann der Bot eine unterschiedliche Antwort senden, je nachdem, ob der Versand erfolgreich ist, fehlschlägt oder eine Zeitüberschreitung eintritt.

Fulfillment-Updates werden nur verwendet, wenn Sie mit einem Bot interagieren, der den [StartConversation](#)Vorgang verwendet. Sie können das Post-Fulfillment-Update verwenden,

wenn Sie mit dem Bot über die [StartConversation](#) Operationen [RecognizeText](#), und [RecognizeUtterance](#) interagieren

Aktualisierungen bei der Auftragsabwicklung

Fulfillment-Updates werden gesendet, während Ihre Lambda-Funktion eine Absicht erfüllt. Wenn Sie Fulfillment-Updates aktivieren, geben Sie eine Startantwort an, die zu Beginn des Fulfillments gesendet wird, und eine Aktualisierungsantwort, die regelmäßig gesendet wird, während der Versand läuft.

Wenn Sie eine Aktualisierungsantwort angeben, geben Sie auch ein Timeout an, das bestimmt, wie lange die Fulfillment-Funktion ausgeführt werden kann. Sie können eine Timeout-Länge von bis zu 15 Minuten (900 Sekunden) angeben.

Wenn Sie Fulfillment-Updates deaktivieren, indem Sie in der Konsole den Wert `active` auf `False` setzen oder die [UpdateIntent](#) Operation [CreateIntent](#) oder verwenden, wird das für die Fulfillment-Updates angegebene Timeout nicht verwendet und stattdessen das Standard-Timeout von 30 Sekunden verwendet.

Wenn bei der Fulfillment-Funktion ein Timeout auftritt, führt Amazon Lex V2 eine der folgenden drei Aktionen aus:

- Die Antwort nach der Erfüllung ist konfiguriert und aktiv — gibt die Timeout-Antwort zurück.
- Die Antwort nach der Erfüllung ist konfiguriert und nicht aktiv — gibt eine Ausnahme zurück.
- Die Antwort nach der Erfüllung ist nicht konfiguriert — es wird eine Ausnahme zurückgegeben.

Antwort starten

Amazon Lex V2 gibt die Startantwort zurück, wenn die Lambda-Fulfillment-Funktion während einer Streaming-Konversation aufgerufen wird. In der Regel wird dem Benutzer mitgeteilt, dass die Erfüllung der Absicht einige Zeit in Anspruch nimmt und dass er warten sollte. Die Startantwort wird nicht zurückgegeben, wenn Sie die `RecognizeUtterance` Operationen `RecognizeText` oder verwenden.

Sie können bis zu fünf Antwortnachrichten angeben. Amazon Lex V2 wählt eine der Nachrichten aus, die dem Benutzer vorgespielt werden soll.

Sie können eine Verzögerung zwischen dem Aufruf der Lambda-Funktion und der Rückgabe der Startantwort konfigurieren. Die Startantwort wird nicht zurückgegeben, wenn die Lambda-Funktion ihre Arbeit abgeschlossen hat, bevor die Verzögerung abgeschlossen ist.

Sie können den `active` Schalter in der Konsole oder in der [FulfillmentUpdatesSpecification](#) Struktur verwenden, um die Startantwort ein- und auszuschalten. Wenn der Wert falsch `active` ist, wird die Startantwort nicht abgespielt.

Antwort aktualisieren

Amazon Lex gibt die Aktualisierungsantwort regelmäßig während einer Streaming-Konversation zurück, während die Lambda-Fulfillment-Funktion ausgeführt wird. Die Aktualisierungsantwort wird nicht abgespielt, wenn Sie die `RecognizeUtterance` Operationen `RecognizeText` oder verwenden. Sie können konfigurieren, wie oft die Antwort auf das Update abgespielt wird. Sie können beispielsweise alle 30 Sekunden eine Antwort auf das Update abspielen, während die Fulfillment-Funktion ausgeführt wird, um den Benutzer darüber zu informieren, dass der Vorgang läuft und dass er weiter warten soll.

Sie können bis zu fünf Aktualisierungsnachrichten angeben. Amazon Lex V2 wählt eine Nachricht aus, die dem Benutzer vorgespielt werden soll. Durch die Verwendung mehrerer Nachrichten wird verhindert, dass sich die Aktualisierungen wiederholen.

Wenn der Benutzer Eingaben per Sprache, DTMF oder Text macht, während die Fulfillment-Lambda-Funktion ausgeführt wird, sendet Amazon Lex V2 die Aktualisierungsantwort an den Benutzer zurück.

Wenn die Lambda-Funktion ihre Arbeit vor Ablauf des ersten Aktualisierungszeitraums abschließt, wird die Aktualisierungsantwort nicht zurückgegeben.

Sie können den `active` Schalter in der Konsole oder in der [FulfillmentUpdatesSpecification](#) Struktur verwenden, um die Aktualisierungsantwort ein- und auszuschalten. Wenn der Wert falsch `active` ist, wird die Aktualisierungsantwort nicht zurückgegeben.

Antwort nach dem Versand

Amazon Lex V2 gibt eine Antwort nach dem Versand zurück, wenn die Fulfillment-Funktion beendet wird. Eine Antwort nach der Erfüllung kann verwendet werden, wenn eine beliebige Absicht erfüllt wird, nicht nur beim Streamen von Konversationen. Die Antwort nach der Erfüllung informiert den Benutzer darüber, dass die Funktion abgeschlossen ist, und informiert über das Ergebnis.

Sie können den `active` Schalter in der Konsole oder in der [PostFulfillmentStatusSpecification](#) Struktur verwenden, um die Antwort nach der Auftragsabwicklung ein- und auszuschalten. Wenn der Wert falsch `active` ist, wird die Antwort nicht abgespielt.

Es gibt drei Arten von Antworten nach der Auftragsabwicklung:

- **Success** — wird zurückgegeben, wenn die Fulfillment-Lambda-Funktion ihre Arbeit erfolgreich abgeschlossen hat. Wenn Antworten nach der Auftragsabwicklung nicht aktiv sind, Amazon Lex V2 führt die nächste konfigurierte Aktion aus.
- **Timeout** — wird zurückgegeben, wenn die Lambda-Funktion ihre Arbeit nicht vor Ablauf des konfigurierten Timeout-Zeitraums abschließt. Wenn Antworten nach der Erfüllung nicht aktiv sind, gibt Amazon Lex V2 eine Ausnahme zurück.
- **Fehler** — wird zurückgegeben, wenn die Lambda-Funktion den Status `Failed` in der Antwort zurückgibt oder wenn Amazon Lex V2 bei der Erfüllung der Absicht auf einen Fehler stößt. Wenn Antworten nach der Erfüllung nicht aktiv sind, gibt Amazon Lex V2 eine Ausnahme zurück.

Sie können bis zu fünf Nachrichten für jeden Typ angeben. Amazon Lex V2 wählt eine der Nachrichten aus, die dem Benutzer vorgespielt werden soll.

Im Gegensatz zu Antworten auf den Versandstart und die Aktualisierung des Versandes werden Antworten nach dem Versand sowohl für Streaming-Konversationen als auch für Nicht-Streaming-Konversationen wiedergegeben.

Sie haben auch die Möglichkeit, diese Nachrichten zu überschreiben, indem Sie die Lambda-Funktion so konfigurieren, dass sie nach der Erfüllung eine Nachricht zurückgibt.

 Note

Wenn die Absicht eine abschließende Antwort hat, wird sie nach der Antwort nach der Erfüllung zurückgegeben.

Beispiel nach der Auftragsabwicklung für Lex V2

Um die Reaktion nach dem Versand besser zu verstehen, nehmen wir als Beispiel einen *BookTrip* Bot, der erstellt wurde, um eine Reise zu planen, mit einer *BookFlight* Absicht, konfiguriert mit einer Fulfillment-Lambda-Funktion, die den Flug des Kunden bei einer Fluggesellschaft reserviert. Sobald die Slots für ausgewählt *BookFlight* wurden, ruft Amazon Lex V2 die Fulfillment-Lambda-

Funktion auf. Während dieses Fulfillment-Prozesses kann eines der folgenden drei Ergebnisse eintreten:

- **Erfolgreich** — Der Flug wurde erfolgreich gebucht.
- **Timeout** — Der Buchungsprozess dauert länger als die konfigurierte Fulfillment-Lambda-Ausführungszeit (z. B. wenn die Fluggesellschaft nicht innerhalb der zugewiesenen Zeit kontaktiert werden kann).
- **Fehlschlag** — Die Buchung schlägt aus einem anderen Grund fehl.

Sie können die Antwort nach der Auftragsabwicklung nutzen, um Ihren Kunden in jeder dieser Situationen eine aussagekräftigere Antwort zu geben. Die Beispiele für jede Situation lauten wie folgt:

- **Erfolgreiche Antwort** — „Wir konnten Ihr Ticket erfolgreich buchen und haben Ihnen eine Bestätigungs-E-Mail gesendet. Bitte zögern Sie nicht, uns über die in dieser E-Mail angegebenen Kontaktinformationen zu kontaktieren, wenn Sie Fragen haben.“
- **Timeout-Antwort** — „Aufgrund des starken Datenverkehrs auf unseren Systemen dauert die Buchung Ihres Tickets länger als erwartet. Wir haben Ihre Anfrage in unserer Warteschlange und haben Ihnen eine E-Mail mit der Referenznummer gesendet, die dieser Anfrage entspricht. Sobald wir das Ticket gebucht haben, senden wir Ihnen eine Reservierungsbestätigung. Bitte zögern Sie nicht, uns über die in dieser E-Mail angegebenen Kontaktinformationen zu kontaktieren, wenn Sie Fragen haben.“

 Note

Wenn Sie keine Timeout-Meldung konfigurieren, wird ein 4XX-Fehler ausgegeben, der dem Anwendungsfall entspricht.

- **Fehlerantwort** — „Leider konnten wir Ihr Ticket nicht buchen. Wir haben eine E-Mail mit Einzelheiten zu dem Problem gesendet, auf das wir bei der Buchung Ihrer Reservierung gestoßen sind.“

Konfiguration von Timeouts für die Erfassung von Benutzereingaben mit einem Lex V2-Bot

Die Amazon Lex V2-Streaming-API ermöglicht es einem Bot, Äußerungen in Benutzereingaben automatisch zu erkennen. Wenn Sie eine Absicht oder einen Slot erstellen, können Sie Aspekte einer

Äußerung konfigurieren, z. B. die maximale Dauer einer Äußerung, eine Zeitüberschreitung beim Warten auf Benutzereingaben oder das Endzeichen für DTMF-Eingaben. Sie können das Verhalten eines Bots an Ihren Anwendungsfall anpassen. Sie können beispielsweise die Anzahl der Ziffern für eine Kreditkartennummer auf 16 beschränken.

Sie können Timeouts auch über Sitzungsattribute konfigurieren, wenn Sie eine Konversation mit einem Bot starten, und sie bei Bedarf in Ihrer Lambda-Funktion überschreiben.

Die Konfigurationsschlüssel für ein Attribut verwenden die folgende Syntax:

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

InputType kann **audio**, **dtmf** oder **text** sein.

Sie können Standardeinstellungen für alle Absichten oder Slots in einem Bot konfigurieren, indem Sie die Absicht oder den Slot-Namen angeben*. Alle Intent- oder Slot-spezifischen Einstellungen haben Vorrang vor den Standardeinstellungen.

Amazon Lex V2 bietet vordefinierte Sitzungsattribute für die Verwaltung der Art und Weise, wie die [StartConversation](#) Operationen mit Text-, Sprach- oder DTMF-Eingaben an Ihren Bot funktionieren. Alle vordefinierten Attribute befinden sich im Namespace `x-amz-lex`.

Sie können Standardeinstellungen für alle Intents, Slots oder Subslots in einem Bot konfigurieren, indem Sie die Absicht oder den Slot-Namen angeben*. Alle Intent- oder Slot-spezifischen Einstellungen haben Vorrang vor den Standardeinstellungen. Verwenden Sie diese Muster für alle unten aufgeführten Timeouts.

Für den Teilsteckplatz eines zusammengesetzten Slots können Sie nach trennen. . Zum Beispiel:

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

Expression	Szenario
Absicht: Steckplatz. SubSlot	Gilt nur für den Untersteckplatz mit dem Namen 'SubSlot' innerhalb des Verbundsteckplatzes mit dem Namen 'Slot'

Expression	Szenario
Absicht: Steckplatz. *	Gilt für jeden Sub-Steckplatz innerhalb eines Verbundsteckplatzes mit dem Namen „Slot“
Absicht: *. SubSlot	Gilt nur für den Untersteckplatz mit dem Namen SubSlot " in einem Verbundsteckplatz
Absicht: *. *	Anwendbar für jeden Sub-Steckplatz in jedem Verbundsteckplatz

So funktioniert das Interrupt-Verhalten in einem Lex V2-Bot

Sie können das Interrupt-Verhalten für den Bot einrichten. Das Attribut wird von Amazon Lex V2 definiert.

Unterbrechung zulassen

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

Definiert, ob der Benutzer die vom Amazon Lex V2-Bot abgespielte Aufforderung unterbrechen kann. Sie können es selektiv ausschalten.

Standard: Wahr

Stellen Sie die Timeouts für die Spracheingabe ein

Mithilfe von Sitzungsattributen können Sie Timeout-Werte für die Sprachinteraktion mit Ihrem Bot festlegen. Die Attribute werden von Amazon Lex V2 definiert. Mit diesen Attributen können Sie angeben, wie lange Amazon Lex V2 wartet, bis ein Kunde seine Rede beendet hat, bevor die Eingabesprache erfasst wird.

Alle diese Attribute befinden sich im `x-amz-lex:audio` Namespace.

Maximale Länge der Äußerung

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

Definiert, wie lange Amazon Lex V2 wartet, bis die Spracheingabe gekürzt und die Sprache an Ihre Anwendung zurückgegeben wird. Sie können die Länge der Eingabe verlängern, wenn Sie lange Antworten erwarten oder wenn Sie den Kunden mehr Zeit für die Bereitstellung von Informationen geben möchten.

Standard: 13.000 Millisekunden (13 Sekunden). Der Höchstwert beträgt 55.000 Millisekunden (55 Sekunden)

Wenn Sie das `max-length-ms` Attribut auf mehr als 55.000 Millisekunden festlegen, wird der Standardwert auf 55.000 Millisekunden gesetzt.

Timeout bei Spracheingabe

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

Wie lange ein Bot wartet, bevor er annimmt, dass der Kunde nicht sprechen wird. Sie können die Zeit in Situationen verlängern, in denen der Kunde möglicherweise mehr Zeit benötigt, um Informationen zu finden oder sich daran zu erinnern, bevor er spricht. Möglicherweise möchten Sie den Kunden Zeit geben, ihre Kreditkarte herauszuholen, damit sie die Nummer eingeben können.

Standard: 4.000 Millisekunden (4 Sekunden)

Timeout für Stille

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

Wie lange ein Bot wartet, nachdem der Kunde aufgehört hat zu sprechen, um anzunehmen, dass die Äußerung beendet ist. Sie können die Zeit in Situationen verlängern, in denen während der Eingabe Ruhephasen erwartet werden.

Standard: 600 Millisekunden (0,6 Sekunden)

Audioeingabe zulassen

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

Sie können dieses Attribut aktivieren, sodass der Bot Benutzereingaben nur über die Audiomodalität akzeptiert. Der Bot akzeptiert keine Audioeingaben, wenn dieses Flag auf False gesetzt ist. Der Wert ist standardmäßig auf true gesetzt.

Standard: Wahr

Timeouts für die Texteingabe

Verwenden Sie das folgende Sitzungsattribut, um anzugeben, wie sich Ihr Bot im Textkonversationsmodus verhält.

Dieses Attribut befindet sich im `x-amz-lex:text` Namespace.

Schwellenwert für das Start-Timeout

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

Wie lange der Bot wartet, bevor er einen Kunden erneut zur Texteingabe auffordert. Sie können die Zeit in Situationen verlängern, in denen Sie dem Kunden mehr Zeit geben möchten, Informationen zu finden oder abzurufen, bevor er eine Texteingabe vornimmt. Möglicherweise möchten Sie Kunden beispielsweise mehr Zeit geben, um Details zu ihrer Bestellung zu finden. Alternativ können Sie den Schwellenwert verringern, um Kunden früher zu benachrichtigen.

Standard: 30.000 Millisekunden (30 Sekunden)

Stellen Sie die Konfiguration für die DTMF-Eingabe ein

Verwenden Sie die folgenden Sitzungsattribute, um anzugeben, wie Ihr Amazon Lex V2-Bot auf DTMF-Eingaben reagiert, wenn Sie eine Audiokonversation verwenden.

Alle diese Attribute befinden sich im `x-amz-lex:dtmf` Namespace.

Löschzeichen

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

Das DTMF-Zeichen, das die gesammelten DTMF-Ziffern löscht und die Eingabe sofort beendet.

Standard: *

Endzeichen

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

Das DTMF-Zeichen, das die Eingabe sofort beendet. Wenn der Benutzer dieses Zeichen nicht drückt, endet die Eingabe nach Ablauf des Timeouts.

Standard: #

Ende-Timeout

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

Wie lange der Bot ab der letzten DTMF-Zeicheneingabe warten sollte, bevor er davon ausgeht, dass die Eingabe abgeschlossen ist.

Standard: 5000 Millisekunden (5 Sekunden)

Maximale Anzahl von DTMF-Ziffern pro Äußerung

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

Die maximale Anzahl von DTMF-Ziffern, die in einer Äußerung zulässig sind. Sie könnten diesen Wert beispielsweise auf 16 festlegen, um die Anzahl der Zeichen zu begrenzen, die für eine Kreditkartennummer eingegeben werden können. Dieser Wert kann nicht erhöht werden.

Standard: 1024 Zeichen

DTMF-Eingabe zulassen

Sie können die Art der Eingabe, die der Bot akzeptieren kann, mithilfe von Sitzungsattributen festlegen. Die Attribute werden von Amazon Lex V2 definiert.

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

Sie können dieses Attribut aktivieren, sodass der Bot Benutzereingaben über die DTMF-Modalität akzeptiert. Der Bot akzeptiert keine DTMF-Eingaben, wenn dieses Flag auf False gesetzt ist. Der Wert ist standardmäßig auf true gesetzt.

Standard: Wahr

Bots in Lex V2 importieren und exportieren

Sie können eine Bot-Definition, ein Bot-Gebietsschema oder ein benutzerdefiniertes Vokabular exportieren und es dann wieder importieren, um eine neue Ressource zu erstellen oder eine vorhandene Ressource in einem Konto zu überschreiben. AWS Sie können beispielsweise einen Bot aus einem Testkonto exportieren und dann eine Kopie des Bots in Ihrem Produktionskonto erstellen. Sie können einen Bot auch von einer AWS Region in eine andere Region kopieren.

Sie können die Ressourcen der exportierten Ressource ändern, bevor Sie sie importieren. Sie können beispielsweise einen Bot exportieren und dann die JSON-Datei für einen Slot bearbeiten, um Äußerungen zur Auslösung von Slot-Werten aus einem bestimmten Slot hinzuzufügen oder zu entfernen. Nachdem Sie die Bearbeitung der Definition abgeschlossen haben, können Sie die geänderte Datei importieren.

Themen

- [Bots aus Lex V2 exportieren](#)
- [Bots in Lex V2 importieren](#)
- [Beim Import oder Export ein Passwort verwenden](#)
- [JSON-Format für den Import und Export von Bots in Lex V2](#)

Bots aus Lex V2 exportieren

Sie exportieren einen Bot, ein Bot-Gebietsschema oder ein benutzerdefiniertes Vokabular mithilfe der Konsole oder der `CreateExport` Operation. Sie geben die zu exportierende Ressource an und können ein optionales Passwort angeben, um die ZIP-Datei zu schützen, wenn Sie einen Export starten. Nachdem Sie die ZIP-Datei heruntergeladen haben, müssen Sie das Kennwort für den Zugriff auf die Datei verwenden, bevor Sie sie verwenden können. Weitere Informationen finden Sie unter [Beim Import oder Export ein Passwort verwenden](#).

Der Export ist ein asynchroner Vorgang. Sobald Sie den Export gestartet haben, können Sie die Konsole oder den `DescribeExport` Vorgang verwenden, um den Fortschritt des Exports zu überwachen. Sobald der Export abgeschlossen ist, zeigt die Konsole oder der `DescribeExport` Vorgang den Status `anCOMPLETED`, und die Konsole lädt die exportierte ZIP-Datei in Ihren Browser herunter. Wenn Sie den `DescribeExport` Vorgang verwenden, stellt Amazon Lex V2 eine vorsignierte Amazon S3 S3-URL bereit, über die Sie die Ergebnisse des Exports herunterladen

können. Die Download-URL ist nur für fünf Minuten verfügbar, aber Sie können eine neue URL erhalten, indem Sie den `DescribeExport` Vorgang erneut aufrufen.

Sie können den Verlauf der Exporte für eine Ressource mit der Konsole oder mit dem `ListExports` Vorgang anzeigen. Die Ergebnisse zeigen die Exporte zusammen mit ihrem aktuellen Status. Ein Export ist sieben Tage lang in der Historie verfügbar.

Wenn Sie die `Draft` Version eines Bots oder eines Bot-Gebietsschemas exportieren, ist es möglich, dass sich die Definition in der JSON-Datei in einem inkonsistenten Zustand befindet, da die `Draft` Version eines Bots oder eines Bot-Gebietsschemas während eines Exports geändert werden kann. Wenn die `Draft` Version während des Exports geändert wird, sind die Änderungen möglicherweise nicht in der Exportdatei enthalten.

Wenn Sie ein Bot-Gebietsschema exportieren, exportiert Amazon Lex alle Informationen, die das Gebietsschema definieren, einschließlich des Gebietsschemas, des benutzerdefinierten Wortschatzes, der Absichten, der Slot-Typen und der Slots.

Wenn Sie einen Bot exportieren, exportiert Amazon Lex alle für den Bot definierten Gebietsschemas, einschließlich der Absichten, Slot-Typen und Slots. Die folgenden Elemente werden nicht mit einem Bot exportiert:

- Bot-Alias
- Mit einem Bot verknüpfter Rollen-ARN
- Mit Bots und Bot-Aliasnamen verknüpfte Tags
- Lambda-Code-Hooks, die mit einem Bot-Alias verknüpft sind

Der Rollen-ARN und die Tags werden als Anforderungsparameter eingegeben, wenn Sie einen Bot importieren. Sie müssen nach dem Import gegebenenfalls Bot-Aliase erstellen und Lambda-Code-Hooks zuweisen.

Sie können einen Export und die zugehörige .zip-Datei mithilfe der Konsole oder des Vorgangs `DeleteExport` entfernen.

Ein Beispiel für das Exportieren eines Bots mithilfe der Konsole finden Sie unter [Einen Lex V2-Bot exportieren \(Konsole\)](#).

Für den Export von Bots in Lex V2 sind IAM-Berechtigungen erforderlich

Um Bots, Bot-Gebietsschemas und benutzerdefinierte Vokabulare zu exportieren, muss der Benutzer, der den Export ausführt, über die folgenden IAM-Berechtigungen verfügen.

API	• Erforderliche IAM-Aktionen	Ressource
CreateExport	• CreateExport	Bot
UpdateExport	• UpdateExport	Bot
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	Bot
DescribeExport für benutzerdefinierte Vokabeln	• DescribeExport • DescribeCustomVocabulary	Bot
DeleteExport	• DeleteExport	Bot
ListExports	• ListExports	*

Eine IAM-Beispielrichtlinie finden Sie unter [Erlaubt einem Benutzer, Bots und Bot-Gebietsschemas zu exportieren](#).

Einen Lex V2-Bot exportieren (Konsole)

Sie können einen Bot aus der Bot-Liste, aus der Versionsliste oder von der Seite mit den Versionsdetails exportieren. Wenn Sie eine Version auswählen, exportiert Amazon Lex V2 diese Version. In den folgenden Anweisungen wird davon ausgegangen, dass Sie den Bot aus der Liste der Bots exportieren. Wenn Sie jedoch mit einer Version beginnen, sind die Schritte dieselben.

Um einen Bot mit der Konsole zu exportieren

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie aus der Liste der Bots den Bot aus, den Sie exportieren möchten.
3. Wählen Sie unter Aktion die Option Exportieren aus.
4. Wählen Sie die Bot-Version, die Plattform und das Exportformat aus.
5. (Optional) Geben Sie ein Passwort für die ZIP-Datei ein. Die Angabe eines Kennworts trägt zum Schutz des Ausgabearchivs bei.
6. Wählen Sie Export aus.

Nachdem Sie den Export gestartet haben, kehren Sie zur Liste der Bots zurück. Verwenden Sie die Liste mit dem Import-/Exportverlauf, um den Fortschritt des Exports zu überwachen. Wenn der Status des Exports abgeschlossen ist, lädt die Konsole die ZIP-Datei automatisch auf Ihren Computer herunter.

Um den Export erneut herunterzuladen, wählen Sie in der import/export Liste den Export und dann Herunterladen aus. Sie können ein Passwort für die heruntergeladene ZIP-Datei angeben.

Um eine Bot-Sprache zu exportieren

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie aus der Liste der Bots den Bot aus, dessen Sprache Sie exportieren möchten.
3. Wählen Sie unter Sprachen hinzufügen die Option Sprachen anzeigen aus.
4. Wählen Sie aus der Liste Alle Sprachen die zu exportierende Sprache aus.
5. Wählen Sie unter Aktion die Option Exportieren aus.
6. Wählen Sie die Bot-Version, die Plattform und das Format aus.

7. (Optional) Geben Sie ein Passwort für die ZIP-Datei ein. Die Angabe eines Kennworts trägt zum Schutz des Ausgabearchivs bei.
8. Wählen Sie Export aus.

Nachdem Sie den Export gestartet haben, kehren Sie zur Liste der Sprachen zurück. Verwenden Sie die Liste mit dem Import-/Exportverlauf, um den Fortschritt des Exports zu überwachen. Wenn der Status des Exports abgeschlossen ist, lädt die Konsole die ZIP-Datei automatisch auf Ihren Computer herunter.

Um den Export erneut herunterzuladen, wählen Sie in der import/export Liste den Export und dann Herunterladen aus. Sie können ein Passwort für die heruntergeladene ZIP-Datei angeben.

Bots in Lex V2 importieren

Um mit der Konsole einen zuvor exportierten Bot, ein Bot-Gebietsschema oder ein benutzerdefiniertes Vokabular zu importieren, geben Sie den Speicherort der Datei auf Ihrem lokalen Computer und das optionale Passwort zum Entsperren der Datei an. Ein Beispiel finden Sie unter [Einen Lex V2-Bot importieren \(Konsole\)](#).

Wenn Sie die API verwenden, erfolgt das Importieren einer Ressource in drei Schritten:

1. Erstellen Sie mithilfe des `CreateUploadUrl` Vorgangs eine Upload-URL. Sie müssen keine Upload-URL erstellen, wenn Sie die Konsole verwenden.
2. Laden Sie die ZIP-Datei hoch, die die Ressourcendefinition enthält.
3. Starten Sie den Import mit dem `StartImport` Vorgang.

Die Upload-URL ist eine vorseignierte Amazon S3 S3-URL mit Schreibberechtigung. Die URL ist nach ihrer Generierung fünf Minuten lang verfügbar. Wenn Sie die ZIP-Datei mit einem Passwort schützen, müssen Sie das Passwort angeben, wenn Sie den Import starten. Weitere Informationen finden Sie unter [Beim Import oder Export ein Passwort verwenden](#).

Ein Import ist ein asynchroner Prozess. Sie können den Fortschritt eines Imports mithilfe der Konsole oder des `DescribeImport` Vorgangs überwachen.

Wenn Sie einen Bot oder ein Bot-Gebietsschema importieren, kann es zu Konflikten zwischen Ressourcennamen in der Importdatei und vorhandenen Ressourcennamen in Amazon Lex V2 kommen. Amazon Lex V2 kann den Konflikt auf drei Arten behandeln:

- Fehler bei Konflikt — Der Import wird gestoppt und es werden keine Ressourcen aus der importierten ZIP-Datei importiert.
- Überschreiben — Amazon Lex V2 importiert alle Ressourcen aus der importierten ZIP-Datei und ersetzt alle vorhandenen Ressourcen durch die Definition aus der Importdatei.
- Anfügen — Amazon Lex V2 importiert alle Ressourcen aus der importierten ZIP-Datei und fügt sie jeder vorhandenen Ressource mit der Definition aus der Importdatei hinzu. Dies ist nur für das Bot-Gebietsschema verfügbar.

Sie können eine Liste der Importe in eine Ressource mithilfe der Konsole oder des `ListImports` Vorgangs anzeigen. Importe bleiben sieben Tage lang in der Liste. Sie können die Konsole oder den `DescribeImport` Vorgang verwenden, um Details zu einem bestimmten Import anzuzeigen.

Sie können einen Import und die zugehörige ZIP-Datei auch mithilfe der Konsole oder des `DeleteImport` Vorgangs entfernen.

Ein Beispiel für das Importieren eines Bots mithilfe der Konsole finden Sie unter [Einen Lex V2-Bot importieren \(Konsole\)](#).

Für den Import sind IAM-Berechtigungen erforderlich

Um Bots, Bot-Gebietsschemas und benutzerdefinierte Vokabulare zu importieren, muss der Benutzer, der den Import ausführt, über die folgenden IAM-Berechtigungen verfügen.

API	Erforderliche IAM-Aktionen	Ressource
CreateUploadUrl	• <code>CreateUploadUrl</code>	*
StartImport für Bot und Bot-Locale	• <code>StartImport</code> • <code>ich bin: PassRole</code> • <code>CreateBot</code> • <code>CreateCustomVocabulary</code> • <code>CreateLocale</code> • <code>CreateIntent</code> • <code>CreateSlot</code> • <code>CreateSlotType</code> • <code>UpdateBot</code>	1. Um einen neuen Bot zu importieren: Bot, Bot-Alias. 2. Um einen vorhandenen Bot zu überschreiben: Bot. 3. Um ein neues Gebietsschema zu importieren: Bot.

API	Erforderliche IAM-Aktionen	Ressource
	• UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary • DeleteLocale • DeleteIntent • DeleteSlot • DeleteSlotType	
StartImport für benutzerdefinierte Vokabulare	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	Bot
DescribeImport	• DescribeImport	Bot
DeleteImport	• DeleteImport	Bot
ListImports	• ListImports	*

Eine IAM-Beispielrichtlinie finden Sie unter [Erlaubt einem Benutzer, Bots und Bot-Gebietsschemas zu importieren](#).

Einen Lex V2-Bot importieren (Konsole)

Um einen Bot mit der Konsole zu importieren

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie unter Aktion die Option Importieren aus.

3. Geben Sie dem Bot unter Eingabedatei einen Namen und wählen Sie dann die ZIP-Datei aus, die die JSON-Dateien enthält, die den Bot definieren.
4. Wenn die .zip-Datei kennwortgeschützt ist, geben Sie das Passwort für die .zip-Datei ein. Der Kennwortschutz des Archivs ist optional, trägt aber zum Schutz des Inhalts bei.
5. Erstellen Sie die IAM-Rolle, die die Berechtigungen für Ihren Bot definiert, oder geben Sie sie ein.
6. Geben Sie an, ob Ihr Bot dem Gesetz zum Schutz der Privatsphäre von Kindern im Internet (Children's Online Privacy Protection Act, COPPA) unterliegt.
7. Geben Sie eine Einstellung für das Leerlauf-Timeout für Ihren Bot an. Wenn Sie keinen Wert angeben, wird der Wert aus der ZIP-Datei verwendet. Wenn die ZIP-Datei keine Timeout-Einstellung enthält, verwendet Amazon Lex V2 die Standardeinstellung von 300 Sekunden (fünf Minuten).
8. (Optional) Fügen Sie Tags für Ihren Bot hinzu.
9. Wählen Sie aus, ob Sie vor dem Überschreiben vorhandener Bots mit demselben Namen warnen möchten. Wenn Sie Warnungen aktivieren und der Bot, den Sie importieren, einen vorhandenen Bot überschreiben würde, erhalten Sie eine Warnung und der Bot wird nicht importiert. Wenn Sie Warnungen deaktivieren, ersetzt der importierte Bot den vorhandenen Bot mit demselben Namen.
10. Wählen Sie Importieren aus.

Nachdem Sie den Import gestartet haben, kehren Sie zur Liste der Bots zurück. Verwenden Sie die Liste mit dem Import-/Exportverlauf, um den Fortschritt des Imports zu überwachen. Wenn der Status des Imports abgeschlossen ist, können Sie den Bot aus der Liste der Bots auswählen, um den Bot zu ändern oder zu erstellen.

Um eine Bot-Sprache zu importieren

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex V2-Konsole zu <https://console.aws.amazon.com/lexv2/Hause>.
2. Wählen Sie aus der Liste der Bots den Bot aus, in den Sie eine Sprache importieren möchten.
3. Wählen Sie unter Sprachen hinzufügen die Option Sprachen anzeigen aus.
4. Wählen Sie unter Aktion die Option Importieren aus.
5. Wählen Sie unter Eingabedatei die Datei aus, die die zu importierende Sprache enthält. Wenn Sie die ZIP-Datei geschützt haben, geben Sie das Passwort im Feld Passwort ein.

6. Wählen Sie unter Sprache die Sprache aus, als die importiert werden soll. Die Sprache muss nicht mit der Sprache in der Importdatei übereinstimmen. Sie können die Absichten von einer Sprache in eine andere kopieren.
7. Wählen Sie in Voice die Amazon Polly Polly-Stimme aus, die für die Sprachinteraktion verwendet werden soll, oder wählen Sie Keine für einen reinen Text-Bot.
8. Geben Sie im Feld Schwellenwert für den Vertrauenswert den Schwellenwert ein, an dem Amazon Lex V2 die `AMAZON.FallbackIntent` `AMAZON.KendraSearchIntent`, die oder beide einfügt, wenn alternative Absichten zurückgegeben werden.
9. Wählen Sie aus, ob vor dem Überschreiben einer vorhandenen Sprache gewarnt werden soll. Wenn Sie Warnungen aktivieren und die Sprache, die Sie importieren, eine bestehende Sprache überschreiben würde, erhalten Sie eine Warnung und die Sprache wird nicht importiert. Wenn Sie Warnungen deaktivieren, ersetzt die importierte Sprache die vorhandene Sprache.
10. Wählen Sie Import, um mit dem Import der Sprache zu beginnen.

Nachdem Sie den Import gestartet haben, kehren Sie zur Liste der Sprachen zurück. Verwenden Sie die Liste mit dem Import-/Exportverlauf, um den Fortschritt des Imports zu überwachen. Wenn der Status des Imports abgeschlossen ist, können Sie aus der Liste der Bots die Sprache auswählen, in der der Bot geändert oder erstellt werden soll.

Beim Import oder Export ein Passwort verwenden

Amazon Lex V2 kann Ihre Exportarchive mit einem Passwort schützen oder Ihre geschützten Importarchive mithilfe der standardmäßigen ZIP-Dateikomprimierung lesen. Sie sollten Ihre Import- und Exportarchive immer mit einem Passwort schützen.

Amazon Lex V2 sendet Ihr Exportarchiv an einen S3-Bucket, und es steht Ihnen mit einer vorsignierten S3-URL zur Verfügung. Die URL ist nur für fünf Minuten verfügbar. Das Archiv steht jedem zur Verfügung, der Zugriff auf die Download-URL hat. Um die Daten im Archiv zu schützen, geben Sie beim Exportieren der Ressource ein Passwort ein. Wenn Sie das Archiv abrufen müssen, nachdem die URL abgelaufen ist, können Sie die Konsole oder den `DescribeExport` Vorgang verwenden, um eine neue URL abzurufen.

Wenn Sie das Kennwort für ein Exportarchiv verlieren, können Sie ein neues Kennwort für eine vorhandene Datei erstellen, indem Sie in der import/export Verlaufstabelle die Option Herunterladen auswählen oder den `UpdateExport` Vorgang verwenden. Wenn Sie in der Verlaufstabelle für einen

Export die Option Herunterladen wählen und kein Passwort angeben, lädt Amazon Lex V2 eine ungeschützte ZIP-Datei herunter.

JSON-Format für den Import und Export von Bots in Lex V2

Sie importieren und exportieren Bots, Bot-Locales oder benutzerdefinierte Vokabulare aus Amazon Lex V2 mithilfe einer ZIP-Datei, die JSON-Strukturen enthält, die die Teile der Ressource beschreiben. Wenn Sie eine Ressource exportieren, erstellt Amazon Lex V2 die .zip-Datei und stellt sie Ihnen mithilfe einer vorsignierten Amazon S3 S3-URL zur Verfügung. Wenn Sie eine Ressource importieren, müssen Sie eine .zip-Datei erstellen, die die JSON-Strukturen enthält, und sie auf eine vorsignierte S3-URL hochladen.

Amazon Lex erstellt die folgende Verzeichnisstruktur in der .zip-Datei, wenn Sie einen Bot exportieren. Wenn Sie ein Bot-Gebietsschema exportieren, wird nur die Struktur unter dem Gebietsschema exportiert. Wenn Sie ein benutzerdefiniertes Vokabular exportieren, wird nur die Struktur unter dem benutzerdefinierten Vokabular exportiert.

```
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
      ...

```

Struktur der Manifestdatei

Die Manifestdatei enthält Metadaten für die Exportdatei.

```

{
  "metadata": {
    "schemaVersion": "1.0",
    "fileFormat": "LexJson",
    "resourceType": "Bot | BotLocale | CustomVocabulary"
  }
}

```

Struktur der Bot-Datei

Die Bot-Datei enthält die Konfigurationsinformationen für den Bot.

```

{
  "name": "BotName",
  "identifier": "identifier",
  "version": "number",
  "description": "description",
  "dataPrivacy": {
    "childDirected": true | false
  },
  "idleSessionTTLInSeconds": seconds
}

```

Struktur der Bot-Locale-Datei

Die Bot-Locale-Datei enthält eine Beschreibung des Gebietsschemas oder der Sprache eines Bots. Wenn Sie einen Bot exportieren, kann die .zip-Datei mehr als eine Bot-Locale-Datei enthalten. Wenn Sie ein Bot-Gebietsschema exportieren, enthält die ZIP-Datei nur ein Gebietsschema.

```

{

```

```

    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural"
    },
    "nluConfidenceThreshold": number
}

```

Struktur der beabsichtigten Datei

Die Absichtsdatei enthält die Konfigurationsinformationen für einen Intent. In der ZIP-Datei gibt es eine Absichtsdatei für jede Absicht in einem bestimmten Gebietsschema.

Im Folgenden finden Sie ein Beispiel für eine JSON-Struktur für die BookCar Absicht im BookTrip Beispiel-Bot. Ein vollständiges Beispiel für die JSON-Struktur für einen Intent finden Sie unter [CreateIntentOperation](#).

```

{
    "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
    }
]
}

```

Struktur der Slot-Datei

Die Slot-Datei enthält die Konfigurationsinformationen für einen Slot in einem Intent. In der ZIP-Datei gibt es für jeden Slot, der für eine Absicht in einem bestimmten Gebietsschema definiert ist, eine Slot-Datei.

Das folgende Beispiel zeigt die JSON-Struktur eines Slots, die es dem Kunden ermöglicht, den Fahrzeugtyp auszuwählen, den er in der BookCar Absicht im BookTrip Beispiel-Bot mieten möchte. Ein vollständiges Beispiel für die JSON-Struktur für einen Slot finden Sie unter [CreateSlot](#) Operation.

```

{
    "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,
}
}

```

Das folgende Beispiel zeigt die JSON-Struktur eines zusammengesetzten Slots.

```

{
  "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)"
}
}

```

Dateistruktur vom Typ Slot

Die Slot-Typ-Datei enthält die Konfigurationsinformationen für einen benutzerdefinierten Slot-Typ, der in einer Sprache oder einem Gebietsschema verwendet wird. In der ZIP-Datei gibt es für jeden benutzerdefinierten Slot-Typ in einem bestimmten Gebietsschema eine Slot-Typ-Datei.

Im Folgenden finden Sie die JSON-Struktur für den Slot-Typ, in der die im BookTrip Beispiel-Bot verfügbaren Fahrzeugtypen aufgeführt sind. Ein vollständiges Beispiel für die JSON-Struktur für einen Slot-Typ finden Sie unter [CreateSlotType](#)Operation.


```
{
  "name": "CarTypeValues",
  "identifier": "T1YUHG9ZR",
  "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
  }
}
```

```
}
}
```

Das folgende Beispiel zeigt die JSON-Struktur für einen zusammengesetzten Slot-Typ.

```
{
  "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
      }
    ]
  }
}
```

Der folgende Slot-Typ verwendet eine benutzerdefinierte Grammatik, um die Äußerungen des Kunden zu verstehen. Weitere Informationen finden Sie unter [Grammatik-Slot-Typ](#).

```
{
  "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"
    }
  }
}
}
}

```

Struktur der benutzerdefinierten Vokabeldatei

Die benutzerdefinierte Vokabeldatei enthält die Einträge in einem benutzerdefinierten Vokabular für eine einzelne Sprache oder ein bestimmtes Gebietsschema. Die ZIP-Datei enthält eine benutzerdefinierte Vokabeldatei für jedes Gebietsschema, das über ein benutzerdefiniertes Vokabular verfügt.

Im Folgenden finden Sie eine benutzerdefinierte Vokabeldatei für einen Bot, der Restaurantbestellungen entgegennimmt. Es gibt eine Datei pro Gebietsschema im 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"  
    }  
  ]  
}
```

Ressourcen in Lex V2 kennzeichnen

Um Ihnen bei der Verwaltung Ihrer Amazon Lex V2-Bots und Bot-Aliase zu helfen, können Sie jeder Ressource Metadaten als Tags zuweisen. Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen. Jedes Tag besteht aus einem Schlüssel und einem Wert.

Mithilfe von Tags können Sie Ihre AWS Ressourcen auf unterschiedliche Weise kategorisieren, z. B. nach Zweck, Eigentümer oder Anwendung. Tags helfen Ihnen bei Folgendem:

- Identifizieren und organisieren Sie Ihre AWS Ressourcen. Viele AWS -Ressourcen unterstützen das Markieren mit Tags (kurz: Tagging). Sie können Ressourcen aus verschiedenen Services dasselbe Tag zuweisen, um anzugeben, dass die Ressourcen identisch sind. Sie können beispielsweise einen Bot und die Lambda-Funktionen, die er verwendet, mit demselben Tag kennzeichnen.
- Zuordnen von Kosten. Sie aktivieren Tags auf dem AWS Fakturierung und Kostenmanagement Dashboard. AWS verwendet die Tags, um Ihre Kosten zu kategorisieren und Ihnen einen monatlichen Kostenverteilungsbericht zu liefern. Für Amazon Lex V2 können Sie die Kosten für jeden Alias mithilfe von Alias-Tags zuordnen. Weitere Informationen finden Sie unter [Use cost allocation tags](#) (Verwendung von Kostenzuordnungs-Tags) im AWS Fakturierung und Kostenmanagement -Benutzerhandbuch.
- Kontrollieren Sie den Zugriff auf Ihre -Ressourcen. Sie können Tags mit Amazon Lex V2 verwenden, um Richtlinien zur Steuerung des Zugriffs auf Amazon Lex V2-Ressourcen zu erstellen. Diese Richtlinien können an eine IAM-Rolle oder einen IAM-Benutzer angehängt werden, um eine Tag-basierte Zugriffskontrolle zu ermöglichen.

Sie können mit Tags arbeiten, indem Sie die AWS Management Console AWS Command Line Interface, oder die Amazon Lex V2-API verwenden.

Markieren Sie Ihre Ressourcen mit der Konsole oder API

Wenn Sie die Amazon Lex V2-Konsole verwenden, können Sie Ressourcen bei ihrer Erstellung taggen oder die Tags später hinzufügen. Sie können die Konsole auch verwenden, um vorhandene Tags zu aktualisieren oder zu entfernen.

Wenn Sie die AWS CLI oder Amazon Lex V2-API verwenden, verwenden Sie die folgenden Operationen, um Tags für Ihre Ressource zu verwalten:

- [CreateBot](#) und [CreateBotAlias](#)— wenden Sie Tags an, wenn Sie einen Bot oder einen Bot-Alias erstellen.
- [ListTagsForResource](#)— die mit einer Ressource verknüpften Tags anzeigen.
- [TagResource](#)— Hinzufügen und Ändern von Tags zu einer vorhandenen Ressource.
- [UntagResource](#)— Tags aus einer Ressource entfernen.

Die folgenden Ressourcen in Amazon Lex V2 unterstützen Tagging:

- Bots — verwenden Sie einen Amazon-Ressourcennamen (ARN) wie den folgenden:
 - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Bot-Aliase — verwende einen ARN wie den folgenden:
 - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

Die `bot-alias-id` Werte `bot-id` und sind alphanumerische Zeichenketten in Großbuchstaben mit einer Länge von 10 Zeichen.

Tag-Einschränkungen bei der Verwendung von Lex V2

Die folgenden grundlegenden Einschränkungen gelten für Tags auf Amazon Lex V2-Ressourcen:

- Maximale Anzahl von Schlüsseln — 50 über die Konsole, 200 über die API
- Maximale Schlüssellänge — 128 Zeichen
- Maximale Länge des Werts — 256 Zeichen
- Gültige Zeichen für Schlüssel und Wert – a–z, A–Z, 0–9, Leerzeichen und die folgenden Zeichen: `_.:/=+-` and `@`
- Bei Schlüsseln und Werten wird die Groß-/Kleinschreibung berücksichtigt.
- Nicht `aws:` als Präfix für Schlüssel verwenden, es ist für die AWS Verwendung reserviert

Ressourcen taggen (Konsole)

Sie können die Konsole verwenden, um Tags für einen Bot oder Bot-Alias zu verwalten. Sie können Tags hinzufügen, wenn Sie die Ressource erstellen, oder Sie können Tags zu vorhandenen Ressourcen hinzufügen, ändern oder entfernen.

So fügen Sie ein Tag hinzu, wenn Sie einen Bot erstellen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie Create bot aus.
3. Wählen Sie unter Bot-Einstellungen konfigurieren im Abschnitt Erweiterte Einstellungen die Option Neues Tag hinzufügen aus. Sie können dem Bot und dem TestBotAlias Alias Tags hinzufügen.
4. Wählen Sie Weiter, um mit der Erstellung Ihres Bots fortzufahren.

So fügen Sie ein Tag hinzu, wenn Sie einen Bot-Alias erstellen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie den Bot aus, dem Sie den Bot-Alias hinzufügen möchten.
3. Wählen Sie im linken Menü Aliase und dann Alias erstellen aus.
4. Wählen Sie unter „Allgemeine Informationen“ unter „Tags“ die Option Neues Tag hinzufügen aus.
5. Wählen Sie Erstellen aus.

So fügen Sie einem vorhandenen Bot ein Tag hinzu, oder entfernen bzw. ändern es:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie den Bot aus, den Sie ändern möchten.
3. Wählen Sie im linken Menü Einstellungen und dann Bearbeiten.
4. Nehmen Sie unter „Tags“ Ihre Änderungen vor.
5. Wählen Sie Speichern, um Ihre Änderungen am Bot zu speichern.

Um ein Tag für einen vorhandenen Alias hinzuzufügen, zu entfernen oder zu ändern

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie den Bot aus, den Sie ändern möchten.

3. Wählen Sie im linken Menü Aliase und dann aus der Aliasliste den Alias aus, den Sie ändern möchten.
4. Wählen Sie unter Aliasdetails unter Tags die Option Tags ändern aus.
5. Nehmen Sie unter Tags verwalten Ihre Änderungen vor.
6. Wählen Sie Speichern, um Ihre Änderungen am Alias zu speichern.

Sicherheit in Amazon Lex V2

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#) . Weitere Informationen zu den Compliance-Programmen, die für Amazon Lex V2 gelten, finden Sie unter [AWS-Services in Umfang nach Compliance-Programm](#) .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Amazon Lex V2 anwenden können. In den folgenden Themen erfahren Sie, wie Sie Amazon Lex V2 konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS-Services nutzen können, mit denen Sie Ihre Amazon Lex V2-Ressourcen überwachen und sichern können.

Themen

- [Datenschutz in Amazon Lex V2](#)
- [Identitäts- und Zugriffsmanagement für Amazon Lex V2](#)
- [Protokollierung und Überwachung in Amazon Lex V2](#)
- [Konformitätsprüfung für Amazon Lex V2](#)
- [Resilienz in Amazon Lex V2](#)
- [Infrastruktursicherheit in Amazon Lex V2](#)
- [Amazon Lex V2 und VPC-Schnittstellen-Endpunkte \(AWS PrivateLink\)](#)

Datenschutz in Amazon Lex V2

Amazon Lex V2 entspricht dem [Modell der AWS gemeinsamen Verantwortung](#), dem der , das Vorschriften und Richtlinien für den Datenschutz umfasst. AWS ist für den Schutz der globalen Infrastruktur verantwortlich, auf der alle AWS Dienste ausgeführt werden. AWS behält die Kontrolle über die auf dieser Infrastruktur gehosteten Daten, einschließlich der Sicherheitskonfigurationen für den Umgang mit Kundeneinhalten und personenbezogenen Daten. AWS Kunden und APN-Partner, die entweder als Datenverantwortliche oder als Datenverarbeiter agieren, sind für alle personenbezogenen Daten verantwortlich, die sie in die AWS Cloud stellen.

Aus Datenschutzgründen empfehlen wir Ihnen, die AWS Kontoanmeldeinformationen zu schützen und individuelle Benutzerkonten mit AWS Identity and Access Management (IAM) einzurichten, sodass jeder Benutzer nur die Berechtigungen erhält, die zur Erfüllung seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden zu schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen innerhalb der AWS Dienste.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu sichern.

Wir empfehlen dringend, in Freitextfeldern wie z. B. im Feld Name keine sensiblen, identifizierenden Informationen wie Kontonummern von Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon Lex V2 oder anderen AWS Diensten über die Konsole AWS CLI, API oder arbeiten AWS SDKs. Alle Daten, die Sie in Amazon Lex V2 oder andere Dienste eingeben, werden möglicherweise zur Aufnahme in Diagnoseprotokolle aufgenommen. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

Weitere Informationen zum Datenschutz enthält der Blog-Beitrag [AWS Shared Responsibility Model and GDPR](#) im AWS -Sicherheitsblog.

Verschlüsselung im Ruhezustand

Amazon Lex V2 verschlüsselt Benutzeräußerungen und andere Informationen, die es speichert.

Themen

- [Beispiele für Äußerungen](#)
- [Sitzungsattribute](#)
- [Attribute anfordern](#)

Beispiele für Äußerungen

Sie können während der Entwicklung eines Bots Beispieläußerungen für jede Absicht und jeden Slot bereitstellen. Sie können auch benutzerdefinierte Werte und Synonyme für Slots bereitstellen. Diese Informationen sind im Ruhezustand verschlüsselt und werden nur verwendet, um den Bot zu erstellen und das Kundenerlebnis zu verbessern.

Sitzungsattribute

Sitzungsattribute enthalten anwendungsspezifische Informationen, die zwischen Amazon Lex V2 und Client-Anwendungen weitergegeben werden. Amazon Lex V2 übergibt Sitzungsattribute an alle für einen Bot konfigurierten AWS Lambda Funktionen. Wenn eine Lambda-Funktion Sitzungsattribute hinzufügt oder aktualisiert, gibt Amazon Lex V2 die neuen Informationen zurück an die Client-Anwendung.

Sitzungsattribute bleiben in einem verschlüsselten Speicher für die Dauer der Sitzung erhalten. Sie können die Sitzung konfigurieren, sodass sie für mindestens eine Minute und höchstens 24 Stunden nach der letzten Äußerung des Benutzers aktiv bleibt. Die Standardsitzungsdauer beträgt 5 Minuten.

Attribute anfordern

Anforderungsattribute enthalten anforderungsspezifische Informationen und gelten nur für die jeweils aktuelle Anforderung. Eine Client-Anwendung verwendet Anforderungsattribute, um zur Laufzeit Informationen an Amazon Lex V2 zu senden.

Verwenden Sie Anforderungsattribute zur Weitergabe von Informationen, die nicht während der ganzen Sitzung erhalten bleiben müssen. Da Anforderungsattribute nicht anforderungsübergreifend erhalten bleiben, werden sie nicht gespeichert.

Verschlüsselung während der Übertragung

Amazon Lex V2 verwendet das HTTPS-Protokoll für die Kommunikation mit Ihrer Client-Anwendung. Es verwendet HTTPS und AWS Signaturen, um mit anderen Diensten wie Amazon Polly und AWS Lambda im Namen Ihrer Anwendung zu kommunizieren.

Identitäts- und Zugriffsmanagement für Amazon Lex V2

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service, den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Amazon Lex V2-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service, das Sie ohne zusätzliche Kosten nutzen können.

Themen

- [Zielgruppe](#)
- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [So funktioniert Amazon Lex V2 mit IAM](#)
- [Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2](#)
- [Beispiele für ressourcenbasierte Richtlinien für Amazon Lex V2](#)
- [AWS verwaltete Richtlinien für Amazon Lex V2](#)
- [Verwenden von serviceverknüpften Rollen für Amazon Lex V2](#)
- [Fehlerbehebung bei Amazon Lex V2: Identität und Zugriff](#)

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Amazon Lex V2 ausführen.

Servicebenutzer — Wenn Sie den Amazon Lex V2-Service für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Da Sie für Ihre Arbeit mehr Amazon Lex V2-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen,

wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Wenn Sie auf eine Funktion in Amazon Lex V2 nicht zugreifen können, finden Sie weitere Informationen unter [Fehlerbehebung bei Amazon Lex V2: Identität und Zugriff](#).

Service-Administrator — Wenn Sie in Ihrem Unternehmen für Amazon Lex V2-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon Lex V2. Es ist Ihre Aufgabe, zu bestimmen, auf welche Funktionen und Ressourcen von Amazon Lex V2 Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM mit Amazon Lex V2 verwenden kann, finden Sie unter [So funktioniert Amazon Lex V2 mit IAM](#).

IAM-Administrator — Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf Amazon Lex V2 zu verwalten. Beispiele für identitätsbasierte Amazon Lex V2-Richtlinien, die Sie in IAM verwenden können, finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2](#)

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter [So melden Sie sich bei Ihrem an AWS-Konto](#) im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode

für die Selbstsignierung von Anforderungen finden Sie unter [AWS Signature Version 4 für API-Anforderungen](#) im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter [Multi-Faktor-Authentifizierung](#) im AWS IAM Identity Center - Benutzerhandbuch und [AWS Multi-Faktor-Authentifizierung \(MFA\) in IAM](#) im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie ein AWS-Konto erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter [Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern](#) im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter [Was ist IAM Identity Center?](#) im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein [IAM-Benutzer](#) ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter [Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern](#) im IAM-Benutzerhandbuch.

Eine [IAM-Gruppe](#) ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdmins und dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter [Anwendungsfälle für IAM-Benutzer](#) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie [von einer Benutzer- zu einer IAM-Rolle \(Konsole\) wechseln](#). Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter [Methoden für die Übernahme einer Rolle](#) im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter [Erstellen von Rollen für externe](#)

[Identitätsanbieter \(Verbund\)](#) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter [Berechtigungssätze](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
- Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).
- Servicerolle – Eine Servicerolle ist eine [IAM-Rolle](#), die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter [Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service](#) im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und

gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden](#).

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter [Übersicht über JSON-Richtlinien](#) im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter [Auswählen zwischen verwalteten und eingebundenen Richtlinien](#) im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#). Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF
Weitere Informationen finden Sie unter [Übersicht über ACLs die Zugriffskontrollliste \(ACL\)](#) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- **Berechtigungsgrenzen** – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter [Berechtigungsgrenzen für IAM-Entitäten](#) im IAM-Benutzerhandbuch.
- **Dienststeuerungsrichtlinien (SCPs)** — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter [Richtlinien zur Servicesteuerung](#) im AWS Organizations Benutzerhandbuch.
- **Ressourcenkontrollrichtlinien (RCPs)** — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter [Resource Control Policies \(RCPs\)](#) im AWS Organizations Benutzerhandbuch.

- **Sitzungsrichtlinien** – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter [Bewertungslogik für Richtlinien](#).

So funktioniert Amazon Lex V2 mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf Amazon Lex V2 zu verwalten, sollten Sie sich darüber informieren, welche IAM-Funktionen für Amazon Lex V2 verfügbar sind.

IAM-Funktionen, die Sie mit Amazon Lex V2 verwenden können

IAM-Feature	Amazon Lex V2-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Ja
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Nein
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Nein

IAM-Feature	Amazon Lex V2-Unterstützung
Prinzipalberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Teilweise

Einen allgemeinen Überblick darüber, wie Amazon Lex V2 und andere AWS Services mit den meisten IAM-Funktionen funktionieren, finden Sie im [IAM-Benutzerhandbuch unter AWS Services, die mit IAM funktionieren](#).

Identitätsbasierte Richtlinien für Amazon Lex V2

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter [Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien](#) im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der [IAM-Referenz für JSON-Richtlinienelemente](#) im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2

Beispiele für identitätsbasierte Amazon Lex V2-Richtlinien finden Sie unter. [Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2](#)

Ressourcenbasierte Richtlinien in Amazon Lex V2

Unterstützt ressourcenbasierte Richtlinien: Ja

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie [einen Prinzipal angeben](#). Zu den Prinzipalen können Benutzer, Rollen, Verbundbenutzer oder AWS-Services gehören.

Sie können keine konto- oder regionsübergreifenden Richtlinien mit verwenden. Wenn Sie eine Richtlinie für eine Ressource mit einem konto- oder regionsübergreifenden ARN erstellen, wird ein Fehler zurückgegeben.

Der Dienst unterstützt ressourcenbasierte Richtlinien, die als Bot-Richtlinie und Bot-Alias-Richtlinie bezeichnet werden und an einen Bot oder Bot-Alias angehängt sind. Diese Richtlinien definieren, welche Principals Aktionen mit dem Bot oder Bot-Alias ausführen können.

Aktionen können nur für bestimmte Ressourcen verwendet werden. Beispielsweise kann die `UpdateBot` Aktion nur für Bot-Ressourcen verwendet werden, die `UpdateBotAlias` Aktion kann nur für Bot-Alias-Ressourcen verwendet werden. Wenn Sie in einer Richtlinie eine Aktion angeben, die nicht für die in der Richtlinie angegebene Ressource verwendet werden kann, wird ein Fehler zurückgegeben. Eine Liste der Aktionen und der Ressourcen, mit denen sie verwendet werden können, finden Sie in der folgenden Tabelle.

Aktion	Unterstützt ressourcenbasierte Richtlinien	Ressource
<code>BuildBotLocale</code>	Unterstützt	<code>BotId</code>
<code>CreateBot</code>	Nein	
<code>CreateBotAlias</code>	Nein	
<code>CreateBotChannel</code> [Nur mit Genehmigung]	Unterstützt	<code>BotId</code>
<code>CreateBotLocale</code>	Unterstützt	<code>BotId</code>
<code>CreateBotVersion</code>	Unterstützt	<code>BotId</code>

Aktion	Unterstützt ressourcenbasierte Richtlinien	Ressource
CreateExport	Unterstützt	BotId
CreateIntent	Unterstützt	BotId
CreateResourcePolicy	Unterstützt	BotId, BotAliasId
CreateSlot	Unterstützt	BotId
CreateSlotType	Unterstützt	BotId
CreateUploadUrl	Nein	
DeleteBot	Unterstützt	BotId, BotAliasId
DeleteBotAlias	Unterstützt	BotAliasId
DeleteBotChannel [Nur mit Genehmigung]	Unterstützt	BotId
DeleteBotLocale	Unterstützt	BotId
DeleteBotVersion	Unterstützt	BotId
DeleteExport	Unterstützt	BotId
DeleteImport	Unterstützt	BotId
DeleteIntent	Unterstützt	BotId
DeleteResourcePolicy	Unterstützt	BotId, BotAliasId
DeleteSession	Unterstützt	BotAliasId
DeleteSlot	Unterstützt	BotId
DeleteSlotType	Unterstützt	BotId
DescribeBot	Unterstützt	BotId

Aktion	Unterstützt ressourcenbasierte Richtlinien	Ressource
DescribeBotAlias	Unterstützt	BotAliasId
DescribeBotChannel [Nur mit Genehmigung]	Unterstützt	BotId
DescribeBotLocale	Unterstützt	BotId
DescribeBotVersion	Unterstützt	BotId
DescribeExport	Unterstützt	BotId
DescribeImport	Unterstützt	BotId
DescribeIntent	Unterstützt	BotId
DescribeResourcePolicy	Unterstützt	BotId, BotAliasId
DescribeSlot	Unterstützt	BotId
DescribeSlotType	Unterstützt	BotId
GetSession	Unterstützt	BotAliasId
ListBotAliases	Unterstützt	BotId
ListBotChannels [Nur mit Genehmigung]	Unterstützt	BotId
ListBotLocales	Unterstützt	BotId
ListBots	Nein	
ListBotVersions	Unterstützt	BotId
ListBuiltInIntents	Nein	
ListBuiltInSlotTypes	Nein	
ListExports	Nein	

Aktion	Unterstützt ressourcenbasierte Richtlinien	Ressource
ListImports	Nein	
ListIntents	Unterstützt	BotId
ListSlots	Unterstützt	BotId
ListSlotTypes	Unterstützt	BotId
PutSession	Unterstützt	BotAliasId
RecognizeText	Unterstützt	BotAliasId
RecognizeUtterance	Unterstützt	BotAliasId
StartConversation	Unterstützt	BotAliasId
StartImport	Unterstützt	BotId, BotAliasId
TagResource	Nein	
UpdateBot	Unterstützt	BotId
UpdateBotAlias	Unterstützt	BotAliasId
UpdateBotLocale	Unterstützt	BotId
UpdateBotVersion	Unterstützt	BotId
UpdateExport	Unterstützt	BotId
UpdateIntent	Unterstützt	BotId
UpdateResourcePolicy	Unterstützt	BotId, BotAliasId
UpdateSlot	Unterstützt	BotId
UpdateSlotType	Unterstützt	BotId
UntagResource	Nein	

Informationen zum Anhängen einer ressourcenbasierten Richtlinie an einen Bot oder Bot-Alias finden Sie unter. [Beispiele für ressourcenbasierte Richtlinien für Amazon Lex V2](#)

Beispiele für ressourcenbasierte Richtlinien in Amazon Lex V2

Beispiele für ressourcenbasierte Amazon Lex V2-Richtlinien finden Sie unter. [Beispiele für ressourcenbasierte Richtlinien für Amazon Lex V2](#)

Politische Maßnahmen für Amazon Lex V2

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Amazon Lex V2-Aktionen finden Sie unter [Von Amazon Lex V2 definierte Aktionen](#) in der Service Authorization Reference.

Richtlinienaktionen in Amazon Lex V2 verwenden das folgende Präfix vor der Aktion:

```
lex
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [  
  "lex:action1",  
  "lex:action2"  
]
```

Beispiele für identitätsbasierte Amazon Lex V2-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2](#)

Richtlinienressourcen für Amazon Lex V2

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein `Resource` oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen [Amazon-Ressourcennamen \(ARN\)](#) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*" 
```

Eine Liste der Amazon Lex V2-Ressourcentypen und ihrer ARNs Typen finden Sie unter [Von Amazon Lex V2 definierte Ressourcen](#) in der Service Authorization Reference. Informationen darüber, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter [Von Amazon Lex V2 definierte Aktionen](#).

Beispiele für identitätsbasierte Amazon Lex V2-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2](#)

Schlüssel für Richtlinienbedingungen für Amazon Lex V2

Unterstützt dienstspezifische Richtlinien-Bedingungsschlüssel: Nein

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition block`) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die [Bedingungsoperatoren](#) verwenden, z. B. `ist gleich` oder `kleiner als`, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die Bedingung mithilfe einer logischen OR-Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter [IAM-Richtlinienelemente: Variablen und Tags](#) im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter [Kontextschlüssel für AWS globale Bedingungen](#) im IAM-Benutzerhandbuch.

Eine Liste der Amazon Lex V2-Bedingungsschlüssel finden Sie unter [Bedingungsschlüssel für Amazon Lex V2](#) in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter [Von Amazon Lex V2 definierte Aktionen](#).

Beispiele für identitätsbasierte Amazon Lex V2-Richtlinien finden Sie unter [Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2](#)

Zugriffskontrolllisten (ACLs) in Amazon Lex V2

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle (ABAC) mit Amazon Lex V2

Unterstützt ABAC (Tags in Richtlinien): Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In werden AWS diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungelement einer [Richtlinie Tag-Informationen](#) an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter [Definieren von Berechtigungen mit ABAC-Autorisierung](#) im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe [Attributbasierte Zugriffskontrolle \(ABAC\)](#) verwenden im IAM-Benutzerhandbuch.

Temporäre Anmeldeinformationen mit Amazon Lex V2 verwenden

Unterstützt temporäre Anmeldeinformationen: Nein

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services [funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit](#) IAM.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter [Wechseln von einer Benutzerrolle zu einer IAM-Rolle \(Konsole\)](#) im IAM-Benutzerhandbuch.

Mithilfe der AWS API, der AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen in IAM](#).

Serviceübergreifende Prinzipalberechtigungen für Amazon Lex V2

Unterstützt Forward Access Sessions (FAS): Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).

Servicerollen für Amazon Lex V2

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine [IAM-Rolle](#), die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter [Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service](#) im IAM-Benutzerhandbuch.

Warning

Das Ändern der Berechtigungen für eine Servicerolle kann die Funktionalität von Amazon Lex V2 beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon Lex V2 eine Anleitung dazu bietet.

Servicebezogene Rollen für Amazon Lex V2

Unterstützt serviceverknüpfte Rollen: Ja

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter [AWS -Services, die mit IAM funktionieren](#). Suchen Sie in der Tabelle nach einem Service mit einem Yes in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon Lex V2

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Amazon Lex V2-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe der AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) oder ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter [Erstellen von IAM-Richtlinien \(Konsole\)](#) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Amazon Lex V2 definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Lex V2](#) in der Service Authorization Reference.

Themen

- [Bewährte Methoden für Richtlinien](#)
- [Verwenden der Amazon Lex V2-Konsole](#)
- [Erlauben Sie Benutzern, einem Bot Funktionen hinzuzufügen](#)
- [Erlaubt Benutzern, Kanäle zu einem Bot hinzuzufügen](#)
- [Erlauben Sie Benutzern, Bots zu erstellen und zu aktualisieren](#)
- [Erlauben Sie Benutzern, den Automated Chatbot Designer zu verwenden](#)
- [Erlaubt Benutzern die Verwendung eines AWS KMS Schlüssels zum Verschlüsseln und Entschlüsseln von Dateien](#)

- [Erlaubt Benutzern das Löschen von Bots](#)
- [Erlauben Sie Benutzern, eine Konversation mit einem Bot zu führen](#)
- [Erlauben Sie einem bestimmten Benutzer, ressourcenbasierte Richtlinien zu verwalten](#)
- [Erlaubt einem Benutzer, Bots und Bot-Gebietsschemas zu exportieren](#)
- [Erlaubt einem Benutzer, ein benutzerdefiniertes Vokabular zu exportieren](#)
- [Erlaubt einem Benutzer, Bots und Bot-Gebietsschemas zu importieren](#)
- [Erlaubt einem Benutzer, ein benutzerdefiniertes Vokabular zu importieren](#)
- [Erlauben Sie einem Benutzer, einen Bot von Amazon Lex zu Amazon Lex V2 zu migrieren](#)
- [Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer](#)
- [Erlauben Sie einem Benutzer, einen Konversationsfluss mit Visual Conversation Builder in Amazon Lex V2 zu zeichnen](#)
- [Erlaubt Benutzern, Bot-Replikate zu erstellen und anzusehen, sie jedoch nicht zu löschen](#)

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Amazon Lex V2-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder diese löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter [AWS -verwaltete Richtlinien](#) oder [AWS -verwaltete Richtlinien für Auftrags-Funktionen](#) im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter [Richtlinien und Berechtigungen in IAM](#) im IAM-Benutzerhandbuch.

- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter [IAM-JSON-Richtlinienelemente: Bedingung](#) im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter [Richtlinienvvalidierung mit IAM Access Analyzer](#) im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter [Sicherer API-Zugriff mit MFA](#) im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter [Bewährte Methoden für die Sicherheit in IAM](#) im IAM-Benutzerhandbuch.

Verwenden der Amazon Lex V2-Konsole

Um auf die Amazon Lex V2-Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon Lex V2-Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Amazon Lex V2-Konsole weiterhin verwenden können, benötigen Benutzer Konsolenzugriff. Weitere Informationen zum Erstellen eines Benutzers mit Konsolenzugriff finden Sie unter [Erstellen eines IAM-Benutzers in Ihrem AWS Konto](#) im IAM-Benutzerhandbuch.

Erlauben Sie Benutzern, einem Bot Funktionen hinzuzufügen

Dieses Beispiel zeigt eine Richtlinie, die es IAM-Benutzern ermöglicht, einem Amazon Lex V2-Bot Amazon Comprehend-, Stimmungsanalyse- und Amazon Kendra Kendra-Abfrageberechtigungen hinzuzufügen.

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*"
    }
  ]
}
```

Erlaubt Benutzern, Kanäle zu einem Bot hinzuzufügen

Dieses Beispiel ist eine Richtlinie, die es IAM-Benutzern ermöglicht, einem Bot einen Messaging-Kanal hinzuzufügen. Ein Benutzer muss über diese Richtlinie verfügen, bevor er einen Bot auf einer Messaging-Plattform einsetzen kann.

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*"
    }
  ]
}
```

Erlauben Sie Benutzern, Bots zu erstellen und zu aktualisieren

Dieses Beispiel zeigt eine Beispielrichtlinie, die es IAM-Benutzern ermöglicht, jeden beliebigen Bot zu erstellen und zu aktualisieren. Die Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder mithilfe der AWS CLI AWS OR-API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateBot",
        "lex:UpdateBot",
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*"]
    }
  ]
}
```

```
}
```

Erlauben Sie Benutzern, den Automated Chatbot Designer zu verwenden

Dieses Beispiel zeigt eine Beispielrichtlinie, die es IAM-Benutzern ermöglicht, den Automated Chatbot Designer auszuführen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::<customer-bucket>/<bucketName>",
        # Resource should point to the bucket or an explicit folder.
        # Provide this to read the entire bucket
        "arn:aws:s3:::<customer-bucket>/<bucketName>/*",
        # Provide this to read a specific folder
        "arn:aws:s3:::<customer-bucket>/<bucketName>/<pathFormat>/*"
      ]
    },
    {
      # Use this if your S3 bucket is encrypted with a KMS key.
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:<Region>:<customerAccountId>:key/<kmsKeyId>"
      ]
    }
  ]
}
```

Erlaubt Benutzern die Verwendung eines AWS KMS Schlüssels zum Verschlüsseln und Entschlüsseln von Dateien

Dieses Beispiel zeigt eine Beispielrichtlinie, die es IAM-Benutzern ermöglicht, einen vom AWS KMS Kunden verwalteten Schlüssel zum Verschlüsseln und Entschlüsseln von Daten zu verwenden.

```
{
  "Version": "2012-10-17",
  "Id": "sample-policy",
  "Statement": [
    {
      "Sid": "Allow Lex access",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": [
        # If the key is for encryption
        "kms:Encrypt",
        "kms:GenerateDataKey"
        # If the key is for decryption
        "kms:Decrypt"
      ],
      "Resource": "*"
    }
  ]
}
```

Erlaubt Benutzern das Löschen von Bots

Dieses Beispiel zeigt eine Beispielpolitlinie, die es IAM-Benutzern ermöglicht, jeden Bot zu löschen. Die Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder mithilfe der AWS CLI AWS OR-API.

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:Region:123412341234:bot/*",
                 "arn:aws:lex:Region:123412341234:bot-alias/*"]
  }
]
}

```

Erlauben Sie Benutzern, eine Konversation mit einem Bot zu führen

Dieses Beispiel zeigt eine Beispielrichtlinie, die es IAM-Benutzern ermöglicht, eine Konversation mit einem beliebigen Bot zu führen. Die Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder mithilfe der AWS CLI AWS OR-API.

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:Region:123412341234:bot-alias/*"
    }
  ]
}

```

Erlauben Sie einem bestimmten Benutzer, ressourcenbasierte Richtlinien zu verwalten

Das folgende Beispiel erteilt einem bestimmten Benutzer die Erlaubnis, die ressourcenbasierten Richtlinien zu verwalten. Es ermöglicht den Konsolen- und API-Zugriff auf die Richtlinien, die Bots und Bot-Aliassen zugeordnet sind.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ResourcePolicyEditor",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ResourcePolicyEditor"
      },
      "Action": [
        "lex:CreateResourcePolicy",
        "lex:UpdateResourcePolicy",
        "lex>DeleteResourcePolicy",
        "lex:DescribeResourcePolicy"
      ]
    }
  ]
}
```

Erlaubt einem Benutzer, Bots und Bot-Gebietsschemas zu exportieren

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, einen Bot oder ein Bot-Gebietsschema zu erstellen, zu aktualisieren und zu exportieren.

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:Region:123456789012:bot/*"]
}
]
}

```

Erlaubt einem Benutzer, ein benutzerdefiniertes Vokabular zu exportieren

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, ein benutzerdefiniertes Vokabular aus einem Bot-Gebietsschema zu exportieren.

JSON

```

{"Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateExport",
        "lex:UpdateExport",
        "lex:DescribeExport",
        "lex:DescribeCustomVocabulary"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
    }
  ]
}

```

Erlaubt einem Benutzer, Bots und Bot-Gebietsschemas zu importieren

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, einen Bot oder ein Bot-Gebietsschema zu importieren und den Status eines Imports zu überprüfen.


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateUploadUrl",
        "lex:StartImport",
        "lex:DescribeImport",
        "lex:CreateBot",
        "lex:UpdateBot",
        "lex>DeleteBot",
        "lex:CreateBotLocale",
        "lex:UpdateBotLocale",
        "lex>DeleteBotLocale",
        "lex:CreateIntent",
        "lex:UpdateIntent",
        "lex>DeleteIntent",
        "lex:CreateSlotType",
        "lex:UpdateSlotType",
        "lex>DeleteSlotType",
        "lex:CreateSlot",
        "lex:UpdateSlot",
        "lex>DeleteSlot",
        "lex:CreateCustomVocabulary",
        "lex:UpdateCustomVocabulary",
        "lex>DeleteCustomVocabulary",
        "iam:PassRole",
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:lex:Region:123456789012:bot/*",
        "arn:aws:lex:Region:123456789012:bot-alias/*"
      ]
    }
  ]
}
```

Erlaubt einem Benutzer, ein benutzerdefiniertes Vokabular zu importieren

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, ein benutzerdefiniertes Vokabular in ein Bot-Gebietsschema zu importieren.

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:Region:123456789012:bot/*"
      ]
    }
  ]
}
```

Erlauben Sie einem Benutzer, einen Bot von Amazon Lex zu Amazon Lex V2 zu migrieren

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, mit der Migration eines Bots von Amazon Lex zu Amazon Lex V2 zu beginnen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "startMigration",
      "Effect": "Allow",
      "Action": "lex:StartMigration",
      "Resource": "arn:aws:lex:>Region<:>123456789012<:bot:*"
    },
  ],
}
```

```

{
  "Sid": "passRole",
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "arn:aws:iam::>123456789012<:role/>v2 bot role<"
},
{
  "Sid": "allowOperations",
  "Effect": "Allow",
  "Action": [
    "lex:CreateBot",
    "lex:CreateIntent",
    "lex:UpdateSlot",
    "lex:DescribeBotLocale",
    "lex:UpdateBotAlias",
    "lex:CreateSlotType",
    "lex>DeleteBotLocale",
    "lex:DescribeBot",
    "lex:UpdateBotLocale",
    "lex:CreateSlot",
    "lex>DeleteSlot",
    "lex:UpdateBot",
    "lex>DeleteSlotType",
    "lex:DescribeBotAlias",
    "lex:CreateBotLocale",
    "lex>DeleteIntent",
    "lex:StartImport",
    "lex:UpdateSlotType",
    "lex:UpdateIntent",
    "lex:DescribeImport",
    "lex:CreateCustomVocabulary",
    "lex:UpdateCustomVocabulary",
    "lex>DeleteCustomvocabulary",
    "lex:DescribeCustomVocabulary",
    "lex:DescribeCustomVocabularyMetadata"
  ],
  "Resource": [
    "arn:aws:lex:>Region<:>123456789012<:bot/*",
    "arn:aws:lex:>Region<:>123456789012<:bot-alias/*/*"
  ]
},
{
  "Sid": "showBots",
  "Effect": "Allow",

```

```

        "Action": [
            "lex:CreateUploadUrl",
            "lex:ListBots"
        ],
        "Resource": "*"
    }
}

```

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie beinhaltet Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der OR-API. AWS CLI AWS

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "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": "*"
    }
  ]
}

```

Erlauben Sie einem Benutzer, einen Konversationsfluss mit Visual Conversation Builder in Amazon Lex V2 zu zeichnen

Die folgende IAM-Berechtigungsrichtlinie ermöglicht es einem Benutzer, den Konversationsfluss mit Visual Conversation Builder in Amazon Lex V2 zu zeichnen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:UpdateIntent ",
        "lex:DescribeIntent "
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
    }
  ]
}

```

Erlaubt Benutzern, Bot-Replikate zu erstellen und anzusehen, sie jedoch nicht zu löschen

Sie können einer IAM-Rolle die folgenden Berechtigungen zuweisen, damit sie nur Bot-Replikate erstellen und anzeigen kann. Wenn Sie diese Option weglassen, verhindern Sie `lex:DeleteBotReplica`, dass die Rolle Bot-Replikate löscht. Weitere Informationen finden Sie unter [Berechtigungen zur Replikation von Bots und zur Verwaltung von Bot-Repliken in Lex V2](#).

```

{
  "Version": "2012-10-17",
  "Statement": [

```

```

    {
      "Effect": "Allow",
      "Action": [
        "lex:CreateBotReplica",
        "lex:DescribeBotReplica",
        "lex:ListBotReplica",
        "lex:ListBotVersionReplicas",
        "lex:ListBotAliasReplicas",
      ],
      "Resource": [
        "arn:aws:lex:*:*:bot/*",
        "arn:aws:lex:*:*:bot-alias/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole"
      ],
      "Resource": [
        "arn:aws:iam:*:*:role/aws-service-role/replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
      ],
      "Resource": [
        "arn:aws:iam:*:*:role/aws-service-role/replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "lexv2.amazonaws.com"
        }
      }
    }
  ]
}

```

Beispiele für ressourcenbasierte Richtlinien für Amazon Lex V2

Eine ressourcenbasierte Richtlinie ist an eine Ressource angehängt, z. B. an einen Bot oder einen Bot-Alias. Mit einer ressourcenbasierten Richtlinie können Sie angeben, wer Zugriff auf die Ressource hat und welche Aktionen sie mit ihr ausführen können. Sie können beispielsweise ressourcenbasierte Richtlinien hinzufügen, die es einem Benutzer ermöglichen, einen bestimmten Bot zu ändern oder einem Benutzer die Verwendung von Laufzeitoperationen für einen bestimmten Bot-Alias zu ermöglichen.

Wenn Sie eine ressourcenbasierte Richtlinie verwenden, können Sie anderen AWS Diensten den Zugriff auf Ressourcen in Ihrem Konto ermöglichen. Sie können Amazon Connect beispielsweise den Zugriff auf einen Bot gestatten.

Informationen zum Erstellen eines Bot oder Bot-Alias finden Sie unter [Arbeiten mit Amazon Lex V2-Bots](#).

Themen

- [Verwenden Sie die Konsole, um eine ressourcenbasierte Richtlinie anzugeben](#)
- [Verwenden Sie die API, um eine ressourcenbasierte Richtlinie anzugeben](#)
- [Erlaubt einer IAM-Rolle, einen Bot zu aktualisieren und Bot-Aliase aufzulisten](#)
- [Erlauben Sie einem Benutzer, eine Konversation mit einem Bot zu führen](#)
- [Erlauben Sie einem AWS Service, einen bestimmten Amazon Lex V2-Bot zu verwenden](#)

Verwenden Sie die Konsole, um eine ressourcenbasierte Richtlinie anzugeben

Sie können die Konsole verwenden, um die ressourcenbasierten Richtlinien für Ihre Bots und Bot-Aliase zu verwalten. Sie geben die JSON-Struktur einer Richtlinie ein und die Konsole ordnet sie der Ressource zu. Wenn einer Ressource bereits eine Richtlinie zugeordnet ist, können Sie die Richtlinie in der Konsole anzeigen und ändern.

Wenn Sie eine Richtlinie mit dem Richtlinien-Editor speichern, überprüft die Konsole die Syntax der Richtlinie. Wenn die Richtlinie Fehler enthält, z. B. dass ein Benutzer nicht existiert oder eine Aktion, die von der Ressource nicht unterstützt wird, wird ein Fehler zurückgegeben und die Richtlinie wird nicht gespeichert.

Im Folgenden wird der ressourcenbasierte Policy-Editor für einen Bot in der Konsole gezeigt. Der Policy-Editor für einen Bot-Alias ist ähnlich.

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

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Sid": "botRunners",  
6       "Effect": "Allow",  
7       "Principal": {  
8         "AWS": "arn:aws:iam::123456789012:user/botRunner"  
9       },  
10      "Action": [  
11        "lex:RecognizeText",  
12        "lex:RecognizeUtterance",  
13        "lex:StartConversaion"  
14      ],  
15      "Resource": [  
16        "arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"  
17      ]  
18    }  
19  ]  
20 }
```

Cancel

Save

Um den Policy-Editor für einen Bot zu öffnen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Bot-Liste den Bot aus, dessen Richtlinie Sie bearbeiten möchten.
3. Wählen Sie im Abschnitt Ressourcenbasierte Richtlinie die Option Bearbeiten aus.

Um den Policy-Editor für einen Bot-Alias zu öffnen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie aus der Bot-Liste den Bot aus, der den Alias enthält, den Sie bearbeiten möchten.
3. Wählen Sie im linken Menü Aliase und dann den Alias aus, den Sie bearbeiten möchten.
4. Wählen Sie im Abschnitt Ressourcenbasierte Richtlinie die Option Bearbeiten aus.

Verwenden Sie die API, um eine ressourcenbasierte Richtlinie anzugeben

Sie können API-Operationen verwenden, um die ressourcenbasierten Richtlinien für Ihre Bots und Bot-Aliase zu verwalten. Es gibt Operationen zum Erstellen, Aktualisieren und Löschen von Richtlinien.

[CreateResourcePolicy](#)

Fügt einem Bot oder Bot-Alias eine neue Ressourcenrichtlinie mit den angegebenen Richtlinienanweisungen hinzu.

[CreateResourcePolicyStatement](#)

Fügt einem Bot oder Bot-Alias eine neue Ressourcenrichtlinien-Anweisung hinzu.

[DeleteResourcePolicy](#)

Entfernt eine Ressourcenrichtlinie von einem Bot oder Bot-Alias.

[DeleteResourcePolicyStatement](#)

Entfernt eine Ressourcenrichtlinien-Erklärung von einem Bot oder Bot-Alias.

[DescribeResourcePolicy](#)

Ruft eine Ressourcenrichtlinie und deren Revision ab.

UpdateResourcePolicy

Ersetzt die bestehende Ressourcenrichtlinie für einen Bot oder Bot-Alias durch eine neue.

Beispiele

Java

Das folgende Beispiel zeigt, wie die ressourcenbasierten Richtlinienoperationen verwendet werden, um eine ressourcenbasierte Richtlinie zu verwalten.

```

/*
 * 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 createStateRequest =
    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(createStateRequest);

/*
 * 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);

```

Erlaubt einer IAM-Rolle, einen Bot zu aktualisieren und Bot-Aliase aufzulisten

Das folgende Beispiel gewährt einer bestimmten IAM-Rolle Berechtigungen zum Aufrufen von Amazon Lex V2-API-Operationen zur Modellerstellung, um einen vorhandenen Bot zu ändern. Der Benutzer kann Aliase für einen Bot auflisten und den Bot aktualisieren, aber den Bot oder die Bot-Aliase nicht löschen.

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:Region:123456789012:bot/MYBOT"
        ]
      }
    ]
  }
}

```

Erlauben Sie einem Benutzer, eine Konversation mit einem Bot zu führen

Das folgende Beispiel gewährt einem bestimmten Benutzer die Erlaubnis, Amazon Lex V2-Runtime-API-Operationen für einen einzelnen Alias eines Bots aufzurufen.

Dem Benutzer wird ausdrücklich die Erlaubnis verweigert, den Bot-Alias zu aktualisieren oder zu löschen.

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:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
    ]
},
{
    "Sid": "botRunners",
    "Effect": "Deny",
    "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/botRunner"
    },
    "Action": [
        "lex:UpdateBotAlias",
        "lex>DeleteBotAlias"
    ],
    "Resource": [
        "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
    ]
}
]
}

```

Erlauben Sie einem AWS Service, einen bestimmten Amazon Lex V2-Bot zu verwenden

Das folgende Beispiel erteilt Amazon Connect die AWS Lambda Erlaubnis, Amazon Lex V2-Runtime-API-Operationen aufzurufen.

Der Bedingungsblock ist für Service Principals erforderlich und muss die globalen Kontextschlüssel `AWS:SourceAccount` und `AWS:SourceArn` verwenden.

Das `AWS:SourceAccount` ist die Konto-ID, die den Amazon Lex V2-Bot anruft.

Das `AWS:SourceArn` ist der Ressourcen-ARN der Amazon Connect Connect-Serviceinstanz oder Lambda-Funktion, von der der Aufruf des Amazon Lex V2-Bot-Alias stammt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "connect-bot-alias",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "connect.amazonaws.com"
        ]
      },
      "Action": [
        "lex:RecognizeText",
        "lex:StartConversation"
      ],
      "Resource": [
        "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
      ],
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "AWS:SourceArn":
            "arn:aws:connect:Region:123456789012:instance/instance-id"
        }
      }
    },
    {
      "Sid": "lambda-function",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com"
        ]
      },
      "Action": [
        "lex:RecognizeText",
        "lex:StartConversation"
      ],
      "Resource": [
```

```

        "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
    ],
    "Condition": {
        "StringEquals": {
            "AWS:SourceAccount": "123456789012"
        },
        "ArnEquals": {
            "AWS:SourceArn":
"arn:aws:lambda:Region:123456789012:function/function-name"
        }
    }
}
]
}

```

AWS verwaltete Richtlinien für Amazon Lex V2

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien dienen dazu, Berechtigungen für viele gängige Anwendungsfälle bereitzustellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie [vom Kunden verwaltete Richtlinien](#) definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter [Von AWS verwaltete Richtlinien](#) im IAM-Benutzerhandbuch.

Von AWS verwaltete Richtlinie: AmazonLexReadOnly

Sie können die AmazonLexReadOnly-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt nur Leseberechtigungen, mit denen Benutzer alle Aktionen im Amazon Lex V2- und Amazon Lex Model Building-Service einsehen können.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `lex`— Schreibgeschützter Zugriff auf Amazon Lex V2- und Amazon Lex Lex-Ressourcen im Modellbau-Service.

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": "*"
}
]
}
```

Von AWS verwaltete Richtlinie: AmazonLexRunBotsOnly

Sie können die AmazonLexRunBotsOnly-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt nur Leseberechtigungen, die den Zugriff auf die Ausführung von Amazon Lex V2- und Amazon Lex Lex-Konversationsbots ermöglichen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `lex`— Schreibgeschützter Zugriff auf alle Aktionen in der Amazon Lex V2- und Amazon Lex Lex-Laufzeit.

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": "*"
    }
  ]
}
```

Von AWS verwaltete Richtlinie: AmazonLexFullAccess

Sie können die AmazonLexFullAccess-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt Administratorberechtigungen, mit denen Benutzer Amazon Lex V2- und Amazon Lex-Ressourcen erstellen, lesen, aktualisieren und löschen sowie Amazon Lex V2- und Amazon Lex Lex-Konversationsbots ausführen können.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `lex`— Ermöglicht Prinzipalen Lese- und Schreibzugriff auf alle Aktionen in den Amazon Lex V2- und Amazon Lex Modellerstellungs- und Runtime-Services.
- `cloudwatch`— Ermöglicht es Prinzipalen, CloudWatch Amazon-Metriken und -Alarmer einzusehen.
- `iam`— Ermöglicht es Prinzipalen, servicebezogene Rollen zu erstellen und zu löschen, Rollen zu übergeben und Richtlinien an eine Rolle anzuhängen und zu trennen. Die Berechtigungen sind auf „lex.amazonaws.com“ für Amazon Lex-Operationen und auf „lexv2.amazonaws.com“ für Amazon Lex V2-Operationen beschränkt.
- `kendra`— Ermöglicht es Prinzipalen, Amazon Kendra Kendra-Indizes aufzulisten.
- `kms`— Ermöglicht Prinzipalen die Beschreibung AWS KMS von Schlüsseln und Aliasnamen.
- `lambda`— Ermöglicht es Prinzipalen, AWS Lambda Funktionen aufzulisten und Berechtigungen zu verwalten, die mit einer beliebigen Lambda-Funktion verknüpft sind.
- `polly`— Ermöglicht Schulleitern, Amazon Polly-Stimmen zu beschreiben und Sprache zu synthetisieren.

```
{
  "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"
        ]
      }
    }
  }
]
}

```

Von AWS verwaltete Richtlinie: AmazonLexReplicationPolicy

Sie können AmazonLexReplicationPolicy nicht an Ihre IAM-Entitäten anhängen. Diese Richtlinie ist mit einer dienstbezogenen Rolle verknüpft, die es Amazon Lex V2 ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter [Verwenden von serviceverknüpften Rollen für Amazon Lex V2](#).

Diese Richtlinie gewährt Administratorberechtigungen, die es Amazon Lex V2 ermöglichen, AWS Ressourcen in Ihrem Namen regionsübergreifend zu replizieren. Sie können diese Richtlinie anhängen, um es einer Rolle zu ermöglichen, Ressourcen wie Bots, Gebietsschemas, Versionen, Aliase, Absichten, Slot-Typen, Slots und benutzerdefinierte Vokabulare einfach zu replizieren.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

- **lex**— Ermöglicht Prinzipalen, Ressourcen in anderen Regionen zu replizieren.
- **iam**— Ermöglicht es Prinzipalen, Rollen von IAM zu übergeben. Dies ist erforderlich, damit Amazon Lex V2 berechtigt ist, Ressourcen in anderen Regionen zu replizieren.

```

{
  "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"
    }
  }
}
]
}

```

Von AWS verwaltete Richtlinie: AmazonLex V2 BedrockAgentPolicy

Richtlinie für Amazon Bedrock-Agenten

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Sid": "BedrockAgentInvokePolicy",
    "Action": [
      "bedrock:InvokeAgent"
    ],
    "Resource": [
      "arn:aws:bedrock:{region}:{accountId}:agent/[agentId]"
    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceAccount": "{accountId}"
      }
    }
  ]
}
```

Antwort

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2TrustPolicy",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

Von AWS verwaltete Richtlinie: AmazonLex V2 BedrockKnowledgeBasePolicy

Richtlinie für Amazon Bedrock-Wissensdatenbanken

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Sid": "BedrockKnowledgeBaseReadWritePolicy",
    "Action": [
      "bedrock:RetrieveAndGenerate",
      "bedrock:Retrieve"
    ],
    "Resource": [
      "arn:aws:bedrock:{region}:{accountId}:knowledge-base/[knowledgeBaseId]"
    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceAccount": "{accountId}"
      }
    }
  ]
}

```

Antwort

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2TrustPolicy",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Von AWS verwaltete Richtlinie: AmazonLex V2 BedrockAgentPolicyInternal

interne Richtlinien für Amazon Bedrock-Agenten

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Sid": "BedrockAgentInvokePolicy",
    "Action": [
      "bedrock:InvokeAgent"
    ],
    "Resource": [
      "arn:aws:bedrock:{region}:{accountId}:agent/{agentId}"
    ],
    "Condition": {
      "StringEquals": {
```

```

        "aws:ResourceAccount": "{accountId}"
    }
}
]
}

```

Antwort

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2InternalTrustPolicy",
      "Principal": {
        "Service": "lexv2.aws.internal"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}

```

Von AWS verwaltete Richtlinie: AmazonLex V2 BedrockKnowledgeBasePolicyInternal

interne Richtlinie für Amazon Bedrock-Wissensdatenbanken

JSON

```

{

```



```

"Version": "2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Sid": "BedrockKnowledgeBaseReadWritePolicy",
  "Action": [
    "bedrock:RetrieveAndGenerate",
    "bedrock:Retrieve"
  ],
  "Resource": [
    "arn:aws:bedrock:{region}:{accountId}:knowledge-base/
[knowledgeBaseId]"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "{accountId}"
    }
  }
}]
}

```

Antwort

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-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon Lex V2 an, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Abonnieren Sie den RSS-Feed auf der Amazon Lex V2-Seite, um automatische Benachrichtigungen über Änderungen an dieser [Dokumentenverlauf für Amazon Lex V2](#) Seite zu erhalten.

Änderung	Beschreibung	Datum
AmazonLexReplicationPolicy — Aktualisierte Richtlinie	Amazon Lex V2 hat die Richtlinie aktualisiert, um die Replikation von Tags und zu ermöglichen ResourceBasedPolicy.	24. Juni 2025
AmazonLexV2 BedrockKnowledgeBasePolicyInternal — Neue Richtlinie	Amazon Lex V2 hat eine neue Richtlinie hinzugefügt, um die Replikation von Amazon Bedrock-Wissensdatenbankressourcen zu ermöglichen.	30. August 2024
AmazonLexV2 BedrockAgentPolicyInternal — Neue Richtlinie	Amazon Lex V2 hat eine neue Richtlinie hinzugefügt, um die Replikation von Amazon Bedrock-Agentenressourcen zu ermöglichen.	30. August 2024
AmazonLexV2 BedrockKnowledgeBasePolicy — Neue Richtlinie	Amazon Lex V2 hat eine neue Richtlinie hinzugefügt, um die Replikation von	30. August 2024

Änderung	Beschreibung	Datum
	Amazon Bedrock-Wissensdatenbankressourcen zu ermöglichen.	
AmazonLexV2 BedrockAgentPolicy — Neue Richtlinie	Amazon Lex V2 hat eine neue Richtlinie hinzugefügt, um die Replikation von Amazon Bedrock-Agentenressourcen zu ermöglichen.	30. August 2024
AmazonLexReadOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um Repliken von Bot-Ressourcen mit schreibgeschütztem Zugriff zu ermöglichen.	10. Mai 2024
AmazonLexFullAccess – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um die Replikation von Bot-Ressourcen in andere Regionen zu ermöglichen.	16. April 2024
AmazonLexFullAccess – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um die Replikation von Bot-Ressourcen in andere Regionen zu ermöglichen.	31. Januar 2024
AmazonLexReplicationPolicy – Neue Richtlinie	Amazon Lex V2 hat eine neue Richtlinie hinzugefügt, um die Replikation von Bot-Ressourcen in andere Regionen zu ermöglichen.	31. Januar 2024

Änderung	Beschreibung	Datum
AmazonLexReadOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf die Liste benutzerdefinierter Vokabelelemente zu ermöglichen.	29. November 2022
AmazonLexFullAccess – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf die Abläufe des Amazon Lex V2-Modell badienstes zu ermöglichen.	18. August 2021
AmazonLexReadOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf Amazon Lex V2 Automated Chatbot Designer-Operationen zu ermöglichen.	1. Dezember 2021
AmazonLexFullAccess – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf die Abläufe des Amazon Lex V2-Modell badienstes zu ermöglichen.	18. August 2021
AmazonLexReadOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf die Abläufe des Amazon Lex V2-Modell badienstes zu ermöglichen.	18. August 2021

Änderung	Beschreibung	Datum
AmazonLexRunBotsOnly – Aktualisierung auf eine bestehende Richtlinie	Amazon Lex V2 hat neue Berechtigungen hinzugefügt, um den schreibgeschützten Zugriff auf Amazon Lex V2- Runtime-Servicevorgänge zu ermöglichen.	18. August 2021
Amazon Lex V2 hat begonnen, Änderungen zu verfolgen	Amazon Lex V2 hat damit begonnen, Änderungen für seine AWS verwalteten Richtlinien nachzuverfolgen.	18. August 2021

Verwenden von serviceverknüpften Rollen für Amazon Lex V2

Amazon Lex V2 verwendet AWS Identity and Access Management (IAM) [serviceverknüpfte Rollen](#). Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit Amazon Lex V2 verknüpft ist. Servicebezogene Rollen sind von Amazon Lex V2 vordefiniert und beinhalten alle Berechtigungen, die der Service benötigt, um andere AWS Services in Ihrem Namen aufzurufen.

Eine serviceverknüpfte Rolle erleichtert die Einrichtung von Amazon Lex V2, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Amazon Lex V2 definiert die Berechtigungen seiner serviceverknüpften Rollen, und sofern nicht anders definiert, kann nur Amazon Lex V2 seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Informationen zu anderen Services, die serviceverknüpfte Rollen unterstützen, finden Sie unter [AWS-Services, die mit IAM funktionieren](#). Suchen Sie nach den Services, für die Ja in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter [serviceverknüpfte Rollenberechtigungen](#) im IAM-Benutzerhandbuch.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem Sie zuerst die zugehörigen Ressourcen gelöscht haben. Dies schützt Ihre Amazon Lex V2-Ressourcen, da Sie die Zugriffsberechtigungen für die Ressourcen nicht versehentlich entfernen können.

Themen

- [Eine serviceverknüpfte Rolle für Amazon Lex V2 erstellen](#)
- [Bearbeiten einer serviceverknüpften Rolle für Amazon Lex V2](#)
- [Löschen einer serviceverknüpften Rolle für Amazon Lex V2](#)
- [Servicebezogene Rollenberechtigungen für Amazon Lex V2](#)
- [Unterstützte Regionen für serviceverknüpfte Amazon Lex V2-Rollen](#)

Eine serviceverknüpfte Rolle für Amazon Lex V2 erstellen

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen, da Amazon Lex V2 die serviceverknüpfte Rolle für Sie erstellt, wenn Sie die entsprechende Aktion (weitere Informationen finden [Servicebezogene Rollenberechtigungen für Amazon Lex V2](#) Sie unter) in der AWS Management Console AWS CLI, oder AWS API ausführen.

Wenn Sie diese serviceverknüpfte Rolle löschen und dann erneut eine erstellen müssen, können Sie dasselbe Verfahren verwenden, um eine neue Rolle in Ihrem Konto zu erstellen.

Bearbeiten einer serviceverknüpften Rolle für Amazon Lex V2

Amazon Lex V2 erlaubt es Ihnen nicht, serviceverknüpfte Rollen zu bearbeiten. Nachdem Sie eine serviceverknüpfte Rolle erstellt haben, können Sie den Namen der Rolle nicht mehr ändern, da verschiedene Entitäten auf die Rolle verweisen könnten. Sie können jedoch die Beschreibung einer Rolle mithilfe von IAM bearbeiten. Weitere Informationen finden Sie unter [Bearbeiten einer serviceverknüpften Rolle](#) im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Rolle für Amazon Lex V2

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch die Ressourcen für Ihre serviceverknüpften Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

 Note

Wenn der Amazon Lex V2-Service die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Die Schritte zum Löschen von Ressourcen für bestimmte serviceverknüpfte Rollen in Amazon Lex V2 finden Sie im [Servicebezogene Rollenberechtigungen für Amazon Lex V2](#) rollenspezifischen Abschnitt unter.

Um eine serviceverknüpfte Rolle mithilfe von IAM manuell zu löschen

Nachdem Sie Ressourcen gelöscht haben, die sich auf eine serviceverknüpfte Rolle beziehen, verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die Rolle zu löschen. Weitere Informationen finden Sie unter [Löschen einer serviceverknüpften Rolle](#) im IAM-Leitfaden.

Servicebezogene Rollenberechtigungen für Amazon Lex V2

Amazon Lex V2 verwendet serviceverknüpfte Rollen mit den folgenden Präfixen.

Themen

- [AWSServiceRoleForLexV2Bots_](#)
- [AWSServiceRoleForLexV2Channels_](#)
- [AWSServiceRoleForLexV2-Replikation](#)

AWSServiceRoleForLexV2Bots_

Die Rolle AWSService RoleForLex V2Bots_ erteilt die Erlaubnis, Ihren Bot mit anderen erforderlichen Diensten zu verbinden. Diese Rolle beinhaltet eine Vertrauensrichtlinie, die es dem Dienst lexv2.amazonaws.com ermöglicht, die Rolle zu übernehmen, und beinhaltet Berechtigungen zur Ausführung der folgenden Aktionen.

- Verwenden Sie Amazon Polly, um Sprache auf allen Amazon Lex V2-Ressourcen zu synthetisieren, die die Aktion unterstützt.
- Wenn ein Bot für die Verwendung der Amazon Comprehend-Stimmungsanalyse konfiguriert ist, ermitteln Sie die Stimmung auf allen Amazon Lex V2-Ressourcen, die die Aktion unterstützt.

- Wenn ein Bot so konfiguriert ist, dass er Audioprotokolle in einem S3-Bucket speichert, platzieren Sie Objekte in einem bestimmten Bucket.
- Wenn ein Bot so konfiguriert ist, dass er Audio- und Textprotokolle speichert, erstellen Sie einen Log-Stream und fügen Sie die Logs einer bestimmten Protokollgruppe hinzu.
- Wenn ein Bot so konfiguriert ist, dass er einen AWS KMS Schlüssel zum Verschlüsseln von Daten verwendet, generieren Sie einen bestimmten Datenschlüssel.
- Wenn ein Bot für die Verwendung der KendraSearchIntent Absicht konfiguriert ist, fragen Sie den Zugriff auf einen bestimmten Amazon Kendra Kendra-Index ab.

Um die Rolle zu erstellen

[Amazon Lex V2 erstellt jedes Mal, wenn Sie einen Bot erstellen, eine neue AWSService RoleForLex V2Bots_-Rolle mit einem zufälligen Suffix in Ihrem Konto.](#) Amazon Lex V2 ändert die Rolle, wenn Sie einem Bot zusätzliche Funktionen hinzufügen. Wenn Sie beispielsweise [Amazon Comprehend Sentiment Analysis zu einem Bot hinzufügen](#), fügt Amazon Lex V2 der Servicerolle die Berechtigung für die `lex:DetectSentiment` Aktion hinzu.

Um die Rolle zu löschen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie im linken Navigationsbereich Bots und dann den Bot aus, dessen mit dem Service verknüpfte Rolle Sie löschen möchten.
3. Wählen Sie eine beliebige Version des Bots aus.
4. Die Runtime-Rolle für IAM-Berechtigungen befindet sich in den Versionsdetails.
5. Kehren Sie zur Seite „Bots“ zurück und wählen Sie das Optionsfeld neben dem Bot, den Sie löschen möchten.
6. Wählen Sie Aktion und dann Löschen.
7. Folgen Sie den Schritten unter [Löschen einer serviceverknüpften Rolle, um die IAM-Rolle zu löschen](#).

AWSServiceRoleForLexV2Channels_

Die Rolle AWSService RoleForLex V2Channels_ erteilt die Erlaubnis, Bots in einem Konto aufzulisten und Konversationen für einen Bot aufzurufen. APIs Diese Rolle beinhaltet eine Vertrauensrichtlinie,

die es dem Dienst `channels.lexv2.amazonaws.com` ermöglicht, die Rolle zu übernehmen. Wenn ein Bot so konfiguriert ist, dass er einen Kanal für die Kommunikation mit einem Messaging-Dienst verwendet, ermöglicht die Rollenberechtigungsrichtlinie `AWSService RoleForLex V2Channels_ Amazon Lex V2`, die folgenden Aktionen durchzuführen.

- Listet die Berechtigungen für alle Bots in einem Konto auf.
- Erkennen Sie Text, rufen Sie eine Sitzung ab und vergeben Sie Sitzungsberechtigungen für einen bestimmten Bot-Alias.

Um die Rolle zu erstellen

Wenn Sie eine Kanalintegration erstellen, um einen Bot auf einer Messaging-Plattform bereitzustellen, erstellt Amazon Lex V2 in Ihrem Konto für jeden Kanal eine neue serviceverknüpfte Rolle mit einem zufälligen Suffix.

Um die Rolle zu löschen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie im linken Navigationsbereich Bots aus.
3. Wählen Sie einen Bot aus.
4. Wählen Sie im linken Navigationsbereich unter Bereitstellungen die Option Kanalintegrationen aus.
5. Wählen Sie einen Channel aus, dessen dienstbezogene Rolle Sie löschen möchten.
6. Die Runtime-Rolle für IAM-Berechtigungen befindet sich in der allgemeinen Konfiguration
7. Wählen Sie Löschen und anschließend erneut Löschen, um den Kanal zu löschen.
8. Folgen Sie den Schritten unter [Löschen einer serviceverknüpften Rolle, um die IAM-Rolle](#) zu löschen.

AWSServiceRoleForLexV2-Replikation

Die Rolle `AWSService RoleForLex V2Replication` erteilt die Erlaubnis, Bots in einer zweiten Region zu replizieren. Diese Rolle beinhaltet eine Vertrauensrichtlinie, die es dem Service `replication.lexv2.amazonaws.com` ermöglicht, die Rolle zu übernehmen, und umfasst auch die [AmazonLexReplicationPolicy](#) AWS verwaltete Richtlinie, die Berechtigungen für die folgenden Aktionen gewährt.

- Übergeben Sie Bot-IAM-Rollen an den Replikat-Bot, um die entsprechenden Berechtigungen für den Replikat-Bot erneut zu duplizieren.
- Erstellen und verwalten Sie Bots und Bot-Ressourcen (Versionen, Aliase, Absichten, Slots, benutzerdefinierte Vokabulare usw.) in anderen Regionen.

Um die Rolle zu erstellen

Wenn Sie Global Resiliency für einen Bot aktivieren, erstellt Amazon Lex V2 die mit dem Dienst AWSService RoleForLex V2Replication verknüpfte Rolle in Ihrem Konto. Stellen Sie sicher, dass Sie über die richtigen [Berechtigungen verfügen](#), um dem Amazon Lex V2-Service Berechtigungen zur Erstellung der serviceverknüpften Rolle zu erteilen.

Um Amazon Lex V2-Ressourcen zu löschen, die von AWSService RoleForLex V2Replication verwendet werden, damit Sie die Rolle löschen können

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Lex Lex-Konsole unter <https://console.aws.amazon.com/lex/>.
2. Wählen Sie einen Bot aus, für den Global Resiliency aktiviert ist.
3. Wählen Sie unter Deployment die Option Global Resiliency aus.
4. Wählen Sie Globale Resilienz deaktivieren aus.
5. Wiederholen Sie den Vorgang für alle Bots, für die Global Resiliency aktiviert ist.
6. Folgen Sie den Schritten unter [Löschen einer serviceverknüpften Rolle, um die IAM-Rolle zu löschen](#).

Unterstützte Regionen für serviceverknüpfte Amazon Lex V2-Rollen

Amazon Lex V2 unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter [AWS -Regionen und Endpunkte](#).

Fehlerbehebung bei Amazon Lex V2: Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon Lex V2 und IAM auftreten können.

Themen

- [Ich bin nicht berechtigt, eine Aktion in Amazon Lex V2 durchzuführen](#)

- [Ich bin nicht berechtigt, iam auszuführen: PassRole](#)
- [Ich bin Administrator und möchte anderen den Zugriff auf Amazon Lex V2 ermöglichen](#)
- [Gewähren Sie einem Benutzer programmatischen Zugriff](#)
- [Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine Amazon Lex V2-Ressourcen ermöglichen](#)

Ich bin nicht berechtigt, eine Aktion in Amazon Lex V2 durchzuführen

Wenn Ihnen AWS Management Console mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion durchzuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven `my-example-widget`-Ressource zu verwenden, jedoch nicht über `lex:GetWidget`-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
lex:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion `my-example-widget` auf die Ressource `lex:GetWidget` zugreifen zu können.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Amazon Lex V2 übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder eine dienstbezogene Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon Lex V2 auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:  
iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin Administrator und möchte anderen den Zugriff auf Amazon Lex V2 ermöglichen

Um anderen den Zugriff auf Amazon Lex V2 zu ermöglichen, müssen Sie den Personen oder Anwendungen, die Zugriff benötigen, die Erlaubnis erteilen. Wenn Sie Personen und Anwendungen verwalten, weisen Sie Benutzern oder Gruppen Berechtigungssätze zu, um deren Zugriffsebene zu definieren. AWS IAM Identity Center Mit Berechtigungssätzen werden automatisch IAM-Richtlinien erstellt und den IAM-Rollen zugewiesen, die der Person oder Anwendung zugeordnet sind. Weitere Informationen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter [Berechtigungssätze](#).

Wenn Sie IAM Identity Center nicht verwenden, müssen Sie IAM-Entitäten (Benutzer oder Rollen) für die Personen oder Anwendungen erstellen, die Zugriff benötigen. Anschließend müssen Sie der Entität eine Richtlinie beifügen, die ihr die richtigen Berechtigungen in Amazon Lex V2 gewährt. Nachdem die Berechtigungen erteilt wurden, geben Sie die Anmeldeinformationen an den Benutzer oder Anwendungsentwickler weiter. Sie werden diese Anmeldeinformationen für den Zugriff verwenden AWS. Weitere Informationen zum Erstellen von IAM-Benutzern, -Gruppen, -Richtlinien und -Berechtigungen finden Sie im [IAM-Benutzerhandbuch unter IAM-Identitäten sowie Richtlinien und Berechtigungen in IAM](#).

Gewähren Sie einem Benutzer programmatischen Zugriff

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des Interagierens möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. - Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. - Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldinformationen im AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine Amazon Lex V2-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Amazon Lex V2 diese Funktionen unterstützt, finden Sie unter [So funktioniert Amazon Lex V2 mit IAM](#).
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter [Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen](#).
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie [AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte](#).

- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter [Gewähren von Zugriff für extern authentifizierte Benutzer \(Identitätsverbund\)](#) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter [Kontoübergreifender Ressourcenzugriff in IAM](#) im IAM-Benutzerhandbuch.

Protokollierung und Überwachung in Amazon Lex V2

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon Lex V2 und Ihren anderen AWS-Lösungen. AWS bietet die folgenden Überwachungstools, um Amazon Lex V2 zu beobachten, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen:

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarmer festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).
- AWS CloudTrail erfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Aufrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im [AWS CloudTrail -Benutzerhandbuch](#).

Konformitätsprüfung für Amazon Lex V2

Externe Prüfer bewerten die Sicherheit und Konformität von Amazon Lex V2 im Rahmen mehrerer AWS Compliance-Programme. Amazon Lex V2 ist ein HIPAA-fähiger Service. Er ist PCI-, SOC- und ISO-konform.

Informationen darüber, ob AWS-Service ein in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter [AWS-Services Umfang nach Compliance-Programm unter Umfang](#)

[nach Compliance-Programm AWS-Services](#) das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#).

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#).

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- [Compliance und Governance im Bereich Sicherheit](#) – In diesen Anleitungen für die Lösungsimplementierung werden Überlegungen zur Architektur behandelt. Außerdem werden Schritte für die Bereitstellung von Sicherheits- und Compliance-Features beschrieben.
- [Referenz für berechnete HIPAA-Services](#) – Listet berechnete HIPAA-Services auf. Nicht alle AWS-Services sind HIPAA-fähig.
- [AWS Compliance-Ressourcen](#) — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- [AWS Leitfäden zur Einhaltung von Vorschriften für Kunden](#) — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- [Evaluierung von Ressourcen anhand von Regeln](#) im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- [AWS Security Hub](#) — Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Die Liste der unterstützten Services und Kontrollen finden Sie in der [Security-Hub-Steuerelementreferenz](#).
- [Amazon GuardDuty](#) — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.

- [AWS Audit Manager](#)— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Resilienz in Amazon Lex V2

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Zusätzlich zur AWS globalen Infrastruktur bietet Amazon Lex V2 mehrere Funktionen, um Ihre Anforderungen an Datenstabilität und Datensicherung zu erfüllen.

Note

[Weitere Informationen zur globalen Resilienz in Amazon Lex V2, mit der Sie einen replizierten Bot in einer zweiten Region in vordefinierten Paaren erstellen können, finden Sie unter Globale Resilienz.](#)

Infrastruktursicherheit in Amazon Lex V2

Als verwalteter Service ist Amazon Lex V2 durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper [Amazon Web Services: Sicherheitsprozesse im Überblick](#) beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Amazon Lex V2 zuzugreifen. Kunden müssen Transport Layer Security (TLS) 1.0 oder neuer unterstützen. Wir empfehlen TLS 1.2 oder höher. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve

Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit [AWS Security Token Service](#) (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Amazon Lex V2 und VPC-Schnittstellen-Endpunkte (AWS PrivateLink)

Sie können eine private Verbindung zwischen Ihrer VPC und Amazon Lex V2 herstellen, indem Sie einen VPC-Schnittstellen-Endpunkt erstellen. Schnittstellenendpunkte werden mit einer Technologie betrieben [AWS PrivateLink](#), die es Ihnen ermöglicht, privat auf Amazon Lex V2 APIs ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder AWS Direct Connect-Verbindung zuzugreifen. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um mit Amazon Lex V2 APIs zu kommunizieren. Der Datenverkehr zwischen Ihrer VPC und Amazon Lex V2 verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere [Elastic-Netzwerk-Schnittstellen](#) in Ihren Subnetzen dargestellt.

Weitere Informationen finden Sie unter [Schnittstellen-VPC-Endpunkte \(AWS PrivateLink\)](#) im Amazon-VPC-Benutzerhandbuch.

Überlegungen zu Amazon Lex V2 VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für Amazon Lex V2 einrichten, sollten Sie die [Eigenschaften und Einschränkungen der Schnittstellen-Endpunkte](#) im Amazon VPC-Benutzerhandbuch lesen.

Amazon Lex V2 unterstützt Aufrufe aller API-Aktionen von Ihrer VPC aus.

Erstellen eines Schnittstellen-VPC-Endpunkts für Amazon Lex V2

Sie können einen VPC-Endpunkt für den Amazon Lex V2-Service entweder mit der Amazon VPC-Konsole oder mit AWS Command Line Interface (AWS CLI) erstellen. Weitere Informationen finden Sie unter [Erstellung eines Schnittstellenendpunkts](#) im Benutzerhandbuch für Amazon VPC.

Erstellen Sie einen VPC-Endpunkt für Amazon Lex V2 mit dem folgenden Servicenamen:

- `com.amazonaws. region.models-v2-lex`
- `com.amazonaws. region.runtime-v2-lex`

Wenn Sie privates DNS für den Endpunkt aktivieren, können Sie API-Anfragen an Amazon Lex V2 stellen, indem Sie den Standard-DNS-Namen für die Region verwenden, `runtime-v2-lex.us-east-1.amazonaws.com` z. B.

Weitere Informationen finden Sie unter [Zugriff auf einen Service über einen Schnittstellenendpunkt](#) im Benutzerhandbuch für Amazon VPC.

Erstellen einer VPC-Endpunktrichtlinie für Amazon Lex V2

Sie können Ihrem VPC-Endpunkt eine Endpunktrichtlinie hinzufügen, die den Zugriff auf Amazon Lex V2 steuert. Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter [Steuerung des Zugriffs auf Services mit VPC-Endpunkten](#) im Amazon-VPC-Benutzerhandbuch.

Beispiel: VPC-Endpunktrichtlinie für Amazon Lex V2-Aktionen

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für Amazon Lex V2. Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die aufgelisteten Amazon Lex V2-Aktionen.

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "lex:RecognizeText",
        "lex:RecognizeUtterance",
        "lex:StartConversation",
```

```
        "lex:DeleteSession",  
        "lex:GetSession",  
        "lex:DeleteSession"  
    ],  
    "Resource": "*"   
}   
]
```

Codebeispiele für Amazon Lex mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon Lex mit einem AWS Software Development Kit (SDK) verwendet wird.

Szenarien sind Code-Beispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Amazon Lex V2 mit einem AWS SDK verwenden](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- [Szenarien für die Verwendung von Amazon Lex AWS SDKs](#)
 - [Erstellen Sie einen Amazon Lex Lex-Chatbot, um die Besucher Ihrer Website anzusprechen](#)

Szenarien für die Verwendung von Amazon Lex AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Lex mit implementieren AWS SDKs. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon Lex oder in Kombination mit anderen aufrufen AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- [Erstellen Sie einen Amazon Lex Lex-Chatbot, um die Besucher Ihrer Website anzusprechen](#)

Erstellen Sie einen Amazon Lex Lex-Chatbot, um die Besucher Ihrer Website anzusprechen

Die folgenden Codebeispiele zeigen, wie Sie einen Chatbot erstellen, um die Besucher Ihrer Website anzusprechen.

Java

SDK für Java 2.x

Zeigt, wie Sie mithilfe der Amazon Lex Lex-API einen Chatbot innerhalb einer Webanwendung erstellen, um die Besucher Ihrer Website anzusprechen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mithilfe der Amazon Lex Lex-API einen Chatbot innerhalb einer Webanwendung erstellen, um die Besucher Ihrer Website anzusprechen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel „[Einen Amazon Lex-Chatbot erstellen](#)“ im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Amazon Lex V2 mit einem AWS SDK verwenden](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Leitlinien und Bewährte Methoden

Beachten Sie die folgenden Richtlinien und bewährten Methoden, um das Verhalten und die Interaktionen Ihres Bots mit Kunden zu optimieren.

Signieren von Anforderungen

Alle Amazon Lex V2-Modellerstellungs- und Runtime-Anfragen in der [API-Referenz](#) verwenden die Signatur V4 für die Authentifizierung von Anfragen. Weitere Informationen zur Authentifizierung von Anfragen finden Sie unter [Signaturprozess für Signature Version 4](#) im. Allgemeine AWS-Referenz

Schutz vertraulicher Informationen

Die Runtime-API-Operationen [RecognizeText](#) und [RecognizeUtterance](#) verwenden eine Sitzungs-ID als erforderlichen Parameter. Developer können diese auf jeden Wert stellen, der den in der API beschriebenen Einschränkungen übereinstimmt. Wir empfehlen, diesen Parameter nicht zum Senden vertraulicher Informationen wie Benutzeranmeldungen, E-Mails oder Sozialversicherungsnummern zu verwenden. Diese ID wird hauptsächlich verwendet, um eine Konversation mit einem Bot eindeutig zu identifizieren.

Erfassung von Slot-Werten aus Benutzeräußerungen

Amazon Lex V2 verwendet die Aufzählungswerte, die Sie in einer Slot-Typdefinition angeben, um seine Machine-Learning-Modelle zu trainieren. Angenommen, Sie definieren eine Absicht, die `GetPredictionIntent` mit der folgenden Beispieläußerung aufgerufen wird:

```
"Tell me the prediction for {sign}"
```

wobei `{sign}` ein Slot mit dem benutzerdefinierten Typ `istZodiacSign`, der 12 Aufzählungswerte hat: `Aries` bis `Pisces`. Nehmen wir nun an, der Benutzer sagt „Erzähl mir die Vorhersage für die Erde“:

- Amazon Lex V2 leitet ab, dass „Erde“ ein `ZodiacSign` Wert ist, wenn Sie eine der folgenden Aktionen ausführen:
 - Stellen Sie das `valueSelectionStrategy` Feld so ein, dass es die `ORIGINAL_VALUE` [CreateSlotType](#) Operation verwendet
 - Wählen Sie in der Konsole Werte erweitern
- Amazon Lex V2 erkennt den Wert „Erde“ nicht, wenn Sie die Erkennung auf die Werte beschränken, die Sie für den Slot-Typ definiert haben, indem Sie eine der folgenden Aktionen ausführen:

- Stellen Sie das `valueSelectionStrategy` Feld so ein, dass es die `CreateSlotType` Operation `TOP_RESOLUTION` verwendet
- Wählen Sie in der Konsole auf Slot-Werte und Synonyme beschränken

Wenn Sie Synonyme für Slot-Werte definieren, werden diese als identisch mit Slot-Werten erkannt. Der Slot-Wert wird jedoch anstelle des Synonyms zurückgegeben.

Da Amazon Lex V2 diesen Wert an Ihre Client-Anwendung oder an die Lambda-Funktion weitergibt, sollten Sie überprüfen, ob die Slot-Werte gültige Werte sind, bevor Sie sie in Ihrer Fulfillment-Aktivität verwenden.

Wenn Amazon Lex V2 eine Lambda-Funktion aufruft oder das Ergebnis einer Sprachinteraktion mit Ihrem Client zurückgibt, kann die Groß- und Kleinschreibung der Slot-Werte nicht garantiert werden. Bei Textinteraktionen entspricht die Groß-/Kleinschreibung der Slot-Werte dem eingegebenen Text oder dem Slot-Wert, abhängig von dem Wert im Feld `valueResolutionStrategy`.

Akronyme in Slot-Werten

Verwenden Sie bei der Definition von Slot-Werten, die Akronyme enthalten, die folgenden Muster:

- Durch Punkte getrennte Großbuchstaben (D.V.D.)
- Durch Leerzeichen getrennte Großbuchstaben (D V D)

Integrierte Steckplätze für Datum und Uhrzeit

Die [Amazon.date](#) und die [Amazon.Zeit](#) integrierten Slot-Typen erfassen Datum und Uhrzeit (sowohl absolut als auch relativ). Relative Daten und Uhrzeiten werden zu dem Zeitpunkt und dem Datum, an dem Amazon Lex V2 die Anfrage empfängt, und in der Region, in der die Anfrage bearbeitet wird, ermittelt.

Wenn der Benutzer beim `AMAZON.Time` integrierten Slot-Typ nicht angibt, dass eine Uhrzeit vor oder nach Mittag liegt, ist die Uhrzeit mehrdeutig. In diesem Fall fordert Amazon Lex V2 den Benutzer erneut auf. Wir empfehlen Aufforderungen, in denen nach der absoluten Zeit gefragt wird. Verwenden Sie beispielsweise eine Aufforderung wie "Wann möchten Sie Ihre Pizza geliefert bekommen? Sie können 6 Uhr morgen oder 6 Uhr abends angeben."

Vermeidung von Mehrdeutigkeiten bei den Trainingsdaten für Ihren Bot

Die Bereitstellung verwirrender Trainingsdaten in Ihrem Bot verringert die Fähigkeit von Amazon Lex V2, Benutzereingaben zu verstehen. Angenommen, Sie haben zwei Absichten (`OrderPizza` und `OrderDrink`) in Ihrem Bot und geben „Ich möchte bestellen“ als Beispiel für eine Äußerung an. Wenn Sie Ihren Bot erstellen, ist Amazon Lex V2 nicht in der Lage, diese Äußerung einer bestimmten Absicht zuzuordnen. Wenn ein Benutzer diese Äußerung zur Laufzeit eingibt, kann Amazon Lex V2 daher keine Absicht mit einem hohen Maß an Sicherheit auswählen.

Wenn Sie zwei Absichten mit derselben Beispieläußerung haben, verwenden Sie Eingabekontexte, damit Amazon Lex V2 zur Laufzeit zwischen den beiden Absichten unterscheiden kann. Weitere Informationen finden Sie unter Absichtskontext [festlegen](#).

Verwenden Sie den TSTALIASID-Alias

- Der TSTALIASID-Alias Ihres Bots verweist auf die Entwurfsversion und sollte nur für manuelle Tests verwendet werden. Amazon Lex begrenzt die Anzahl der Runtime-Anfragen, die Sie an den TSTALIASID-Alias des Bots stellen können.
- Wenn Sie die Entwurfsversion des Bots aktualisieren, beendet Amazon Lex alle laufenden Konversationen für jede Client-Anwendung, die den TSTALIASID-Alias des Bots verwendet. Im Allgemeinen sollten Sie den TSTALIASID-Alias eines Bots in der Produktion nicht verwenden, da die Entwurfsversion aktualisiert werden kann. Sie sollten eine Version und einen Alias veröffentlichen und diese stattdessen verwenden.
- Wenn Sie einen Alias aktualisieren, benötigt Amazon Lex einige Minuten, um die Änderungen zu übernehmen. Wenn Sie die Entwurfsversion des Bots ändern, wird die Änderung sofort vom TSTALIASID-Alias übernommen.

Kontingente

Servicekontingenten, auch Limits genannt, sind die maximale Anzahl von Serviceressourcen, die für Ihr AWS Konto zulässig sind. Weitere Informationen finden Sie unter [AWS-Servicekontingente](#) in der AWS allgemeinen Referenz.

Einige Service Quotas können angepasst oder erhöht werden. In der Spalte „Anpassbar“ in den folgenden Tabellen finden Sie Informationen darüber, ob ein Kontingent angepasst werden kann, und in der Spalte Self-Service, ob Sie eine Kontingentanpassung über die [Service-Kontingente-Konsole](#) beantragen können. Wenden Sie sich an uns Support , um ein Kontingent zu erhöhen, das zwar anpassbar ist, jedoch nicht über Self-Service. Die Erhöhung eines Servicekontingents kann einige Tage dauern. Wenn Sie Ihr Kontingent im Rahmen eines größeren Projekts erhöhen, sollten Sie diese Zeit unbedingt zu Ihrem Plan hinzufügen.

Note

Zeichenbeschränkungen werden als Anzahl der [Unicode-Codeeinheiten](#) berechnet. In den meisten Fällen entspricht ein Unicode-Zeichen einer Unicode-Codeeinheit. Einige Sonderzeichen können mehr als eine Einheit umfassen, und die Anzahl kann je nach Kodierung unterschiedlich sein. Weitere Informationen zur Berechnung der Zeichenkettenlänge finden Sie in [dieser Dokumentation](#).

Kontingente für die Erstellung

Die folgenden Höchstkontingente werden durchgesetzt, wenn Sie einen Bot erstellen.

Beschreibung	Standard	Anpassbar	Self-Service
Bots pro Konto AWS	100	Ja	Ja
Bot-Kanalzuordnungen pro AWS Konto	5,000	Nein	N/A
Parallele Gebietssemaphora-Builds pro Konto AWS	5	Ja	Nein

Beschreibung	Standard	Anpassbar	Self-Service
Bots pro Bot-Netzwerk	5	Nein	N/A
Bot-Netzwerke pro Bot	25	Nein	N/A
Versionen pro Bot	100	Nein	N/A
Absichten pro Gebietsschema in jedem Bot	• 1.000 in en-AU, en-GB und en-US • 250 in allen anderen Regionen	Ja	Nein
Steckplätze pro Gebietsschema in jedem Bot	• 4.000 in en-AU, en-GB und en-US • 2.000 in allen anderen Regionen	Nein	N/A
Benutzerdefinierte Slot-Typen pro Bot-Gebietsschema	• 250 in en-AU, en-GB und en-US • 100 in allen anderen Ländern	Nein	N/A
Benutzerdefinierte Slot-Typ-Werte und Synonyme pro Gebietsschema in jedem Bot	50 000	Nein	N/A
Gesamtzahl der Zeichen in Beispielaussagen pro Gebietsschema in jedem Bot	• 2.000.000 in en-AU, en-GB und en-US • 200.000 in allen anderen Regionen	Nein	N/A

Beschreibung	Standard	Anpassbar	Self-Service
Kanalzuordnungen pro Bot-Alias	10	Nein	N/A
Steckplätze pro Absicht	100	Nein	N/A
Beispiele für Äußerungen pro Absicht	1.500	Ja	Ja
Zeichen pro Beispieläußerung	500	Nein	N/A
Länge der Textantwort	4.000	Nein	N/A
Beispiele für Äußerungen pro Slot	10	Ja	Ja
Zeichen pro Muster-Slot-Äußerung	500	Nein	N/A
Eingabeaufforderungen pro Slot	30	Nein	N/A
Werte und Synonyme pro benutzerdefinierter Slot-Typ	10.000	Nein	N/A
Zeichen pro benutzerdefiniertem Slot-Typ-Wert	500	Nein	N/A
Zeichen in einem Kanalzuordnungsnamen	100	Nein	N/A

Beschreibung	Standard	Anpassbar	Self-Service
Anzahl gleichzeitiger automatisierter Chatbot-Designer-Analysejobs für alle Bots in Ihrem Konto pro Region	10	Nein	N/A
Größe der XML-Datei vom Typ „Benutzer definierter Grammatik slot“	100 KB	Nein	N/A

Laufzeitkontingente

Die folgenden maximalen Kontingente werden zur Laufzeit durchgesetzt.

Beschreibung	Standard	Anpassbar	Self-Service
Eingabe-Textgröße für und Recognize TextUtterance	1024 Zeichen	Nein	N/A
Länge der Spracheingabe für den Recognize Utterance Betrieb	15 Sekunden	Ja	Nein
Größe der Recognize Utterance Header	16 KB	Nein	N/A

Beschreibung	Standard	Anpassbar	Self-Service
Größe der kombinierten Anfrage- und Sitzungsheader für Recognize Utterance	12 KB	Nein	N/A
Maximale Anzahl gleichzeitiger Konversationen im Textmodus für Recognize Text ,Recognize Utterance , oder StartConversation für TestBotAlias	2	Nein	N/A
Maximale Anzahl gleichzeitiger Konversationen im Textmodus für Recognize Text ,Recognize Utterance , oder für andere Aliase StartConversation	50	Ja	Nein
Maximale Anzahl gleichzeitiger Konversationen im Sprachmodus für für Recognize Utterance TestBotAlias	2	Nein	N/A

Beschreibung	Standard	Anpassbar	Self-Service
Maximale Anzahl gleichzeitiger Sprachmodus-Konversationen für andere Aliase Recognize Utterance	125	Ja	Nein
Maximale Anzahl gleichzeitiger Sprachmodus-Konversationen für StartConversation TestBotAlias	2	Nein	N/A
Maximale Anzahl gleichzeitiger Sprachmodus-Konversationen für andere Aliase StartConversation	200	Ja	Nein
Maximale Anzahl gleichzeitiger Sitzungsverwaltungsvorgänge (PutSession ,GetSession , oder) bei Verwendung von DeleteSession TestBotAlias	2	Nein	N/A

Beschreibung	Standard	Anpassbar	Self-Service
Maximale Anzahl gleichzeitiger Sitzungsverwaltungsvorgänge (PutSession, GetSession, oder DeleteSession) bei Verwendung anderer Aliase	50	Ja	Nein
Maximale Eingabegröße für eine Lambda-Funktion	12 KB	Nein	N/A
Maximale Ausgabegröße einer Lambda-Funktion	50 KB	Nein	N/A
Maximale Größe von Sitzungsattributen in der Lambda-Funktionsausgabe (nach Base-64-Codierung)	12 KB	Nein	N/A
Maximaler Timeout einer Lambda-Funktion	30 Sekunden	Ja	Nein
Maximale Dauer für eine einzelne Konversation	15 Minuten	Nein	Nein
Maximale Audiolänge	55 Sekunden	Nein	Nein

Leitfaden zur Migration von Amazon Lex V1 zu V2

Die Amazon Lex V2-Konsole und APIs erleichtern das Erstellen und Verwalten von Bots. In diesem Handbuch erfahren Sie mehr über die Verbesserungen der Amazon Lex V2-API bei der Migration von Bots.

Sie migrieren einen Bot mithilfe der Amazon Lex Lex-Konsole oder API. Weitere Informationen finden Sie unter [Einen Bot migrieren](#) im Amazon Lex Developer Guide.

Überblick über Amazon Lex V2

Einem Bot können mehrere Sprachen hinzugefügt werden, sodass Sie sie als eine einzige Ressource verwalten können. Mit einer vereinfachten Informationsarchitektur können Sie Ihre Bot-Versionen effizient verwalten. Funktionen wie ein „Konversationsfluss“, teilweises Speichern der Bot-Konfiguration und Massenupload von Äußerungen bieten Ihnen mehr Flexibilität.

Mehrere Sprachen in einem Bot

Mit der Amazon Lex V2-API können Sie mehrere Sprachen hinzufügen. Sie fügen jede Sprache unabhängig hinzu, ändern und erstellen sie. Ressourcen wie Slot-Typen werden auf Sprachebene abgegrenzt. Sie können schnell zwischen verschiedenen Sprachen wechseln, um die Konversationen zu vergleichen und zu verfeinern. Sie können ein Dashboard in der Konsole verwenden, um Äußerungen für alle Sprachen zu überprüfen, um Analysen und Wiederholungen zu beschleunigen. Ein Bot-Operator kann Berechtigungen und Protokollierungsvorgänge für alle Sprachen mit einer Bot-Konfiguration verwalten. Sie müssen eine Sprache als Laufzeitparameter angeben, um mit einem Amazon Lex V2-Bot zu kommunizieren. Weitere Informationen finden Sie unter [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#).

Vereinfachte Informationsarchitektur

Die Amazon Lex V2-API folgt einer vereinfachten Informationsarchitektur (IA) mit Intent- und Slot-Typen, die auf eine Sprache beschränkt sind. Sie versionieren auf Bot-Ebene, sodass Ressourcen wie Intents und Slot-Typen nicht einzeln versioniert werden. Standardmäßig wird ein Bot mit einer Entwurfsversion erstellt, die veränderbar ist und zum Testen von Änderungen verwendet wird. Sie können nummerierte Schnappschüsse aus der Entwurfsversion erstellen. Sie wählen die Sprachen aus, die in eine Version aufgenommen werden sollen. Alle Ressourcen innerhalb des Bots

(Sprachen, Absichten, Slot-Typen) werden im Rahmen der Erstellung einer Bot-Version archiviert. Weitere Informationen finden Sie unter [Versionen](#).

Verbesserte Produktivität des Builders

Sie verfügen über zusätzliche Tools und Funktionen für Builder-Produktivität, die Ihnen mehr Flexibilität und Kontrolle über Ihren Bot-Designprozess bieten.

Speichern Sie die Teilkonfiguration

Die Amazon Lex V2-API ermöglicht es Ihnen, teilweise Änderungen während der Entwicklung zu speichern. Sie können beispielsweise einen Slot speichern, der auf einen gelöschten Slot-Typ verweist. Diese Flexibilität ermöglicht es Ihnen, Ihre Arbeit zu speichern und später darauf zurückzukommen. Sie können diese Änderungen beheben, bevor Sie den Bot erstellen. In Amazon Lex V2 kann das teilweise Speichern auf Slots, Versionen und Aliase angewendet werden.

Ressourcen umbenennen

Mit Amazon Lex V2 können Sie eine Ressource umbenennen, nachdem sie erstellt wurde. Verwenden Sie einen Ressourcennamen, um jeder Ressource benutzerfreundliche Metadaten zuzuordnen. Die Amazon Lex V2-API weist jeder Ressource eine eindeutige 10-stellige Ressourcen-ID zu. Alle Ressourcen haben einen Ressourcennamen. Sie können die folgenden Ressourcen umbenennen:

- Bot
- Absicht
- Slot-Typ
- Slot
- Alias

Sie können Ressourcen verwenden IDs , um Ihre Ressourcen zu lesen und zu ändern. Wenn Sie die AWS Command Line Interface oder die Amazon Lex V2-API für die Arbeit mit Amazon Lex V2 verwenden, IDs sind für bestimmte Befehle Ressourcen erforderlich.

Vereinfachtes Management von Lambda-Funktionen

In der Amazon Lex V2-API definieren Sie eine Lambda-Funktion pro Sprache statt einer Funktion für jede Absicht. Die Lambda-Funktion ist im Alias für die Sprache konfiguriert und wird sowohl für den

Dialog- als auch für den Fulfillment-Code-Hook verwendet. Sie können weiterhin wählen, ob Sie den Dialog- und den Fulfillment-Code-Hooks für jede Absicht unabhängig voneinander aktivieren oder deaktivieren möchten. Weitere Informationen finden Sie unter [Eine AWS Lambda Funktion in deinen Bot integrieren](#).

Granulare Einstellungen

Die Amazon Lex V2-API verschiebt den Schwellenwert für die Vertrauensbewertung bei der Klassifizierung von Sprache und Absicht vom Bot auf den Sprachbereich. Das Kennzeichen für die Stimmungsanalyse wechselt vom Bot-Bereich zum Alias-Bereich. Sitzungs-Timeout und Datenschutzeinstellungen im Bot-Bereich sowie Konversationsprotokolle im Alias-Bereich bleiben unverändert.

Standard-Fallback-Absicht

Die Amazon Lex V2-API fügt eine Standard-Fallback-Absicht hinzu, wenn Sie eine Sprache erstellen. Verwenden Sie es, um die Fehlerbehandlung für Ihren Bot zu konfigurieren, anstatt spezifische Eingabeaufforderungen zur Fehlerbehandlung zu verwenden.

Optimiertes Update der Sitzungsvariablen

Mit der Amazon Lex V2-API können Sie den Sitzungsstatus direkt mit den [RecognizeUtterance](#) Operationen [RecognizeText](#) und aktualisieren, ohne von der Sitzung abhängig zu sein APIs.

Erstellen von Amazon Lex V2-Ressourcen mit AWS CloudFormation

Amazon Lex V2 ist integriert mit AWS CloudFormation, ein Service, der Sie bei der Modellierung und Einrichtung Ihrer AWS Ressourcen unterstützt, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle gewünschten AWS Ressourcen beschreibt (z. B. Amazon Lex V2-Chatbots) und diese Ressourcen für Sie AWS CloudFormation bereitstellt und konfiguriert.

Wenn Sie Ihre Vorlage verwenden AWS CloudFormation, können Sie sie wiederverwenden, um Ihre Amazon Lex V2-Ressourcen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren AWS-Konten Regionen bereit.

Amazon Lex V2 und AWS CloudFormation Vorlagen

Um Ressourcen für Amazon Lex V2 und verwandte Services bereitzustellen und zu konfigurieren, müssen Sie [AWS CloudFormation Vorlagen](#) verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation -Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie AWS CloudFormation Designer verwenden, um Ihnen bei den ersten Schritten mit AWS CloudFormation Vorlagen zu helfen. Weitere Informationen finden Sie unter [Was ist AWS CloudFormation Designer?](#) im AWS CloudFormation Benutzerhandbuch.

Amazon Lex V2 unterstützt die Erstellung der folgenden Ressourcen in AWS CloudFormation:

- `AWS::Lex::Bot`
- `AWS::Lex::BotAlias`
- `AWS::Lex::BotVersion`
- `AWS::Lex::ResourcePolicy`

Weitere Informationen, einschließlich Beispielen für JSON- und YAML-Vorlagen für diese Ressourcen, finden Sie in der [Referenz zum Amazon Lex V2-Ressourcentyp](#) im AWS CloudFormation Benutzerhandbuch.

Erfahren Sie mehr über AWS CloudFormation

Weitere Informationen AWS CloudFormation dazu finden Sie in den folgenden Ressourcen:

- [AWS CloudFormation](#)
- [AWS CloudFormation benutzerhandbuch](#)
- [AWS CloudFormation API-Referenz](#)
- [AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle](#)

Dokumentenverlauf für Amazon Lex V2

- Letzte Aktualisierung der Dokumentation: 3. Juli 2025

In der folgenden Tabelle werden wichtige Änderungen in den einzelnen Versionen von Amazon Lex V2 beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Neues Feature	Sie können den AMAZON entIntent eingebaute Absicht, Ihren Amazon Lex V2-Bot mit Amazon Bedrock Agents und Knowledge Bases zu verbinden. Weitere Informationen finden Sie unter AMAZON. BedrockAgentIntent eingebaute Absicht .	3. Juli 2025
Aktualisierung der AWS verwalteten Richtlinie	Amazon Lex V2 hat die Richtlinie aktualisiert, um die Replikation von Tags zu ermöglichen. ResourceBasedPolicy Bei Verwendung von globaler Resilienz siehe AmazonLexReplicationPolicy .	24. Juni 2025
Neues Feature	Sie können jetzt Assisted NLU verwenden, um die Absichtsklassifizierung und die Slot-Auflösung in Amazon Lex V2 zu verbessern. Diese Funktion verwendet Large Language Models (LLMs), um die	11. Juni 2025

Genauigkeit zu erhöhen und gleichzeitig die konfigurierten Absichten und Slots Ihres Bots einzuhalten. Weitere Informationen finden Sie unter [Assisted NLU](#).

[Aktualisierung der Funktion \(03.06.2025\)](#)

Sie können 17 neue Sprachen mit Ihren benutzerdefinierten Vokabularen in Amazon Lex V2 verwenden. Weitere Informationen finden Sie unter [Benutzerdefinierte Vokabulare](#).

3. Juni 2025

[Neue Funktion \(2024-12-01\)](#)

Sie können den QinConnect integrierten Intent verwenden, um Ihren Amazon Lex V2-Bot mit Amazon Connect zu verbinden. Weitere Informationen finden Sie unter [QinConnect Integrierte Absicht](#).

01. Dezember 2024

[Neue Funktion \(2024-10-01\)](#)

Sie können den integrierten Amazon.Currency Slot für US-amerikanische und spanische Gebietsschemas in Amazon Lex V2 verwenden. [Weitere Informationen finden Sie unter Amazon.Currency](#).

1. Oktober 2024

[Aktualisierung der Funktion \(26.07.2020\)](#)

Die integrierte QnA-Absicht für generative KI unterstützt jetzt Bedrock Knowledge Base und Guardrails in Amazon Lex V2. [Weitere Informationen finden Sie unter Amazon.qn. Alntent](#)

26. Juli 2024

[Aktualisierung der Funktion \(10.07.2020\)](#)

Die integrierte QnA-Absicht für generative KI unterstützt jetzt die Modelle Anthropic Claude 3 Haiku und Anthropic Claude 3 Sonnet in Amazon Lex V2. Weitere Informationen finden Sie unter [Optimieren Sie die Erstellung und Leistung von Bots](#) mit generativer KI.

10. Juli 2024

[Aktualisierung der AWS verwalteten Richtlinie \(2024-05-10\)](#)

Amazon Lex V2 hat der [AmazonLexReadOnly](#) verwalteten Richtlinie neue Berechtigungen hinzugefügt, um Lesezugriff auf Bot-Ressourcen zu ermöglichen, die in anderen Regionen repliziert wurden.

10. Mai 2024

[Aktualisierung der AWS verwalteten Richtlinie \(15.04.2020\)](#)

Amazon Lex V2 hat der [AmazonLexFullAccess](#) verwalteten Richtlinie neue Berechtigungen hinzugefügt, um die Aktualisierung replizierter Bot-Ressourcen in anderen Regionen zu ermöglichen.

15. April 2024

[Regionale Erweiterung](#)

Amazon Lex V2 ist jetzt in AWS GovCloud (US-West) (us-gov-west-1) verfügbar.

22. März 2024

[Aktualisierung der AWS
verwalteten Richtlinie
\(2024-03-07\)](#)

Amazon Lex V2 hat der [AmazonLexReplicationPolicy](#) verwalteten Richtlinie neue Berechtigungen hinzugefügt, um die Aktualisierung replizierter Bot-Ressourcen in anderen Regionen zu ermöglichen.

7. März 2024

[Neue Funktion](#)

Sie können die Funktion `fn.Length ()` verwenden, um die Wertelänge eines Zeichenkettenwerts in Amazon Lex V2 zu bestimmen. Weitere Informationen finden Sie unter [Bedingte Verzweigung](#) — Funktionen.

4. März 2024

[Aktualisierung der Funktion
\(28.02.2021\)](#)

Der integrierte QnA-Steckplatz für generative KI-Funktionen in Amazon Lex V2 ist jetzt GA. Weitere Informationen finden Sie unter [Optimieren Sie die Erstellung und Leistung von Bots mit generativer KI.](#)

28. Februar 2024

[Aktualisierung der AWS
verwalteten Richtlinie
\(08.02.2020\)](#)

Amazon Lex V2 hat der [AmazonLexFullAccess](#) verwalteten Richtlinie neue Berechtigungen hinzugefügt, um die Replikation von Bot-Ressourcen in andere Regionen zu ermöglichen.

8. Februar 2024

Neue verwaltete Richtlinie	Amazon Lex V2 hat eine verwaltete Richtlinie hinzugefügt, die Berechtigungen zur Replikation von Bot-Ressourcen in anderen Regionen bietet. Weitere Informationen finden Sie unter AmazonLex ReplicationPolicy .	8. Februar 2024
Neue Funktion (2024-02-08)	Sie können Global Resiliency verwenden, um Ihren Bot in einer zweiten AWS-Region in Amazon Lex V2 zu replizieren. Weitere Informationen finden Sie unter Globale Resilienz .	8. Februar 2024
Neue Funktion (2023-11-29)	Sie können jetzt die generativen KI-Funktionen in Amazon Lex V2 nutzen. Weitere Informationen finden Sie unter Optimieren Sie die Erstellung und Leistung von Bots mit generativer KI .	29. November 2023
Neue Funktion (2023-11-08)	Amazon Lex V2 kann jetzt die selektive Protokollierung verwenden, um and/or Textaudio auf Intent- oder Slot-Ebene aufzunehmen. Weitere Informationen finden Sie unter Selektive Protokollierung .	8. November 2023

Neue Funktion (2023-08-17)	Amazon Lex V2 kann jetzt einen integrierten Steckplatz verwenden, um Antworten zu ermitteln, die Yes/No/Maybe/Don't know sind. Weitere Informationen finden Sie unter Integrierte Steckplatztypen .	17. August 2023
Neue Funktion (2023-07-18)	Sie können die Leistungskennzahlen von Intents und Slots sowie andere Konversationsmetriken im Analyse-Dashboard einsehen. Weitere Informationen finden Sie unter Analytics .	18. Juli 2023
Neue Funktion (2023-06-06)	Mit der Test Workbench können Sie die Genauigkeit und die Erfolgserfüllung Ihres Bots verbessern. Weitere Informationen finden Sie unter Test Workbench .	6. Juni 2023
Neue Funktion (2023-02-23)	Sie können jetzt Bots aus einer Vorlage für einige beliebte Geschäftsbereiche erstellen. Weitere Informationen finden Sie unter Bot-Vorlagen .	23. Februar 2023

Neue Funktion (2023-02-09)	Sie können jetzt mehrere Bots zu einem Netzwerk von Bots kombinieren, um ein integriertes Kundenerlebnis zu schaffen. Weitere Informationen finden Sie unter Netzwerk von Bots .	9. Februar 2023
Neue Funktion (06.12.2022)	Amazon Lex V2 unterstützt jetzt die Gebietsschemas Golfarabisch (Vereinigte Arabische Emirate), Kantonesisch (Hongkong), Finnisch (Finnland), Norwegisch (Norwegen), Polnisch (Polen) und Schwedisch (Schweden). Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	6. Dezember 2022
Auf AWS verwaltete Richtlinie aktualisiert	Amazon Lex V2 hat der AmazonLexReadOnly verwalteten Richtlinie neue Berechtigungen hinzugefügt, um die Anzeige von benutzerdefinierten Vokabelementen zu ermöglichen.	29. November 2022

[Neue Funktion \(07.11.2022\)](#)

Amazon Lex V2 kann mithilfe der Konsole oder zur Anpassung der APIs Sprache-zu-Text-Ausgabe eine alternative Darstellung für eine Phrase oder ein Wort anzeigen. Weitere Informationen finden Sie unter [Benutzerdefinierte s Vokabular für die Spracherkennung erstellen](#).

7. November 2022

[Neue Funktion \(28.10.2022\)](#)

Amazon Lex V2 kann einem Artikelelement ein Gewichtungsattribut hinzufügen, das den Grad angibt, in dem die Phrase bei der Spracherkennung verstärkt wird. Weitere Informationen finden Sie unter [Grammatikgewichte](#).

28. Oktober 2022

[Neue Funktion \(19.10.2022\)](#)
[\(19.10.2022\) - 2](#)

Amazon Lex V2 kann eine alternative Darstellung für eine Phrase oder ein Wort in der endgültigen Sprach- und Textausgabe anzeigen. Weitere Informationen finden Sie unter [Benutzerdefinierte s Vokabular für die Spracherkennung erstellen](#).

19. Oktober 2022

[Neue Funktion \(19.10.2022\)](#) [\(19.10.2022\)](#)

Amazon Lex V2 kann verwendet werden, um formlose Eingaben des Endbenutzers, die aus Wörtern oder Zeichen bestehen, mithilfe von zu erfassen. `AMAZON.FreeFormInput`. Weitere Informationen finden Sie unter [Integrierte Steckplatztypen](#).

19. Oktober 2022

[Neue Funktion \(14.10.2022\)](#)

Amazon Lex V2 unterstützt jetzt die Sprachen Hindi (Indien) und Niederländisch (Niederlande). Weitere Informationen finden Sie unter [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#).

14. Oktober 2022

[Neue Funktion \(22.09.2022\)](#)

Die Art und Weise, wie Amazon Lex V2 Benutzereingaben verwaltet, wurde aktualisiert. Jetzt können Sie wählen, ob Amazon Lex V2 Text-, Audio- oder DTMF-Eingaben zu einem beliebigen Zeitpunkt im Konversationsablauf akzeptiert. Weitere Informationen finden Sie unter [Konfigurierbare Attribute](#).

22. September 2022

[Neue Funktion \(14.09.2022\)](#)

Die Art und Weise, wie Amazon Lex V2 Konversationsflüsse verwaltet, wurde aktualisiert. Visual Conversation Builder ist ein Drag-and-Drop-Konversationsgenerator, mit dem Konversationspfade einfach entworfen und visualisiert werden können. Weitere Informationen finden Sie unter [Visual Conversation Builder](#).

14. September 2022

[Neue Funktion \(09.09.2022\)](#)

Die Art und Weise, wie Amazon Lex V2 komplexe Slots erstellt, wurde aktualisiert. Sie können jetzt komplexe Teilbereiche innerhalb von Slots erstellen, um Absichten in komplexen Konversationsdesigns zu verwalten. Weitere Informationen finden Sie unter [Zusammengesetzte Slots erstellen](#).

09. September 2022

[Neue Funktion \(17.08.2022\)](#)

Die Art und Weise, wie Amazon Lex V2 den Fluss der Konversationspfade mit Ihren Benutzern verwaltet, wurde aktualisiert. Sie können jetzt komplexe Konversationspfade erstellen, indem Sie den nächsten Schritt in der Konversation anordnen. Weitere Informationen finden Sie unter [Konversationspfade erstellen](#).

17. August 2022

[Neue Funktion \(05.07.2022\)](#)

Die Art und Weise, wie Amazon Lex V2 den Konversationsfluss mit Ihren Benutzern verwaltet, wurde aktualisiert. Sie können jetzt komplexe Konversationen erstellen, indem Sie die Eingabeaufforderungen ordnen. Weitere Informationen finden Sie unter [Prompts konfigurieren](#).

5. Juli 2022

[Neue Funktion \(03.05.2022\)](#)

Die Art und Weise, wie Amazon Lex V2 den Konversationsfluss mit Ihren Benutzern verwaltet, wurde aktualisiert. Sie können jetzt komplexe Konversationen mithilfe von Bedingungen erstellen. Weitere Informationen finden Sie unter [Grundlegendes zu den neuen Konversationsabläufen](#).

3. Mai 2022

[Neue Funktion \(22.03.2022\)](#)

Branchenspezifische Beispiellammatiken für den integrierten Grammatik-Slot-Typ hinzugefügt. Weitere Informationen finden Sie unter [Branchengrammatiken](#).

22. März 2022

[Neue Funktion \(18.03.2022\)](#)

Dokumentation zur Integration von Amazon Lex V2 mit dem Amazon Chime SDK hinzugefügt. Weitere Informationen finden Sie unter [Amazon Chime SDK](#).

18. März 2022

[Neue Funktion \(27.01.2022\)](#)

Amazon Lex V2 bietet jetzt Konfidenzwerte für Sprachtranskriptionen. Verwenden Sie die Punktzahl, um die richtige Antwort des Benutzers zu ermitteln. Weitere Informationen finden Sie unter [Verwenden von Konfidenzwerten für die Sprachtranskription](#).

27. Januar 2022

[Neue Funktion \(13.01.2022\)](#)

Du kannst jetzt kontextuelle und dynamische Hinweise zu Slots hinzufügen, um die Genauigkeit deines Bots zu verbessern. Weitere Informationen findest du unter [Hinweise verwenden, um die Genauigkeit zu verbessern](#).

13. Januar 2022

[Neue Funktion \(12.01.2022\)](#)

Amazon Lex V2 bietet Unterstützung für benutzerdefinierte Vokabulare, um die Spracherkennung für Audioeingaben zu verbessern. Weitere Informationen finden Sie unter [Benutzerdefiniertes Vokabular zur Verbesserung der Spracherkennung erstellen](#).

12. Januar 2022

[Neue Funktion \(07.01.2022\)](#)

Amazon Lex V2 unterstützt jetzt AWS PrivateLink. Weitere Informationen finden Sie unter [VPC-Endpoints \(AWS PrivateLink\)](#).

7. Januar 2022

Neue Funktion (03.01.2022)	Amazon Lex V2 unterstützt jetzt das Gebietsschema Katalanisch (Spanien). Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	3. Januar 2022
Neue Funktion (20.12.2021) (20.12.2021) — 2	AWS CloudFormation unterstützt jetzt Amazon Lex V2. Weitere Informationen finden Sie unter AWS CloudFormation Ressourcen .	20. Dezember 2021
Neue Funktion (20.12.2021) (20.12.2021)	Sie können jetzt Slot-Typen mit Ihrer eigenen benutzerdefinierten Grammatik erstellen. Weitere Informationen finden Sie unter Verwenden eines benutzerdefinierten Grammatik-Slot-Typs .	20. Dezember 2021
Neue Funktion (16.12.2021)	Amazon Lex V2 unterstützt jetzt die Gebietsschemas Portugiesisch (Brasilien), Portugiesisch (Portugal) und Mandarin (VR China). Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	16. Dezember 2021

[Neue Funktion \(01.12.2021\)](#)

Amazon Lex V2 bietet jetzt eine Vorschau des Automated Chatbot Designers, um Ihnen den Einstieg in die Erstellung eines Chatbots aus Contact-Center-Protokollen zu erleichtern. Weitere Informationen finden Sie unter [Verwenden des automatisierten Chatbot-Designers \(Vorschau\)](#).

1. Dezember 2021

[Neue Funktion \(19.11.2021\)](#)
[\(19.11.2021\) — 2](#)

Sie können jetzt Amazon Polly Neural Text to Speech (NTTS)-Stimmen für Audiogespäche mit Ihren Benutzern verwenden. Weitere Informationen finden Sie unter [Stimmen in Amazon Polly](#).

19. November 2021

[Neue Funktion \(19.11.2021\)](#)
[\(19.11.2021\)](#)

Sie können jetzt spell-by-word Stile für die Eingabe von Slot-Werten verwenden spell-by-letter, die Amazon Lex V2 nur schwer verstehen kann. Weitere Informationen finden Sie unter [Verwenden von Rechtschreibstilen zur Erfassung von Slot-Werten](#).

19. November 2021

[Neue Funktion \(09.11.2021\)](#)

Amazon Lex V2 unterstützt jetzt das Gebietsschema Englisch (Südafrika). Weitere Informationen finden Sie unter [Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas](#).

9. November 2021

Neue Funktion (05.11.2021)	Amazon Lex V2 unterstützt jetzt das deutsche (Österreich) Gebietsschema. Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	5. November 2021
Neue Funktion (07.10.2021)	Sie können Benutzern jetzt Aktualisierungsnachrichten zur Verfügung stellen, die zu Beginn einer Fulfillment-Funktion und in regelmäßigen Abständen, während die Funktion ausgeführt wird, abgespielt werden. Sie können auch Nachrichten erstellen, die den Benutzer über den Status der Ausführung informieren, wenn die Funktion abgeschlossen ist. Weitere Informationen finden Sie unter Updates zum Status der Auftragsabwicklung konfigurieren .	7. Oktober 2021
Erweiterung der Region (22.09.2021)	Amazon Lex V2 ist jetzt in Afrika (Kapstadt) (af-south-1) und im asiatisch-pazifischen Raum (Seoul) (ap-northeast-2) verfügbar.	22. September 2021
Neue Funktion (22.09.2021)	Sie können jetzt Statistiken für die Äußerungen einsehen, die Ihre Benutzer an Ihren Bot senden. Weitere Informationen finden Sie unter Statistiken zu Äußerungen anzeigen .	22. September 2021

Neue Funktion (09.09.2021)	Amazon Lex V2 unterstützt jetzt das Gebietsschema Koreanisch (Korea). Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	9. September 2021
Neue Funktion (27.07.2021)	Amazon Lex V2 bietet jetzt einen integrierten Steckplatztyp für britische Postleitzahlen. Weitere Informationen finden Sie unter AMAZON.UKPostalCode .	27. Juli 2021
Neue Funktion (15.07.2021)	Amazon Lex V2 unterstützt jetzt das englische (indische) Gebietsschema. Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	15. Juli 2021
Neue Funktion (13.07.2021)	Amazon Lex V2 bietet jetzt ein Tool zur Migration eines Bots von Amazon Lex V1 zur Amazon Lex V2-API. Weitere Informationen finden Sie unter Einen Bot migrieren im Amazon Lex Developer Guide.	13. Juli 2021

[Neue Funktion \(26.06.2021\)](#)

Sie können Ihren Amazon Lex V2-Bot jetzt so konfigurieren, dass er bereits gefüllte Slots erneut auslöst, indem Sie die Slot-Werte auf Null setzen und den Konversationsfluss so konfigurieren, dass er zurückläuft. Diese Funktion ermöglicht dynamischere Konversationen auf der Grundlage von Kundenantworten. Weitere Informationen finden Sie unter Slots [neu entlocken](#).

26. Juni 2021

[Neue Dokumentation](#)

Amazon Lex V2 bietet jetzt umfassende Richtlinien und bewährte Methoden für die Bot-Entwicklung, mit denen Sie effektivere Konversationserlebnisse schaffen können. Weitere Informationen finden Sie unter [Richtlinien für die Entwicklung von Amazon Lex V2-Bots](#).

26. Juni 2021

[Neue Funktion \(15.06.2021\)](#)

Mit Amazon Lex V2 können Sie jetzt mehrere Werte für einen einzelnen Slot in englischer Sprache (USA) akzeptieren. Weitere Informationen finden Sie unter [Verwenden mehrerer Werte in einem Slot](#).

15. Juni 2021

Neue Funktion (2021-06-11)	Sie können jetzt größere Bots für englische Sprachen erstellen. Weitere Informationen finden Sie unter Kontingente .	11. Juni 2021
Neue Funktion (20.05.2021)	Verwenden Sie ressourcenbasierte Richtlinien von Amazon Lex V2, um den Zugriff auf Ihre Bots und Bot-Aliase zu verwalten. Weitere Informationen finden Sie unter Ressourcenbasierte Richtlinien in Amazon Lex V2 .	20. Mai 2021
Neue Funktion (18.05.2021)	Mit Amazon Lex V2 können Sie jetzt Bots und Bot-Gebietsschemas importieren und exportieren. Sie können diese Funktion verwenden, um Bots und Bot-Gebietsschemas zwischen Konten und AWS Regionen zu kopieren. Weitere Informationen finden Sie unter Importieren und Exportieren .	18. Mai 2021
Erweiterung der Region (17.05.2021)	Amazon Lex V2 ist jetzt in Kanada (Central) (ca-central-1) verfügbar.	17. Mai 2021
Neue Funktion (2021-04-01)	Amazon Lex V2 unterstützt jetzt das Gebietsschema Japanisch (Japan). Weitere Informationen finden Sie unter Von Amazon Lex V2 unterstützte Sprachen und Gebietsschemas .	1. April 2021

[Neue Funktion \(12.03.2021\)](#)

Amazon Lex V2 unterstützt jetzt drei neue integrierte Steckplatztypen: AMAZON.City, AMAZON.Country, und AMAZON.State.

12. März 2021

[Neues Handbuch](#)

Dies ist die erste Version des Amazon Lex V2-Benutzerhandbuchs.

21. Januar 2021

API-Referenz

Die [API-Referenz](#) ist jetzt ein separates Dokument.

AWS Glossar

Die neueste AWS Terminologie finden Sie im [AWS Glossar](#) in der AWS-Glossar Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.